Multivariate Time-Series Forecasting with Temporal Polynomial Graph Neural Networks

Yijing Liu[1], **Qinxian Liu**[1], **Jian-Wei Zhang**[1]
Haozhe Feng[1], **Zhongwei Wang**[1], **Zihan Zhou**[1], **Wei Chen**[1*]
[1] State Key Lab of CAD&CG, Zhejiang University, Hangzhou, China
{3150105531,22021050,zjw.cs,fenghz,wzw09,12121109,chenvis}@zju.edu.cn

Abstract

Modeling multivariate time series (MTS) is critical in modern intelligent systems. The accurate forecast of MTS data is still challenging due to the complicated latent variable correlation. Recent works apply the Graph Neural Networks (GNNs) to the task, with the basic idea of representing the correlation as a static graph. However, predicting with a static graph causes significant bias because the correlation is time-varying in the real-world MTS data. Besides, there is no gap analysis between the actual correlation and the learned one in their works to validate the effectiveness. This paper proposes a temporal polynomial graph neural network (TPGNN) for accurate MTS forecasting, which represents the dynamic variable correlation as a temporal matrix polynomial in two steps. First, we capture the overall correlation with a static matrix basis. Then, we use a set of time-varying coefficients and the matrix basis to construct a matrix polynomial for each time step. The constructed result empirically captures the precise dynamic correlation of six synthetic MTS datasets generated by a non-repeating random walk model. Moreover, the theoretical analysis shows that TPGNN can achieve perfect approximation under a commutative condition. We conduct extensive experiments on two traffic datasets with prior structure and four benchmark datasets. The results indicate that TPGNN achieves the state-of-the-art on both short-term and long-term MTS forecastings. [1]

1 Introduction

The wide deployment of sensors in modern societies records tremendous time-series data, which boosts the application of energy dispatch, traffic control, etc. Commonly, there are numerous distributed sensors in a monitoring system, e.g., the temperature control system of a computing center. Univariate time-series data recorded by these sensors formulate the multivariate time-series (MTS) data. Forecasting over MTS data has been widely studied [4, 21, 41] in recent years, as it provides essential information for strategy formulation and resource schedule [30, 52].

One basic premise of modeling the MTS data is that the variables interact with each other, which is fulfilled in most cases. Therefore, capturing the variable correlation is essential for MTS forecasting. Previous deep-learning-based works like FC-LSTM [36] and TPA-LSTM [34] model the correlation with an implicit recurrent process, which is inefficient in prediction and hard to optimize. The development of graph neural networks (GNNs) [17] brings an innovative way to capture the variable dependence. Recent works [12, 25, 44] regard the MTS data as a series of graph signals and process them with GNNs, where the nodes are variables and the edge weight quantifies the dependence. Although GNNs-based methods achieve promising efficiency and accuracy in MTS forecasting,

*Corresponding author.
[1]Code is available at `https://github.com/zyplanet/TPGNN`.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

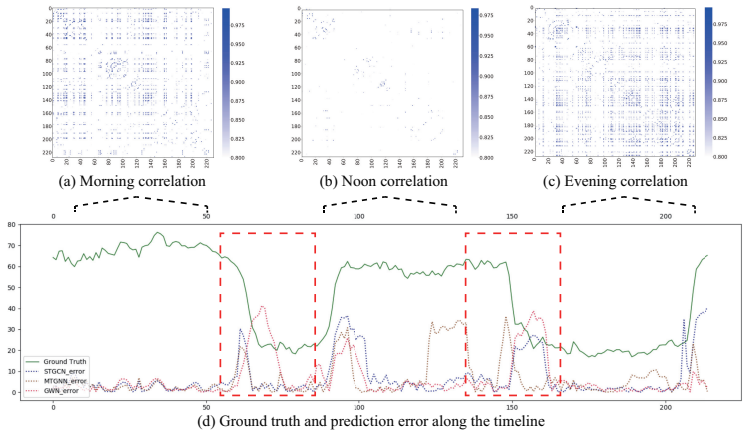


Figure 1: The sensor correlation and model prediction error in 24 hours. In (a)-(c), we apply the Pearson correlation coefficient to derive the sensor correlation of different periods, the results show time-varying patterns. We present the prediction error of three static-graph-based models in (d), the errors have steep increases between two periods, we mark them with red rectangles.

there are two notable challenges to the current approaches. 1) The MTS data have no prior variable dependence, which is required for determining the edge weight in graph construction. The existent methods compute the weight according to the physical distance among sensors [23, 35, 44] or utilize parameter matrices to learn the weight from MTS data [27, 41, 42]. As a result, the edge weights are constant in their methods. However, according to the observations from the real-world dataset, the variable dependence changes with time. Therefore, using static-weight graphs in prediction causes significant bias. Figure 1 illustrates the first challenge with a traffic dataset (PEMS-D7). It demonstrates that the correlation has distinct patterns in different periods. Besides, three static-graph-based methods [41, 42, 44] show error increases between two consecutive periods, which is caused by dependence variation. 2) Many researchers point out a gap between the actual variable dependence and the constructed correlation [25, 42, 49], i.e., the edge weight does not accurately describe the variable correlation. Although they propose various methods to bridge the gap, there is no theoretical/empirical analysis of the gap. Therefore, the method's effectiveness is still unknown.

To address the above two challenges, we propose Temporal Polynomial Graph Neural Network (TPGNN), a novel GNNs-based forecasting model for MTS data. TPGNN has an encoder-decoder structure, we encode historical MTS data with a novel GNN module and make predictions by decoding the encoded results auto-regressively. The core of TPGNN is a temporal polynomial graph (TPG) module that learns the dynamic variable dependence end-to-end. The main idea is to represent the correlation as a time-varying matrix polynomial. Firstly, we define an adjacency matrix basis $\mathbf{A}$ to capture the overall dependence for each variable pair. Then, we use a matrix polynomial of $\mathbf{A}$ to represent the correlation of each time step, where the coefficients are determined by time. To generate coefficients for MTS data of arbitrary length, we propose to use a set of cyclic timestamp embedding to index the time step. As a result, TPGNN captures the dynamic correlation with a temporal matrix polynomial, which solves the first challenge effectively. For the second challenge, we analyze the approximation error of the TPG module and provide a theoretical bound on the error. Furthermore, we evaluate the existent dependence learning methods on six synthetic MTS datasets with ground-truth variable dependence. The main contributions of the work are as follows:

- We propose a novel dependence learning module to capture MTS data's dynamic correlation, which represents the variable dependence as a temporal matrix polynomial. The resultant model achieves state-of-the-art forecasting performance on 5 of 6 benchmark datasets with significant improvements.

- We investigate the dependence approximation gap on six synthetic datasets with different levels of dependence complexity. On average, the proposed TPGNN outperforms the best baseline by 23.41% on the approximation error. A theoretical result further demonstrates the dependence learning ability of TPGNN.

2 Related Works

MTS Forecasting There are two main categories of methods for MTS forecasting: implicit-dependence approaches and structural-dependence approaches. One representative method of the first category is LSTNet [20], it captures the variable dependence with convolution over variables. The structural-dependence approaches represent the variable correlation as explicit graphs and predict with GNNs. Graph Wavenet [42] (GWN) attempts to learn the variable correlation with static node representations and captures the temporal pattern with convolution neural networks (CNNs). MTGNN [41] follows the idea of GWN to learn variable correlation statically and proposes a new graph convolution module for MTS forecasting. Although works like GMAN [50], SLC [49], and StemGNN [4] model a dynamic correlation among variables with self-attention, the learned graph is sensitive to the input, causing a significant forecasting variance. Our construction is based on a set of time-variant coefficients, the result is independent of the input and robust in forecasting. Some works that focus on learning time-varying graph structure [26, 33, 45]. However, these graph structures are not captured from the forecast task directly. They thus cause bias and increase the computation cost. There are some transformer-based works [43, 46, 48], but their methods do not thoroughly discuss the dynamic-dependence issues.

GNNs and Polynomial Graph Filters (PGFs) GNNs are based on the message-passing mechanism. One representative work is GraphSAGE [14], which learns a sample-aggregation function to generate the node embedding. Michael et al. [6] generalize the convolutional neural networks (CNNs) to the graph structure. Many filters of GNNs belong to PGFs, and Gama et al. [11] discussed the connections. In order to incorporate the node feature of high-order neighbors, many works introduce PGFs to their methods. APPNP [18] and SGC [40] adopt fixed-weight PGFs and use hyperparameters to define the coefficients. However, the constructed fixed-weight PGFs are biased because the coefficients are task-irrelevant. GPR-GNN [5] constructs a PGF from the graph dataset, where the coefficients are learnable. Besides, there are also some PGFs-based methods for MTS data learning [13, 15, 16]. Although these works are competitive in capturing graph structure, they fail in learning the dynamic correlation, we illustrate the point in Section 5.4. Moreover, the missing/noisy/incomplete adjacency matrix causes difficulties in applying these methods. Our approach introduces a self-adaptive graph to the PGFs construction, which improves the dependence capturing ability and the model versatility.

3 The Framework of TPGNN

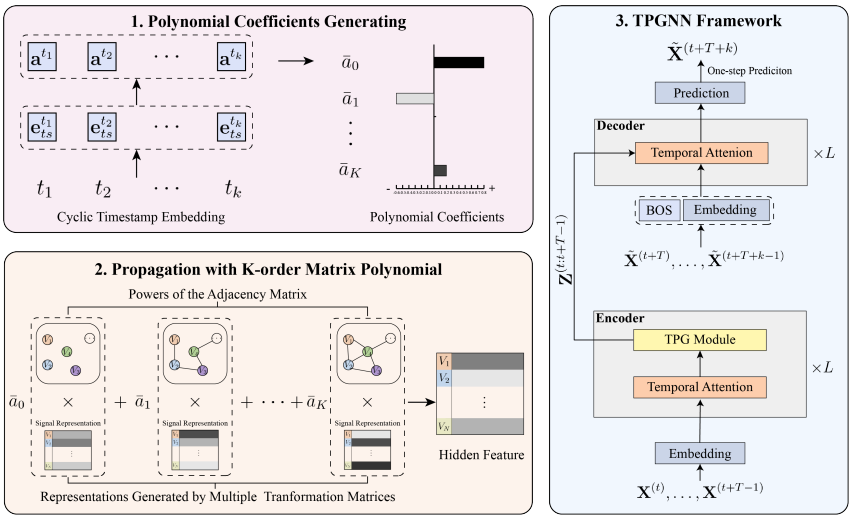


Figure 2: Overview of the proposed TPGNN. (1) The TPG module generates a set of coefficients according to the time steps. (2) We define a matrix polynomial with the coefficients to capture the dependence. (3) TPGNN has an encoder-decoder structure and predicts in an auto-regressive manner.

In this section, we first define the MTS forecasting task in the context of graph signal processing. We then introduce the TPG module and illustrate the inference pipeline of our framework.

