Spectral Temporal Graph Neural Network for Multivariate Time-series Forecasting

Defu Cao[1,*,†], **Yujing Wang**[1,2,†], **Juanyong Duan**[2], **Ce Zhang**[3], **Xia Zhu**[2]
Conguri Huang[2], **Yunhai Tong**[1], **Bixiong Xu**[2], **Jing Bai**[2], **Jie Tong**[2], **Qi Zhang**[2]
[1]Peking University [2]Microsoft [3]ETH Zürich
{cdf, yujwang, yhtong}@pku.edu.cn ce.zhang@inf.ethz.ch
{juaduan, zhuxia, conhua, bix, jbai, jietong, qizhang}@microsoft.com

Abstract

Multivariate time-series forecasting plays a crucial role in many real-world applications. It is a challenging problem as one needs to consider both intra-series temporal correlations and inter-series correlations simultaneously. Recently, there have been multiple works trying to capture both correlations, but most, if not all of them only capture temporal correlations in the time domain and resort to pre-defined priors as inter-series relationships.

In this paper, we propose Spectral Temporal Graph Neural Network (StemGNN) to further improve the accuracy of multivariate time-series forecasting. StemGNN captures inter-series correlations and temporal dependencies *jointly* in the *spectral domain*. It combines Graph Fourier Transform (GFT) which models inter-series correlations and Discrete Fourier Transform (DFT) which models temporal dependencies in an end-to-end framework. After passing through GFT and DFT, the spectral representations hold clear patterns and can be predicted effectively by convolution and sequential learning modules. Moreover, StemGNN learns inter-series correlations automatically from the data without using pre-defined priors. We conduct extensive experiments on ten real-world datasets to demonstrate the effectiveness of StemGNN. Code is available at `https://github.com/microsoft/StemGNN/`

1 Introduction

Time-series forecasting plays a crucial role in various real-world scenarios, such as traffic forecasting, supply chain management and financial investment. It helps people to make important decisions if the future evolution of events or metrics can be estimated accurately. For example, we can modify our driving route or reschedule an appointment if there is a severe traffic jam anticipated in advance. Moreover, if we can forecast the trend of COVID-19 in advance, we are able to reschedule important events and take quick actions to prevent the spread of epidemic.

Making accurate forecasting based on historical time-series data is challenging, as both intra-series temporal patterns and inter-series correlations need to be modeled *jointly*. Recently, deep learning models shed new lights on this problem. On one hand, Long Short-Term Memory (LSTM) [12], Gated Recurrent Units (GRU) [6], Gated Linear Units (GLU) [8] and Temporal Convolution Networks (TCN) [3] have achieved promising results in temporal modeling. At the same time, Discrete Fourier Transform (DFT) is also useful for time-series analysis. For instance, State Frequency Memory (SFM) network [39] combines the advantages of DFT and LSTM jointly for stock price prediction; Spectral Residual (SR) model [28] leverages DFT and achieves state-of-the-art performances

[*] The work was done when the author did internship at Microsoft.
[†] Equal Contribution

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

in time-series anomaly detection. Another important aspect of multivariate time-series forecasting is to model the correlations among multiple time-series. For example, in the traffic forecasting task, adjacent roads naturally interplay with each other. Current state-of-the-art models highly depend on Graph Convoluational Networks (GCNs) [16] originated from the theory of Graph Fourier Transform (GFT). These models [38, 20] stack GCN and temporal modules (e.g., LSTM, GRU) directly, which only capture temporal patterns in the time domain and require a pre-defined topology of inter-series relationships.

In this paper, our goal is to better model the intra-series temporal patterns and inter-series correlations jointly. Specifically, we hope to combine *both* the advantages of GFT and DFT, and model multivariate time-series data entirely in the *spectral domain*. The intuition is that after GFT and DFT, the spectral representations could hold clearer patterns and can be predicted more effectively. It is non-trivial to achieve this goal. The key technical contribution of this work is a carefully designed StemGNN (Spectral Temporal Graph Neural Network) block. Inside a StemGNN block, GFT is first applied to transfer structural multivariate inputs into spectral time-series representations, while different trends can be decomposed to *orthogonal* time-series. Furthermore, DFT is utilized to transfer each univariate time-series into the frequency domain. After GFT and DFT, the spectral representations become easier to be recognized by convolution and sequential modeling layers. Moreover, a latent correlation layer is incorporated in the end-to-end framework to learn inter-series correlations *automatically*, so it does not require multivariate dependencies as priors. Moreover, we adopt both forecasting and backcasting output modules with a shared encoder to facilitate the representation capability of multivariate time-series.

The **main contributions** of this paper are summarized as follows:

- To the best of our knowledge, StemGNN is the first work that represents both intra-series and inter-series correlations jointly in the *spectral domain*. It encapsulates the benefits of GFT, DFT and deep neural networks simultaneously and collaboratively. Ablation studies further prove the effectiveness of this design.
- StemGNN enables a data-driven construction of dependency graphs for different time-series. Thereby the model is general for all multivariate time-series without pre-defined topologies. As shown in the experiments, automatically learned graph structures have good interpretability and work even better than the graph structures defined by humans.
- StemGNN achieves state-of-the-art performances on nine public benchmarks of multivariate time-series forecasting. On average, it outperforms the best baseline by 8.1% on MAE an 13.3% on RMSE. A case study on COVID-19 further shows its feasibility in real scenarios.

2 Related Work

Time-series forecasting is an emerging topic in machine learning, which can be divided into two major categories: *univariate techniques* [25, 27, 23, 33, 39, 24, 23] and *multivariate techniques* [29, 26, 20, 38, 3, 35, 30, 19, 18] *Univariate techniques* analyze each individual time-series separately without considering the correlations between different time-series [27]. For example, FC-LSTM [36] forecasts univariate time-series with LSTM and fully-connected layers. SMF [39] improves the LSTM model by breaking down the cell states of a given univariate time-series into a series of different frequency components through Discrete Fourier Transform (DFT). N-BEATS [24] proposes a deep neural architecture based on a deep stack of fully-connected layers with basis expansion.

Multivariate techniques consider a collection of multiple time-series as a unified entity [29, 11, 21]. TCN [3] is a representative work in this category, which treats the high-dimensional data entirely as a tensor input and considers a large receptive field through dilated convolutions. LSTNet [17] uses convolution neural network (CNN) and recurrent neural network (RNN) to extract short-term local dependence patterns among variables and discover long-term patterns of time series. DeepState [26] marries state space models with deep recurrent neural networks and learns the parameters of the entire network through maximum log likelihood. DeepGLO [30] leverages both global and local features during training and forecasting. The global component in DeepGLO is based on matrix factorization and is able to capture global patterns by representing each time-series as a linear

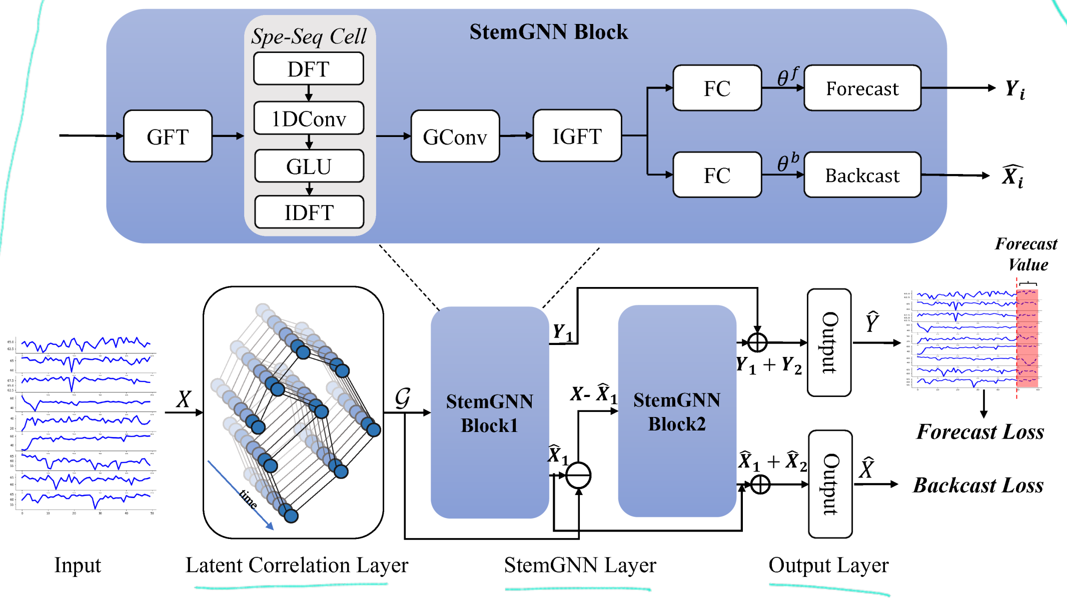


Figure 1: The overall architecture of Spectral Temporal Graph Neural Network.

combination of basis components. There is another category of works using graph neural networks to capture the correlations between different time-series explicitly. For instance, DCRNN [20] incorporates both spatial and temporal dependencies in the convolutional recurrent neural network for traffic forecasting. ST-GCN [38] is another deep learning framework for traffic prediction, integrating graph convolution and gated temporal convolution through spatio-temporal convolutional blocks. GraphWaveNet [35] combines graph convolutional layers with adaptive adjacency matrices and dilated casual convolutions to capture spatio-temporal dependencies. However, most of them either ignore the inter-series correlations or require a dependency graph as priors. In addition, Fourier transform has showed its advantages in previous work, especially in Joint Fourier Transform (JFT) [10], and its application to forecasting can be found in [14, 21]. In spite of this, none of existing solutions capture temporal patterns and multivariate dependencies jointly in the *spectral domain*. In this paper, StemGNN is proposed to address these issues. We refer you to recent surveys [34, 41, 40] for more details about related works.

