SpecSTG: A Fast Spectral Diffusion Framework for Probabilistic Spatio-Temporal Traffic Forecasting

Lequan Lin[1*†] , **Dai Shi**[1†] , **Andi Han**[2] and **Junbin Gao**[1]
[1]**The University of Sydney**
[2]**Riken AIP**
{lequan.lin, dai.shi, junbin.gao}@sydney.edu.au, andi.han@riken.jp

Abstract

Traffic forecasting, a crucial application of spatio-temporal graph (STG) learning, has traditionally relied on deterministic models for accurate point estimations. Yet, these models fall short of quantifying future uncertainties. Recently, many probabilistic methods, especially variants of diffusion models, have been proposed to fill this gap. However, existing diffusion methods typically deal with individual sensors separately when generating future time series, resulting in limited usage of spatial information in the probabilistic learning process. In this work, we propose SpecSTG, a novel spectral diffusion framework, to better leverage spatial dependencies and systematic patterns inherent in traffic data. More specifically, our method generates the Fourier representation of future time series, transforming the learning process into the spectral domain enriched with spatial information. Additionally, our approach incorporates a fast spectral graph convolution designed for Fourier input, alleviating the computational burden associated with existing models. Compared with state-of-the-arts, SpecSTG achieves up to 8% improvements on point estimations and up to 0.78% improvements on quantifying future uncertainties. Furthermore, SpecSTG's training and validation speed is 3.33× of the most efficient existing diffusion method for STG forecasting. The source code for SpecSTG is available at https://anonymous.4open.science/r/SpecSTG.

1 INTRODUCTION

Traffic forecasting on road networks is an essential application domain of spatio-temporal graph (STG) learning [Yuan and Li(2021); Jin et al.(2023b); Jin et al.(2023a)]. As a crucial component of the Intelligence Transportation System (ITS), accurate and informative prediction of future traffic dynamics provides essential guidelines to traffic authorities in decision-making [Lana et al.(2018); Boukerche et al.(2020)].

[*]The corresponding author.
[†]Equal contributions.

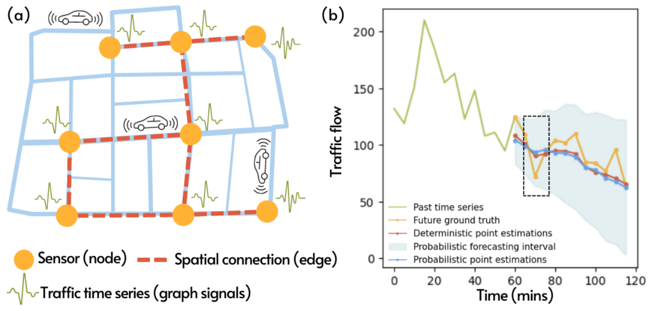

Figure 1: Illustrations: (a) an example of traffic STG; (b) traffic flow forecasting in future 60 minutes with GMAN (deterministic) and SpecSTG (probabilistic) on PEMS04.

Since traffic data, such as vehicle speed and traffic flow, are collected from sensors in the continuous space of road networks, they present strong spatio-temporal dependencies, especially at neighbouring locations and time windows [Guo et al.(2019)]. This naturally leads to their representation as STGs: the traffic network is modelled as a graph, in which nodes are sensors and edges are decided with some criteria such as geographic distances. Hence, temporal records are stored as graph signals, while spatial information is encapsulated in the graph structure. In **Figure 1 (a)**, we provide a visualized example of traffic STG.

STG traffic forecasting aims at predicting future values at all sensors based on past time series and spatial connections in the traffic network. This task has traditionally relied on deterministic models such as DCRNN [Li et al.(2018)] and GMAN [Zheng et al.(2020)] to produce accurate point estimations. Nevertheless, these models may fall short in identifying unexpected variations that lead to consequential change in traffic regulations [Pal et al.(2021); Wen et al.(2023); Hu et al.(2023)]. This limitation can be overcome by probabilistic methods, which alternatively approximate the distribution of future time series, thus leading to more uncertainty-aware predictions. For example, **Figure 1 (b)** shows the results of deterministic and probabilistic models on the PEMS04 traffic flow forecasting task [Guo et al.(2019)]. The deterministic model can only provide point estimations (red), while the probabilistic model is capable of generating both point estimations and the forecasting interval (blue) which captures some abrupt fluctuations in traffic flow (black box).

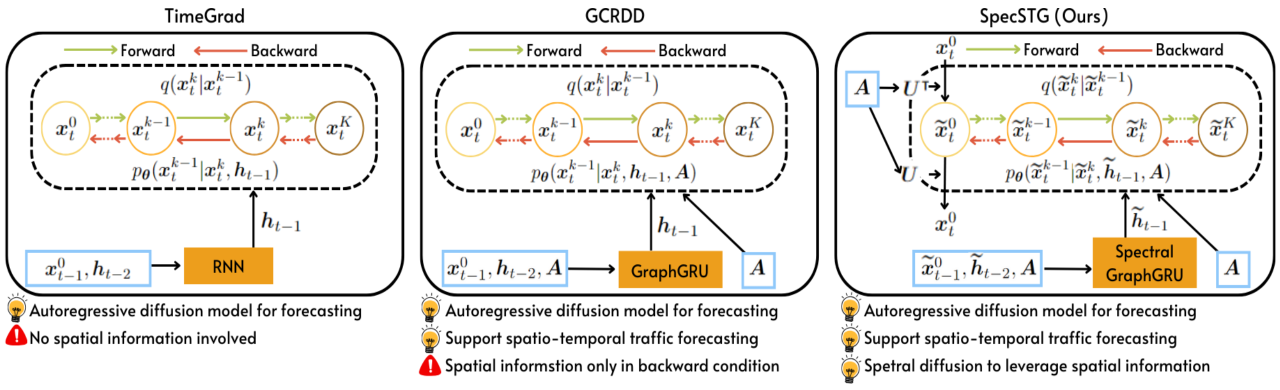
TimeGrad
Forward
Backward
$q(x_t^k|x_t^{k-1})$
x_t^0
x_t^{k-1}
x_t^k
x_t^K
$p_\theta(x_t^{k-1}|x_t^k, h_{t-1})$
h_{t-1}
x_{t-1}^0, h_{t-2}
RNN
Autoregressive diffusion model for forecasting
No spatial information involved
GCRDD
$p_\theta(x_t^{k-1}|x_t^k, h_{t-1}, A)$
x_{t-1}^0, h_{t-2}, A
GraphGRU
A
Support spatio-temporal traffic forecasting
Spatial informstion only in backward condition
SpecSTG (Ours)
$U^\top$
U
$q(\tilde{x}_t^k|\tilde{x}_t^{k-1})$
$p_\theta(\tilde{x}_t^{k-1}|\tilde{x}_t^k, \tilde{h}_{t-1}, A)$
$\tilde{h}_{t-1}$
$\tilde{x}_{t-1}^0, \tilde{h}_{t-2}, A$
Spectral GraphGRU
Spetral diffusion to leverage spatial information

Figure 2: The overview of SpecSTG. Illustrations of TimeGrad and GCRDD are provided to show the novelty and advantage of our approach.

Among all the probabilistic models applicable to STG traffic forecasting, we specifically focus on diffusion models [Yang et al.(2023); Ho et al.(2020); Lin et al.(2023)]. Classic diffusion models for time series forecasting, such as TimeGrad [Rasul et al.(2021a)], generally follow a forward-backward training process: the generative target (i.e., future time series) is first turned into white noise and then recovered by a learnable backward kernel. The backward kernel is a conditional distribution which is similar to traditional diffusion models except for the inclusion of encoded past temporal information in its conditions. For prediction, samples from the distribution of future time series are generated by denoising white noise with the learned backward kernel. Since such methods are incapable of handling spatial information in STGs, diffusion models for STG forecasting, such as DiffSTG [Wen et al.(2023)] and GCRDD [Li et al.(2023)], incorporate graph structure in the backward kernel condition. More specifically, the backward kernel is usually approximated by a denoising network, which takes graph structure and other conditions as input to predict noises injected in the forward process.

Our work considers two limitations of existing diffusion methods for STG forecasting. (**Limitation #1**) Although these methods emphasize on the importance of spatial information in STG forecasting, they only use spatial information in the backward kernel condition. Consequently, the involvement of spatial information in the overall forward-backward diffusion learning process is limited. (**Limitation #2**) The denoising network of existing methods usually relies substantially on graph convolution to encode spatial information. This often introduces a complexity of $\mathcal{O}(N^2)$, quadratic in the number of sensors N in a traffic STG (see Subsection 4.1 for more details). Thus the computational cost is high especially for large traffic networks, leading to slow training and sampling.

In this work, we propose a novel spectral diffusion framework (SpecSTG), which adopts the graph Fourier representation of time series as the generative target. According to the spectral graph theory, the graph Fourier representation is a measurement of variations in graph signals guided by the graph structure [Chung(1997); Kreuzer et al.(2021)]. In the context of STGs, we may treat time points as features, thereby the graph Fourier representation can be considered as a new time series of systematic fluctuations enriched by spatial information. Hence, **Limitation #1** is resolved by generating the Fourier representation rather than the original time series, transforming the entire diffusion process into the spectral domain. This effectively leverages the graph structure to construct a more comprehensive diffusion base with additional systematic and spatial patterns. Besides, with no loss of information, the generated data can be converted back to the original domain via the inverse Fourier transform for prediction. **Limitation #2** is mitigated by replacing the graph convolution with a light-complexity alternative, which only works for the Fourier input (details in Subsection 4.1). An overview of SpecSTG can be found in **Figure 2**. We also provide illustrations of TimeGrad and GCRDD for comparison. The contributions of this paper are three-fold:

(1) To our best knowledge, this is the first work that explores probabilistic STG forecasting on the graph spectral domain;

(2) SpecSTG achieves up to 8% improvements on point estimations and up to 0.78% improvements on generating compatible forecasting intervals.

(3) SpecSTG's training and validation speed is 3.33× of the most efficient existing diffusion method for STG forecasting. Additionally, SpecSTG significantly accelerates the sampling process, particularly for large sample sizes.

2 PRELIMINARIES

2.1 Spatio-Temporal Graphs

STGs can be considered as a multidimensional graph representation of entities in a system with time series as graph signals. In traffic forecasting, we model sensors as nodes and then create edges based on some spatial relationships such as geographic distances. The average traffic records in observation periods are modelled as graph signals. For a traffic network with N sensors, the corresponding STG can be denoted as $\mathcal{G}\{\mathcal{V}, \mathcal{E}, \boldsymbol{A}\}$, where $\mathcal{V}$ is the set of nodes/sensors, $\mathcal{E}$ is the set of edges, and $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ is the adjacency matrix. $\boldsymbol{A}$ is assumed to be undirected and can be either weighted or unweighted. The graph signals are denoted as $\boldsymbol{X}_\mathcal{G} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_t, ...|\boldsymbol{x}_t \in \mathbb{R}^{N\times D_x}\}$, where D_x is the number of variables. In traffic forecasting, it is common that only one variable such as speed or flow is of interest [Li et al.(2018); Guo et al.(2019)], thus we have $D_x = 1$ such that $\boldsymbol{x}_t \in \mathbb{R}^N$.