3.1 Problem Defintion

At time step t, the MTS data is a signal set $\mathcal{G}^{(t)} = \{\mathcal{V}, \mathcal{E}^{(t)}, \mathbf{X}^{(t)}, \mathbf{W}^{(t)}\}$. The node set $\mathcal{V}$ ($|\mathcal{V}| = N$) contains N variables of MTS data, $\mathcal{E}^{(t)}$ is the edge set, $\mathbf{X}^{(t)} \in \mathbb{R}^{N\times 1}$ records the signal of N variables, and $\mathbf{W}^{(t)} \in \mathbb{R}^{N\times N}$ is a weighted adjacency matrix. If variables i and j are dependent, then an edge connects the two nodes, and entry $\mathbf{W}_{ij}^{(t)}$ indicates their correlation. Otherwise, $\mathbf{W}_{ij}^{(t)}$ equals to zero. Given T observations of the signal sets, our problem is to find a function F that can forecast the variable state of the subsequent T' signals:

$$(\mathcal{G}^{(t)}, \mathcal{G}^{(t+1)}, \ldots, \mathcal{G}^{(t+T-1)}) \xrightarrow{F} \mathbf{X}^{(t+T)}, \mathbf{X}^{(t+T+1)}, \ldots, \mathbf{X}^{(t+T+T'-1)}. \tag{1}$$

The setting regards the graph structure as a latent feature of MTS data, and the forecast target is the same as the standard setting. Prevailing works [25, 41, 42] regard $\mathbf{W}^{(t)}$ as a time-invariant matrix, while in our work, $\mathbf{W}^{(t)}$ changes with the time.

3.2 Represent the Correlation as a Temporal Matrix Polynomial

To capture $\mathcal{E}$ and $\mathbf{W}^t$ for MTS data, we construct a series of matrix polynomials that share a matrix basis. The TPG module controls the influence of each polynomial term temporally with a set of adaptive coefficients. As a result, we can approximate a wide range of dynamic graph structures.

Counstructing the adjacency matrix basis. We firstly define an initial adjacency matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$ following the self-adaptive graph proposed by Wu et al. [42]. They learn variable embeddings $\mathbf{E} \in \mathbb{R}^{N\times c}$ (c is the embedding dimension) from MTS data and define the variable dependence with embedding similarity. Besides, they further remove weak connections with $\mathrm{ReLU}(\cdot)$ and normalize the result with $\mathrm{SoftMax}(\cdot)$.

$$\mathbf{A} = \mathrm{SoftMax}(\mathrm{ReLU}(\mathbf{E}\mathbf{E}^{\mathbf{T}})) \tag{2}$$

For MTS data that have a prior structure, e.g., the physical distance among sensors, we denote the corresponding adjacency matrix as $\mathbf{W} \in \mathbb{R}^{N\times N}$. To incorporate $\mathbf{W}$ into the encoding process, we first calculate its symmetric normalized Laplacian [17] $\mathbf{L} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{I} + \mathbf{W})\mathbf{D}^{-\frac{1}{2}}$ ($\mathbf{I}$ is the identity matrix and $\mathbf{D}$ is the degree matrix), then combine the two results to define the initial adjacency matrix:

$$\mathbf{A} = \mathrm{SoftMax}(\mathrm{ReLU}(\mathbf{E}\mathbf{E}^{\mathbf{T}})) + \mathbf{L} \tag{3}$$

Generating the temporal polynomial coefficients. At time step t, we represent the variable dependence as a K-order matrix polynomial of $\mathbf{A}$:

$$\mathbf{W}^{(t)} = \Sigma_{k=0}^{K} a_k^{(t)} \mathbf{A}^k \tag{4}$$

Due to the polynomial coefficients being time-variant, $\mathbf{W}^{(t)}$ shows dynamic behavior. However, learning coefficients for every time step is impractical because the sequence length grows infinitely. Fortunately, the periodicity property is common in MTS data [8, 29]. Therefore, we propose to use a set of cyclic timestamp embeddings to index the time step. Firstly, we define T_p timestamp embeddings $(\mathbf{e}_{ts}^{(1)}, \ldots, \mathbf{e}_{ts}^{(T_p)}) \in \mathbb{R}^{D_e}$, where T_p is the cycle and D_e is the embedding dimension. Following a cyclic order, we then assign the $(t\%T_p)$-th embedding to time step t, where $\%$ is the remainder operator. Finally, we generate the coefficients for time step t by the $\mathbf{e}_{ts}^{(t\%T_p)}$. As shown in Figure 2 (1), Given T historical graph signals start from time step t, we calculate the coefficients $\mathbf{a} \in \mathbb{R}^{K+1}$ for each graph signal with a coefficient matrix $\mathbf{W}_c \in \mathbb{R}^{D_e\times(K+1)}$.

$$(\mathbf{a}^{(t)}, \ldots, \mathbf{a}^{(t+T-1)}) = (\mathbf{e}_{ts}^{(t\%T_p)}, \ldots, \mathbf{e}_{ts}^{(t+T-1\%T_p)})\mathbf{W}_c \tag{5}$$

According to Equation 4, each $\mathbf{a}^{(t)}$ defines a K-order matrix polynomial of $\mathbf{A}$, representing variable dependence of time step t. Instead of predicting with T different matrix polynomials, we use an average result to improve the prediction efficiency and robustness. Due to the linearity of the polynomial, the average polynomial is equivalent to the polynomial with average coefficients. We calculate the final coefficients $\bar{\mathbf{a}}$ with another parameter matrix $\mathbf{W}_a \in \mathbb{R}^{T\times 1}$ (average matrix).

$$\bar{\mathbf{a}} = (\bar{a}_0, \bar{a}_1, \ldots, \bar{a}_K) = (\mathbf{a}^{(t)}, \ldots, \mathbf{a}^{(t+T-1)})\mathbf{W}_a \tag{6}$$

In Figure 2 (2), we illusrate the TPG module's propagation process with $\bar{\mathbf{a}}$. Let $\mathbf{X} \in \mathbb{R}^{N \times D_e}$ denote the matrix signal embedding, $\mathbf{W}_k \in \mathbb{R}^{D_e \times D_e}, k = 0, \ldots, K$ denote the model parameter matrices, $||\cdot||_F$ denote the Frobenius norm of matrix, we derive the hidden feature $\mathbf{Z}^{(t)} \in \mathbb{R}^{N \times D_e}$ as follows:

$$\mathbf{Z}^{(t)} = \Sigma_{k=0}^{K} \bar{a}_k \mathbf{A}^k \mathbf{X}^{(t)} \frac{\mathbf{W}_k}{||\mathbf{W}_k||_F} \tag{7}$$

TPG module follows the diffusion graph convolution layer [23] to introduce $(K+1)$ parameter matrices for enhancing representation diversity. Besides, we decouple the parameter matrix norm and coefficient by normalizing $\mathbf{W}_k$ with a factor of $\frac{1}{||\mathbf{W}_k||_F}$, since $\mathbf{W}_k$ with a large norm increases the k-th term's contribution. As a result, $\bar{\mathbf{a}}$ completely controls the contribution of each term.

3.3 Inference Pipeline

In Figure 2 (3), TPGNN has an encoder-decoder structure. First, we derive the historical data embedding with a linear transformation, then feed the results into the encoder. The encoder generates data encoding $\mathbf{Z}^{(t:t+T-1)} \in \mathbb{R}^{T \times N \times D_e}$ for time step t to $t+T-1$ with the TPG module and temporal attention layer. The temporal attention layer follows the design of Transformer [37], which uses self-attention to capture intra-series patterns for each variable. Finally, the decoder forecasts with the data encoding and temporal attention layer. For the first prediction of time step $t+T$, the decoder queries the data encoding with a learnable beginning of sentence (BOS) token $\mathbf{E}_{BOS} \in \mathbb{R}^{N \times D_e}$. The query result represents the first prediction, denoted as $\mathbf{E}_X^{(t+T)} \in \mathbb{R}^{N \times D_e}$. We use a prediction matrix $\mathbf{W}_{pred} \in \mathbb{R}^{D_e \times 1}$ to get the forecasting result $\tilde{\mathbf{X}}^{(t+T)} \in \mathbb{R}^{N \times 1}$.

$$\begin{aligned} \mathbf{E}_X^{(t+T)} &= \mathrm{Decoder}(\mathbf{E}_{\mathbf{BOS}}, \mathbf{Z}^{(t:t+T-1)}), \\ \tilde{\mathbf{X}}^{(t+T)} &= \mathbf{E}_X^{(t+T)} \mathbf{W}_{pred}, \end{aligned} \tag{8}$$

After generating the subsequent k query results $(\mathbf{E}_X^{(t+T)}, \ldots, \mathbf{E}_X^{(t+T+k-1)})$, the decoder derives the $(k+1)$-th result $\mathbf{E}_X^{(t+T+k)}$ by querying $\mathbf{Z}^{(t:t+T-1)}$ with the latest L_{max} query results. The forecast basically follows an auto-regressive (AR) mechanism, where L_{max} is the maximum query length.

$$\begin{aligned} \mathbf{E}_X^{(t+T+k)} &= \mathrm{Decoder}((\mathbf{E}_X^{(t+T+k-L_{max})}, \ldots, \mathbf{E}_X^{(t+T+k-1)}), \mathbf{Z}), \\ \tilde{\mathbf{X}}^{(t+T+k)} &= \mathbf{E}_X^{(t+T+k)} \mathbf{W}_{pred}. \end{aligned} \tag{9}$$

The above process continues until getting the subsequent T' forecasts, such AR mechanism helps TPGNN to capture long-term intra-series dependence with informative context.

4 Theoretical Properties of TPGNN

To explore the dependence learning capability of TPGNN, we focus on analyzing the theoretical gap between the optimal graph structure and the TPG module's approximation in this section.

TPGNN is able to achieve perfect approximation. Unlike the real function, the approximation of graph structure largely depends on the topology property of the target graph, where the lower bound of the approximation error is nonzero in many cases. However, TPGNN mitigates the gap for a wide range of dynamic graph structures with a small group of parameters, which leads to a robust and accurate forecasting result. The approximation capability of TPGNN is concluded as the following result, we show the detailed statement and proofs in Appendix A.9 due to the space limitation.

Theorem 1. Let $\mathbf{G}^{(1)}, \ldots, \mathbf{G}^{(T)} \in \mathbb{R}^{N \times N}$ be the symmetric normalized Laplacian of the optimal graph structure for time step 1 to T, $\mathbf{A} \in \mathbb{R}^{N \times N}$ is the initial adjacency matrix of TPGNN. We model these Laplacians with Equation 4, and the corresponding approximation error $\mathrm{e}^{(1:T)} = \frac{1}{T}\Sigma_{t=1}^{T}||\mathbf{W}^{(t)} - \mathbf{G}^{(t)}||_F^2$, where the $||\cdot||_F$ is the Frobenius norm of the matrix. Suppose all the matrices are symmetric (undirected graph), $\mathbf{G}_i\mathbf{G}_j = \mathbf{G}_j\mathbf{G}_i, \forall i, j$, $\mathbf{A}$ has N different singular values, and the polynomial's order is large enough. Then we have the following estimation for $\mathrm{e}^{(1:T)}$.

$$(1 - \lambda_{max})\mathbb{E}_t||\mathbf{G}^{(t)}||_F^2 \leq \mathrm{e}^{(1:T)} \leq (1 - \lambda_{min})\mathbb{E}_t||\mathbf{G}^{(t)}||_F^2, \tag{10}$$

where $\mathbb{E}_t||\mathbf{G}^{(t)}||_F^2 = \frac{1}{T}\Sigma_{t=1}^{T}||\mathbf{G}^{(t)}||_F^2$ is the average norm of the Laplacians, $\lambda_{min}, \lambda_{max} \in [0, 1]$ are the minimum and maximum eigenvalues of a constant matrix. If $\mathbf{A}$ is commutative with $\mathbf{G}^{(t)}$, then $\lambda_{max} = \lambda_{min} = 1$, i.e., we achieve a uniform perfect approximation. Although the zero error depends on the commutative condition, $\mathbf{A}$ is learnable with Equation 2, which relaxes the restriction. Moreover, TPGNN empirically achieves the best approximation compared with other methods.