3 Problem Definition

In order to emphasize the relationships among multiple time-series, we formulate the problem of multivariate time-series forecasting based on a data structure called *multivariate temporal graph*, which can be denoted as $\mathcal{G} = (X, W)$. $X = \{x_{it}\} \in \mathbb{R}^{N \times T}$ stands for the multivariate time-series input, where N is the number of time-series (nodes), and T is the number of timestamps. We denote the observed values at timestamp t as $X_t \in \mathbb{R}^N$. $W \in \mathbb{R}^{N \times N}$ is the adjacency matrix, where $w_{ij} > 0$ indicates that there is an edge connecting nodes i and j, and w_{ij} indicates the strength of this edge.

Given observed values of previous K timestamps $X_{t-K}, \cdots, X_{t-1}$, the task of *multivariate time-series forecasting* aims to predict the node values in a multivariate temporal graph $\mathcal{G} = (X, W)$ for the next H timestamps, denoted by $\hat{X}_t, \hat{X}_{t+1}, \cdots, \hat{X}_{t+H-1}$. These values can be inferred by the forecasting model F with parameter Φ and a graph structure $\mathcal{G}$, where $\mathcal{G}$ can be input as prior or automatically inferred from data.

$$\hat{X}_t, \hat{X}_{t+1} ..., \hat{X}_{t+H-1} = F(X_{t-K}, ..., X_{t-1}; \mathcal{G}; \Phi). \qquad (1)$$

4 Spectral Temporal Graph Neural Network

4.1 Overview

Here, we propose Spectral Temporal Graph Neural Network (StemGNN) as a general solution for multivariate time-series forecasting. The overall architecture of StemGNN is illustrated in Figure 1. The multivariate time-series input X is first fed into a latent correlation layer, where the graph structure and its associated weight matrix W can be inferred automatically from data.

Next, the graph $\mathcal{G} = (X, W)$ serves as input to the StemGNN layer consisting of two residual StemGNN blocks. A StemGNN block is by design to model structural and temporal dependencies inside multivariate time-series jointly in the spectral domain (as visualized in the top diagram of Figure 1). It contains a sequence of operators in a well-designed order. First, a Graph Fourier Transform (GFT) operator transforms the graph $\mathcal{G}$ into a spectral matrix representation, where the univariate time-series for each node becomes linearly independent. Then, a Discrete Fourier Transform (DFT) operator transforms each univariate time-series component into the frequency domain. In the frequency domain, the representation is fed into 1D convolution and GLU sub-layers to capture feature patterns before transformed back to the time domain through inverse DFT. Finally, we apply graph convolution on the spectral matrix representation and perform inverse GFT.

After the StemGNN layer, we add an output layer composed of GLU and fully-connected (FC) sub-layers. There are two kinds of outputs in the network. The forecasting outputs Y_i are trained to generate the best estimation of future values, while the backcasting outputs $\hat{X}_i$ are used in an auto-encoding fashion to enhance the representation power of multivariate time-series. The final loss function can be formulated as a combination of both forecasting and backcasting losses:

$$\mathcal{L}(\hat{X}, X; \Delta_\theta) = \sum_{t=0}^{T} ||\hat{X}_t - X_t||_2^2 + \sum_{t=K}^{T} \sum_{i=1}^{K} ||B_{t-i}(X) - X_{t-i}||_2^2 \qquad (2)$$

where the first term represents for the forecasting loss and the second term denotes the backcasting loss. For each timestamp t, $\{X_{t-K}, ..., X_{t-1}\}$ are input values within a sliding window, and X_t is the ground truth value to forecast; $\hat{X}_t$ is the forecasted value for the timestamp t, and $\{B_{t-K}(X), ..., B_{t-1}(X)\}$ are reconstructed values from the backcasting module. B indicates the entire network that generates backcasting output, Δ_θ denotes all parameters in the network.

In the inference phase, we adopt a rolling strategy for multi-step prediction. First, $\hat{X}_t$ is predicted by taking $\{X_{t-K}, ..., X_{t-1}\}$ as input. Then, the input will be changed to $\{X_{t-K+1}, ..., X_{t-1}, \hat{X}_t\}$ for predicting the next timestamp $\hat{X}_{t+1}$. By applying this rolling strategy consecutively, we can obtain forecasting values of the next H timestamps.

4.2 Latent Correlation Layer

GNN-based approach requires a graph structure when modeling multivariate time-series. It can be constructed by human knowledge (such as road network in traffic forecasting), but sometimes we do not have a pre-defined graph structure as prior. In order to serve general cases, we leverage the self-attention mechanism to learn latent correlations between multiple time-series automatically. In this way, the model emphasizes task-specific correlations in a data-driven fashion.

First, the input $X \in \mathbb{R}^{N \times T}$ is fed into a Gated Recurrent Unit (GRU) layer, which calculates the hidden state corresponding to each timestamp t sequentially. Then, we use the last hidden state R as the representation of entire time-series and calculate the weight matrix W by the self-attention mechanism as follows,

$$Q = RW^Q, K = RW^K, W = \text{Softmax}(\frac{QK^T}{\sqrt{d}}) \qquad (3)$$

where Q and K denote the representation for query and key, which can be calculated by linear projections with learnable parameters W^Q and W^K in the attention mechanism, respectively; and d is the hidden dimension size of Q and K. The output matrix $W \in \mathbb{R}^{N \times N}$ is served as the adjacency weight matrix for graph $\mathcal{G}$. The overall time complexity of self-attention is $O(N^2 d)$.

4.3 StemGNN Block

The StemGNN layer is constructed by stacking multiple *StemGNN blocks* with skip connections. A *StemGNN block* is designed by embedding a Spectral Sequential (Spe-Seq) Cell into a Spectral Graph Convolution module. In this section, we first introduce the motivation and architecture of the StemGNN block, and then briefly describe the Spe-Seq Cell and Spectral Graph Convolution module separately.

StemGNN Block Spectral Graph Convolution has been widely used in time-series forecasting task due to its extraordinary capability of learning latent representations of multiple time-series in the spectral domain. The key component is applying Graph Fourier Transform (GFT) to capture inter-series relationships. It is worth noting that the output of GFT is also a multivariate time-series while GFT does not learn intra-series temporal relationships explicitly. Therefore, we can utilize Discrete Fourier Transform (DFT) to learn the representations of the input time-series on the trigonometric basis in the frequency domain, which captures the repeated patterns in the periodic data or the auto-correlation features among different timestamps. Motivated by this, we apply the Spe-Seq Cell on the output of GFT to learn temporal patterns in the frequency domain. Then the output of the Spe-Seq Cell is processed by the rest components of Spectral Graph Convolution.

Our model can also be extended to multiple channels. We apply GFT and Spe-Seq Cell on each individual channel X_i of input data and sum the results after graph convolution with kernel $\Theta_{\cdot j}$. Next, Inverse Graph Fourier Transform (IGFT) is applied on the sum to obtain the jth channel Z_j of the output, which can be written as follows,

$$Z_j = \mathcal{GF}^{-1}\left(\sum_i g_{\Theta_{ij}}(\Lambda_i)\mathcal{S}(\mathcal{GF}(X_i))\right). \tag{4}$$

Here $\mathcal{GF}$, $\mathcal{GF}^{-1}$ and $\mathcal{S}$ denote GFT, IGFT, and Spe-Seq Cell respectively, Θ_{ij} is the graph convolution kernel corresponding to the ith input and the jth output channel, and Λ_i is the eigenvalue matrix of normalized Laplacian and the number of eigenvectors used in GFT is equivalent to the multivariate dimension (N) without dimension reduction. After that we concatenate each output channel Z_j to obtain the final result Z.

Following [24], we use learnable parameters to represent basis vectors V and a fully-connected layer to generate basis expansion coefficients θ based on Z. Then the output can be calculated by a combination of different bases: $Y = V\theta$. We have two branches of this module in the StemGNN block, one to forecast future values, namely forecasting branch, and the other to reconstruct history values, namely backcasting branch (denoted by B). The backcasting branch helps regulate the functional space for the block to represent time-series data.

Furthermore, we employ residual connections to stack multiple StemGNN blocks to build deeper models. In our case, we use two StemGNN blocks. The second block tries to approximate the residual between the ground-truth values and the reconstructed values from the first block. Finally, the outputs from both blocks are concatenated and fed into GLU and fully-connected layers to generate predictions.