2.2 Spatio-Temporal Graph Forecasting

The objective of STG traffic forecasting is to predict a future time series window $\boldsymbol{X}_f = \{\boldsymbol{x}_{t_0+1}, \boldsymbol{x}_{t_0+2}, ..., \boldsymbol{x}_{t_0+f}\}$ given the past context window $\boldsymbol{X}_c = \{\boldsymbol{x}_{t_0-c+1}, \boldsymbol{x}_{t_0-c+2}, ..., \boldsymbol{x}_{t_0}\}$, where f and c are the length of future and past windows. We denote the combination of past and future time series as $\boldsymbol{X} = \{\boldsymbol{x}_{t_0-c+1}, \boldsymbol{x}_{t_0-c+2}, ..., \boldsymbol{x}_{t_0+f}\}$. Normally, the target distribution of generative models depends on the sampling methods: one-shot methods produce all future predictions together from the distribution $q(\boldsymbol{X}_f|\boldsymbol{X}_c, \boldsymbol{A})$ [Wen et al.(2023); Liu et al.(2023)], while autoregressive methods generate samples from $q(\boldsymbol{x}_t|\boldsymbol{X}_{t_0-c+1:t-1}, \boldsymbol{A})$ for $t = t_0+1, t_0+2, ..., t_0+f$ successively, where $\boldsymbol{X}_{t_0-c+1:t-1} = \{\boldsymbol{x}_{t_0-c+1}, ..., \boldsymbol{x}_{t-1}\}$ [Li et al.(2023)]. Autoregressive methods often capture the sequential information in consecutive time points more closely. However, they are associated with higher time costs because of the non-parallel step-by-step sampling process. Our method, SpecSTG, is formulated within the autoregressive framework but is equipped with a specially designed spectral graph convolution to mitigate computational inefficiency.

2.3 Denoising Diffusion Probabilistic Model

Denoising diffusion probabilistic models (DDPMs) learn how to generate samples from the target distribution via a pair of forward-backward Markov chains [Ho et al.(2020); Yang et al.(2023)]. Assuming that $\boldsymbol{x}^0 \sim q(\boldsymbol{x}^0)$ is the original data, for diffusion step $k = 0, 1, ..., K$, the forward chain injects Gaussian noises to $\boldsymbol{x}^k$ until $q(\boldsymbol{x}^K) := \int q(\boldsymbol{x}^K|\boldsymbol{x}^0)q(\boldsymbol{x}^0)\mathrm{d}\boldsymbol{x}^0 \approx \mathcal{N}(\boldsymbol{x}^K; \boldsymbol{0}, \boldsymbol{I})$. As a special property, given a noise schedule $\beta = \{\beta_1, \beta_2, ..., \beta_K\}$, we may directly compute the disturbed data at step k as $\boldsymbol{x}^k = \sqrt{\tilde{\alpha}_k}\boldsymbol{x}^0 + \sqrt{1-\tilde{\alpha}_k}\boldsymbol{\epsilon}$, where $\tilde{\alpha}_k = \prod_{i=1}^k(1-\beta_i)$ and $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. Next, the backward chain denoises from $\boldsymbol{x}^K$ to recover $p_{\boldsymbol{\theta}}(\boldsymbol{x}^0)$ through a probabilistic backward kernel $p_{\boldsymbol{\theta}}(\boldsymbol{x}^{k-1}|\boldsymbol{x}^k)$, where $\boldsymbol{\theta}$ denotes all learnable parameters. In practice, the backward kernel is usually optimized with a denoising network $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ by minimizing the loss function

$$\mathcal{L}_{DDPM}(\boldsymbol{\theta}) = \mathbb{E}_{k,\boldsymbol{x}^0,\boldsymbol{\epsilon}} \left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}\left(\boldsymbol{x}^k, k\right)\right\|^2, \tag{1}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ represents the noises injected in the forward diffusion steps. Please refer to **Appendix A** for more details on DDPMs.

3 SPATIO-TEMPORAL GRAPH FOURIER TRANSFORM

Given a traffic STG $\mathcal{G}\{\mathcal{V}, \mathcal{E}, \boldsymbol{A}\}$ with N sensors, the normalized graph Laplacian is computed as $\boldsymbol{L} = \boldsymbol{I}_N - \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}$, where $\boldsymbol{I}_N \in \mathbb{R}^{N\times N}$ is the identity matrix, and $\boldsymbol{D} \in \mathbb{R}^{N\times N}$ is the diagonal degree matrix. We denote the eigendecomposition of the graph Laplacian as $\boldsymbol{L} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\intercal$, where $\boldsymbol{U} \in \mathbb{R}^{N\times N}$ and $\boldsymbol{\Lambda} \in \mathbb{R}^{N\times N}$ are the corresponding eigenvector and eigenvalue matrices, respectively. Considering the univariate graph signal $\boldsymbol{X}_{\mathcal{G}} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_t, ...|\boldsymbol{x}_t \in \mathbb{R}^N\}$, the Fourier transform for each time point t is given by $\tilde{\boldsymbol{x}}_t = \boldsymbol{U}^\intercal\boldsymbol{x}_t$, known as the Fourier representation of $\boldsymbol{x}_t$ in the spectral domain. The Fourier reconstruction is $\boldsymbol{x}_t = \boldsymbol{U}\tilde{\boldsymbol{x}}_t$. The orthonormal $\boldsymbol{U}$ ensures a lossless reconstruction for the temporal information. We may compute in matrix form for all time points as $\tilde{\boldsymbol{X}} = \boldsymbol{U}^\intercal\boldsymbol{X}$ and $\boldsymbol{X} = \boldsymbol{U}\tilde{\boldsymbol{X}}$. In **Appendix B**, we also discuss how to naturally extend the method to multivariate traffic STGs with $\boldsymbol{x}_t \in \mathbb{R}^{N\times D_x}$, $D_x \geq 2$.

The graph Fourier transform can be understood as a projection of graph signals onto the spectral domain spanned by the eigenvector basis of graph Laplacian. The operator $\boldsymbol{U}$ brings rich positional information for graph signals and offers a platform to investigate the variations among signals through a global perspective on the graph. In particular, when the input is a traffic STG, the Fourier representation measures how graph signals (time series values) fluctuate across the network. This effectively integrates spatial connectivity into time series, leading to a spatial-aware forecasting paradigm.

4 THE PROPOSED METHOD

SpecSTG assumes that the Fourier representation of future time series follows the distribution

$$q(\tilde{\boldsymbol{X}}_f^0|\tilde{\boldsymbol{X}}_c^0, \boldsymbol{A}) \approx \prod_{t=t_0+1}^{t_0+f} p_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}_t^0|\tilde{\boldsymbol{h}}_{t-1}, \boldsymbol{A}), \tag{2}$$

where $\tilde{\boldsymbol{X}}_c^0$ and $\tilde{\boldsymbol{X}}_f^0$ are the noise-free Fourier representations of past and future time series, respectively. $\tilde{\boldsymbol{h}}_{t-1}$ represents past spatio-temporal condition encoded by a spectral recurrent encoder. For each time point $t = t_0+1, ..., t_0+f$, the diffusion process will learn the corresponding backward kernel $p_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}_t^{k-1}|\tilde{\boldsymbol{x}}_t^k, \tilde{\boldsymbol{h}}_{t-1}, \boldsymbol{A})$ with $k = 1, ..., K$. The objective function of SpecSTG is given by

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{t,k,\tilde{\boldsymbol{x}}_t^0,\boldsymbol{\epsilon}_t} \left\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}\left(\tilde{\boldsymbol{x}}_t^k, k, \tilde{\boldsymbol{h}}_{t-1}, \boldsymbol{A}\right)\right\|^2, \tag{3}$$

where t denotes future time points, k denotes diffusion steps, $\tilde{\boldsymbol{x}}_t^k$ is the disturbed data at time point t and step k, and $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. It is worth noting that t may also start with t_0-c+1 instead of t_0+1 to facilitate the learning of autoregressive dependencies by considering both past and future instances. Lastly, we employ a graph-modified WaveNet architecture [van den Oord et al.(2016)] for the denoising network $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\cdot)$, which is specially designed for Fourier input. In the rest of this section, we will focus on details of SpecSTG components, training, and inference. The visualization of SpecSTG architecture can be found in **Figure 2**.

4.1 Light-Complexity Spectral Graph Convolution

Diffusion models for STG forecasting usually rely on graph convolution to encode spatial information in the condition of backward kernel [Li et al.(2023); Wen et al.(2023); Liu et al.(2023)]. Straightforwardly, we adopt the spectral convolution, which typically transforms graph signals to the spectral domain and then processes the data via frequency filtering before they are converted back to the original domain [Defferrard et al.(2016); Kipf and Welling(2016)]. However, with

SpecSTG, the Fourier representation is already formed as input, so the transform is no longer required in the convolution. In addition, to ensure that the model pipeline flows in the spectral domain throughout the diffusion learning process, we do not apply reconstruction either.

We choose the Chebyshev convolution [Defferrard et al.(2016)], whose filters are formed with Chebyshev polynomials to accelerate computation as $\mathrm{ChebConv}(\boldsymbol{X}) = \boldsymbol{U}\sum_{j=0}^{J-1}\phi_j \mathcal{T}_j(\tilde{\boldsymbol{\Lambda}})\boldsymbol{U}^\intercal \boldsymbol{X}$, for polynomial degrees up to $J-1$, where $\mathcal{T}_j(\tilde{\boldsymbol{\Lambda}}) \in \mathbb{R}^{N\times N}$ is the j-th order Chebyshev polynomial evaluated at $\tilde{\boldsymbol{\Lambda}} = 2\boldsymbol{\Lambda}/\lambda_{max} - \boldsymbol{I}_N$ with ϕ_j being a learnable coefficient. With Fourier representation $\tilde{\boldsymbol{X}} = \boldsymbol{U}^\intercal \boldsymbol{X}$ as input, we define the modified spectral graph convolution as

$$\mathrm{SpecConv}(\tilde{\boldsymbol{X}}) := \sum_{j=0}^{J-1}\phi_j \mathcal{T}_j(\tilde{\boldsymbol{\Lambda}})\tilde{\boldsymbol{X}}. \tag{4}$$

Remark 1 (Complexity Analysis). Our spectral convolution has only $\mathcal{O}(N)$ complexity, contrasting to the $\mathcal{O}(N^2)$ complexity of the classic Chebyshev convolution. This significant improvement endows SpecSTG with the advantage of time efficiency over methods using Chebyshev convolution, such as GCRDD. Other methods such as DiffSTG may use the graph convolution in [Kipf and Welling(2016)] as $\mathrm{GCNConv}(\boldsymbol{X}) = \boldsymbol{A}\boldsymbol{X}$, whose complexity is $\mathcal{O}(|\mathcal{E}|)$ where $|\mathcal{E}|$ is the number of edges. As we see in Section 5.1, $|\mathcal{E}|$ is often much larger than N in traffic STGs.