5 Experiments

Table 1: Dataset statistics.

Datasets	# Samples	# Nodes	Sample Rate	Input Length	Output Length
Traffic	17,544	862	1 hour	168	1
Solar-Energy	52,560	137	10 minutes	168	1
Electricity	26304	321	1 hour	168	1
Exchange-Rate	7,588	8	1 day	168	1
PEMS-D7	12672	228	5 minutes	12	12
PEMS-Bay	52116	325	5 minutes	12	12

5.1 Experimental Setup

We validate the performance of TPGNN on two tasks: single-step and multi-step MTS forecasting. In Table 1, we present the statistics of six benchmark datasets. The first four benchmark datasets [2, 19] aim to predict a single future step; the last two datasets [23,44] record a distance matrix among sensors, where the goal is to forecast multiple future steps. More details about the datasets are given in the AppendixA.3. For single-step prediction, the baselines are VAR-MLP [47], GP [31], RNN-GRU [41], LSTNet [20], TPA-LSTM [34], and MTGNN [41]. MTGNN firstly introduces GNNs to general MTS data and achieves state-of-the-art (SOTA) performance; other baselines are representative

Table 2: We forecast 3, 6, 12, and 24 horizons for the first four real-world datasets, and models with high CORR and low RSE are preferred. TPGNN outperforms other models on three of the first four datasets. For the remaining two datasets, we predict the next 15, 30, and 60 minutes based on one-hour observation, and the three metrics represent the prediction error. The results show that TPGNN achieves state-of-the-art performance on both short-term and long-term predictions.

Dataset		Solar-Energy				Traffi				Electricity				Exchange-Rate			
		Horizon				Horizon				Horizon				Horizon			
Methods	Metric	3	6	12	24	3	6	12	24	3	6	12	24	3	6	12	24
VARMLP	RSE	0.1922	0.2679	0.4244	0.6841	0.5582	0.6579	0.6023	0.6146	0.1392	0.1620	0.1557	0.1274	0.0265	0.0394	0.0407	0.0578
VARMLP	CORR	0.9829	0.9655	0.9058	0.7149	0.8245	0.7695	0.7929	0.7891	0.8708	0.8389	0.8192	0.8679	0.8609	0.8725	0.8280	0.7675
GP	RSE	0.2259	0.3286	0.5200	0.7973	0.6082	0.6772	0.6406	0.5995	0.1500	0.1907	0.1621	0.1273	0.0239	0.0272	0.0394	0.0580
GP	CORR	0.9751	0.9448	0.8518	0.5971	0.7831	0.7406	0.7671	0.7909	0.8670	0.8334	0.8394	0.8818	0.8713	0.8193	0.8484	0.8278
RNN-GRU	RSE	0.1932	0.2628	0.4163	0.4852	0.5358	0.5522	0.5562	0.5633	0.1102	0.1144	0.1183	0.1295	0.0192	0.0264	0.0408	0.0626
RNN-GRU	CORR	0.9823	0.9675	0.9150	0.8823	0.8511	0.8405	0.8345	0.8300	0.8597	0.8623	0.8472	0.8651	0.9786	**0.9712**	0.9513	0.9223
LSTNet	RSE	0.1843	0.2559	0.3254	0.4643	0.4777	0.4893	0.4950	0.4973	0.0864	0.0931	0.1007	0.1007	0.0226	0.0280	0.0356	0.0449
LSTNet	CORR	0.9843	0.9690	0.9467	0.8870	0.8721	0.8690	0.8614	0.8588	0.9283	0.9135	0.9077	0.9119	0.9735	0.9658	0.9511	0.9354
TPA-LSTM	RSE	0.1803	**0.2347**	0.3234	0.4389	0.4487	0.4658	0.4641	0.4765	0.0823	0.0916	0.0964	0.1006	0.0174	**0.0241**	**0.0341**	**0.0444**
TPA-LSTM	CORR	0.9850	**0.9742**	0.9487	0.9081	0.8812	0.8717	0.8717	0.8629	0.9439	0.9337	0.9250	0.9133	0.9790	0.9709	**0.9564**	**0.9381**
MTGNN	RSE	**0.1778**	0.2348	0.3109	0.4270	0.4162	0.4754	**0.4461**	**0.4535**	0.0745	0.0878	0.0916	0.0953	0.0194	0.0259	0.0349	0.0456
MTGNN	CORR	**0.9852**	0.9726	0.9509	0.9031	0.8963	0.8667	0.8794	0.8810	**0.9474**	0.9316	0.9278	0.9234	0.9786	0.9708	0.9551	0.9372
TPGNN	RSE	0.1850	0.2412	**0.3059**	**0.3498**	**0.3989**	**0.4715**	0.4476	0.4696	**0.0627**	**0.0685**	**0.0699**	**0.0936**	**0.0174**	0.0250	0.0350	0.0458
TPGNN	CORR	0.9840	0.9716	**0.9529**	**0.9710**	**0.9232**	**0.8945**	**0.9028**	**0.8858**	0.9417	**0.9362**	**0.9285**	**0.9293**	**0.9792**	0.9687	0.9509	0.9306

Model	PEMS-BAY (Horizon 3/6/12)			PEMS-D7 (Horizon 3/6/12)		
	MAE	MAPE(%)	RMSE	MAE	MAPE(%)	RMSE
ARIMA	1.62/2.33/3.38	3.50/5.40/8.30	3.30/4.76/6.50	5.55/5.86/6.27	12.92/13.94/15.20	9.00/9.13/9.38
FC-LSTM	2.05/2.20/2.37	4.80/5.20/5.70	4.19/4.55/4.96	3.57/3.92/4.16	8.60/9.55/10.10	6.20/7.03/7.51
STGCN	1.39/1.84/2.42	3.00/4.22/5.58	2.92/4.12/5.33	2.25/3.03/4.02	5.26/7.33/9.85	4.04/5.70/7.64
DCRNN	1.38/1.74/2.07	2.90/3.90/4.90	2.95/3.97/4.74	2.25/2.98/3.83	5.30/7.39/9.85	4.04/5.58/7.19
StemGNN	1.52/1.94/2.45	3.38/4.58/6.03	3.06/4.07/5.04	2.94/3.66/4.66	7.63/9.66/12.58	5.05/6.35/8.00
Graph WaveNet	1.30/**1.63**/1.95	2.73/3.67/4.63	2.74/3.70/4.52	2.18/2.95/3.88	5.02/7.22/10.03	4.18/5.82/7.61
Informer	2.30/2.40/2.55	5.02/5.32/5.73	4.21/4.49/4.85	3.64/3.77/4.09	8.66/9.07/9.87	6.02/6.34/6.85
MTGNN	1.32/1.65/**1.94**	2.77/3.69/4.53	2.79/3.74/**4.49**	2.17/2.89/4.02	5.03/6.93/9.93	**4.01**/5.84/8.78
TPGNN	**1.26**/1.65/2.05	**2.56/3.47/4.40**	**2.64/3.65**/4.58	**2.12/2.72/3.22**	**5.00/6.73/8.22**	4.05/**5.45/6.56**

non-GNNs-based methods. For multi-step prediction, the baselines are ARIMA [23],FC-LSTM [36], DCRNN [22], STGCN [44], StemGNN [4], Graph WaveNet, Informer [51], and MTGNN; all the baselines are GNN-based except ARIMA, FC-LSTM and Informer. These GNNs-based baselines are representative methods for MTS data with a prior structure. Besides, the Informer is a novel Transformer-based model that achieves SOTA performance on long-term forecasting. We list the paramter scale of each model in Appendix A.4 to show our method is light-weight. We use five evaluation metrics [41]: Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), Root Mean Squared Error (RMSE), Root Relative Squared Error (RRSE), and Empirical Correlation Coefficient (CORR). For CORR, higher values are better. For the other metrics, lower values are better. We tune the hyperparameters on the validation dataset, the results are presented in the Appendix A.7. We divide the dataset into three parts for training, validation, and testing with a ratio of 7:1:2. More details about the baselines, metrics, and experiment settings are in Appendix A.4A.5A.6.

5.2 Main Results

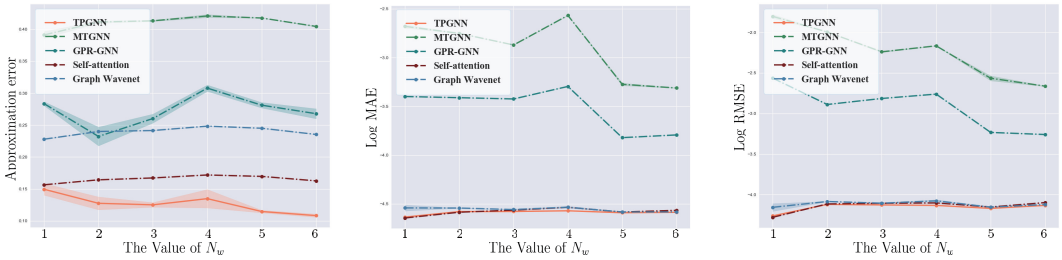
Table 2 summarizes the experimental results of TPGNN. Generally, TPGNN achieves the state-of-the-art on most of the datasets. Our framework has an on-par performance for the single-step forecasting task with the SOTA methods like MTGNN and TPA-LSTM. Moreover, TPGNN makes a significant improvement on Traffic and Electricity datasets. On the Solar-Energy, TPGNN lowers RSE by 1.61%, 18.08% and increases CORR by 1.47%, 6.93% over the horizons of 12, 24. Our method improves the CORR by 2.21% on average over the four horizons of the Traffic. Moreover, we lower the RSE by 15.82% on average over the four horizons of Electricity. Our method fails to achieve SOTA performance on the exchange-rate data. We think the main reason is the small sample size, which causes difficulties in capturing the dynamic variable dependence. The second row of Table 2 concludes the results of the multi-step forecasting task, which demonstrates that TPGNN achieves SOTA performance on the two datasets with prior structure. On the PEMS-BAY, TPGNN lowers the MAPE by 5.47% on average over the three horizons. Besides, The RMSE is reduced by $4.30\%, 2.41\%$ over the horizons of 3 and 6. TPGNN reduces the MAE/MAPE by $8.04\%/6.61\%$ on average over the three horizons on the PEMS-D7 dataset. The RMSE is lowered by $2.33\%, 4.23\%$ over the horizon of 6 and 12. Although the prior structure is provided to each baseline, the results on the last two traffic datasets demonstrate that TPGNN can capture more precise variable dependence and intra-series patterns than other GNNs-based methods.