Spectral Sequential Cell (Spe-Seq Cell) The Spe-Seq Cell $\mathcal{S}$ aims to decompose each individual time-series after GFT into frequency basis and learn feature representations on them. It consists of four components in order: Discrete Fourier Transform (DFT, $\mathcal{F}$), 1D convolution, GLU and Inverse Discrete Fourier Transform (IDFT, $\mathcal{F}^{-1}$), where DFT and IDFT transforms time-series data between temporal domain and frequency domain, while 1D convolution and GLU learn feature representations in the frequency domain. Specifically, the output of DFT has real part ($\hat{X}_u^r$) and imaginary part ($\hat{X}_u^i$), which are processed by the same operators with different parameters in parallel. The operations can be formulated as:

$$M^*(\hat{X}_u^*) = \text{GLU}(\theta_\tau^*(\hat{X}_u^*), \theta_\tau^*(\hat{X}_u^*)) = \theta_\tau^*(\hat{X}_u^*) \odot \sigma^*(\theta_\tau^*(\hat{X}_u^*)), * \in \{r, i\} \tag{5}$$

where θ_τ^* is the convolution kernel with the size of 3 in our experiments, $\odot$ is the Hadamard product and nonlinear sigmoid gate σ^* determines how much information in the current input is closely related to the sequential pattern. Finally, the result can be obtained by $M^r(\hat{x}_u^r) + iM^i(\hat{x}_u^i)$, and IDFT is applied on the final output.

Spectral Graph Convolution The Spectral Graph Convolution [16] is composed of three steps. (1) The multivariate time-series input is projected to the spectral domain by GFT. (2) The spectral representation is filtered by a graph convolution operator with learnable kernels. (3) Inverse Graph Fourier Transform (IGFT) is applied on the spectral representation to generate final output.

Graph Fourier Transform (GFT) [9] is a basic operator for Spectral Graph Convolution. It projects the input graph to an orthonormal space where the bases are constructed by eigenvectors of the normalized graph Laplacian. The normalized graph Laplacian [1] can be computed as: $L = I_N - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$, where $I_N \in \mathbb{R}^{N \times N}$ is the identity matrix and D is the diagonal degree matrix with $D_{ii} = \sum_j W_{ij}$. Then, we perform eigenvalue decomposition on the Laplacian matrix, forming $L = U \Lambda U^T$, where $U \in \mathbb{R}^{N \times N}$ is the matrix of eigenvectors and Λ is a diagonal matrix of eigenvalues. Given multivariate time-series $X \in \mathbb{R}^{N \times T}$, the operators of GFT and IGFT are defined as $\mathcal{GF}(X) = U^T X = \hat{X}$ and $\mathcal{GF}^{-1}(\hat{X}) = U\hat{X}$ respectively. The graph convolution operator is implemented as a function $g_\Theta(\Lambda)$ of eigenvalue matrix Λ with parameter Θ. The overall time complexity is $O(N^3)$

5 Experiments

5.1 Setup

Table 1: Summary of Datasets

	METR-LA	PEMS-BAY	PEMS07	PEMS03	PEMS04	PEMS08	Solar	Electricity	ECG5000	COVID-19
# of nodes	207	325	228	358	307	170	137	321	140	25
# of timesteps	34,272	52,116	12,672	26,209	16,992	17,856	52,560	26,304	5,000	110
Granularity	5min	5min	5min	5min	5min	5min	10min	1hour	-	1day
Start time	9/1/2018	1/1/2018	7/1/2016	5/1/2012	7/1/2017	3/1/2012	1/1/2006	1/1/2012	-	1/22/2020

We compare the performances of StemGNN on nine public datasets, ranging from traffic, energy and electrocardiogram domains with other state-of-the-art models, including FC-LSTM [32], SFM [39], N-BEATS [24], DCRNN [20], LSTNet [17], ST-GCN [38], DeepState [26], TCN [3], Graph Wavenet [35] and DeepGLO [30]. We tune the hyper-parameters on the validation data by grid search for StemGNN. Finally, the channel size of each graph convolution layer is set as 64 and the kernel size of 1D convolution is 3. Following [38], we adopt the RMSprop optimizer, and the number of training epochs is set as 50. The learning rate is initialized by 0.001 and decayed with rate 0.7 after every 5 epochs. We use the Mean Absolute Errors (MAE) [13], Mean Absolute Percentage Errors (MAPE) [13], and Root Mean Squared Errors (RMSE) [13] to measure the performances, which are averaged by H steps in multi-step prediction. We report the performances of baseline models in their original publications unless otherwise stated. The dataset statistics are summarized in Table 1.

We conduct the dataset into three part for training, validation and testing with a ratio of 6:2:2 on PEMS03, PMES04, PEMS08, and 7:2:1 on META-LA, PEMS-BAY, PEMS07, Solar, Electricity and ECG. The inputs of ECG are normalized by min-max normalization following [5]. Besides, the inputs are normalized by Z-Score method [24]. That means StemGNN is trained on normalized input where each time-series in the training set is re-scaled as $X_{in} = (X_{in} - \mu(X_{in}))/\sigma(X_{in})$, where μ and σ denote the mean and standard deviation respectively. More details descriptions about datasets, evaluation metrics, and experimental settings can be found in Appendix B, C and D.

5.2 Results

The evaluation results are summarized in Table 2, and more details can be found in Appendix E.1.Generally, StemGNN establishes a new state-of-the-art on most of the datasets. Furthermore, the model does not need apriori topology and demonstrates the feasibility of learning latent correlations automatically. In particular, on all datasets, StemGNN improves an average of 8.1% on MAE and 13.3% on RMSE compared to the best baseline for each dataset. In terms of baseline models, FC-LSTM only takes temporal information into consideration and performs estimation in the time domain. SFM models the time-series data in the frequency domain and shows stable improvement over FC-LSTM. Besides, N-BEATS, TCN and DeepState are state-of-the-art deep learning models specialized for sequential modeling. A common limitation is that they do not capture the correlations among multiple time-series explicitly, hindering their application to multivariate time-series

Table 2: Forecasting results on different datasets

	MAE	RMSE	MAPE(%)	MAE	RMSE	MAPE(%)	MAE	RMSE	MAPE(%)
	METR-LA [15]			PEMS-BAY [4]			PEMS07 [4]		
FC-LSTM [32]	3.44	6.3	9.6	2.05	4.19	4.8	3.57	6.2	8.6
SFM [39]	3.21	6.2	8.7	2.03	4.09	4.4	2.75	4.32	6.6
N-BEATS [24]	3.15	6.12	7.5	1.75	4.03	4.1	3.41	5.52	7.65
DCRNN [20]	2.77	5.38	7.3	1.38	2.95	2.9	2.25	4.04	5.30
LSTNet [17]	3.03	5.91	7.67	1.86	3.91	3.1	2.34	4.26	5.41
ST-GCN [38]	2.88	5.74	7.6	1.36	2.96	2.9	2.25	4.04	5.26
TCN [3]	2.74	5.68	6.54	1.45	3.01	3.03	3.25	5.51	6.7
DeepState [26]	2.72	5.24	6.8	1.88	3.04	2.8	3.95	6.49	7.9
GraphWaveNet [35]	2.69	5.15	6.9	1.3	2.74	2.7	-	-	-
DeepGLO [30]	2.91	5.48	6.75	1.39	2.91	3.01	3.01	5.25	6.2
StemGNN (ours)	**2.56**	**5.06**	**6.46**	**1.23**	**2.48**	**2.63**	**2.14**	**4.01**	**5.01**
	PEMS03 [4]			PEMS04 [4]			PEMS08 [4]		
FC-LSTM [32]	21.33	35.11	23.33	27.14	41.59	18.2	22.2	34.06	14.2
SFM [39]	17.67	30.01	18.33	24.36	37.10	17.2	16.01	27.41	10.4
N-BEATS [24]	18.45	31.23	18.35	25.56	39.9	17.18	19.48	28.32	13.5
DCRNN [20]	18.18	30.31	18.91	24.7	38.12	17.12	17.86	27.83	11.45
LSTNet [17]	19.07	29.67	17.73	24.04	37.38	17.01	20.26	31.96	11.3
ST-GCN [38]	17.49	30.12	17.15	22.70	35.50	14.59	18.02	27.83	11.4
TCN [3]	18.23	25.04	19.44	26.31	36.11	15.62	15.93	25.69	16.5
DeepState [26]	15.59	**20.21**	18.69	26.5	33.0	15.4	19.34	27.18	16
GraphWaveNet [35]	19.85	32.94	19.31	26.85	39.7	17.29	19.13	28.16	12.68
DeepGLO [30]	17.25	23.25	19.27	25.45	35.9	12.2	**15.12**	25.22	13.2
StemGNN (ours)	**14.32**	21.64	**16.24**	**20.24**	**32.15**	**10.03**	15.83	**24.93**	**9.26**
	Solar [17]			Electricity [2]			ECG [5]		
FC-LSTM [32]	0.13	0.19	27.01	0.62	0.2	24.39	0.32	0.54	31.0
SFM [39]	0.05	0.09	13.4	0.08	0.13	17.3	0.17	0.58	11.9
N-BEATS [24]	0.09	0.15	23.53	-	-	-	0.08	0.16	12.428
LSTNet [17]	0.07	0.19	19.13	0.06	0.07	14.97	0.08	0.12	12.74
TCN [3]	0.06	0.06	21.1	0.072	0.51	16.44	0.1	0.3	19.03
DeepState [26]	0.06	0.25	19.4	0.065	0.67	15.13	0.09	0.76	19.21
GraphWaveNet [35]	0.05	0.09	18.12	0.071	0.53	16.49	0.19	0.86	19.67
DeepGLO [30]	0.09	0.14	21.6	0.08	0.14	15.02	0.09	0.15	12.45
StemGNN (ours)	**0.03**	**0.07**	11.55	**0.04**	**0.06**	**14.77**	**0.05**	**0.07**	**10.58**

Table 3: Results for ablation study of the PEMS07 dataset

	StemGNN	w/o LC	w/o Spe-Seq Cell	w/o DFT	w/o GFT	w/o Residual	w/o Backcasting
MAE	**2.144**	2.158	2.612	2.299	2.237	2.256	2.203
RMSE	**4.010**	4.017	4.692	4.170	4.068	4.155	4.077
MAPE(%)	**5.010**	5.113	6.180	5.336	5.222	5.230	5.130

forecasting. Therefore, it is natural that StemGNN shows much better performances against these baselines.