4.2 Spectral Recurrent Encoder

We design SG-GRU as the spectral version of Graph GRU [Seo et al.(2018)] to encode past time series and spatial information in graph Fourier domain as follows. For $t = t_0 - c + 1, ..., t_0$:

$$\boldsymbol{z} = \sigma(\mathrm{SpecConv}(\tilde{\boldsymbol{x}}_t^0)\boldsymbol{W}_{z_1} + \mathrm{SpecConv}(\tilde{\boldsymbol{h}}_{t-1})\boldsymbol{W}_{z_2}) \tag{5}$$

$$\boldsymbol{r} = \sigma(\mathrm{SpecConv}(\tilde{\boldsymbol{x}}_t^0)\boldsymbol{W}_{r_1} + \mathrm{SpecConv}(\tilde{\boldsymbol{h}}_{t-1})\boldsymbol{W}_{r_2}) \tag{6}$$

$$\boldsymbol{\zeta} = \tanh(\mathrm{SpecConv}(\tilde{\boldsymbol{x}}_t^0)\boldsymbol{W}_{\zeta_1} + \mathrm{SpecConv}(\boldsymbol{r}\odot\tilde{\boldsymbol{h}}_t)\boldsymbol{W}_{\zeta_2}) \tag{7}$$

$$\tilde{\boldsymbol{h}}_{t+1} = \boldsymbol{z}\odot\tilde{\boldsymbol{h}}_t + (1-\boldsymbol{z})\odot\boldsymbol{\zeta}, \tag{8}$$

where $\tilde{\boldsymbol{h}}_0 \in \mathbb{R}^{D_h}$ is a vector of zeors with D_h being the hidden size, $\boldsymbol{W}_{z_1}, \boldsymbol{W}_{r_1}, \boldsymbol{W}_{\zeta_1} \in \mathbb{R}^{1\times D_h}$, $\boldsymbol{W}_{z_2}, \boldsymbol{W}_{r_2}, \boldsymbol{W}_{\zeta_2} \in \mathbb{R}^{D_h\times D_h}$ are learnable weights included in $\boldsymbol{\theta}$, and σ is the sigmoid activation function. $\boldsymbol{z}$ and $\boldsymbol{r}$ are known as update gate and reset gate. $\boldsymbol{\zeta}$ is the candidate state storing current information to be updated in the hidden state. In the implementation, we may also input time features $\boldsymbol{\Gamma} = \{\gamma_{t_0-c+1}, \gamma_{t_0-c+2}, ..., \gamma_{t_0+f}\}$ such as day of week and week of month by concatenating them with $\tilde{\boldsymbol{X}}$.

4.3 Denoising Network: Spectral Graph WaveNet

In reminiscent of TimeGrad, we design <u>S</u>pectral <u>G</u>raph <u>**Wave**</u>Net (SG-Wave) as the $\boldsymbol{\epsilon_\theta}$ of SpecSTG (see details in **Figure 3**). Besides, we replace some Conv1d layers with fully connected linear layers for efficient training and sampling. The network takes disturbed data $\tilde{\boldsymbol{x}}_t^k$, diffusion step k, hidden condition $\tilde{\boldsymbol{h}}_{t-1}$, and graph adjacency $\boldsymbol{A}$ as inputs, and aims at predicting the noise $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ at step k for time point t.

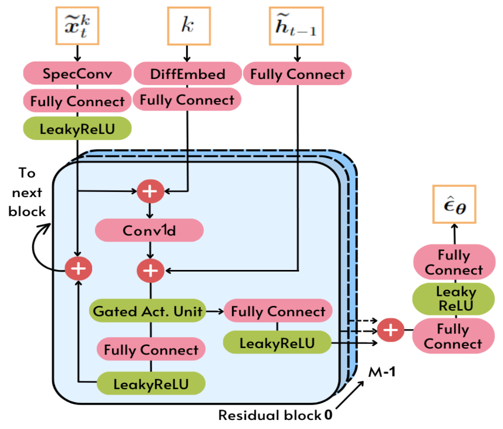

Figure 3: The denoising network $\boldsymbol{\epsilon_\theta}$ is a modified WaveNet structure.

Algorithm 1 SpecSTG training

Input: The distribution of training data after Fourier transform: $q(\{\tilde{\boldsymbol{X}}_c, \tilde{\boldsymbol{X}}_f\})$; hidden states: $\tilde{\boldsymbol{h}}$; number of diffusion steps: K; noise schedule $\{\beta_1, \beta_2, ..., \beta_K\}$, graph: $\mathcal{G}(\mathcal{V}, \mathcal{E}, \boldsymbol{A})$.

Output: Optimized denoising network $\boldsymbol{\epsilon_\theta}$.

```
1: Sample $\{\tilde{\boldsymbol{X}}_c, \tilde{\boldsymbol{X}}_f\} \sim q(\{\tilde{\boldsymbol{X}}_c, \tilde{\boldsymbol{X}}_f\})$
2: while Not Convergence do
3:     for $t = t_0+1, t_0+2, ..., t_0+f$ do
4:         $k \sim \mathrm{Uniform}(1, K)$, $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
5:         Compute $\tilde{\boldsymbol{x}}_t^k = \sqrt{\tilde{\alpha}_k}\tilde{\boldsymbol{x}}_t^0 + \sqrt{1-\tilde{\alpha}_k}\boldsymbol{\epsilon}_t$
6:     end for
7:     Take gradient step on and do gradient descent for $\theta$
           $$\nabla_{\boldsymbol{\theta}} \mathbb{E}_{t,k,\tilde{\boldsymbol{x}}_t^0,\boldsymbol{\epsilon}_t} \left\| \boldsymbol{\epsilon}_t - \boldsymbol{\epsilon_\theta}\left(\tilde{\boldsymbol{x}}_t^k, k, \tilde{\boldsymbol{h}}_{t-1}, \boldsymbol{A}\right)\right\|^2$$
8: end while
```

4.4 Training and Inference

We compute the Fourier representation of all training data before the training process. Next, we input randomly sampled $\{\tilde{\boldsymbol{X}}_c, \tilde{\boldsymbol{X}}_f\}$ to SG-GRU$_{\boldsymbol{\theta}}$ to obtain hidden states $\tilde{\boldsymbol{h}} = \{\tilde{\boldsymbol{h}}_{t_0}, ..., \tilde{\boldsymbol{h}}_{t_0+f-1}\}$. During training, with a pre-specified noise schedule $\{\beta_1, \beta_2, ..., \beta_K\}$, we randomly sample noise $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and step $k \sim \mathrm{Uniform}(0, K)$ to compute disturbed data $\tilde{\boldsymbol{x}}_t^k$ for $t = t_0+1, t_0+2, ..., t_0+f$. Finally, we take a gradient step on the objective function in Equation (3). The training algorithm is provided in **Algorithm 1**. With the denoising network, we can generate samples and make predictions for the forecasting task. The generation process adopts autoregressive sampling, which means we generate samples for each time point one by one. For example, after we generate samples for $t = t_0 + 1$, we feed the sample mean back to the SG-GRU module to compute $\tilde{\boldsymbol{h}}_{t_0+1}$, and

Algorithm 2 SpecSTG sampling and prediction for $\boldsymbol{x}_t^0$

Input: Hidden state: $\tilde{\boldsymbol{h}}_{t-1}$; Fourier operator $\boldsymbol{U}$; number of samples: S; variance hyperparameter: σ_k.
Output: Prediction $\hat{\boldsymbol{x}}_t^0$.

```
1: Randomly generate $S$ samples $\{\tilde{\boldsymbol{x}}_{t,s}^K\}_{s=1}^S \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and do the follows in parallel for all $s$.
2: for $k = K, K-1, ..., 1$ do
3:     $\boldsymbol{e} = \boldsymbol{0}$ if $k = 1$ else $\boldsymbol{e} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
4:     Compute and update $\tilde{\boldsymbol{x}}_{t,s}^{k-1}$ as
           $\frac{1}{\sqrt{1-\beta_k}}\big(\tilde{\boldsymbol{x}}_{t,s}^k - \frac{\beta_k}{\sqrt{1-\tilde{\alpha}_k}}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}_{t,s}^k, k, \tilde{\boldsymbol{h}}_{t-1}, \boldsymbol{A})\big) + \sigma_k \boldsymbol{e}$
5: end for
6: Take average on $S$ samples $\tilde{\boldsymbol{x}}_t^0 = \frac{1}{S}\sum_{s=1}^S \tilde{\boldsymbol{x}}_{t,s}^0$
7: Compute predictions $\hat{\boldsymbol{x}}_t^0 = \boldsymbol{U}\tilde{\boldsymbol{x}}_t^0$
```

then use it to generate samples for $t = t_0 + 2$. Lastly, we convert the predictions back to the original domain via Fourier reconstruction. The sampling and prediction algorithm for a one-time point $\boldsymbol{x}_t^0$ is presented in **Algorithm 2**.

5 EXPERIMENTS

5.1 Datasets and Baseline Models

We validate our model on two traffic datasets, PEMS04 and PEMS08 [Guo et al.(2019)], collected by the California Transportation Agencies' (CalTrans) Performance Measurement System (PEMS) [Chen et al.(2001)]. PEMS04 comprises traffic records from 307 sensors in California's District 04 from Jan 1st to Feb 28th, 2018, while PEMS08 includes data from 170 sensors in District 08 from July 1st to 31st August 2018. Our experiments focus on traffic flow and speed available in both datasets, denoted as "F" and "S" respectively. For instance, "PEMS04F" denotes traffic flow records in the PEMS04 dataset. Each time point covers 5 minutes, with the corresponding value representing the average records during that interval. The speed data are continuous, allowing us to introduce random Gaussian noises. Although traffic flow (i.e., the number of vehicles) is a discrete variable, we still treat it as continuous considering the fact that it contains numerous unique values. More details about the datasets can be found in **Table 1**.

Table 1: Dataset details.

Dataset	Type	#Nodes	#Edges	#Time points
PEMS04F PEMS04S	Flow Speed	307	680	16992
PEMS08F PEMS08S	Flow Speed	170	548	17856

Six probabilistic baselines are considered in our experiments, including four diffusion methods: TimeGrad [Rasul et al.(2021a)], GCRDD [Li et al.(2023)], DiffSTG [Wen et al.(2023)], and PriSTI [Liu et al.(2023)]; and two non-diffusion methods, DeepVAR [Salinas et al.(2020)] and TransNVP [Rasul et al.(2021b)]. All baselines are state-of-the-art diffusion models proposed in recent years. For more discussions on the baseline models, please see **Appendix C**.