5.3 Ablation Experiments

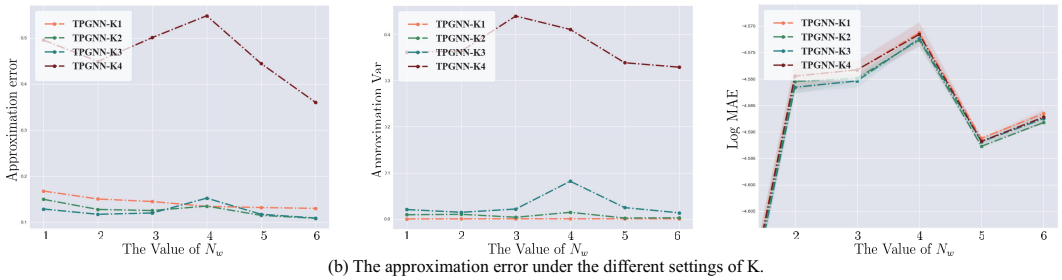
Table 3: Long-term prediction results for ablation study on the PEMS-D7 dataset.

Metrics	TPGNN	w/o TPG	w/o dynamic	w/o overview	w/o normalize	w/o K-matrices
MAE	**3.215±0.014**	3.701±0.021	3.480±0.047	3.330±0.031	3.261±0.035	3.294±0.036
MAPE(%)	**8.221±0.035**	9.553±0.106	8.873±0.014	8.539±0.118	8.347±0.061	8.406±0.245
RMSE	**6.559±0.023**	7.473±0.033	6.989±0.084	6.806±0.098	6.671±0.033	6.701±0.065

The PEMS-D7 includes a prior structure and has complex inter-series and intra-series dependence, we conduct ablation experiments on the PEMS-D7 to evaluate the effectiveness of each design in our method. Due to the importance and complexity of the long-term prediction, we set the horizon as 12. More results of other horizons can be found in Appendix A.8. There are five experiments in total, we repeat each experiment 5 times and report the average and standard deviation (std). **w/o TPG**: TPGNN without the TPG module. We replace the graph constructed by TPG with the prior structure of PEMS-D7. **w/o dynamic**: TPGNN without the time-varying coefficients. We adopt a set of learnable static coefficients to construct the graph. **w/o overview**: TPGNN does not use average coefficients in Equation 6. We predict with T graphs defined by Equation 5. **w/o normalize**: TPGNN without the normalization factor in Equation 7. **w/o K-matrices**: TPGNN without multiple parameter matrices in Equation 7. We use an identical normalized matrix to derive the hidden feature. Table 3 summarizes the experimental results, which indicates that all the designs are indispensable. The result of w/o TPG illustrates the TPG module's critical contribution to our method. Although we replace the graph constructed by the TPG module with the prior structure given by PEMS-D7, the performance has a steep decline by $13.13\%, 13.94\%, 12.23\%$ on the three metrics. w/o dynamic follows the strategy of GPR-GNN [5] that uses static coefficients to capture graph structure from data. However, the results indicate the importance of introducing time-variant coefficients, which



(a) The approximation and prediction error of different methods.



(b) The approximation error under the different settings of K.

Figure 3: The approximation error and prediction error of different graph approximation methods. (a) compares TPGNN with other baselines, the results indicate TPGNN significantly reduces the approximation error and achieves the highest prediction accuracy. (b) investigates the effect of polynomial order to the dependence approximation error. The high-order polynomials impairs the precision and robustness of the dependence learning.

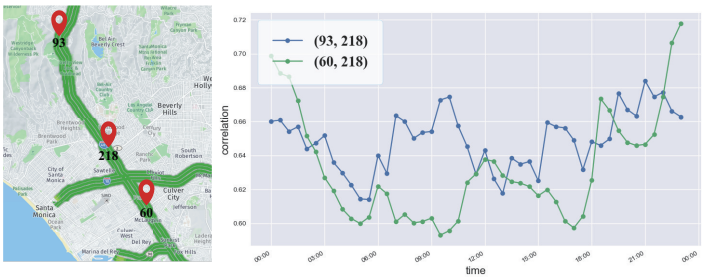
significantly improve the long-term prediction accuracy by 7.04% on average. w/o overview and w/o K-matrices show large variance compared with other results, indicating the two modules are essential for improving model robustness. Besides, the w/o overview's performance is close to the w/o dynamic, demonstrating the necessity of using the average coefficients. w/o normalize has comparable performance to the TPGNN, possibly due to the weight decay causing similar norms. Nevertheless, it demonstrates that the normalization helps TPGNN capture the variable dependence.

Table 4: Average results on six synthetic datasets configured with different N_w.

Method	TPGNN	Graph Wavenet	MTGNN	GPR-GNN	Self-attention
MFE	**0.1266**$\pm 7.057\times 10^{-3}$	$0.2397\pm 3.476\times 10^{-4}$	$0.4098\pm 1.981\times 10^{-3}$	$0.2721\pm 7.060\times 10^{-3}$	$0.1653\pm 1.469\times 10^{-6}$
MAE	**0.01015**$\pm 1.442\times 10^{-5}$	$0.01049\pm 10.30\times 10^{-5}$	$0.05669\pm 40.29\times 10^{-5}$	$0.03008\pm 1.638\times 10^{-5}$	$0.01026\pm 1.289\times 10^{-5}$
RMSE	**0.01568**$\pm 1.986\times 10^{-5}$	$0.01632\pm 2.276\times 10^{-4}$	$0.1114\pm 1.344\times 10^{-3}$	$0.05561\pm 3.537\times 10^{-5}$	$0.01593\pm 2.051\times 10^{-5}$

5.4 Study of the Graph Structure Approximation Gap

Synthetic data. To investigate the empirical gap between the optimal graph structure and the approximation result, we propose to generate MTS data with ground-truth dynamic dependence using a non-repeating random walk (NRW) model [7], which is widely adopted in time-series data generation [1, 32]. At time step t, we synthesize the matrix signal $\mathbf{X}^{(t)} \in \mathbb{R}^{N\times 1}$ with a dynamic weighted adjacency matrix $\mathbf{W}^{(t)}$ by $\mathbf{X}^{(t)} = \mathcal{N}(\mathbf{W}^{(t-1)}\mathbf{X}^{(t-1)}, \sigma)$, where $\mathcal{N}(\cdot,\cdot)$ is the normal distribution, $\sigma \in \mathbb{R}$ controls the variance, and $\mathbf{X}^{(0)}$ is sampled from a discrete uniform distribution. We define $\mathbf{W}^{(t)}$ by traversing N_w constant matrices $(\mathbf{G}^{(1)}, \ldots, \mathbf{G}^{(N_w)})$ in a cyclic order, where each $\mathbf{G}^{(i)}$ is a Laplacian of sparsified random adjacency matrix. Given a period of $T_p, T_p \geq N_w$, we partition the period into N_w even intervals of length T_s, and each interval shares one of the matrices, i.e., $\mathbf{W}^{(t)} = \mathbf{G}^{([(t\%T_p)/T_s])}$, where $[\cdot]$ is the rounding function. We point out that N_w controls the complexity of dynamic dependence, and $N_w = 1$ corresponds to a static-dependence case. To increase the data diversity, we randomly initialize the $X^{(t)}$ for every T_p step. We set $T_p = 120, \sigma = 0.001$ in our evaluation and generate six MTS data of length 2400 with $N_w = 1, \ldots, 6$. Besides, each



(a) The physical locations of nodes 218, 93, and 60. (b) The dynamic dependence learned by TPGNN of (93, 218) and (60, 218).

Figure 4: Three nodes are highly correlated in the TPG's learned result but independent in the prior structure. We find they are on the same road and have different dynamic correlation trends.

synthetic dataset is divided into three parts for training, validation, and testing with a ratio of 7:1:2. More details about the synthetic algorithm, configuration, and synthetic results are in Appendix A.3.

Predicting with different graph approximation methods. We evaluate the dependence capturing ability of different methods on the six synthetic datasets, where the task is the next-step prediction based on current variable states. Suppose that $\tilde{\mathbf{W}}^{(t)} \in \mathbb{R}^{N \times N}, t = 0, \ldots, T-1$ is the learned adjacency matrix series, we measure the approximation gap with a Mean Frobenius Error (MFE) metric, defined by $MFE = \frac{1}{TN^2} \Sigma_{t=0}^{T-1} ||\mathbf{W}^t - \tilde{\mathbf{W}}^{(t)}||_F$. Besides, we use MAE/RMSE to measure the forecasting accuracy. The baselines are Graph Wavenet (GWN), MTGNN, GPR-GNN, and Self-attention. The first three methods are static-dependence-based. Self-attention constructs a dynamic graph, which is adopted in many works like GMAN [50] and StemGNN [4]. We follow the implementation of StemGNN to learn the graph structure with a GRU module and self-attention operator in the evaluation. The polynomial order K is fixed as 2 in the evaluation, we repeat each experiment 5 times and report the average/std, more details about the settings are in Appenedix A.6. The main results are summarized in Figure 3 (a) and Table 4, demonstrating that TPGNN outperforms other baselines on the six synthetic datasets. In Table 4, TPGNN reduces the graph approximation error by 23.41% on average over the six datasets. The first chart of Figure 3 (a) also illustrates the improvement, we consistently achieve the best approximation under different N_w. Moreover, the MFE of TPGNN has a descending trend with the increase of N_w while other methods' approximation errors are increasing. We think the TPG module overfits on datasets with small N_w, since their behavior tends to have static dependence. TPGNN achieves the lowest prediction error on average, the improvement in the accuracy is slight compared with MFE because the task is a simple next-step prediction. Furthermore, we observe that the ranking on MFE is identical to the MAE/RMSE, which demonstrates the importance of capturing precise variable dependence for MTS forecasting.

The effect of polynomial order. We further investigate the relation between approximation error and the polynomial order K, where the K ranges from 1 to 4, and $K = 1$ corresponds to a linear approximation. The first two charts of Figure 3 (b) present the average MFE and corresponding std, respectively. The results show that the approximation variance increases significantly with K, especially when $K = 4$. Moreover, TPGNN-K4 has a large MFE on the six datasets compared with other configurations. Therefore, we think a high-order matrix polynomial impairs both the precision and robustness of the dependence capturing ability of TPGNN. We point out that the observation is non-trivial since GPR-GNN achieves good results with a large K ($K \geq 10$) on normal graph datasets, possibly due to the difference between MTS forecasting and node classification. The third chart shows the prediction error, it also validates that model with low MFE has a low prediction error.

Case study on the real-world dataset. Due to the absence of ground-truth dependence, we illustrate the correlation modeling ability of TPGNN on real-world datasets with an example of PEMS-D7. We find three highly correlated nodes in the learned result but independent in the prior structure, where the node IDs are 60, 93, and 218. In Figure 4 (a), we mark the physical location of the three nodes according to PEMS [3]. The three nodes are on the same road, which illustrates that TPGNN can learn reliable dependence from data. In Figure 4 (b), we show the learned correlation of pairs $(60, 218)$ and $(93, 218)$. We observe that $(93, 218)$ has a stable dependence and $(60, 218)$'s correlation increases in the evening, the difference possibly caused by the branch road between 218 and 60.

6 Conclusion

We proposes a novel GNNs-based method for MTS forecasting, TPGNN, which represents the variable dependence as a temporal polynomial matrix. The results on six real-world datasets illustrate the outstanding forecasting performance of TPGNN. Besides, extensive experiments on synthetic datasets demonstrate that our method achieves the best dependence approximation than other baselines.