On the other hand, spatial and temporal correlations can be modeled in GNN-based approaches, such as DCRNN, ST-GCN and GraphWaveNet. However, most of them need a pre-defined topology of different time-series and are not applicable to Solar, Electricity and ECG datasets. GraphWaveNet is able to work without a pre-defined structure but the performance is not satisfied. For traffic forecasting tasks, StemGNN outperforms these models consistently without any prior knowledge of the road network. It is convincing that a data-driven latent correlation layer works more effectively than human defined priors. Moreover, DeepGLO is a hybrid method that enables the model to focus both on local properties of individual time-series as well as global properties, while multivariate correlations are encoded by a matrix factorization module. It shows competitive performances on some datasets like solar and PEMS08, but StemGNN is generally more advantageous. Arguably, it is beneficial to recognize both structural and sequential patterns jointly in the spectral domain.

5.3 Ablation Study

To better understand the effectiveness of different components in StemGNN, we design six variants of the model and conduct ablation study on several datasets. Table 3 summarizes the results obtained on PEMS07 [4], and more results on other datasets can be found in Appendix E.2.

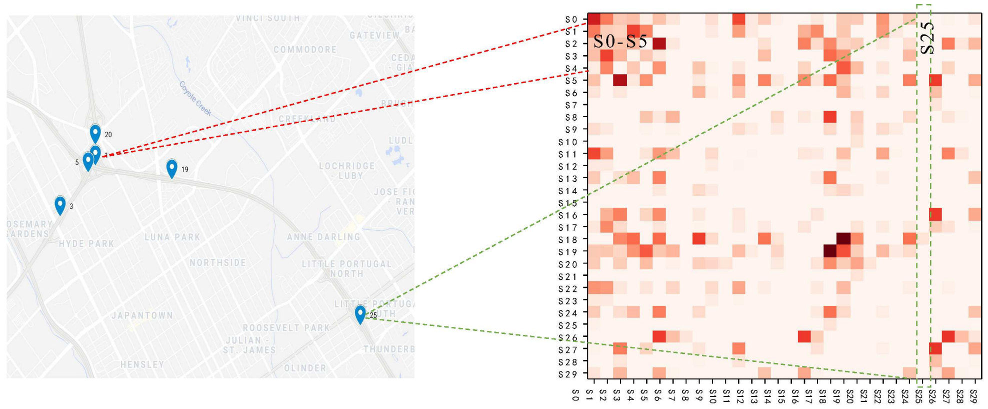


Figure 2: The adjacent matrix obtained from latent correlation layer.

The results show that all the components are indispensable. Specifically, **w/o Spe-Seq Cell** indicates the importance of temporal patterns for multivariate time-series forecasting. The Discrete Fourier Transform inside the cell also brings benefits as verified by **w/o DFT**. Furthermore, **w/o Residual** and **w/o Backcasting** demonstrate that both residual and backcasting designs can learn supplementary information and enhance time-series representation. **w/o GFT** shows the advantages of leveraging GFT to capture structural information in a graph. Moreover, we use a pre-defined topology instead of correlations learned by the *Latent Correlation Layer* in **w/o LC**, which indicates the superiority of StemGNN for learning inter-series correlations automatically.

6 Analysis

6.1 Traffic Forecasting

To investigate the validity of our proposed latent correlation layer, we perform a case study in the traffic forecasting scenarios. We choose 6 detectors from PEMS-BAY and show the average correlation matrix learned from the training data (the right part in Figure 2). Each column represents a sensor in the real world. As shown in the figure, column i represents the correlation strength between detector i and other detectors. As we can see, some columns have a higher value like column s_1 , and some have a smaller value like column s_{25} . This indicates that some nodes are closely related to each other while others are weakly related. This is reasonable, since detector s_1 is located near the intersection of main roads, while detector s_{25} is located on a single road, as shown in the left part of Figure 2. Therefore, our model not only obtains an outstanding forecasting performance, but also shows an advantage of interpretability.

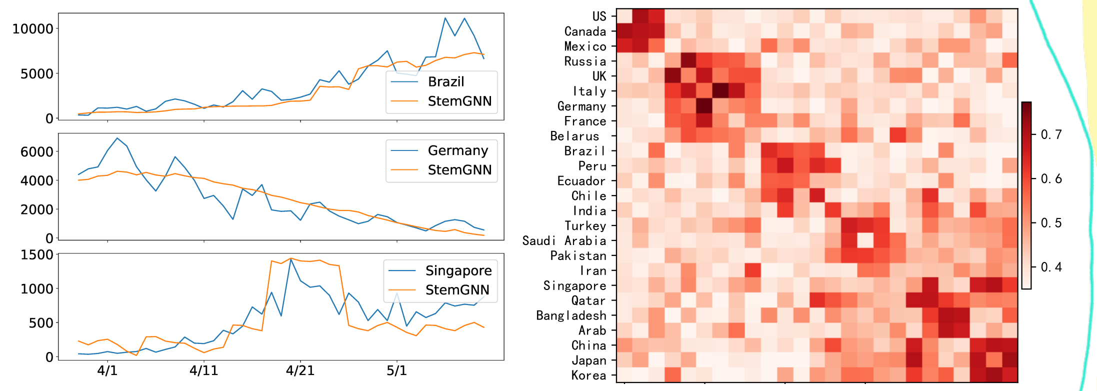
6.2 COVID-19

Table 4: Forecasting results (MAPE%) on COVID-19

	FC-LSTM [32]	SFM [39]	N-BEATS [24]	TCN [3]	DeepState [26]	GraphWaveNet [35]	DeelpGLO [30]	**StemGNN (ours)**
7 Day	20.3	19.6	16.5	18.7	17.3	18.9	17.1	**15.5**
14 Day	22.9	21.3	18.5	23.1	20.4	24.4	18.9	**17.1**
28 Day	27.4	22.7	20.4	26.1	24.5	25.2	23.1	**19.3**

To investigate the feasibility of StemGNN for real problems, we conduct additional analyses on daily number of newly confirmed COVID-19 cases. We select the time-series of 25 countries with severe COVID-19 outbreak from 1/22/2020 to 5/10/2020 (110 days). Specifically, we use the first 60 days for training and the rest 50 days for testing. In this analysis, we forecast the values of H days in the future, where H is set to be 7, 14 and 28 separately. Table 4 shows the evaluation results where we can see that StemGNN outperforms other state-of-the-art solutions in different horizons.

Figure 3(a) illustrates the forecasting results of Brazil, Germany and Singapore in advance of 28 days. Specifically, we set $H = 28$ and take the predicted value of the 28th day for visualization. Each timestamp is predicted with the historical data four weeks before that timestamp. As shown in



(a) Forecasting result for the 28th day

(b) Inter-country correlations

Figure 3: Analysis on COVID-19

the figure, the predicted value is consistent with the ground truth. Taking Singapore as an example, after 4/14/2020, the volume has rapidly increased. StemGNN forecasts such trend successfully in advance of four weeks.

The dependencies among different countries learned by the *Latent Correlation Layer* are visualized in Figure 3(b). Larger numbers indicate stronger correlations. We observe that the correlations captured by StemGNN model are in line with human intuition. Generally, countries adjacent to each other are highly correlated. For example, as expected, US, Canada and Mexico are highly correlated to each other, so are China, Japan and Korea.

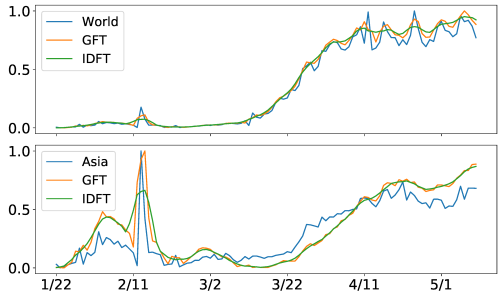


Figure 4: Effectiveness of GFT and DFT.