5.2 Metrics

We use three metrics to evaluate the performance of SpecSTG, including two deterministic metrics, Root Mean Squred Error (RMSE) and Mean Absolute Error (MAE), and one probabilistic metric, Continuous Ranked Probability Score (CRPS). RMSE and MAE are adopted to measure the distance between predictions and the ground truth. We use the mean of generated samples as predictions to calculate RMSE and MAE. Given predictions $\hat{\boldsymbol{X}}_f$ at time t_0 (after the Fourier reconstruction) and ground truth $\boldsymbol{X}_f$ of one future window, the formulas can be written as:

$$\mathrm{RMSE}(\hat{\boldsymbol{X}}_f, \boldsymbol{X}_f) = \sqrt{\frac{1}{f}\sum_{t=t_0+1}^{t_0+f}(\boldsymbol{x}_t - \hat{\boldsymbol{x}}_t)^2}, \tag{9}$$

$$\mathrm{MAE}(\hat{\boldsymbol{X}}_f, \boldsymbol{X}_f) = \frac{1}{f}\sum_{t=t_0+1}^{t_0+f}|\boldsymbol{x}_t - \hat{\boldsymbol{x}}_t|. \tag{10}$$

The final results reported in our experiments are the averages from all available predictive windows in the test set. CRPS is a probabilistic metric that measures the compatibility of the learned probabilistic distribution at each observation [Matheson and Winkler(1976)]. Given the cumulative distribution function (CDF) F of the distribution estimated at observation x, CRPS is defined as

$$\mathrm{CRPS}(F^{-1}, x) = \int_0^1 2\left(\kappa - \mathbb{I}_{x<F^{-1}(\kappa)}\right)\left(x - F^{-1}(\kappa)\right)\mathrm{d}\kappa, \tag{11}$$

where $\kappa \in [0,1]$, F^{-1} is the quantile function, and $\mathbb{I}_{x<F^{-1}(\kappa)}$ is an indicator function which equals to 1 when $x < F^{-1}(\kappa)$ and 0 otherwise. To calculate the integral, we use 100 samples generated at each time point and sensor/node to approximate the corresponding distribution and calculate CRPS following the way defined in [Tashiro et al.(2021)]. For each future window, we may compute the normalized CRPS at time point $t = t_0 + 1, ..., t_0 + f$ as $\frac{\sum_n \mathrm{CRPS}(F_{t,n}^{-1}, x_{t,n})}{\sum_n |x_{t,n}|}$, where $n = 1, ..., N$ denotes each sensor/node, and $x_{t,n}$ is the value of sensor/node n at time t. Likewise, the "CRPS Avg." is computed as $\frac{\sum_{t,n} \mathrm{CRPS}(F_{t,n}^{-1}, x_{t,n})}{\sum_{t,n} |x_{t,n}|}$. We do not adopt CRPS_{sum} [Rasul et al.(2021a); Tashiro et al.(2021)] as a metric because sensors are not regarded as features in our experiments. The results reported for CRPS are also averages over all available predictive windows in the test set.

5.3 Implementation Details

We implement SpecSTG on a single NVIDIA 4090 GPU with 64GB of memory. The model is trained with the Adam optimizer with a learning rate schedule from $5e-4$ to $1e-2$. The maximum number of epochs is 300 for flow data and 50 for speed data with batch size 64. Validation loss is used for

Table 2: The results of traffic forecasting experiments in a future window of 60 minutes. Average RMSE, MAE, CRPS, and their point values at 15/30/60 minutes are reported. Lower values indicate better forecasting performance. The best results are marked in **bold** and the second best results are $\underline{\text{underlined}}$. Improvements of SpecSTG on existing methods are shown in percentage.

Models	RMSE				MAE				CRPS			
	Avg.	15min	30min	60min	Avg.	15min	30min	60min	Avg.	15min	30min	60min
PEMS04F												
DeepVAR	50.59	43.90	48.76	60.46	37.74	32.97	36.72	45.87	0.2094	0.1997	0.2108	0.2209
TransNVP	82.74	68.26	81.70	99.81	61.85	53.06	62.25	73.49	0.2359	0.2008	0.2377	0.2819
TimeGrad	35.58	33.22	35.24	38.95	$\underline{21.70}$	20.26	$\underline{21.56}$	$\underline{24.04}$	0.0801	0.0747	0.0795	0.0887
GCRDD	36.28	$\underline{31.94}$	45.31	41.99	22.16	$\underline{19.48}$	21.87	26.18	0.0779	$\underline{0.0689}$	0.0768	0.0982
DiffSTG	37.62	34.99	36.68	43.04	24.90	22.53	24.65	29.24	0.0904	0.0815	0.0894	0.1077
PriSTI	$\underline{33.74}$	33.56	$\underline{33.71}$	$\underline{37.31}$	22.46	21.65	22.32	25.19	$\underline{0.0772}$	0.0751	$\underline{0.0764}$	$\underline{0.0870}$
SpecSTG	**33.15**	**30.07**	**32.81**	**37.29**	**21.53**	**19.29**	**21.39**	**23.29**	**0.0766**	**0.0683**	**0.0761**	**0.0866**
Improve.	1.75%	5.86%	2.67%	0.54%	0.78%	0.98%	0.79%	3.12%	0.78%	0.87%	0.39%	0.46%
PEMS08F												
DeepVAR	41.43	35.83	39.88	49.41	27.86	23.19	27.03	34.89	0.1291	0.1219	0.1269	0.1412
TransNVP	67.69	60.48	68.48	76.16	51.37	49.08	52.31	58.38	0.1802	0.1601	0.1822	0.2073
TimeGrad	33.09	30.17	32.53	37.51	20.47	18.24	20.06	24.24	0.0705	0.0618	0.0695	0.0843
GCRDD	28.83	$\underline{23.91}$	28.10	35.68	18.72	$\underline{15.52}$	18.35	23.99	0.0626	$\underline{0.0517}$	0.0617	0.0833
DiffSTG	28.26	25.04	27.54	34.32	18.99	16.66	18.63	23.68	0.0692	0.0609	0.0679	0.0872
PriSTI	$\underline{26.35}$	24.58	$\underline{26.93}$	$\underline{29.91}$	$\underline{17.30}$	15.98	$\underline{17.32}$	$\underline{20.67}$	$\underline{0.0576}$	0.0539	$\underline{0.0581}$	$\underline{0.0688}$
SpecSTG	**25.59**	**22.23**	**24.77**	**29.90**	**17.06**	**14.93**	**16.70**	**20.25**	**0.0572**	**0.0500**	**0.0558**	**0.0680**
Improve.	2.88%	7.03%	8.02%	0.03%	1.39%	3.80%	3.58%	2.03%	0.69%	3.29%	3.96%	1.16%
PEMS04S												
DeepVAR	6.23	5.72	6.11	6.93	2.76	2.52	2.72	3.13	0.0490	0.0450	0.0488	0.0549
TransNVP	6.25	5.73	6.27	6.98	3.36	3.12	3.39	3.74	0.0408	0.0382	0.0412	0.0446
TimeGrad	5.92	5.62	5.91	6.35	2.38	2.19	2.37	2.66	0.0307	0.0282	0.0308	0.0345
GCRDD	$\underline{4.33}$	$\underline{3.10}$	$\underline{4.30}$	5.63	$\underline{1.94}$	$\underline{1.51}$	**1.97**	$\underline{2.58}$	**0.0245**	**0.0189**	**0.0248**	$\underline{0.0329}$
DiffSTG	4.46	3.24	4.46	5.72	2.15	1.66	2.20	2.83	0.0264	0.0206	0.0267	0.0340
PriSTI	4.42	3.31	4.67	$\underline{5.60}$	1.96	1.54	$\underline{1.99}$	2.62	$\underline{0.0252}$	0.0198	0.0258	$\underline{0.0329}$
SpecSTG	**4.06**	**3.01**	**4.09**	**5.15**	**1.93**	**1.50**	**1.97**	**2.51**	**0.0245**	$\underline{0.0192}$	$\underline{0.0253}$	**0.0319**
Improve.	6.24%	2.90%	4.88%	8.04%	0.52%	0.66%	0.00%	2.71%	0.00%	-	-	3.04%
PEMS08S												
DeepVAR	5.73	5.55	5.70	6.05	2.56	2.42	2.57	2.79	0.0544	0.0534	0.0543	0.0558
TransNVP	5.41	5.12	5.54	5.74	2.76	2.64	2.83	2.91	0.0349	0.0334	0.0358	0.0368
TimeGrad	4.98	4.93	4.97	5.03	1.98	1.95	1.97	$\underline{2.12}$	0.0267	0.0262	0.0268	$\underline{0.0272}$
GCRDD	$\underline{3.75}$	$\underline{2.74}$	$\underline{3.77}$	4.89	1.72	1.35	$\underline{1.75}$	2.32	$\underline{0.0223}$	0.0171	$\underline{0.0226}$	0.0301
DiffSTG	3.97	3.20	4.07	$\underline{4.82}$	2.36	1.91	2.43	3.04	0.0325	0.0259	0.0335	0.0423
PriSTI	4.22	3.02	4.39	5.20	$\underline{1.70}$	$\underline{1.29}$	1.80	2.15	**0.0217**	**0.0162**	0.0230	$\underline{0.0272}$
SpecSTG	**3.45**	**2.58**	**3.46**	**4.36**	**1.63**	**1.27**	**1.67**	**2.02**	**0.0217**	$\underline{0.0170}$	**0.0219**	**0.0268**
Improve.	8.00%	5.84%	8.22%	9.54%	4.12%	1.55%	4.57%	4.72%	0.00%	-	0.10%	1.47%

model selection. The hyperparameters specific to diffusion models are set as follows. We use the quadratic scheme for noise level β_k starting from $\beta_1 = 1e - 4$ and tune β_K in $[0.1, 0.2, 0.3, 0.4]$. The number of diffusion steps K is selected from $[50, 100, 200]$. The maximum polynomial order in $\mathrm{SpecConv}$ is set as 2. The hidden size D_h is tuned in $[64, 96]$. In SG-Wave, the number of residual blocks $M = 8$ and the residual channel $D_r = 8$. Finally, the number of samples S is set as 100 for all models. For all experiments, we split datasets with 60%/20%/20% train/validation/test proportions and apply Z-score normalization before the Fourier transform. The graph structure is constructed depending on the distance between sensors following [Guo et al.(2019)]. More implementation details are presented in **Appendix D**.

5.4 Experiment Results

Table 2 presents the results of our traffic forecasting experiments, where the prediction task involves forecasting future time series for a 60-minute horizon based on observations from the past 60 minutes. The table includes numerical values for the average RMSE, MAE, and CRPS over the entire forecast window. Additionally, it provides corresponding point evaluations assessed at 15, 30, and 60 minutes such that we can evaluate the short and long-term forecasting performance of the models.