Limitations and future works Although TPGNN achieves outstanding performance on MTS forecasting and dependence capturing, our methods have limitations. First, the dependence captured by TPGNN is not a strict causal structure. Therefore, some correlations are unreliable and decrease the model's robustness. Second, MTS forecasting in the real world is affected by rare events like weather disasters, which may significantly impair prediction accuracy. The future works focus on causal learning and transfer learning [9, 24, 28, 38] on MTS data to solve the above limitations.

7 Acknowledgment

The work of the authors was supported by the National Natural Science Foundation of China (No. 62132017).

References

[1] ADHIKARI, R., AND AGRAWAL, R. K. A combination of artificial neural network and random walk models for financial time series forecasting. *Neural Computing and Applications 24* (2013), 1441–1449.

[2] ASUNCION, A., AND NEWMAN, D. J. Uci machine learning repository. [EB/OL], 2007. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

[3] CA.GOV. Pems. [EB/OL], 2010. `https://pems.dot.ca.gov/`.

[4] CAO, D., WANG, Y., DUAN, J., ZHANG, C., ZHU, X., HUANG, C., TONG, Y., XU, B., BAI, J., TONG, J., ET AL. Spectral temporal graph neural network for multivariate time-series forecasting. *arXiv preprint arXiv:2103.07719* (2021).

[5] CHIEN, E., PENG, J., LI, P., AND MILENKOVIC, O. Adaptive universal generalized pagerank graph neural network. *arXiv: Learning* (2021).

[6] DEFFERRARD, M., BRESSON, X., AND VANDERGHEYNST, P. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS* (2016).

[7] DENTON, A. M. Kernel-density-based clustering of time series subsequences using a continuous random-walk noise model. *Fifth IEEE International Conference on Data Mining (ICDM'05)* (2005), 8 pp.–.

[8] DLOTKO, P., QIU, W., AND RUDKIN, S. Cyclicality, periodicity and the topology of time series. *ArXiv abs/1905.12118* (2019).

[9] FENG, H., YOU, Z., CHEN, M., ZHANG, T., ZHU, M., WU, F., WU, C., AND CHEN, W. KD3A: unsupervised multi-source decentralized domain adaptation via knowledge distillation. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event* (2021), M. Meila and T. Zhang, Eds., vol. 139 of *Proceedings of Machine Learning Research*, PMLR, pp. 3274–3283.

[10] FRIGOLA, R. *Bayesian time series learning with Gaussian processes*. PhD thesis, University of Cambridge, 2015.

[11] GAMA, F., ISUFI, E., LEUS, G., AND RIBEIRO, A. Graphs, convolutions, and neural networks: From graph filters to graph neural networks. *IEEE Signal Processing Magazine 37* (2020), 128–138.

[12] GUO, S., LIN, Y., FENG, N., SONG, C., AND WAN, H. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2019), vol. 33, pp. 922–929.

[13] HADOU, S., KANATSOULIS, C. I., AND RIBEIRO, A. Space-time graph neural networks. *ArXiv abs/2110.02880* (2021).

[14] HAMILTON, W. L., YING, R., AND LESKOVEC, J. Inductive representation learning on large graphs. *arXiv preprint arXiv:1706.02216* (2017).

[15] ISUFI, E., LOUKAS, A., PERRAUDIN, N., AND LEUS, G. Forecasting time series with varma recursions on graphs. *IEEE Transactions on Signal Processing 67* (2019), 4870–4885.

[16] ISUFI, E., AND MAZZOLA, G. Graph-time convolutional neural networks. *2021 IEEE Data Science and Learning Workshop (DSLW)* (2021), 1–6.

[17] KIPF, T. N., AND WELLING, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[18] KLICPERA, J., BOJCHEVSKI, A., AND GÜNNEMANN, S. Predict then propagate: Graph neural networks meet personalized pagerank. In *ICLR* (2019).

[19] LAI, G., CHANG, W.-C., YANG, Y., AND LIU, H. Modeling long- and short-term temporal patterns with deep neural networks. *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval* (2018).

[20] LAI, G., CHANG, W.-C., YANG, Y., AND LIU, H. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval* (2018), pp. 95–104.

[21] LEE, W. K. Partial correlation-based attention for multivariate time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2020), vol. 34, pp. 13720–13721.

[22] LI, Y., YU, R., SHAHABI, C., AND LIU, Y. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926* (2017).

[23] LI, Y., YU, R., SHAHABI, C., AND LIU, Y. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv: Learning* (2018).

[24] LIANG, X. S. Normalized multivariate time series causality analysis and causal graph reconstruction. *Entropy 23* (2021).

[25] MENGZHANG, L., AND ZHANXING, Z. Spatial-temporal fusion graph neural networks for traffic flow forecasting. *arXiv preprint arXiv:2012.09641* (2020).

[26] NATALI, A., ISUFI, E., COUTIÑO, M., AND LEUS, G. Learning time-varying graphs from online data. *IEEE Open Journal of Signal Processing 3* (2022), 212–228.

[27] ORESHKIN, B. N., AMINI, A., COYLE, L., AND COATES, M. J. Fc-gaga: Fully connected gated graph architecture for spatio-temporal traffic forecasting. *ArXiv abs/2007.15531* (2021).

[28] PAN, S. J., AND YANG, Q. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering 22* (2010), 1345–1359.

[29] PUECH, T., BOUSSARD, M., D'AMATO, A., AND MILLERAND, G. A fully automated periodicity detection in time series. In *AALTD@PKDD/ECML* (2019).

[30] QING, L., YONGQIN, T., YONG-GUO, H., AND QINGMING, Z. The forecast and the optimization control of the complex traffic flow based on the hybrid immune intelligent algorithm. *The Open Electrical & Electronic Engineering Journal 8* (2014), 245–251.

[31] ROBERTS, S., OSBORNE, M., EBDEN, M., REECE, S., GIBSON, N., AND AIGRAIN, S. Gaussian processes for time-series modelling. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 371*, 1984 (2013), 20110550.

[32] SARIA, S., DUCHI, A., AND KOLLER, D. Discovering deformable motifs in continuous time series data. In *IJCAI* (2011).

[33] SHAFIPOUR, R., SEGARRA, S., MARQUES, A. G., AND MATEOS, G. Identifying the topology of undirected networks from diffused non-stationary graph signals. *IEEE Open Journal of Signal Processing 2* (2021), 171–189.

[34] SHIH, S.-Y., SUN, F.-K., AND LEE, H.-Y. Temporal pattern attention for multivariate time series forecasting. *Machine Learning 108*, 8 (2019), 1421–1441.

[35] SONG, C., LIN, Y., GUO, S., AND WAN, H. Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2020), vol. 34, pp. 914–921.

[36] SUTSKEVER, I., VINYALS, O., AND LE, Q. V. Sequence to sequence learning with neural networks. *arXiv preprint arXiv:1409.3215* (2014).

[37] VASWANI, A., SHAZEER, N., PARMAR, N., USZKOREIT, J., JONES, L., GOMEZ, A. N., KAISER, L., AND POLOSUKHIN, I. Attention is all you need. *arXiv preprint arXiv:1706.03762* (2017).

[38] WEN, T., AND KEYES, R. Time series anomaly detection using convolutional neural networks and transfer learning. *ArXiv abs/1905.13628* (2019).

[39] WU, C.-H., HO, J.-M., AND LEE, D.-T. Travel-time prediction with support vector regression. *IEEE transactions on intelligent transportation systems 5*, 4 (2004), 276–281.

[40] WU, F., ZHANG, T., DE SOUZA, A. H., FIFTY, C., YU, T., AND WEINBERGER, K. Q. Simplifying graph convolutional networks. *ArXiv abs/1902.07153* (2019).

[41] WU, Z., PAN, S., LONG, G., JIANG, J., CHANG, X., AND ZHANG, C. 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* (2020), pp. 753–763.

[42] WU, Z., PAN, S., LONG, G., JIANG, J., AND ZHANG, C. Graph wavenet for deep spatial-temporal graph modeling. *arXiv preprint arXiv:1906.00121* (2019).

[43] XU, M., DAI, W., LIU, C., GAO, X., LIN, W., QI, G.-J., AND XIONG, H. Spatial-temporal transformer networks for traffic flow forecasting. *ArXiv abs/2001.02908* (2020).

[44] YU, B., YIN, H., AND ZHU, Z. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875* (2017).

[45] ZAMAN, B., RAMOS, L. M. L., ROMERO, D., AND BEFERULL-LOZANO, B. Online topology identification from vector autoregressive time series. *IEEE Transactions on Signal Processing 69* (2021), 210–225.

[46] ZERVEAS, G., JAYARAMAN, S., PATEL, D., BHAMIDIPATY, A., AND EICKHOFF, C. A transformer-based framework for multivariate time series representation learning. *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining* (2021).

[47] ZHANG, G. P. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing 50* (2003), 159–175.

[48] ZHANG, J.-W., SUN, Y., YANG, Y., AND CHEN, W. Feature-Proxy Transformer for Few-Shot Segmentation. In *NeurIPS* (2022).

[49] ZHANG, Q., CHANG, J., MENG, G., XIANG, S., AND PAN, C. Spatio-temporal graph structure learning for traffic forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2020), vol. 34, pp. 1177–1185.

[50] ZHENG, C., FAN, X., WANG, C., AND QI, J. Gman: A graph multi-attention network for traffic prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2020), vol. 34, pp. 1234–1241.

[51] ZHOU, H., ZHANG, S., PENG, J., ZHANG, S., LI, J., XIONG, H., AND ZHANG, W. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *AAAI* (2021).

[52] ZHOU, X., YAN, H., ZHANG, H., AND PENG, C. Model predictive control with feedback correction for optimal energy dispatch of a networked microgrid. *Transactions of the Institute of Measurement and Control 41* (2019), 1540 – 1552.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] see Section 6
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] see Appendix A.1
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...

(a) Did you state the full set of assumptions of all theoretical results? [Yes]
(b) Did you include complete proofs of all theoretical results? [Yes] see Appendix A.9

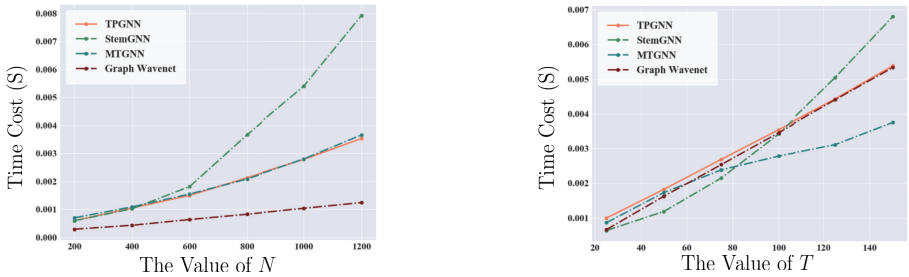
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] see Appendix A.6
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] see Appendix A.6
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] see Section 5.3 5.4 and Appendix A.8
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] see Appendix A.6
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] see Appendix A.3
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] see Appendix A.3
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] see Appendix A.3
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] not applicable
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A] not applicable
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] not applicable

A Appendix

A.1 Limitation

Potential negative societal impacts. Accurate MTS forecasting benefits the application of intelligent systems. Since such systems contain many distributed sensors and produce a large amount of time-series data, predicting the systematic trend is crucial for controlling the system. However, suppose the algorithm is used for financial crimes or other illegal activities. In that case, it will have a bad influence on society. Fortunately, we can address such problems with privacy-preserving methods to improve data safety. Then, we can restrict the negative impacts since the data access is limited. Besides, the privacy-preserving approach also lowers the risk of personal data leakage.