We further analyze the effect of GFT and DFT in StemGNN. We choose the top two eigenvectors obtained by eigenvalue decomposition of the normalized Laplacian matrix L and visualize their corresponding time-series after GFT in Figure 4. As encoded by the eigenvectors, the first time-series captures a common trend in the world and the second time-series captures a common trend from Asian countries. For a clear comparison, we also visualize the ground truth of daily number of newly confirmed in the whole world and Asian countries. As shown in Figure 4, the time-series after GFT capture these two major trends obviously. Moreover, the time-series data in the spectral space becomes smoother, which increases the generalization capability and reduces the difficulty of forecasting. We also draw the time-series after processed by the *Spectral Sequential Cell* (denoted by IDFT in Figure 4), which recognizes the data patterns in a frequency domain. Compared to the ones after GFT, the result time-series turn to be smoother and more feasible for forecasting.

7 Conclusion

In this paper, we propose a novel deep learning model, namely Spectral Temporal Graph Neural Network (StemGNN), to take the advantages of both inter-series correlations and temporal dependencies by modeling them jointly in the spectral domain. StemGNN outperforms existing approaches consistently in a variety of multivariate time-series forecasting applications. Future works are considered in two directions. First, we will investigate approximation method to reduce the time complexity of StemGNN, because directly applying eigenvalue decomposition is prohibitive for very large graphs of high-dimensional time-series. Second, we will look for its application to more real-world scenarios, such as product demand, stock price prediction and budget analysis. We also plan to apply StemGNN for predictive maintenance, which is an important topic in AIOps.

Broader Impact

Time-series analysis is an important research domain for machine learning, while multivariate time-series forecasting is one of the most prevalent tasks in this domain. This paper proposes a novel model, StemGNN, for the task of multivariate time-series forecasting. For the first time, we model the inter-series correlations and temporal patterns jointly in the spectral domain, which improves the representation power of multivariate time-series. Signals in the time domain can be easily restored by the orthogonal basis in the frequency domain, so we could leverage the rich information beneath the hood of the frequency domain to improve the forecasting results. StemGNN is neat yet powerful as proved by extensive experiments and analyses. It is one of the first attempts that incorporate Discrete Fourier Transform with Graph Neural Networks. We believe it will motivate more exploration along this direction in other related domains with temporal features, such as social graph mining and sentiment analysis. Moreover, StemGNN adopts a latent correlation layer in an end-to-end framework to learn relationships among multivariate signals automatically. This makes StemGNN a general approach that could be applied to a wide range of applications, including surveillance of traffic flows, healthcare data monitoring, natural disaster forecasting and economy.

Multivariate time-series forecasting has significant societal implications as well. A sophisticated supply chain management system may be built if we can predict market trend precisely. It also brings benefit to our daily life. For example, there is a real case about ‘Flooding Risk Analysis’. The task is to predict when there will be a flooding in certain areas near the city. The prediction mainly depend on two external factors, tides and rainfalls. Accurate prediction can alert people to keep away from the area at the corresponding time to avoid unnecessary losses. For COVID-19, accurate prediction of the trend may help the government make suitable decisions to control the spread of the epidemic. According to a case study on COVID-19 in this paper, we can reasonably forecast the daily number of newly confirmed cases four weeks in advance based on historical data. Nevertheless, how to predict the trend from the beginning without sufficient historical data is more challenging and remained to be investigated. Moreover, we are aware of the negative impact of this technique to infringement of personal privacy. Customers’ behavior may be predicted by unscrupulous business persons on historical records, which provides a convenient way to send spam information. Hackers may also use the predicted data to avoid surveillance of a bank’s security system for fraud credit card transactions.

Although current models are still far away from predicting future data absolutely correct, we do believe that the margin is decreasing rapidly. We hope that researchers could understand and mitigate the potential risks in this domain. We would like to mention the concept of responsible AI, which guides us to integrate fairness, interpretability, privacy, security, accountability into the design of AI systems. We suggest researchers to take a people-centered approach to research, development, and deployment of AI and cultivate a responsible AI-ready culture.

References

[1] Rie K Ando and Tong Zhang. Learning on graph with laplacian regularization. In *Advances in Neural Information Processing Systems*, pages 25–32, 2007.

[2] Arthur Asuncion and David Newman. UCI machine learning repository, 2007.

[3] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[4] Chao Chen, Karl Petty, Alexander Skabardonis, Pravin Varaiya, and Zhanfeng Jia. Freeway performance measurement system: mining loop detector data. *Transportation Research Record*, 1748(1):96–102, 2001.

[5] Yanping Chen, Eamonn Keogh, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, and Gustavo Batista. The UCR time series classification archive. 2015.

[6] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. In Dekai Wu, Marine Carpuat, Xavier Carreras, and Eva Maria Vecchi, editors, *Proceedings of SSST@EMNLP 2014, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, Doha, Qatar, 25 October 2014*, pages 103–111. Association for Computational Linguistics, 2014.

[7] JHU CSSE. Covid-19 data repository by the center for systems science and engineering (csse) at johns hopkins university. `https://github.com/CSSEGISandData/COVID-19/`. Accessed may 28, 2020.

[8] Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 933–941. JMLR. org, 2017.

[9] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852, 2016.

[10] F. Grassi, A. Loukas, N. Perraudin, and B. Ricaud. A time-vertex signal processing framework: Scalable processing and meaningful representations for time-series on graphs. *IEEE Transactions on Signal Processing*, 66(3):817–829, 2018.

[11] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 922–929, 2019.

[12] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[13] Rob J Hyndman and Anne B Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4):679–688, 2006.

[14] E. Isufi, A. Loukas, N. Perraudin, and G. Leus. Forecasting time series with varma recursions on graphs. *IEEE Transactions on Signal Processing*, 67(18):4870–4885, 2019.

[15] HV Jagadish, Johannes Gehrke, Alexandros Labrinidis, Yannis Papakonstantinou, Jignesh M Patel, Raghu Ramakrishnan, and Cyrus Shahabi. Big data and its technical challenges. *Communications of the ACM*, 57(7):86–94, 2014.

[16] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations*, ICLR '17, 2017.

[17] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 95–104, 2018.

[18] Jiachen Li, Hengbo Ma, Zhihao Zhang, and Masayoshi Tomizuka. Social-wagdat: Interaction-aware trajectory prediction via wasserstein graph double-attention network. *arXiv preprint arXiv:2002.06241*, 2020.

[19] Jiachen Li, Fan Yang, Masayoshi Tomizuka, and Chiho Choi. Evolvegraph: Multi-agent trajectory prediction with dynamic relational reasoning. In *Proceedings of the Neural Information Processing Systems (NeurIPS)*, 2020.

[20] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations (ICLR '18)*, 2018.

[21] Andreas Loukas and Nathanaël Perraudin. Stationary time-vertex signal processing. *EURASIP journal on advances in signal processing*, 2019(1):1–19, 2019.

[22] Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27(1):415–444, 2001.

[23] Pablo Montero-Manso, George Athanasopoulos, Rob J Hyndman, and Thiyanga S Talagala. Fforma: Feature-based forecast model averaging. *International Journal of Forecasting*, 36(1):86–92, 2020.

[24] Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-BEATS: neural basis expansion analysis for interpretable time series forecasting. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[25] Jigar Patel, Sahil Shah, Priyank Thakkar, and Ketan Kotecha. Predicting stock market index using fusion of machine learning techniques. *Expert Systems with Applications*, 42(4):2162–2172, 2015.

[26] Syama Sundar Rangapuram, Matthias W Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. Deep state space models for time series forecasting. In *Advances in Neural Information Processing Systems*, pages 7785–7794, 2018.

[27] Akhter Mohiuddin Rather, Arun Agarwal, and VN Sastry. Recurrent neural network and a hybrid model for prediction of stock returns. *Expert Systems with Applications*, 42(6):3234–3241, 2015.

[28] Hansheng Ren, Bixiong Xu, Yujing Wang, Chao Yi, Congrui Huang, Xiaoyu Kou, Tony Xing, Mao Yang, Jie Tong, and Qi Zhang. Time-series anomaly detection service at microsoft. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3009–3017, 2019.

[29] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 2019.

[30] Rajat Sen, Hsiang-Fu Yu, and Inderjit S Dhillon. Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. In *Advances in Neural Information Processing Systems*, pages 4838–4847, 2019.

[31] Chao Song, Youfang Lin, Shengnan Guo, and Huaiyu Wan. Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, 2020.

[32] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112, 2014.

[33] Peter R Winters. Forecasting sales by exponentially weighted moving averages. *Management Science*, 6(3):324–342, 1960.

[34] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[35] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. Graph wavenet for deep spatial-temporal graph modeling. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 1907–1913. International Joint Conferences on Artificial Intelligence Organization, 7 2019.

[36] SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems*, pages 802–810, 2015.

[37] Bing Yu, Mengzhang Li, Jiyong Zhang, and Zhanxing Zhu. 3d graph convolutional networks with temporal graphs: A spatial information free framework for traffic forecasting. *arXiv preprint arXiv:1903.00919*, 2019.

[38] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 3634–3640. International Joint Conferences on Artificial Intelligence Organization, 7 2018.

[39] Liheng Zhang, Charu Aggarwal, and Guo-Jun Qi. Stock price prediction via discovering multi-frequency trading patterns. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2141–2149, 2017.