Analysis of deterministic results SpecSTG consistently achieves top-tier deterministic results across various tasks. In comparison to the second-best model, SpecSTG exhibits an 8.00% improvement in average RMSE for PEMS08S and a 6.24% improvement for PEMS04S. Similarly, the average MAE sees enhancements of 4.12% for PEMS08S and 1.39% for PEMS08F. In addition, our method shows proficiency in both short and long-term deterministic forecasting. Particularly, the RMSE improvement of SpecSTG achieves 7.03% on PEMS08F for short-term 15-minute forecasting and 9.54% on PEMS08S for long-term 60-minute forecasting. We also observe improvements in MAE for most results in the table, especially for traffic flow tasks. The superior deterministic performance stems from SpecSTG's unique probabilistic learning process in the spectral domain, lever-

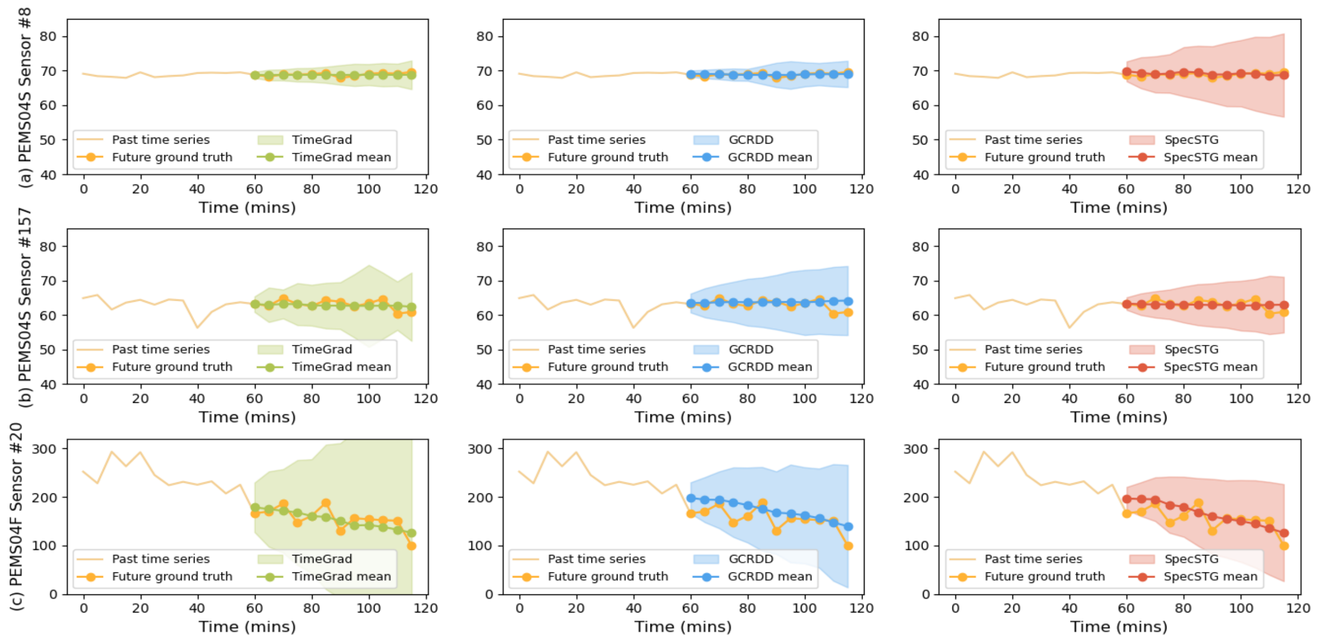

Figure 4: Forecasting visualizations of TimeGrad (green), GCRDD (blue), and SpecSTG (red). (a) and (b) are results on speed data (PEMS04S), while (c) presents results on flow data (PEMS04F).

aging rich global spatial information, which is an acknowledged crucial component in deterministic traffic forecasting [Yu et al.(2018); Fang et al.(2019)].

Analysis of probabilistic results Regarding the probabilistic metric CRPS, SpecSTG demonstrates an advantage in traffic flow forecasting but not in vehicle speed forecasting. The average CRPS is improved by 0.78% and 0.69% with SpecSTG on PEMS04F and PEMS08F, respectively. However, little improvement is observed on PEMS04S and PEMS08S. Since speed and flow datasets share the same graph structure (e.g., PEMS04S and PEMS04F), we suggest that the inferior probabilistic performance is related to the time series data. A possible explanation is that flow data is often associated with higher variations, thus the systematic variations measured by the Fourier representation are more informative than the speed data. For instance, the standard deviations of data in PEMS08S and PEMS08F are 6.65 and 146.22, respectively. We will further investigate this observation with forecasting visualizations in Subsection 6.1.

Comments on baseline models Non-diffusion models such as DeepVAR and TransNVP clearly show inferior performance compared to diffusion models. PriSTI excels in predicting traffic flow, leveraging the attention mechanism to process temporal dynamics. On the other hand, TimeGrad and GCRDD exhibit comparably better performance in forecasting vehicle speeds, utilizing a recurrent structure that emphasizes consecutive temporal connectivity. We suppose this distinction is caused by the attention mechanism's effectiveness in capturing global temporal variations and the recurrent structure's aptness for learning highly correlated speed patterns. Our approach, with a recurrent encoder, combines the strengths of both methodologies. It preserves the ability to learn continuous time patterns while enhancing the identification of variations through the incorporation of spectral variation measurements.

6 DISCUSSIONS

In this section, we will visualize SpecSTG's forecasting outcomes compare with two baseline diffusion models. In addition, we will provide analyses on time efficiency and sensitivity to hyperparameters to further show the advantage of SpecSTG. More supplementary discussions can be found in **Appendices E**, **F**, and **G**.

6.1 Forecasting Visualizations

Recall that in Subsection 5.4, SpecSTG performs better in probabilistic forecasting on traffic flow data than speed data. Here we further explore this observation by visualizing the forecasting outcomes of TimeGrad, GCRDD, and SpecSTG on PEMS04S and PMES04F (**Figure 4**). The figure displays the mean and 95% confidence interval (adjusted by time) of estimated future distributions. Our primary objective is to assess whether the forecasting intervals produced by various methods are compatible with the actual future time series. An appropriate interval should capture the future variations while remaining sufficiently narrow to provide meaningful insights.

In traffic speed forecasting, SpecSTG's mean estimation is closer to future time series, but the intervals generated by TimeGrad and GCRDD sometimes better fit the variations in future values. Upon closer examination of the data patterns, we observe that this impact is particularly pronounced in windows with very small variations (**Figure 4 (a)**). In contrast, distributions estimated by SpecSTG at sensors with larger variations (**Figure 4 (b)**) exhibit a better ability to capture

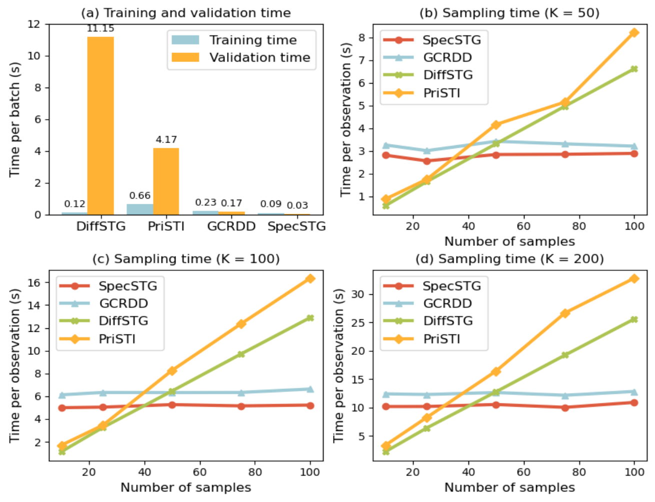

Figure 5: Time efficiency of training, validation, and sampling.

future uncertainty compared to TimeGrad and GCRDD. This matches our previous hypothesis in Subsection 5.4 that **STG forecasting with larger variations benefits more from the spectral diffusion process**. Because more systematic fluctuations exist in such data, and thus more information can be captured by the Fourier representation, and eventually learned by the diffusion process.

Analogously, in traffic flow forecasting on PEMS04F, a dataset characterized by high systematic variation, SpecSTG demonstrates promising performance by generating both more accurate deterministic predictions and more compatible distributions (**Figure 4 (c)**). This observation reflects the experiment results that SpecSTG achieves outstanding performance with PEMS04F in terms of all metrics.

6.2 Time Efficiency

In **Figure 5**, we compare the training, validation, and sampling time of SpecSTG with three diffusion models for STG forecasting, including GCRDD, DiffSTG, and PriSTI. The training and validation of SpecSTG are clearly faster than other diffusion baselines. Particularly, SpecSTG's training plus validation time is 3.33× of GCRDD, the most efficient method among other existing state-of-the-arts. The validation time of DiffSTG is significantly high because it requires sampling and prediction during validation. To show sampling efficiency, we plot the sampling time per observation (i.e., a future window of 60 minutes) when diffusion steps $K = 50, 100, 200$. We set the batch size of one-shot methods, DiffSTG and PriSTI, as 8 and 16, and report the best results. For autoregressive methods, SpecSTG shows a notable time advantage over GCRDD in sampling. Besides, their sampling time does not vary much with the number of samples S. By contrast, the time cost of one-shot methods increases rapidly with the increase of S. Although DiffSTG and PriSTI are more efficient when S is small, a small number of samples often cannot present a clear picture of future data distribution.

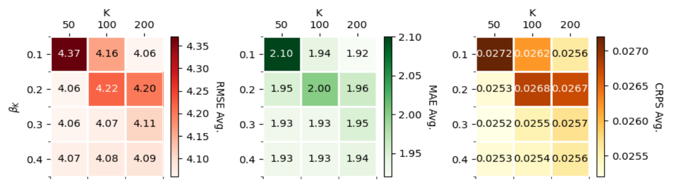

Figure 6: Sensitivity analysis of SpecSTG on key hyperparameters: diffusion steps K and the end of beta schedule β_K.

6.3 Sensitivity Analysis on Hyperparameters

In this sensitivity analysis, we focus on the combination of two very important diffusion hyperparameters: the number of diffusion steps, K, and the end of noise schedule, β_K. Theoretically, the choices of K and β_K are relevant to each other. Since the noise level gradually increases from $\beta_1 = 1e - 4$ to β_K in K diffusion steps, these two hyperparameters control the changing speed and level of noises in the diffusion process, which is essential for the white noise assumption of diffusion models. The same as the implementation of SpecSTG in our experiments, we set the search spaces of β_K and K as $[0.1, 0.2, 0.3, 0.4]$ and $[50, 100, 200]$, respectively. In **Figure 6**, we use heatmaps to show the change in model performance in terms of RMSE, MAE, and CRPS on PEMS04S with different hyperparameter combinations. We observe that SpecSTG typically performs better with a larger β_K (for instance 0.3 or 0.4). The best results appear when $\beta_K = 0.3$ and $K = 50$. It is also worth noting that the forecasting performance of SpecSTG does not vary dramatically with different hyperparameter combinations. This means our method is not very sensitive to hyperparameter selection, alleviating the burden of hyperparameter tuning.

7 RELATED WORKS

Generative Diffusion Models The initial idea of diffusion models was introduced by [Sohl-Dickstein et al.(2015)]. Then, some improvements proposed by [Ho et al.(2020)] endowed them with remarkable practical value, contributing to their conspicuous popularity nowadays. In recent years, diffusion models have demonstrated their power over many existing generative techniques in various real-world applications such as image synthesis [Austin et al.(2021); Dhariwal and Nichol(2021); Ho et al.(2022a)], video generation [Harvey et al.(2022); Ho et al.(2022b); Yang et al.(2022)], natural language processing [Li et al.(2022b); Nikolay et al.(2022); Yu et al.(2022)], and time series prediction [Rasul et al.(2021a); Li et al.(2022a); Alcaraz and Strodthoff(2023)].