A.2 Time Complexity Analysis



(a) The time complexity to varaible number.

(b) The time complexity to sequence length.

Figure 5: The efficiency of different methods on the synthetic datasets. The results demonstrate that TPGNN is efficient in processing MTS data.

Suppose that the MTS data has N variables, the input sequence length is T, the feature dimension is D_e, and the polynomial order is K. The TPGNN is composed of self-attention and TPG module, and it is known that self-attention has a time complexity as follows:

$$\mathcal{O}(NT^2D_e + NTD_e^2) \tag{11}$$

As the for TPG module, the time complexity for constructing the initial adjacency matrix is $\mathcal{O}(N^2)$, and the time complexity of Equation (7) is $\mathcal{O}(KT(ND_e^2 + N^2D_e))$. As a result, The overall time complexity of TPGNN is $\mathcal{O}(KTN^2D_e + KTND_e^2 + NT^2D_e)$. Since the K, D_E is small in practice, the efficiency of TPGNN is mainly decided by N and T.

In Figure 5, we investigate the efficiency of several SOTA methods on the synthetic datasets. The results show that TPGNN has competitive efficiency with MTGNN/Graph Wavenet, though these methods do not contain self-attention modules.

A.3 Data

We summarize the primary information of benchmark datasets in Table 1. All the six datasets come from the real-world application, we list the details as follows. All the data follow the MIT license.

A.3.1 Single-step forecasting

- Traffic: the traffic dataset from the California Department of Transportation contains road occupancy rates measured by 862 sensors in San Francisco Bay area freeways during 2015 and 2016.
- Solar-Energy: the solar-energy dataset from the National Renewable Energy Laboratory contains the solar power output collected from 137 PV plants in Alabama State in 2007.
- Electricity: the electricity dataset from the UCI Machine Learning Repository contains electricity consumption for 321 clients from 2012 to 2014.

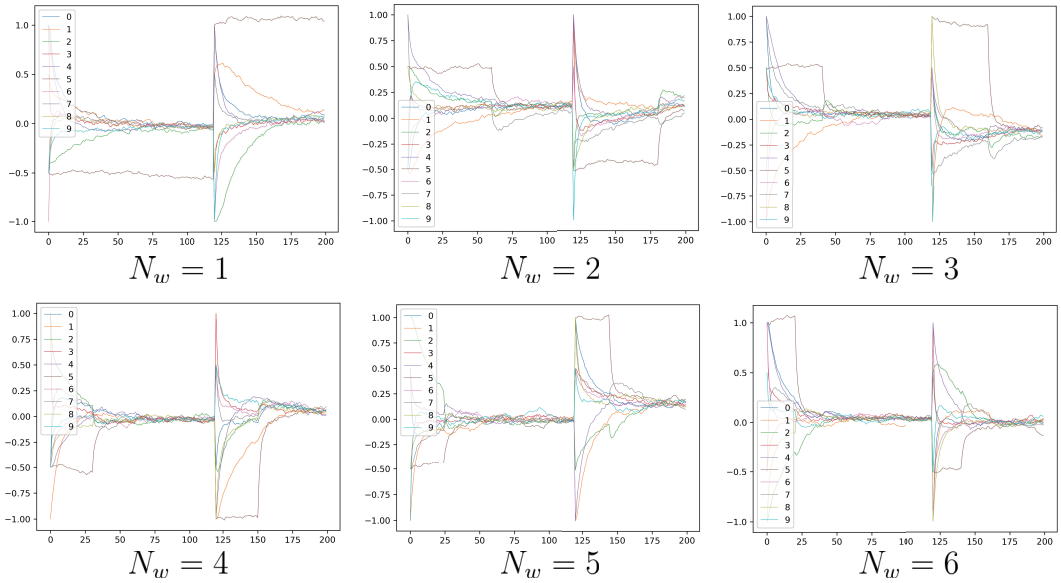


Figure 6: The first ten time-series data of the synthetic results, we show the intial 200 time steps. We observe that the increases of the N_w introduce complex behaviors in the MTS data.

- Exchange-Rate: the exchange-rate dataset contains the daily exchange rates of eight foreign countries, including Australia, British, Canada, Switzerland, China, Japan, New Zealand, and Singapore, ranging from 1990 to 2016.

The input sequence length is 168 and the output sequence length is 1. Models are trained to predict the target future step (horizon) 3, 6, 12, and 24.

A.3.2 Multi-step forecasting

- PEMS-D7: 44-day traffic data collected by 228 sensors in the California state highway system during the weekdays from May through June in 2012.
- PEMS-BAY: 6-month traffic data collected by 325 sensors in the Bay Area of California from January 1st through May 31th in 2017.

For the two traffic datasets, we forecast the subsequent 12 steps by observing a sequence of length 12 and evaluate the model performance at 3, 6, 12 steps.

A.3.3 Synthetic data with the NPR model

For clarification, we illustrate the synthetic algorithm for generating the MTS data in this section, the pseudo-code of our algorithm is showed in Algorithm 1. We firstly generate a ranomd orthogonal matrix $\mathbf{P} \in \mathbb{R}^{N \times N}$ by decomposing a random matrix with SVD decomposition. Then, we define N_w constant matrices with random positive eigenvalues. Moreover, we sparsify the results to the given sparisty threshold to remove unnecessary connections. We then generate the required dynamic dependence MTS data with an NRW model, illustrated in lines 12 to 20 of the pseudo-code.

We further show some examples of the synthetic results in Figure 6, it illustrates that the dependence complexity increases with the N_w. In the evaluation, we set $\sigma = 0.01$, $T = 2400$, $T_p = 120$, and $\delta = 0.05$.

A.4 Baselines

There are several state-of-the-art baseline methods, we summarize them as follows.

Algorithm 1 Generating MTS data with the NPR model

Require: Total length T of the MTS data, number of variables N, number of constant matrices N_w.
Require: Length of the cycle T_p, stand deviation of the random walk σ, matrix sparsity δ.
Ensure: Synthetic MTS data $\mathbf{X} \in \mathbb{R}^{T \times N}$

```
Generate a random orthogonal matrix P ∈ R^{N×N};
for i = 1 → N_w do
    Σ_i = diag(|N(0,1)|, ..., |N(0,1)|)
    G_i = P^T Σ_i P, α = 0
    G_i[G_i < α] = 0
    while sparsity of G_i > δ do
        α = α + 0.02
        G_i[G_i < α] = 0
    end while
    G_i = Symmetric normalized Laplacian of G_i
end for
X = 0 ∈ R^{T×N}
T_l = T/T_p
for t = 1 → T do
    if t-1% T_p=0 then
        initialize x ∈ R^N randomly from {−1, −0.5, 0.5, 1}
    else
        x = N(G_{(t−1%T_p)/T_l} X[t − 1], σ)
    end if
    X[t] = x
end for
```

A.4.1 Single-step forecasting

- VAR-MLP: A model utilizes both of the multilayer perception (MLP) and auto-regressive model (VAR) [47].
- GP: Modeling time-series data with non-parametric method and Gaussian processes [10, 31].
- RNN-GRU: A model combines the recurrent unit and fully connected GRU.
- LSTNet: A deep neural network, which combines convolutional neural networks and recurrent neural networks [20].
- TPA-LSTM: A model utilizes convolution layers to extract temporal patterns and captures time series correlation with attention mechanism [34].
- MTGNN: MTGNN learns a static graph to represent the inter-series correlation and proposes a novel mix-hop propagation layer to capture the inter-series relation [41].

A.4.2 Multi-step forecasting

- ARIMA: A representative univariate time-series forecasting method based on the Kalman filter [23].
- FC-LSTM: Long Short-Term Memory Network, which is a recurrent neural network with fully connected LSTM hidden units [36].
- LSVR: A linear support vector regression (LSVR) model for travel-time predictions [39].
- STGCN: A spatial-temporal graph convolutional network, which incorporates graph convolutions with 1×1-kernel convolution layers [44].
- DCRNN: A diffusion convolutional recurrent neural network, which utilizes the diffusion graph convolution layer and reccurent unit [22].

Table 5: Results for selecting layers of encoder and decoder L and state embedding dimension D_e. We find that $L = 1, D_e = 64$ is an optimal configuration.

L	1	2	4	1	2	4	1	2	4
D_e	MAE			MAPE(%)			RMSE		
16	2.15/2.76/3.29	2.14/2.73/3.23	2.15/2.75/3.25	5.06/6.86/8.52	5.08/6.88/8.48	5.08/6.90/8.49	4.10/5.55/6.75	4.09/5.51/6.61	4.12/5.57/6.972
32	**2.11**/2.72/3.26	2.12/**2.70**/3.22	2.14/2.73/3.24	4.99/6.79/8.49	5.00/6.73/8.35	5.04/6.80/8.38	**4.07**/5.52/6.68	4.07/5.48/6.67	4.08/5.49/6.63
64	2.12/2.73/**3.22**	2.13/2.73/3.36	2.12/2.71/3.29	5.00/6.74/**8.25**	5.02/6.76/8.40	**4.97/6.73**/8.34	4.08/**5.45/6.56**	4.08/5.49/6.66	4.09/5.52/6.79

- StemGNN: A graph neural network, which combines Graph Fourier Transform (GFT) and Discrete Fourier Transform (DFT) together to capture inter-series correlations and temporal dependencies jointly in the spectral domain [4].
- Graph WaveNet: A spatial-temporal graph convolutional network, which learns a static graph to capture the spatial correlations [42].
- MTGNN: The same model to single-step forecasting, which is trained to predict multiple steps.

Parameter scale. We further list the model size of SOTA methods to illustrate our method is lightweight in the model scale. TPGNN: 0.31M, MTGNN: 0.44M, Graph WaveNet: 0.25M, STGCN:0.33M, StemGNN:1.22M, TPA-LSTM:0.12M, DCRNN:0.37M.