[40] Ziwei Zhang, Peng Cui, and Wenwu Zhu. Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[41] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018.

A Notation

Table 5: Notations

$\mathcal{G}$	multivariate temporal graph
X	multivariate time-series input, $X_t \in \mathbb{R}^N$ is observed values at timestamp t
Δ_θ	all parameters in the network
W	adjacency matrix, where $w_{ij} \in W$ indicates the strength of edge ij
N	the number of time-series
K	the number of previous time steps
H	the number of future time steps to forecast (horizon)
B	the entire network that generates backcasting output
$\hat{X}$	forecasted time-series output, $\hat{X}_t \in \mathbb{R}^N$ is the value at timestamp t
R	the last hidden state of attention mechanism
Q, K	query and key in the attention mechanism
W^Q, W^K	learnable parameters for query and key projections
$\Theta_{\cdot j}$	graph convolution kernel
$\mathcal{GF}$	Graph Fourier Transform
$\mathcal{GF}^{-1}$	Inverse Graph Fourier Transform
$\mathcal{S}$	the Spe-Seq Cell
V	basis vectors
Z	the output after IGFT
Y	the forecasting output
$\mathcal{F}$	Discrete Fourier Transform
$\hat{X}_u^*$	the real part $\hat{X}_u^r$ and imaginary part $\hat{X}_u^i$ after DFT
$\mathcal{F}^{-1}$	Inverse Discrete Fourier Transform
θ_τ^*	the convolution kernel of Spe-Seq Cell
L	the normalized graph Laplacian
U	the matrix of eigenvectors
Λ	the diagonal matrix of eigenvalue

B Reproduction details for StemGNN

B.1 Datasets

We compare the performance of StemGNN with other state-of-the-art models on ten public datasets, ranging from traffic, energy, electrocardiogram to COVID-19 domain. Among all the datasets, only the datasets from traffic domain provide apriori topology. Table 1 shows the statistics of these datasets.

Traffic Forecasting. These datasets are collected by the Caltrans Performance Measurement System (PeMS) [4] and the loop detectors in the highway of Los Angeles County (METR) [15]. The monitoring data is aggregated by 5 minutes from 30-second data samples, which means there are 12 points in the flow data for each hour. We evaluate the performance of traffic flow forecasting on PEMS03, PEMS07, PEMS08 and traffic speed forecasting on PEMS04, PEMS-BAY and METR-LA.

Energy Forecasting. We consider two datasets in this perspective. (1) Solar. It contains photovoltaic production of 137 stations in Alabama State [17], which is sampled every 10 minutes. (2) Electricity. It contains hourly time-series of electricity consumption from 370 customers [2].

Electrocardiogram Forecasting. We adopt the ECG5000 dataset from the UCR time-series Classification Archive [5], and this dataset is composed of 140 electrocardiograms (ECG) with a length of 5000.

COVID-19 Trend Forecasting. This dataset is provided by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University[3], which contains daily case reports including con-

[3] https://github.com/CSSEGISandData/COVID-19

firmed, deaths and recovered number. We use the daily number of newly confirmed COVID-19 cases as the time-series and select the time-series of 25 countries with severe COVID-19 outbreak from 1/22/2020 to 5/10/2020 (totally 110 days). Specifically, we use the first 60 days for training and the rest 50 days for testing.

B.2 Metrics

Let $\hat{X}_t$ and X_t be the predicted and ground truth values at timestamp t respectively, T is the total number of timestamps. The evaluation metrics we use in the experiments can be computed by:

$$MAE = \frac{1}{T}\sum_{t=1}^{T}|X_t - \hat{X}_t|, \tag{6}$$

$$MAPE = \frac{1}{T}\sum_{t=1}^{T}\left|\frac{X_t - \hat{X}_t}{X_t}\right| \times 100\%, \tag{7}$$

$$RMSE = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(X_t - \hat{X}_t)^2}. \tag{8}$$

C Reproduction details for baselines

FC-LSTM [32]: FC-LSTM can forecast univariate time-series with fully-connected LSTM hidden units. The source code can be found at `https://github.com/farizrahman4u/seq2seq`. We use 4 stacked LSTM cells of 1000 hidden size and other detailed settings can be referred to [32].

SMF [39]: SMF improves the LSTM model to be able to break down the cell states of a given univariate time-series into a series of different frequency components. We use SMF by setting hidden dimension as 50, frequence dimension as 10. Other default configurations are given in the source code: `https://github.com/z331565360/State-Frequency-Memory-stock-prediction`.

N-BEATS [24]: N-BEATS proposes a deep neural architecture based on backward and forward residual links and a very deep stack of fully-connected layers without using time-series domain knowledge. We use the open source code from: `https://github.com/philipperemy/n-beats`, and only modify the data I/O interface for different shapes of inputs. In our experiments, the backcast length is 10 and the hidden units number is 128. According to the recommendation, we turn on the 'share_weights_in_stack' option.

LSTNet [17]: takes advantage of the convolution layer to discover the local dependence patterns among multi-dimensional input variables, and the recurrent layer to captures the complex long-term dependency patterns. We use the open source code from: `https://github.com/fbadine/LSTNet`, and modify the data shapes of inputs. In our experiments, the number of output filters in the CNN layer is 100 and the CNN filter size is 6. Other experimental settings we refer to papers and code default values.

DCRNN [20]: DCRNN is a deep learning framework for traffic forecasting that incorporates both spatial and temporal dependencies in the traffic flow. Some of the results of DCRNN are directly reported in [20, 31], and we use the source code at `https://github.com/liyaguang/DCRNN` when reproduction is necessary. The horizon size is 12 and the RNN layer number is 2 with 64 units in our experiments. DCRNN is not applicable in scenarios without a priori topology. Thus, DCRNN is only used to forecast the traffic data.

STGCN [38]: STGCN is a novel deep learning framework for traffic prediction, integrating graph convolution and gated temporal convolution through spatio-temporal convolutional blocks. The performances of STGCN can be found at [38] and the source code is available at `https://github.com/VeritasYin/STGCN_IJCAI-18`. We use 12 history steps to forecast future data with batch size as 50, epoch number as 50, and learning rate as 0.001. STGCN is not applicable to scenarios without a priori topology.

TCN [3]: TCN combines best practices such as dilations and residual connections with the causal convolutions for autoregressive prediction. We take the source code at https://github.com/locuslab/TCN. We use a configuration similar to polyphonic music task mentioned in this paper, where the kernel size is 5, the gradient clip is 0.2, the upper epoch limit is 100 and the initial learning rate is 0.001.

DeepState [26]: This model marries state space models with deep recurrent neural networks. First, it uses recurrent neural networks to calculate $h_t = RNN(h_{t-1}, x_t)$. Then, this model uses h_t to calculate the parameters of state space $\Theta_t = \Phi(h_t)$. Finally, the likelihood $pss(z_{1:T}|\Theta_{1:T})$ are calculated and the parameters are learned through maximum log likelihood. DeepState is integrated in Gluon Time Series (GluonTS), which is the Gluon toolkit for probabilistic time series modeling. The tutorials can be found at https://gluon-ts.mxnet.io/. We use the default configuration given by the tool and only change its sampling frequency as given in Table 1.

GraphWaveNnet [35]: GraphWaveNet is a method that represents each node's network neighborhood via a low-dimensional embedding by leveraging heat wavelet diffusion patterns. The results of Graph Wavenet are reported at [35, 37, 31], and the source code can be found at https://github.com/nnzhan/Graph-WaveNet. We turn on the 'add graph convolution layer' option when reproduce on some datasets. We set the weight decay rate as 0.0001 and dropout rate as 0.3. Other configurations follow the options recommended in the paper. We use the priori topology when it is available (traffic forecasting), and this method also works in scenarios without a priori topology (energy forecasting, electrocardiogram forecasting and COVID-19 forecasting).

DeepGLO [30] : This model leverages both global and local features during training and forecasting. The global component, TCN regularized Matrix Factorization (TCN-MF), captures global patterns by representing each of the original time-series as a linear combination of k basis time-series, and we set $k = 128$ in our experiments. We use the default setting of DeepGLO provided by https://github.com/rajatsen91/deepglo. It has two batch sizes: horizontal batch size (set to 256) and vertical batch size (set to 128). Besides, we change the start time and frequency of different datasets. The kernel size is set as 7 for both hybrid model and local model, and the learning rate is set to be 0.005. We report the best results from the normalized and unnormalized settings in the paper.

Please refer to their publications for more detailed descriptions and settings.

D Experiment Details

We conduct all our experiments using one NVIDIA GeForce GTX 1080 GPU. We divide the dataset into three part for training, validation and testing according to [11] (PEMS03, PMES04, PEMS08), [38] (PEMS07), and [20] (META-LA, PEMS-BAY, Solar, Electricity, ECG). The inputs of ECG are normalized by min-max normalization following [5]. Besides, the inputs are normalized by Z-Score method [24]. That means StemGNN is trained on normalized input where each time-series in the training set is re-scaled as $X_{in} = (X_{in} - \mu(X_{in}))/\sigma(X_{in})$, where μ and σ denote the mean and standard deviation respectively. The evaluation of Solar, Electricity and ECG datasets is performed on the re-scaled data following [5] and [26], i.e., first using the normalization algorithm to transform Solar, Electricity and ECG into a value range of $[0, 1]$, and then applying StemGNN to generate the forecasting values. Afterwards, the predictions are transformed back to the original scale, and the metrics are calculated on the original data.