Diffusion Models for Time Series and STGs Pioneering diffusion models for time series such as TimeGrad [Rasul et al.(2021a)] and TimeDiff [Shen and Kwok(2023)] were originally tailored for multivariate time series forecasting, utilizing sensors as variables but only exploring general dependencies without incorporating graph structural information. Recently, several diffusion models were proposed specifically for STG forecasting. DiffSTG [Wen et al.(2023)] incorporates graph structure into the backward kernel with a graph-modified Unet [Ronneberger et al.(2015)] architecture. GCRDD [Li et al.(2023)], designed in reminiscent of TimeGrad, adopts a graph-enhanced recurrent encoder to produce hidden states from past time series as conditions. Additionally, USTD [Hu et al.(2023)] introduces a pre-trained encoder that better captures deterministic patterns via an unsupervised reconstruction task. DVGNN [Liang et al.(2023)] is a deterministic model but with a diffusion module to generate dynamic adjacency matrices in its pre-training process. Furthermore, PriSTI [Liu et al.(2023)] was initially developed from CSDI [Tashiro et al.(2021)] for STG imputation, but with potential for forecasting tasks by masking future data as missing values. We highlight that SpecSTG's novelty lies in its unique spectral diffusion framework that generates graph Fourier representation of future time series, which leverages systematic fluctuations in time series data guided by graph structure to boost forecasting accuracy.

Spectral diffusion on graphs and time series The idea of spectral diffusion has been applied in generating graph structure and classic time series. GSDM [Luo et al.(2023)] explores the generation of spectral graph structure, i.e., the eigenvalues of graph adjacency matrices, to enhance graph generation quality. Besides, research has shown that generating classic time series in the Fourier domain facilitates diffusion models to better capture the training distribution [Crabbé et al.(2024)]. Our method, SpecSTG, is the first endeavour to investigate spectral diffusion in generating graph signals and spatio-temporal data.

8 CONCLUDING REMARK

In this paper, we proposed SpecSTG, a spectral diffusion approach for fast probabilistic spatio-temporal traffic forecasting. Our method transforms the entire diffusion learning process to the spectral domain by generating the Fourier representation of future time series instead of the original data. Although we have introduced the autoregressive architecture of SpecSTG, the idea of spectral diffusion can be straightforwardly applied to one-shot methods as well by altering the generative target and graph convolution. Hence, SpecSTG can be regarded as an effective framework for STG forecasting. Experiment results confirm the superior performance of SpecSTG, demonstrating more efficient training and sampling compared to state-of-the-art diffusion methods. Nevertheless, we highlight that SpecSTG may fall short of predicting compatible future distributions when the data have low variations, diminishing the efficacy of spectral measurements of systematic fluctuations.

References

[Alcaraz and Strodthoff(2023)] Juan Lopez Alcaraz and Nils Strodthoff. 2023. Diffusion-based Time Series Imputation and Forecasting with Structured State Space Models. *Transactions on Machine Learning Research* 3 (2023).

[Austin et al.(2021)] Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. 2021. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems* 34 (2021), 17981–17993.

[Boukerche et al.(2020)] Azzedine Boukerche, Yanjie Tao, and Peng Sun. 2020. Artificial intelligence-based vehicular traffic flow prediction methods for supporting intelligent transportation systems. *Computer Networks* 182 (2020), 107484.

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

[Chung(1997)] Fan RK Chung. 1997. *Spectral graph theory*. Vol. 92. American Mathematical Society.

[Coletta et al.(2023)] A. Coletta, S. Gopalakrishan, D. Borrajo, and S. Vyetrenko. 2023. On the Constrained Time-Series Generation Problem. *arXiv preprint* 2307.01717 (2023).

[Crabbé et al.(2024)] Jonathan Crabbé, Nicolas Huynh, Jan Stanczuk, and Mihaela van der Schaar. 2024. Time series diffusion in the frequency domain. *International Conference on Machine Learning (ICML)* (2024).

[Defferrard et al.(2016)] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 29.

[Dhariwal and Nichol(2021)] Prafulla Dhariwal and Alexander Nichol. 2021. Diffusion models beat GANs on image synthesis. In *Advances in Neural Information Processing Systems*, Vol. 34. 8780–8794.

[Dinh et al.(2017)] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. 2017. Density estimation using Real NVP. In *International Conference on Learning Representations (ICLR)*.

[Fang et al.(2019)] Shen Fang, Qi Zhang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. 2019. GSTNet: Global Spatial-Temporal Network for Traffic Flow Prediction.. In *International Joint Conference on Artificial Intelligence (IJCAI)*. 2286–2293.

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

[Harvey et al.(2022)] William Harvey, Saeid Naderiparizi, Vaden Masrani, Christian Weilbach, and Frank Wood. 2022. Flexible diffusion modeling of long videos. In *Advances in Neural Information Processing Systems*, Vol. 35. 27953–27965.

[Ho et al.(2020)] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 33. 6840–6851.

[Ho et al.(2022a)] Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. 2022a. Cascaded Diffusion Models for High Fidelity Image Generation. *Journal of Machine Learning Research* 23, 47 (2022), 1–33.

[Ho et al.(2022b)] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. 2022b. Video Diffusion Models. In *Advances in Neural Information Processing Systems*, Vol. 35. PMLR, 8633–8646.

[Hu et al.(2023)] Junfeng Hu, Xu Liu, Zhencheng Fan, Yuxuan Liang, and Roger Zimmermann. 2023. Towards Unifying Diffusion Models for Probabilistic Spatio-Temporal Graph Learning. *arXiv:2310.17360* (2023).

[Jin et al.(2023b)] Guangyin Jin, Yuxuan Liang, Yuchen Fang, Zezhi Shao, Jincai Huang, Junbo Zhang, and Yu Zheng. 2023b. Spatio-temporal graph neural networks for predictive learning in urban computing: A survey. *IEEE Transactions on Knowledge and Data Engineering* (2023).

[Jin et al.(2023a)] Ming Jin, Huan Yee Koh, Qingsong Wen, Daniele Zambon, Cesare Alippi, Geoffrey I Webb, Irwin King, and Shirui Pan. 2023a. A survey on graph neural networks for time series: Forecasting, classification, imputation, and anomaly detection. *arXiv:2307.03759* (2023).

[Kipf and Welling(2016)] Thomas N. Kipf and Max Welling. 2016. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*.

[Kreuzer et al.(2021)] Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. 2021. Rethinking graph transformers with spectral attention. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 34. 21618–21629.

[Lana et al.(2018)] Ibai Lana, Javier Del Ser, Manuel Velez, and Eleni I Vlahogianni. 2018. Road traffic forecasting: Recent advances and new challenges. *IEEE Intelligent Transportation Systems Magazine* 10, 2 (2018), 93–109.

[Li et al.(2023)] Ruikun Li, Xuliang Li, Shiying Gao, ST Boris Choy, and Junbin Gao. 2023. Graph convolution recurrent denoising diffusion model for multivariate probabilistic temporal forecasting. In *International Conference on Advanced Data Mining and Applications (ADMA)*. Springer, 661–676.

[Li et al.(2022b)] Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. 2022b. Diffusion-LM improves controllable text generation. In *Advances in Neural Information Processing Systems*, Vol. 35. 4328–4343.

[Li et al.(2022a)] Yan Li, Xinjiang Lu, Yaqing Wang, and Dejing Dou. 2022a. Generative Time Series Forecasting with Diffusion, Denoise, and Disentanglement. In *Advances in Neural Information Processing Systems*, Vol. 35. 23009–23022.

[Li et al.(2018)] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2018. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations (ICLR)*. 1–8.

[Liang et al.(2023)] Guojun Liang, Prayag Tiwari, Sławomir Nowaczyk, Stefan Byttner, and Fernando Alonso-Fernandez. 2023. Dynamic Causal Explanation Based Diffusion-Variational Graph Neural Network for Spatio-temporal Forecasting. *arXiv:2305.09703* (2023).

[Lin et al.(2023)] Lequan Lin, Zhengkun Li, Ruikun Li, Xuliang Li, and Junbin Gao. 2023. Diffusion models for time-series applications: a survey. *Frontiers of Information Technology and Electronic Engineering* (2023), 1–23.

[Liu et al.(2023)] Mingzhe Liu, Han Huang, Hao Feng, Leilei Sun, Bowen Du, and Yanjie Fu. 2023. PriSTI: A Conditional Diffusion Framework for Spatiotemporal Imputation. In *International Conference on Data Engineering (ICDE)*. 1–10.

[Luo et al.(2023)] Tianze Luo, Zhanfeng Mo, and Sinno Jialin Pan. 2023. Fast graph generation via spectral diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023).

[Matheson and Winkler(1976)] James E Matheson and Robert L Winkler. 1976. Scoring rules for continuous probability distributions. *Management Science* 22, 10 (1976), 1087–1096.

[Nikolay et al.(2022)] Savinov Nikolay, Chung Junyoung, Binkowski Mikolaj, Elsen Erich, and Oord Aäron van den. 2022. Step-unrolled Denoising Autoencoders for Text Generation. In *International Conference on Learning Representations*. 1–8.

[Pal et al.(2021)] Soumyasundar Pal, Liheng Ma, Yingxue Zhang, and Mark Coates. 2021. RNN with particle flow for probabilistic spatio-temporal forecasting. In *International Conference on Machine Learning (ICML)*. PMLR, 8336–8348.

[Papamakarios et al.(2017)] George Papamakarios, Theo Pavlakou, and Iain Murray. 2017. Masked autoregressive flow for density estimation. *Advances in Neural Information Processing Systems (NeurIPS)* 30 (2017).

[Rasul et al.(2021a)] Kashif Rasul, Calvin Seward, Ingmar Schuster, and Roland Vollgraf. 2021a. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In *International Conference on Machine Learning (ICML)*. 8857–8868.

[Rasul et al.(2021b)] Kashif Rasul, Abdul-Saboor Sheikh, Ingmar Schuster, Urs Bergmann, and Roland Vollgraf. 2021b. Multi-variate Probabilistic Time Series Forecasting via Conditioned Normalizing Flows. In *International Conference on Learning Representations*. 1–8.

[Ronneberger et al.(2015)] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention*. Springer, 234–241.

[Salinas et al.(2020)] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. 2020. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting* 36, 3 (2020), 1181–1191.

[Seo et al.(2018)] Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. 2018. Structured sequence modeling with graph convolutional recurrent networks. In *International Conference on Neural Information Processing (ICONIP)*. Springer, 362–373.

[Shao et al.(2022)] Zezhi Shao, Zhao Zhang, Wei Wei, Fei Wang, Yongjun Xu, Xin Cao, and Christian S. Jensen. 2022. Decoupled dynamic spatial-temporal graph neural network for traffic forecasting. *Proceedings of the VLDB Endowment* 15, 11 (2022), 2733–2746. https://doi.org/10.14778/3551793.3551827

[Shen and Kwok(2023)] L. F. Shen and J. Kwok. 2023. Non-autoregressive Conditional Diffusion Models for Time Series Prediction. In *International Conference on Machine Learning (ICML)*. PMLR.