A.5 Metrics

Let $\tilde{\mathbf{Y}}^{(t)}$ and $\mathbf{Y}^{(t)}$ be the predicted and ground truth signal matrix at step t respectively, N is variable number. The evaluation metrics we use in the experiments are computed by:

$$
\begin{aligned}
MAE &= \frac{1}{N}\Sigma_{i=1}^{N}|\tilde{\mathbf{Y}}^{(t)}[i] - \mathbf{Y}^{(t)}[i]| \\
MAPE &= \frac{1}{N}\Sigma_{i=1}^{N}|\frac{\tilde{\mathbf{Y}}^{(t)}[i] - \mathbf{Y}^{(t)}[i]}{\mathbf{Y}^{(t)}[i]}| \times 100\% \\
RMSE &= \sqrt{\frac{1}{N}\Sigma_{i=1}^{N}(\tilde{\mathbf{Y}}^{(t)}[i] - \mathbf{Y}^{(t)}[i])^2} \\
RRSE &= \frac{\sqrt{\frac{1}{N}\Sigma_{i=1}^{N}(\tilde{\mathbf{Y}}^{(t)}[i] - \mathbf{Y}^{(t)}[i])^2}}{\mathrm{std}(\mathbf{Y}^{(t)}) * \sqrt{N/(N-1)}} \\
CORR &= \frac{\Sigma_{i=1}^{N}(\tilde{\mathbf{Y}}^{(t)}[i] - \mathrm{mean}(\tilde{\mathbf{Y}}^{(t)}))(\mathbf{Y}^{(t)}[i] - \mathrm{mean}(\mathbf{Y}^{(t)}))}{N\,\mathrm{std}(\tilde{\mathbf{Y}}^{(t)})\,\mathrm{std}(\mathbf{Y}^{(t)})}
\end{aligned}
\tag{12}
$$

A.6 Experimental Settings

For the datasets with a prior distance matrix D, we follow the pre-processing method of STGCN [44]. It constructs the adjacency matrix W as follows, and they select the hyperparameters as $\sigma^2 = 10, \epsilon = 0.5$:

$$
W_{ij} = \begin{cases} \exp(\frac{-D_{ij}^2}{\sigma^2}), & i \neq j \text{ and } \exp(\frac{-D_{ij}^2}{\sigma^2}) \geq \epsilon; \\ 0, & \text{otherwise} \end{cases}
\tag{13}
$$

We conduct all experiments under the environment of one Intel(R) Xeon(R) Gold 6254 CPU @ 3.10GHz and NVIDIA GeForce RTX 2080Ti GPU. We repeat the experiment 10 times and report the average of evaluation metrics. In the training stage, we adopt L1 loss and Adam optimizer. The initial learning rate is 10^-3 and it decays by the ratio of 0.3 for every 300 epochs. The layer number L is 1, the embedding dimension D_e is 64, the polynomial order K is 2, and the dimension of variable embedding is 10. The training epoch is 1000, 1500 for single-step and multi-step forecasting, respectively. The batch size is 50. We use dropout to improve the model robustness to the noisy data. The rate for dropout is 0.2.

Table 6: Results for selecting the polynomial order of the TPG module K. We find that $K = 2$ is an optimal configuration.

K	1	2	3	4
MAE	2.13/2.75/3.37	**2.12/2.72/3.22**	2.13/2.84/3.29	2.16/2.88/3.44
MAPE(%)	5.10/6.78/8.53	**5.00/6.73/8.22**	5.04/6.77/8.26	5.12/6.98/8.57
RMSE	4.11/5.53/6.79	**4.05/5.45/6.56**	4.06/5.48/6.59	4.19/5.84/6.98

Table 7: Results for selecting the dimension of node (variable) embedding C. We find that $C = 10$ is an optimal configuration.

C	5	10	20
MAE	2.12/2.72/3.27	**2.12/2.72/3.22**	2.14/2.79/3.26
MAPE(%)	5.00/6.75/8.39	**5.00/6.73/8.22**	5.04/6.81/8.35
RMSE	4.07/5.49/6.71	**4.05/5.45/6.56**	4.06/5.47/6.63

A.7 Parameter Study

This section investigates the model configuration on the PEMSD7(M) dataset. The first two hyperparameters are the layer number of encoder and decoder L and the embedding dimension of the matrix signal D_e, we conclude the results in Table 5, which indicates that the configuration $L = 1, D_e = 64$ is a good setting since it has good performance and efficiency.

Another two crucial hyperparameters are the K and C. The first one decides the polynomial order of the TPGNN, the second one controls the embedding dimension of the variable embedding. We show the results in Table 6 and Table 7, which implies the $K = 2, C = 10$ is the best parameter selection.

A.8 More results for ablation study

This section shows more ablation results on the PEMS-D7 of the different horizons to illustrate that our designs are indispensable. In Table 8 and Table 9, we find TPGNN outperforms other variants consistently, which demonstrate the effectiveness of our components. Although w/o normalize achieves better performance on MAPE, it performs badly in the long-term forecasting.

Table 8: Prediction results for ablation study on the PEMS-D7 dataset over the horizon of 3.

Metrics	TPGNN	w/o TPG	w/o dynamic	w/o overview	w/o normalize	w/o K-matrices
MAE	**2.124±0.012**	2.168±0.028	2.169±0.035	2.138±0.029	2.132±0.039	2.143±0.033
MAPE(%)	5.003±0.032	5.017±0.077	5.051±0.026	5.057±0.089	**5.000±0.067**	5.017±0.112
RMSE	**4.049±0.027**	4.127±0.043	4.106±0.078	4.072±0.101	4.074±0.046	4.077±0.056

Table 9: Prediction results for ablation study on the PEMS-D7 dataset over the horizon of 6.

Metrics	TPGNN	w/o TPG	w/o dynamic	w/o overview	w/o normalize	w/o K-matrices
MAE	**2.716±0.021**	2.845±0.023	2.823±0.065	2.748±0.057	2.727±0.044	2.748±0.029
MAPE(%)	6.728±0.052	6.998±0.087	6.842±0.034	6.825±0.082	**6.714±0.055**	6.751±0.119
RMSE	**5.452±0.048**	5.662±0.067	5.594±0.056	5.492±0.071	5.471±0.046	5.511±0.062

A.9 Theoretical Analysis

Lemma 1 *Let $F = \{\mathbf{A}_1, \ldots, \mathbf{A}_k\}, \mathbf{A}_i \in \mathbb{R}^{N\times N}$, be a set of diagbolizable matrices. Then F is a commuting($\mathbf{A}_i\mathbf{A}_j = \mathbf{A}_j\mathbf{A}_i, \forall 1 \leq i, j \leq k$), if and only if there exists and ivertible $\mathbf{S} \in \mathbb{R}^{N\times N}$ such that $\mathbf{S}^{-1}\mathbf{A}_i\mathbf{S}, \mathbf{A}_i \in F$ is diagonal.*

Proof. This is a known result, which indicates that the commutation and simultaneously diagonalization are equivalent properties. We briefly conclude the proving process. If F can be diagonalized by a $\mathbf{S}$, it is trivial to prove that F is commuting due to the diagonal matrices are commutative. We

inductively prove that F can be diagonalized by a $\mathbf{S}$. If $N = 1$, then it's a trivial result. Suppose the theorem holds for matrices of size $N \leq k-1$, we prove that the theorem is true for $N = k$. Let $N = k$, $\mathbf{A} \in F$, and $\mathbf{A}$ is diagnolizable with eigenvalues $\{\lambda_1, \ldots, \lambda_r\}$ where $r \geq 2$ and $\mathbf{AB} = \mathbf{BA}$ for each $\mathbf{B} \in F$. $\mathbf{A}$ is similar to a diagonal matrix, so without loss of generality assume $\mathbf{A}$ is diagonal. Since each $\mathbf{B}$ commutes with the diagonal matrix $\mathbf{A}$, each $\mathbf{B} \in F$ is a block diagnoal matrix. Since $\mathbf{A}$ has at least two distinct entries, each block of each $\mathbf{B}$ has size $k-1$ or less. By the block-wise commutation property, together with the inductive hypothesis, all the blocks in all $\mathbf{B} \in F$ are simultaneously diagonalizable. Thus there exist fixed matrices $T_1, T_2, \ldots, T_r$ such that conjugating with each $\mathbf{B} \in F$ with $T = diag(T_1, T_2, \ldots, T_r)$ gives a block matrix in which the blocks are diagonal, which is a diagonal matrix.

Lemma 2 *Let $\mathbf{G}, \mathbf{A} \in \mathbb{R}^{N \times N}$ are symmetric matrices. Since they are real symmetric matrices, $\mathbf{G}, \mathbf{A}$ can be diagonalized by orthogonal matrices. Let $\mathbf{P}(\mathbf{A}), \mathbf{P}(\mathbf{G})$ are the corresponding orthogonal matrices, we have $\mathbf{P}(\mathbf{A})^T \mathbf{A} \mathbf{P}(\mathbf{A}) = diag(\lambda_1, \ldots, \lambda_N), \mathbf{P}(\mathbf{G})^T \mathbf{G} \mathbf{P}(\mathbf{G}) = diag(\sigma_1, \ldots, \sigma_N)$. Let $\mathbf{G_A} = \mathbf{P}(\mathbf{A})^T \mathbf{G} \mathbf{P}(\mathbf{A})$, then we have the following results:*

$$||\mathbf{G} - \Sigma_{k=0}^{K} a_k \mathbf{A}^k||_F^2 = tr((\mathbf{G_A} - diag(\mathbf{G_A}))^2) + tr((diag(\mathbf{G_A}) - p_K(\mathbf{D}))^2) \tag{14}$$

where $tr(\cdot)$ is the trace of matrix, $(a_0, \ldots, a_K)$ is the polynomial coefficients, $p_K(\mathbf{D}) = \Sigma_{k=0}^{K} a_k \operatorname{diag}(\lambda_1, \ldots, \lambda_N)^k$ is the matrix polynomial of $\mathbf{A}$'s eigenvalues.

Proof. Let $\mathbf{E} = \mathbf{G} - \Sigma_{k=0}^{K} a_k \mathbf{A}^k$, we have the following equations:

$$\begin{aligned}
\mathbf{E} &= \mathbf{P}(\mathbf{A})\mathbf{P}(\mathbf{A})^{\mathbf{T}}\mathbf{G}\mathbf{P}(\mathbf{A})\mathbf{P}(\mathbf{A})^{\mathbf{T}} - \mathbf{P}(\mathbf{A})(\Sigma_{k=0}^{K} a_k \operatorname{diag}(\lambda_1, \ldots, \lambda_N)^k)\mathbf{P}(\mathbf{A})^{\mathbf{T}} \\
&= \mathbf{P}(\mathbf{A})(\mathbf{G_A} - \Sigma_{k=0}^{K} a_k \operatorname{diag}(\lambda_1, \ldots, \lambda_N)^k))\mathbf{P}(\mathbf{A})^{\mathbf{T}} \\
&= \mathbf{P}(\mathbf{A})(\mathbf{G_A} - p_K(\mathbf{D}))\mathbf{P}(\mathbf{A})^{\mathbf{T}}
\end{aligned} \tag{15}$$

Moreover, $||\mathbf{G} - \Sigma_{k=0}^{K} a_k \mathbf{A}^k||_F^2 = tr(\mathbf{E}^2)$ ($\mathbf{E}$ is symmetric). Therefore, we have:

$$\begin{aligned}
||\mathbf{G} - \Sigma_{k=0}^{K} a_k \mathbf{A}^k||_F^2 &= tr((\mathbf{G_A} - \operatorname{diag}(\mathbf{G_A}) + \operatorname{diag}(\mathbf{G_A}) - p_K(\mathbf{D}))^2) \\
&= tr((\mathbf{G_A} - \operatorname{diag}(\mathbf{G_A}))^2) + 2tr((\mathbf{G_A} - \operatorname{diag}(\mathbf{G_A}))(\operatorname{diag}(\mathbf{G_A}) - p_K(\mathbf{D}))) \\
&+ tr((\operatorname{diag}(\mathbf{G_A}) - p_K(\mathbf{D}))^2) \\
&= tr((\mathbf{G_A} - diag(\mathbf{G_A}))^2) + tr((diag(\mathbf{G_A}) - p_K(\mathbf{D}))^2)
\end{aligned} \tag{16}$$

Review. Lemma 2 clearly illustrates the approximation ability of a matrix polynomial. The error is controlled by two terms. The value of the first term depends on whether $\mathbf{P}(\mathbf{A})$ is able to diagonalize $\mathbf{G}$. As for the second term, it corresponds to a least square regression problem.