In StemGNN, the dimension of self-attention layer is 32, which is chosen from a search space of [16, 32, 64, 128] on the validation data. The channel size of each graph convolution layer is 64 chosen from a search space of [16, 32, 64, 128] and the kernel size of 1D convolution is 3, selected from a search space of [3, 6, 9, 12]. The batch size is 50. The learning rate is initialized as 0.001 and decays with rate 0.7 after every 5 epochs. The total number of training epochs is set as 50.

In the traffic datasets (METR-LA, PEMS-BAY, PEMS07, PEMS07, PEMS04, PEMS08), the data is aggregated by 5 minutes, so the number of timestamps per day is 288. For traffic speed forecasting task, we use the one-hour historical data to predict the next 15 minutes data [20, 38]; for traffic flow forecasting task, we use one-hour historical data to predict the values in the next hour [11]. The solar dataset is aggregated every 10 minutes, according to [26, 30], we forecast the trend in future 0.5 hour with 4-hour historical data. For the electricity data, we follow [26, 30] which use 24-hour historical data to infer the values in next 3 hours. For ECG5000 dataset, according to [24], we set the forecasting step as 3 and the sliding window size as 12. For COVID-19 dataset, we forecast the future 4 weeks' trend, which means the max forecasting step is 28 (days). Besides, we use averaging MAE, MAPE, RMSE over the predicted time period to evaluate StemGNN and all baselines.

E Results

E.1 More results on METR-LA and COVID-19

Table 6: Forecasting Results on METR-LA and COVID-19

		MAE	RMSE	MAPE(%)	MAE	RMSE	MAPE(%)	MAE	RMSE	MAPE(%)
			15min			30min			1hour	
METR-LA [15]	FC-LSTM [32]	3.44	6.3	9.60	3.77	7.23	10.90	4.37	8.69	13.20
	SFM [39]	3.21	6.2	8.7	3.37	6.68	9.62	3.47	7.61	10.15
	N-BEATS [24]	3.15	6.12	7.5	3.62	7.01	9.12	4.12	8.04	11.5
	DCRNN [20]	2.77	5.38	7.30	3.15	6.45	8.80	3.6	7.59	10.50
	STGCN [38]	2.88	5.74	7.60	3.47	7.24	9.60	4.59	9.4	12.70
	TCN [3]	2.74	5.68	6.54	-	-	-	-	-	-
	DeepState [26]	2.72	5.24	6.8	3.13	6.16	8.31	3.61	7.42	10.8
	GraphWaveNet [35]	2.69	5.15	6.90	3.07	6.22	8.40	3.53	7.37	10
	DeepGLO [30]	2.91	5.48	6.75	3.36	6.42	8.33	3.66	7.39	10.3
	StemGNN (ours)	**2.56**	**5.063**	**6.46**	**3.011**	**6.03**	**8.23**	**3.43**	**7.23**	**9.85**
			7Day			14Day			28Day	
COVID-19 [7]	FC-LSTM [32]	1803.65	3284.77	20.3	2135.54	3855.75	22.9	2554.07	4318.4	27.4
	SFM [39]	1699.85	3499.25	19.6	1812.82	3589	21.3	1851	3720	22.7
	N-BEATS [24]	594.43	928.37	16.5	847.14	1286.36	18.5	882.42	1349.46	20.4
	TCN [3]	662.24	2363.95	18.7	1307	2871.17	23.1	2117.34	3419.3	26.1
	DeepState [26]	922.87	1982.32	17.3	1852.73	2091.32	20.4	2345.3	2386.4	24.5
	GraphWaveNet [35]	1056.1	1227.3	18.9	1899.5	2125.7	24.4	2331.5	2451.9	25.2
	DeepGLO [30]	1131.23	1023.19	17.1	1718.69	1734.67	18.9	2084.51	2291.19	23.1
	StemGNN (ours)	**462.24**	**718.11**	**15.5**	**533.67**	**871.17**	**17.1**	**662.24**	**1023.19**	**19.3**

In order to prove that there is a steady improvement for multi-step forecasting, we choose METR-LA and COVID-19 for an evaluation of longer time span. As shown in Table 6, StemGNN achieves excellent performance in multi-steps forecasting scenarios. In particular, we use COVID-19 data to forecast the number of infected people in the next 1-4 weeks which is of great significance to help relevant departments make decisions. Compared to other solutions, StemGNN reduces the time-dependent error accumulation and improves the performance of long-term forecasting.

E.2 More results for ablation study

Table 7: Ablation results

		15min			30min			45min		
		MAE	RMSE	MAPE(%)	MAE	RMSE	MAPE(%)	MAE	RMSE	MAPE(%)
PEMS07 [4]	**StemGNN**	**2.144**	**4.01**	**5.01**	**2.994**	**5.35**	**7.25**	**3.158**	**6.34**	**8.43**
	w/o Latent Correlations	2.158	4.017	5.113	3.004	5.525	7.303	3.214	6.496	8.672
	w/o Spe-Seq Cell	2.612	4.692	6.189	3.459	6.257	8.448	4.505	8.241	11.343
	w/o DFT	2.299	4.17	5.336	3.183	5.945	7.532	3.817	7.145	9.058
	w/o GFT	2.237	4.068	5.222	3.065	5.755	7.355	3.691	6.922	8.899
	w/o Residual	2.256	4.155	5.23	3.073	5.854	7.357	3.684	7.021	8.918
	w/o Backcasting	2.203	4.077	5.13	3.034	5.641	7.316	3.394	6.912	8.681
		15min			30min			1hour		
		MAE	RMSE	MAPE(%)	MAE	RMSE	MAPE(%)	MAE	RMSE	MAPE(%)
METR-LA [15]	**StemGNN**	**2.56**	**5.063**	**6.46**	**3.011**	**6.03**	**8.23**	**3.43**	**7.23**	**9.85**
	w/o Latent Correlations	2.79	5.24	6.867	3.122	6.922	8.36	3.568	7.462	9.97
	w/o Spe-Seq Cell	3.077	5.71	6.99	3.491	7.072	8.83	3.905	7.906	10.163
	w/o DFT	2.81	5.37	6.93	3.24	6.95	8.52	3.717	7.571	9.99
	w/o GFT	2.867	5.25	6.891	3.201	6.92	8.41	3.701	7.552	10.02
	w/o Residual	2.83	5.29	6.71	3.228	6.57	8.27	3.724	7.471	9.95
	w/o Backcasting	2.85	5.219	6.57	3.06	6.233	8.51	3.56	7.72	10.03

- **w/o Latent Correlations (LCs).** We use a priori topology instead of automatic correlations. As shown in Table 7, dynamic latent correlations performs even better than a static priori topology. The reason may be that a priori topology is static, but the StemGNN is capable of building a topology for each sliding window dynamically, which captures the newest knowledge about the interaction between different time-series.
- **w/o Spe-Seq Cell.** This setting does not equip with the *Spe-Seq Cell*. It performs the worst among all settings, indicating that temporal dependency is the most important clue for time-series forecasting.

- **w/o DFT.** It removes DFT and inverse DFT operators in the Spectral Sequential Cell. Thus, temporal dependencies are modeled in the time domain. It shows improvement over the naive baseline without *Spe-Seq Cell*, but under-performs StemGNN by a large margin, which proves the benefit of DFT.
- **w/o GFT.** We no longer use the entire StemGNN cell, but only take the Spectral Sequential Cell. The performance drops significantly, which shows a necessity of capturing latent correlations through graph Fourier transform.
- **w/o Residual.** This setting has two stacked StemGNN blocks without residual connection. It verifies that the second block learns supplement information through a residual connection.
- **w/o Backcasting.** This model disables the backcasting branch, showing the benefit of backcasting module for enhancing time-series representation.

F Analysis

F.1 Efficiency Analysis

Table 8: Results of efficiency analysis

	Training time (in seconds)			
	StemGNN	SFM	N-BEATS	STGCN
PEMS07 [4]	459	1013	1251	**352**
METR-LA [15]	1137	2284	2561	**1035**

Although the time complexity is $O(N^3)$ w.r.t. the multivariate dimension N, training process on all the datasets can be finished in a reasonable time. For a concrete comparison, we summarize the training time of StemGNN, SFM, N-BEATS and STGCN on the PEMS07 and METR-LA datasets separately. The results are shown at Table 8. StemGNN is similar to the first-order approximate graph convolution model (STGCN) in time, but our performance is improved significantly. Comparing to other baselines, StemGNN has a superior training speed, and inference speed shows the same conclusion.