[Sohl-Dickstein et al.(2015)] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning (ICML)*. PMLR, 2256–2265.

[Sutskever et al.(2014)] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems (NeurIPS)* 27 (2014).

[Tashiro et al.(2021)] Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. 2021. CSDI: Conditional score-based diffusion models for probabilistic time series imputation. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 34. 24804–24816.

[van den Oord et al.(2016)] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. 2016. WaveNet: A Generative Model for Raw Audio. In *The 9th ISCA Speech Synthesis Workshop*. 125.

[Vaswani et al.(2017)] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 30. 6000–6010.

[Wen et al.(2023)] Haomin Wen, Youfang Lin, Yutong Xia, Huaiyu Wan, Qingsong Wen, Roger Zimmermann, and Yuxuan Liang. 2023. DiffSTG: Probabilistic spatio-temporal graph forecasting with denoising diffusion models. In *ACM International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL)*. 1–12.

[Yang et al.(2023)] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. 2023. Diffusion models: A comprehensive survey of methods and applications. *Comput. Surveys* 56, 4 (2023), 1–39.

[Yang et al.(2022)] Ruihan Yang, Prakhar Srivastava, and Stephan Mandt. 2022. Diffusion probabilistic modeling for video generation. *arXiv:2203.09481* 1 (2022).

[Yu et al.(2018)] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2018. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *International Joint Conference on Artificial Intelligence (IJCAI)* (2018).

[Yu et al.(2022)] Peiyu Yu, Sirui Xie, Xiaojian Ma, Baoxiong Jia, Bo Pang, Ruigi Gao, Yixin Zhu, Song-Chun Zhu, and Ying Nian Wu. 2022. Latent Diffusion Energy-Based Model for Interpretable Text Modelling. In *International Conference on Machine Learning*, Vol. 162. PMLR, 25702–25720.

[Yuan and Li(2021)] Haitao Yuan and Guoliang Li. 2021. A survey of traffic prediction: from spatio-temporal data to intelligent transportation. *Data Science and Engineering* 6 (2021), 63–85.

[Zheng et al.(2020)] Chuanpan Zheng, Xiaoliang Fan, Cheng Wang, and Jianzhong Qi. 2020. Gman: A graph multi-attention network for traffic prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 1234–1241.

[Zivot and Wang(2006)] Eric Zivot and Jiahui Wang. 2006. Vector autoregressive models for multivariate time series. *Modeling Financial Time Series with S-PLUS®* (2006), 385–429.

APPENDICES

Appendix A. Denoising Diffusion Probabilistic Model

Denoising diffusion probabilistic model (DDPM) is one of the classic formulations of diffusion models [Sohl-Dickstein et al.(2015); Ho et al.(2020); Yang et al.(2023)]. With its high flexibility in modelling target distributions and outstanding capability of capturing complex generative patterns, DDPM is widely used in various time series tasks [Rasul et al.(2021a); Tashiro et al.(2021); Coletta et al.(2023); Lin et al.(2023)]. Unlike traditional probabilistic models that learn the target distribution $q(x)$ explicitly to generate samples, DDPM learns how to generate samples directly via a pair of forward-backward Markov chains without accessing the actual target distribution function. Assuming that $\boldsymbol{x}^0$ is the original data, for $k = 0, 1, ..., K$, the forward chain injects Gaussian noises to the generative target with a probabilistic transition kernel

$$q(\boldsymbol{x}^k|\boldsymbol{x}^{k-1}) = \mathcal{N}\left(\boldsymbol{x}^k; \sqrt{1-\beta_k}\boldsymbol{x}^{k-1}, \beta_k \boldsymbol{I}\right),$$

where $\beta_k \in (0, 1)$ is a hyperparameter controlling the noise level at each forward step. The design of forward chain enables us to derive the disturbed data at a particular step k directly from $\boldsymbol{x}^0$ as

$$q(\boldsymbol{x}^k|\boldsymbol{x}^0) = \mathcal{N}\left(\boldsymbol{x}^k; \sqrt{\tilde{\alpha}_k}\boldsymbol{x}^0, (1-\tilde{\alpha}_k)\boldsymbol{I}\right),$$

where $\tilde{\alpha}_k := \prod_{i=1}^{k}(1-\beta_i)$. This means $\boldsymbol{x}^k = \sqrt{\tilde{\alpha}_k}\boldsymbol{x}^0 + \sqrt{1-\tilde{\alpha}_k}\boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. To ensure that the forward chain transits eventually to white noises, one shall have $\tilde{\alpha}_K \approx 0$ such that $q(\boldsymbol{x}^K) := \int q(\boldsymbol{x}^K|\boldsymbol{x}^0)q(\boldsymbol{x}^0)\mathrm{d}\boldsymbol{x}^0 \approx \mathcal{N}(\boldsymbol{x}^K; 0, \boldsymbol{I})$. Next, the backward chain recovers white noises to original data through a Gaussian transition kernel $p_{\boldsymbol{\theta}}(\boldsymbol{x}^{k-1}|\boldsymbol{x}^k) = \mathcal{N}\left(\boldsymbol{x}^{k-1}; \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}^k, k), \sigma_k \mathbf{I}\right)$, where $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}^k, k)$ is usually parameterized with a neural network with learnable parameters $\boldsymbol{\theta}$, and σ_k is a variance hyperparameter. These parameters are optimized by minimizing the negative evidence lower-bound (ELBO):

$$\mathcal{L}_E(\boldsymbol{\theta}) = \mathbb{E}_{q(\boldsymbol{x}^{0:K})}\left[-\log p(\boldsymbol{x}^K) - \sum_{k=1}^{K}\log\frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}^{k-1}|\boldsymbol{x}^k)}{q(\boldsymbol{x}^k|\boldsymbol{x}^{k-1})}\right].$$

The learning process is simplified with DDPM by [Ho et al.(2020)]. Instead of learning $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}^k, k)$, a denoising network $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}\left(\boldsymbol{x}^k, k\right)$ is learned with the following objective function:

$$\mathcal{L}_{DDPM}(\boldsymbol{\theta}) = \mathbb{E}_{k,\boldsymbol{x}^0,\boldsymbol{\epsilon}}\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}\left(\boldsymbol{x}^k, k\right)\right\|^2.$$

This denoising network $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}^k, k)$ takes noised data $\boldsymbol{x}^k$ and step index k as inputs to predict the noise injected at step k in the forward chain. Finally, DDPM generates samples by eliminating the noises in random white noise $\boldsymbol{x}^K \sim \mathcal{N}(\boldsymbol{x}^K; 0, \boldsymbol{I})$. For backward step $k = K, K-1, ..., 1$, DDPM updates the sample as

$$\boldsymbol{x}^{k-1} \leftarrow \frac{1}{\sqrt{1-\beta_k}}\left(\boldsymbol{x}^k - \frac{\beta_k}{\sqrt{1-\tilde{\alpha}_k}}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}^k, k)\right) + \sigma_k \boldsymbol{e}$$

where σ_k is a hyperparameter of variance in the backward transition kernel with $\boldsymbol{e} = \boldsymbol{0}$ for $k = 1$, and $\boldsymbol{e} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ otherwise. Eventually, $\boldsymbol{x}^0$ will be a sample from the same distribution of the generative target.

Appendix B. Generalized Fourier Transform

The Fourier transform can be naturally generalized to multivariate traffic STGs. Assume that we have a traffic STG with graph signals $\boldsymbol{X}_{\mathcal{G}} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_t, ...|\boldsymbol{x}_t \in \mathbb{R}^{N\times D_x}\}$, where D_x is the number of variables. The graph is a univariate STG when $D_x = 1$, and a multivariate STG when $D_x \geq 2$. For a sampled past-future window $\boldsymbol{X} = \{\boldsymbol{x}_{t_0-c+1}, ..., \boldsymbol{x}_{t_0+f}\} \in \mathbb{R}^{N\times D_x\times(c+f)}$, we first split it according to its variables and rewrite it as $\boldsymbol{X} = \{\boldsymbol{x}'_1, ..., \boldsymbol{x}'_d, ..., \boldsymbol{x}'_{D_x}|\boldsymbol{x}'_d \in \mathbb{R}^{N\times(c+f)}\}$. Then, with the Fourier operator $\boldsymbol{U}$, we conduct the transform on each variable matrix $\boldsymbol{x}'_d$ as

$$\tilde{\boldsymbol{x}}'_d = \boldsymbol{U}^{\intercal}\boldsymbol{x}'_d.$$

Hence, the Fourier representation of the sampled window $\boldsymbol{X}$ is $\tilde{\boldsymbol{X}} = \{\tilde{\boldsymbol{x}}'_1, ..., \tilde{\boldsymbol{x}}'_d, ..., \tilde{\boldsymbol{x}}'_{D_x}\} \in \mathbb{R}^{N\times D_x\times(c+f)}$. To convert the Fourier representation of a single variable matrix back to the original domain, we may apply Fourier reconstruction as

$$\boldsymbol{x}'_d = \boldsymbol{U}\tilde{\boldsymbol{x}}'_d,$$

so $\boldsymbol{X} = \{\boldsymbol{U}\tilde{\boldsymbol{x}}'_1, ..., \boldsymbol{U}\tilde{\boldsymbol{x}}'_d, ..., \boldsymbol{U}\tilde{\boldsymbol{x}}'_{D_x}\} \in \mathbb{R}^{N\times D_x\times(c+f)}$.

Appendix C. Baselines

In the main experiment, we compare SpecSTG with four state-of-the-art diffusion baselines:

- **TimeGrad** [Rasul et al.(2021a)]: an autoregressive diffusion model using long short-term memory (LSTM) or gated recurrent units (GRU) to encode temporal dynamics and a time-modified WavaNet [van den Oord et al.(2016)] as $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$.
- **GCRDD** [Li et al.(2023)]: an autoregressive diffusion model for spatio-temporal forecasting, which adopts a graph-modified GRU to encode past time series and spatial connectivity as conditions. Developed from TimeGrad, GCRDD also employs a WaveNet architecture for its $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ but with graph convolution to process spatial information.
- **DiffSTG** [Wen et al.(2023)]: an one-shot diffusion model for spatio-temporal forecasting with graph-modified UNet [Ronneberger et al.(2015)] as $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$.
- **PriSTI** [Liu et al.(2023)]: a one-shot diffusion model for spatio-temporal imputation. It is equipped with a feature extraction mechanism to construct conditions. An attention-based [Vaswani et al.(2017)] WaveNet is adopted as its $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$.

In addition, we also compare SpecSTG with two non-diffusion probabilistic methods, including

- **DeepVAR** [Salinas et al.(2020)]: an autoregressive RNN-based model for multivariate time series forecasting, known as the multivariate variant of DeepAR.

- **Transformer Normalizing Flow** [Rasul et al.(2021b)]: an autoregressive model for multivariate time series forecasting that approximates the target distribution with a normalizing flow such as Real-NVP [Dinh et al.(2017)] and MAF [Papamakarios et al.(2017)]. Here we choose Real-NVP, and we call this model TransNVP.