Theorem 2 *Given the setting same as Lemma 2. Let $\mathbf{Q} = \mathbf{P}(\mathbf{A})^T \mathbf{G}$, we define a discrepancy matrix $\mathbf{D}_{AG}, (\mathbf{D}_{AG})_{ij} = \mathbf{Q}_{ij}^2$. Let $\mathbf{d}_k$ is the k-th row vector of $\mathbf{D}_{AG}$, and $\lambda_{min}, \lambda_{max}$ be the minimum and maximum eigenvalues of matrix $\Sigma_{k=1}^{N} \mathbf{d}_k \mathbf{d}_k^T$. We then have the following estimation of the frist term of Equation 16:*

$$(1 - \lambda_{max}) tr(\mathbf{G}^2) \leq tr((\mathbf{G_A} - diag(\mathbf{G_A}))^2) \leq (1 - \lambda_{min}) tr(\mathbf{G}^2) \tag{17}$$

Proof. We first expand the first term of Equation 16.

$$\begin{aligned}
tr((\mathbf{G_A} - diag(\mathbf{G_A}))^2) &= tr(\mathbf{G}_A^2 + diag(\mathbf{G_A})^2 - 2diag(\mathbf{G_A})\mathbf{G_A}) \\
&= tr(\mathbf{G}^2) + tr(diag(\mathbf{G_A})^2 - 2diag(\mathbf{G_A})\mathbf{G_A})
\end{aligned} \tag{18}$$

Besides, we observe that:

$$(diag(\mathbf{G_A})\mathbf{G_A})_{ij} = (\mathbf{G_A})_{ii}(\mathbf{G_A})_{ij} \tag{19}$$

Therefore,

$$\begin{aligned}
tr(diag(\mathbf{G_A})\mathbf{G_A}) &= \Sigma_i (diag(\mathbf{G_A})\mathbf{G_A})_{ii} \\
&= \Sigma_i (\mathbf{G_A})_{ii}^2 \\
&= tr(diag(\mathbf{G_A})^2)
\end{aligned} \tag{20}$$

Combining the result with Equation 18, we derive the following result:

$$\begin{aligned} tr((\mathbf{G_A} - diag(\mathbf{G_A}))^2) &= tr(\mathbf{G}^2) + tr(diag(\mathbf{G_A})^2 - 2diag(\mathbf{G_A})\mathbf{G_A}) \\ &= tr(\mathbf{G}^2) + tr(diag(\mathbf{G_A})^2) - 2tr(diag(\mathbf{G_A})^2) \\ &= tr(\mathbf{G}^2) - tr(diag(\mathbf{G_A})^2) \end{aligned} \tag{21}$$

Since $\mathbf{G_A} = \mathbf{P}(\mathbf{A})^T\mathbf{G}\mathbf{P}(\mathbf{A})$. Let $\mathbf{u}_1, \ldots, \mathbf{u}_N$ are the column vectors of $\mathbf{P}(\mathbf{A})$, we have $(\mathbf{G_A})_{ij} = \mathbf{u}_i^T\mathbf{G}\mathbf{u}_j$. Therefore, we derive the following equation:

$$\begin{aligned} tr(diag(\mathbf{G_A})^2) &= \Sigma_{k=1}^N (\mathbf{G_A})_{kk}^2 \\ &= \Sigma_{k=1}^N (\mathbf{u}_k^T\mathbf{G}\mathbf{u}_k)^2 \end{aligned} \tag{22}$$

Let $\mathbf{Q} = \mathbf{P}(\mathbf{A})^T\mathbf{P}(\mathbf{G})$, $\mathbf{v}_1, \ldots, \mathbf{v}_N$ are the column vectors of $\mathbf{P}(\mathbf{G})$. We then have $\mathbf{P}(\mathbf{A}) = \mathbf{P}(\mathbf{G})\mathbf{Q}^T$, and $\mathbf{u}_k = \Sigma_{s=1}^m Q_{sk}\mathbf{v}_s$. Therefore, we derive the following result:

$$\begin{aligned} tr(diag(\mathbf{G_A})^2) &= \Sigma_{k=1}^m (\mathbf{u}_k^T\mathbf{G}\mathbf{u}_k)^2 \\ &= \Sigma_{k=1}^m ((\Sigma_{s=1}^m Q_{sk}\mathbf{v}_s^T)\mathbf{G}(\Sigma_{s=1}^m Q_{sk}\mathbf{v}_s))^2 \end{aligned} \tag{23}$$

Because $\mathbf{v}_s^T\mathbf{G}\mathbf{v}_s = \sigma_s, \mathbf{v}_s^T\mathbf{G}\mathbf{v}_t = 0, if\ s \neq t$, we have $tr(diag(\mathbf{G_A})^2) = \Sigma_{k=1}^m (\Sigma_{s=1}^m \sigma_s Q_{sk}^2)^2$. We then define a discrepancy $\mathbf{D}_{AG}$, $(\mathbf{D}_{AG})_{ij} = (Q)_{ij}^2$. Let $\mathbf{d}_k$ be the k-th column vector of $\mathbf{D}_{AG}$, $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_N)^T$ is a vector composed of $\mathbf{G}$'s eigenvalues. Therefore, we have the following equation:

$$\begin{aligned} tr(diag(\mathbf{G_A})^2) &= \Sigma_{k=1}^m (\sigma^T\mathbf{d}_k)^2 \\ &= \Sigma_{k=1}^m \sigma^T\mathbf{d}_k\mathbf{d}_k^T\sigma \\ &= \sigma^T(\Sigma_{k=1}^N \mathbf{d}_k\mathbf{d}_k^T)\sigma \end{aligned} \tag{24}$$

Due to $\Sigma_{k=1}^N \mathbf{d}_k\mathbf{d}_k^T$ is a symmetric matrix, according the property of Rayleigh quotient, we know:

$$\lambda_{min} \leq \frac{\sigma^T(\Sigma_{k=1}^m \mathbf{d}_k\mathbf{d}_k^T)\sigma}{\sigma^T\sigma} \leq \lambda_{max} \tag{25}$$

Where $\lambda_{min}, \lambda_{max}$ are the minimum and maximum eigenvalues of $\Sigma_{k=1}^N \mathbf{d}_k\mathbf{d}_k^T$. Besides, according to the property of matrix trace, $tr(\mathbf{G}^2) = tr(\mathbf{P}(\mathbf{G})diag(\sigma_1, \ldots, \sigma_N)^2\mathbf{P}(\mathbf{G})^T) = \sigma^T\sigma$. Therefore, we have the following result:

$$\lambda_{min}tr(\mathbf{G}^2) \leq tr(diag(\mathbf{G_A})^2) \leq \lambda_{max}tr(\mathbf{G}^2) \tag{26}$$

According to Equation 21 and Equation 26, we finish the proof of Theorem 2.

Lemma 3 *Given the setting same as Lemma 2, we have the following estimation of the second term of Equation 16:*

$$0 \leq tr((diag(\mathbf{G_A}) - p_K(\mathbf{D}))^2) \leq tr(diag(\mathbf{G_A})^2) - \frac{1}{N}(tr(\mathbf{G_A}))^2 - \frac{tr(\mathbf{D}\mathbf{G_A}) - \frac{1}{N}tr(\mathbf{D})tr(\mathbf{G_A})}{tr(\mathbf{D}^2) - \frac{1}{N}tr(\mathbf{D})^2} \tag{27}$$

Specifically, if $\mathbf{D}$ *has* N *different values and* $K = N - 1$*, then the upper bound is zero.*

Proof. Since $tr((diag(\mathbf{G_A}) - p_K(\mathbf{D}))^2) = \Sigma_{i=1}^N (\mathbf{G_A}_{ii} - \Sigma_{k=0}^K a_k\lambda_i^k)^2$, it corresponds to a least square problem. Equation 3 is a known result of least square optimization under $K = 1$. Due to $K \geq 1$, Equation 3 holds obviously. Besides, if $\mathbf{D}$ has N different values, i.e., $\mathbf{A}$ has N different eigenvalues, and $K = N - 1$. Then the upper bound of $tr((diag(\mathbf{G_A}) - p_K(\mathbf{D}))^2)$ is zero with Lagrange interpolation.

Theorem 3 ***(Formal statement of Theorem 1).****Let* $G = \{\mathbf{G}^{(1)}, \ldots, \mathbf{G}^{(T)}\}, \mathbf{G}^{(t)} \in \mathbb{R}^{N\times N}$ *be the symmetric normalized Laplacian of the optimal structures for time step* 1 *to* T*,*$\mathbf{A} \in \mathbb{R}^{N\times N}$ *is the initial adjacency of TPGNN. Suppose* G *is commuting,* $\mathbf{A}$ *and elements of* G *are symmetric. Then, there is a* $\mathbf{P}(\mathbf{G}) \in \mathbb{R}^{N\times N}$ *such that* $\mathbf{P}(\mathbf{G})^T\mathbf{G}^{(t)}\mathbf{P}(\mathbf{G})$ *is diagonal matrix for each* t*. Let*

$\mathbf{P}(\mathbf{A})^T\mathbf{A}\mathbf{P}(\mathbf{A})$ *be a diagonal matrix, we define* $\mathbf{D}_{AG}$ *the same as Theorem 2, and* $\lambda_{min}, \lambda_{max}$ *are the minimum and maximum eigenvalues of* $\Sigma_{k=1}^{N}\mathbf{d}_k\mathbf{d}_k^T$. *If* $\mathbf{A}$ *has* N *different singular values, the polynomial's order* $K = N-1$. *Then the approximation error* $e^{(1:T)} = \frac{1}{T}\Sigma_{t=1}^{T}||\mathbf{W}^{(t)} - \mathbf{G}^{(t)}||_F^2$ *satisfies the following estimation:*

$$(1-\lambda_{max})\mathbb{E}_t||\mathbf{G}^{(t)}||_F^2 \leq \mathrm{e}^{(1:T)} \leq (1-\lambda_{min})\mathbb{E}_t||\mathbf{G}^{(t)}||_F^2, \tag{28}$$

where $\mathbb{E}_t||\mathbf{G}^{(t)}||_F^2 = \frac{1}{T}\Sigma_{t=1}^{T}||\mathbf{G}^{(t)}||_F^2$ *is the average norm of the Laplacians.*

Proof. Firstly, G is diagonalizable since its elements are symmetric. According to Lemma 1, we can find a $\mathbf{S} \in \mathbb{R}^{N\times N}$ such that $\mathbf{S}^{-1}\mathbf{G}^{(t)}\mathbf{S}$ is diagonal for each t. Therefore, we can find an othogonal matrix $\mathbf{P}(\mathbf{G})$ such that $\mathbf{P}(\mathbf{G})^T\mathbf{G}^{(t)}\mathbf{P}(\mathbf{G})$ is diagonal using Gram-Schmidt orthogonalization. We now consider each $||\mathbf{W}^{(t)} - \mathbf{G}^{(t)}||_F^2$. According to Theorem 2 and Lemma 3, we have the following estimation:

$$(1-\lambda_{max})||\mathbf{G}^{(t)}||_F^2 \leq ||\mathbf{W}^{(t)} - \mathbf{G}^{(t)}||_F^2 \leq (1-\lambda_{min})||\mathbf{G}^{(t)}||_F^2 \tag{29}$$

We thus prove the Theorem 3 with the Equation 29.