F.2 Learning Curve Comparison

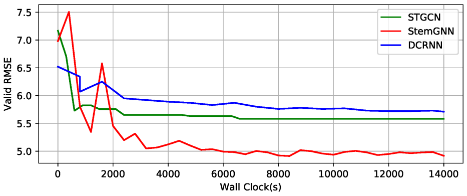


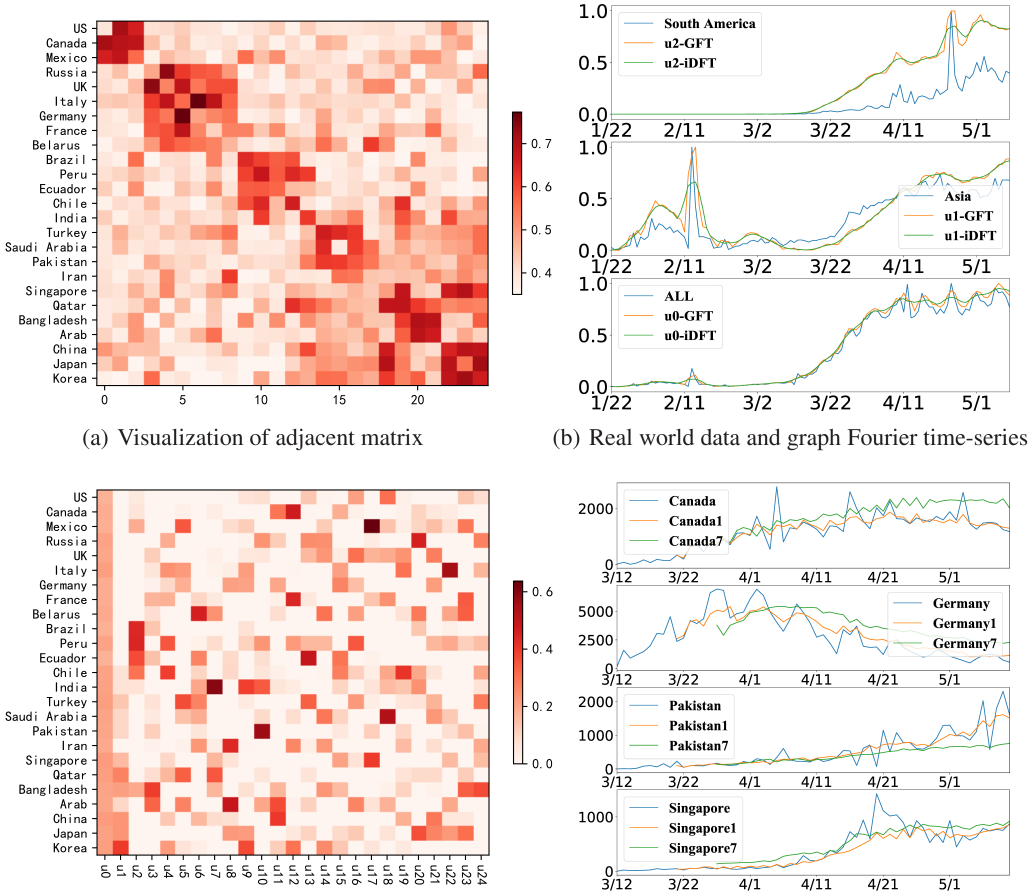
Figure 5: Learning curves on METR-LA.

The learning curves for StemGNN and major baselines on METR-LA dataset are shown in Figure 5, where the x-axis denotes wall-clock time and the y-axis denotes validation RMSE. It shows that StemGNN has an effective training procedure and achieves a better RMSE score at convergence than other SOTAs with comparable training times.

F.3 Case Study on COVID-19

To investigate the usability and interpretability of StemGNN, we conduct a detailed analysis on COVID-19 data. We assume the daily number of newly confirmed cases as time-series, and choose 25 countries with severe outbreak as multivariate input. Figure 6(a) shows the visualization of the inter-series correlations captured automatically by our model. In this figure, row i represents the correlation strength between country i and other countries. As we can see, correlations are not uniform across countries. This indicates that some nodes are closely related to neighboring countries while weakly related to others. This is reasonable since countries on the same continent have higher correlations in population mobility and related policies. Therefore, our model not only obtains the best forecasting performance, but also shows the advantage of interpretability.

To prove that the conversion of graph Fourier transform over multivariate time-series data is effective, we first visualize the matrix of eigenvectors ($\mathbf{U}$) obtained by the decomposition of the normalized Laplace matrix $\mathbf{L}$ on Figure 6(c). Each column of $\mathbf{U}$ represents an eigenvector corresponding to a eigenvalue sorted from the highest to the lowest. We select three eigenvectors with the largest eigenvalues ($\mathbf{u_0} - \mathbf{u_2}$) and visualize the corresponding time-series after GFT for further analysis.



(a) Visualization of adjacent matrix

(b) Real world data and graph Fourier time-series

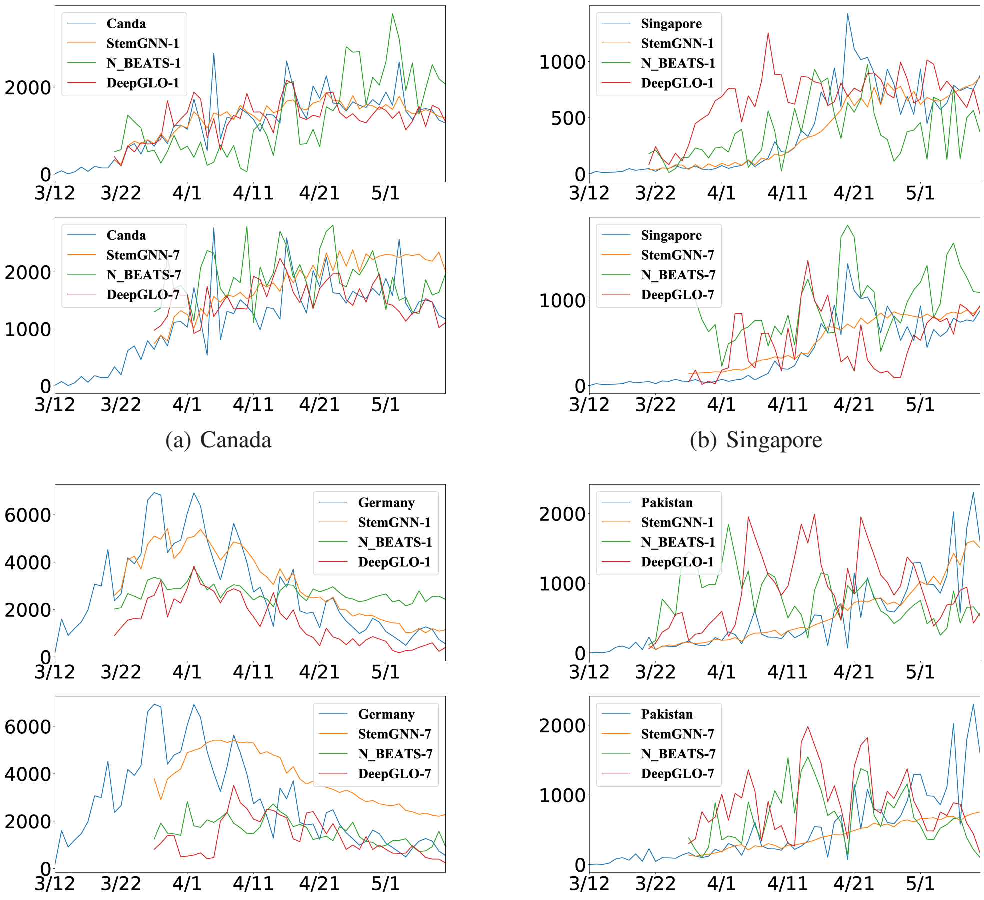
(c) Visualization of eigenvector

(d) Forecasting results on COVID-19

Figure 6: The latent correlations and forecasting results on COVID-19

As shown in Figure 6(c), $\mathbf{u_0}$ captures the general trend countries across the world, $\mathbf{u_1}$ learns the major trend of Asian countries, and $\mathbf{u_2}$ learns the common trend of South American countries. As illustrated in Figure 6(b), the three components capture these three trends respectively, and the time-series in the spectral space is relatively smooth [22] compared to the original data, reducing the difficulty of forecasting. Thus, it is clear that graph Fourier transform can better leverage the relationships learned by the latent correlation layer and make the forecasting easier through feature smoothness. Moreover, IDFT also helps to improve the smoothness of time-series and lead to better generalization (Figure 6(b)). Finally, Figure 6(d) shows the forecasting results for several exemplar countries and demonstrate the feasibility of StemGNN. In the figure, 'Canada' represents for ground truth; 'Canada1' means forecasting in advance of 1 day; 'Canada7' means forecasting in advance of 7 days. It is similar for other countries.

Figure 6 and Figure 7 show the result comparison of stemGNN and two major baselines which respectively use historical data to forecast the number of confirmed people in advance of one day (denoted by *-1) and one week (denoted by *-7). We select several typical countries, and the features show that StemGNN can predict the future trends more accurately. Thanks to graph Fourier transform and Spectral Sequential Cell, StemGNN captures the major trends more smoothly and predict the changes of data more timely. The turning points predicted by other baselines have larger time delays compared to StemGNN.



(a) Canada

(b) Singapore

(c) Germany

(d) Pakistan

Figure 7: The latent correlations and forecasting results on COVID-19