In addition, in **Appendix E**, we provide supplementary comparisons with non-diffusion probabilistic methods, including

- **Historical Average (HA)**: a deterministic method that regards time series as periodic processes and predicts future time series with weighted averages from past observations.
- **Vector Auto-Regressive model (VAR)** [Zivot and Wang(2006)]: a deterministic model for multivariate time series forecasting that assumes time series are stationary and predicts with lagged observations.
- **FC-LSTM** [Sutskever et al.(2014)]: an LSTM model with fully connected (FC) hidden units that performs well in capturing sequential temporal dependencies.
- **STGCN** [Yu et al.(2018)]: a graph convolutional network for spatio-temporal traffic forecasting, equipped with gated temporal convolution and graph convolution to process temporal and spatial information.
- **DCRNN** [Li et al.(2018)]: a recurrent graph neural network that integrates graph diffusion and GRU under an encoder-decoder structure.
- **ASTGCN** [Guo et al.(2019)]: an attention-based graph convolutional neural network that adopts both graph attention and temporal attention to learn spatio-temporal patterns.
- **GMAN** [Zheng et al.(2020)]: a graph neural network with multiple spatio-temporal attention blocks in its encoder-decoder architecture to enhance the learning of spatio-temporal patterns.

Appendix D. Implementation Details

Diffusion baselines Diffusion models are implemented with similar diffusion process hyperparameters as SpecSTG. For all models that utilize WaveNet architecture in their denoising networks, we fix the number of residual blocks and residual channels both as 8. Other model-specific implementation details follow their original papers along with the default settings in their codes.

Non-diffusion probabilistic baselines DeepVAR is implemented with `PyTorchTS`[1]. We choose the LSTM with 2 layers as its recurrent structure. The size of hidden states is fixed at 1024, because a smaller hidden size leads to unsatisfactory performance, and a larger hidden size will not improve the results much. TransNVP is also implemented with `PyTorchTS`. Given that details of TransNVP are not mentioned in its paper, we set its hyperparameters majorly according to the settings of Transformer MAF, which is a similar model but uses MAF to model the target distribution. We also tune the hyperparameters when needed to improve the performance of TransNVP.

Deterministic baselines The results of deterministic baselines are retrieved directly from the experiments in [Shao et al.(2022)], which adopts similar experiment settings (e.g., 60%/20%/20% data split) as our experiments.

Table 3: Comparison between SpecSTG and classic deterministic models on PEMS04F and PEMS08F. Partial results are retrieved from [Shao et al.(2022)]. The best results are marked in **bold**, and the second best results are underlined.

Model	RMSE			MAE		
	15min	30min	60min	15min	30min	60min
			PEMS04F			
HA	42.69	49.37	67.43	28.92	33.73	46.97
VAR	34.30	36.58	40.28	21.94	23.72	26.76
FC-LSTM	33.37	39.1	50.73	21.42	25.83	36.41
STGCN	30.76	34.43	41.11	19.35	21.85	26.97
DCRNN	31.94	36.15	44.81	20.34	23.21	29.24
ASTGCN	31.43	34.34	40.02	20.15	22.09	26.03
GMAN	**29.32**	**30.77**	**30.21**	**18.28**	**18.75**	**19.95**
SpecSTG	30.17	32.81	37.29	19.29	21.39	23.29
			PEMS08F			
HA	34.96	40.89	56.74	23.52	27.67	39.28
VAR	29.73	30.30	38.97	19.52	22.25	26.17
FC-LSTM	26.27	34.53	47.03	17.38	21.22	30.69
STGCN	25.03	27.27	34.21	15.30	17.69	25.46
DCRNN	25.48	27.63	34.21	15.64	17.88	22.51
ASTGCN	25.09	28.17	33.68	16.48	18.66	22.83
GMAN	22.88	**24.02**	**25.96**	**13.80**	**14.62**	**15.72**
SpecSTG	**22.23**	24.77	29.90	14.93	16.70	20.25

Appendix E. Comparison with Classic Deterministic Baselines

Table 3 presents a comparison between SpecSTG and deterministic methods. Notably, SpecSTG outperforms most classic deterministic approaches, including traditional statistical models like VAR and graph neural networks such as DCRNN and ASTGCN. However, GMAN consistently demonstrates superior performance, particularly in long-term forecasting. Here are two plausible explanations. Firstly, being a deterministic model, GMAN is trained with MAE loss, strengthening the generation of accurate deterministic predictions compared to SpecSTG, which optimizes an implicit probabilistic objective. Secondly, GMAN's sophisticated encoder-decoder architecture and multiple spatio-temporal attention blocks effectively integrate spatial and temporal information, while SpecSTG relies on a single graph-modified GRU encoder for processing spatio-temporal patterns. This observation could guide future work to enhance our model.

Appendix F. Supplementary Forecasting Visualizations

In **Figure 7**, we provide more forecasting visualizations of TimeGrad, GCRDD, and SpecSTG. Consistent with our conclusion in Subsection 6.1, SpecSTG generates more compatible forecasting intervals for data with more fluctuations (see **Figure 7 (a)**, **(d)**, **(e)**, and **(f)**). On the contrary, when data variations are small, the method is less effective due to the

[1] Source: https://github.com/zalandoresearch/pytorch-ts

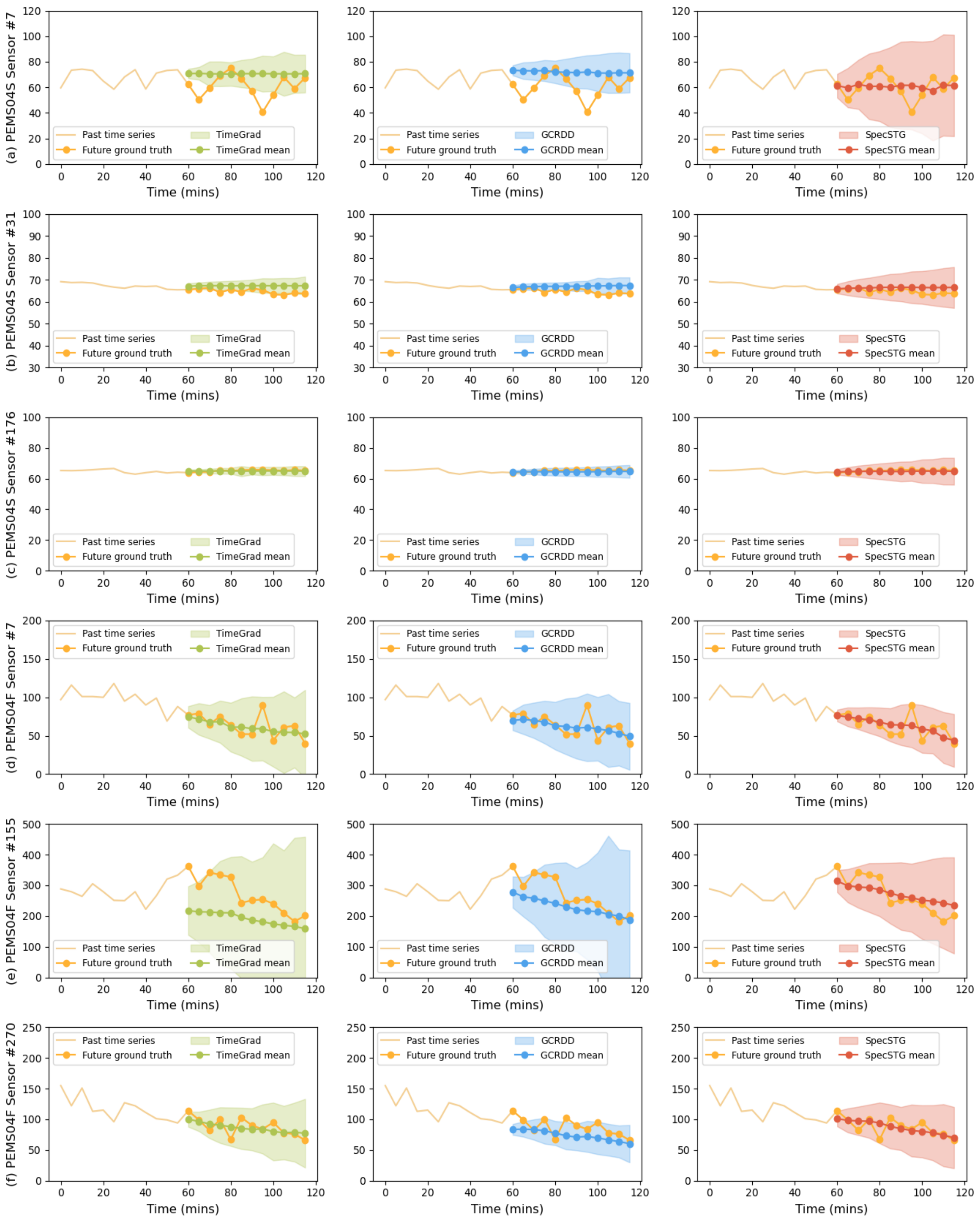

Figure 7: Supplementary forecasting visualizations on traffic speed data ((a), (b), (c)) and traffic flow data ((d), (e), (f)).

lack of systematic information in the Fourier representation (see **Figure 7 (b)** and **(c)**).

Appendix G. Extended Sensitivity Analysis

In **Figure 8**, we show extended sensitivity analysis results for point evaluation at 15/30/60 minutes. The implementation details of this extended sensitivity analysis are consistent with those described in Subsection 6.3. As previously observed, SpecSTG demonstrates better performance with relatively larger values of β_K. The number of diffusion steps K shows less impact on performance compared to β_K. Additionally, the forecasting outcomes exhibit minimal variation across different hyperparameter configurations. This indicates that SpecSTG is not highly sensitive to hyperparameters, thereby simplifying the hyperparameter selection process during training.

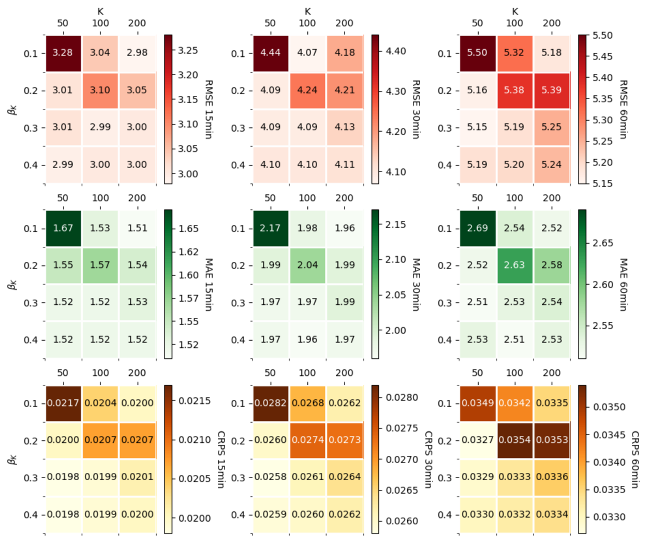

Figure 8: Extended sensitivity analysis on key hyperparameters: diffusion step K and the end of beta schedule β_K (point evaluation results at 15/30/60 minutes).