SPATIOTEMPORAL REPRESENTATION LEARNING ON TIME SERIES WITH DYNAMIC GRAPH ODES

Anonymous authors
Paper under double-blind review

ABSTRACT

Spatiotemporal representation learning on multivariate time series has received tremendous attention in forecasting traffic and energy data. Recent works either rely on complicated discrete neural architectures or graph priors, hindering their effectiveness and applications in the real world. In this paper, inspired by neural ordinary differential equations and graph structure learning, we propose a fully continuous model named Dynamic Graph ODE (`DyG-ODE`) to capture both long-range spatial and temporal dependencies to learn expressive representations on arbitrary multivariate time series data without being restricted by rigid preconditions (e.g., graph priors). For modeling the continuous dynamics of spatiotemporal clues, we design a simple yet powerful dynamic graph ODE by coupling the proposed spatial and temporal ODEs, which not only allows the model to obtain infinite spatial and temporal receptive fields but also reduces numerical errors and model complexity significantly. Our empirical evaluations demonstrate the superior effectiveness and efficiency of `DyG-ODE` on a number of benchmark datasets.

1 INTRODUCTION

Spatiotemporal representation learning is a nontrivial problem in the real world that has been studied in multiple fields, such as sequential recommendation (Zhang et al., 2021; Song et al., 2019), time series forecasting (Wu et al., 2020; Liu et al., 2020), and video representation learning (Hou et al., 2021), where the modeling of temporal and spatial clues are complementary to each other.

In this paper, we focus on learning the spatiotemporal representation of multivariate time series. While earlier methods are based on autoregressive models, recent methods take a deep learning-based approach. In particular, techniques based on graph neural networks (GNNs) (Jiang & Luo, 2021; Skardinga et al., 2021) have demonstrated great potential in simultaneously capturing the temporal and spatial interdependencies among multiple signals over time. For instance, the spatiotemporal graph neural networks (STGNNs) (Yu et al., 2018; Li et al., 2018; Wu et al., 2019b; Zheng et al., 2020), have improved the forecasting precision significantly by leveraging the message passing mechanism on predefined sensor graphs, allowing them to capture not only the temporal but also spatial dependencies among multiple series. However, such topological information is typically static, and may not be available in the real world (Wu et al., 2019b).

To bridge this gap, recent works (Wu et al., 2020; Shang et al., 2020) have explored several ways to address the missing dynamic graph structures. A concrete example is MTGNN (Wu et al., 2020), which proposes a convolution-based STGNN together with a graph construction module, showing competitive performances on multivariate time series forecasting. In many existing spatiotemporal models, including MTGNN, let $\mathbf{X}$ denote the input series, $\mathbf{A}$ represent the predefined or learned graph structure, and $\mathbf{H}_0 = \mathrm{Conv}_{1\times1}(\mathbf{X}, \mathbf{\Theta}_{sc})$ be a transformation of the input feature, these methods can be abstracted as the combination of multiple residual blocks:

$$\begin{cases} \mathbf{H}_{l+1} = \widetilde{\mathbf{H}}_l + \mathrm{GNN}\left(\mathrm{TCN}(\mathbf{H}_l, \mathbf{\Theta}_l), \mathbf{A}, \mathbf{\Phi}_l\right), \\ \mathbf{H}_{out} = \mathbf{H}_L, \end{cases} \tag{1}$$

where $\widetilde{\mathbf{H}}_l$ denotes the truncated representation at the l-th layer, and $\mathbf{H}_{out}$ is the output representation. $\mathrm{TCN}(\cdot, \mathbf{\Theta}_l)$ and $\mathrm{GNN}(\cdot, \mathbf{\Phi}_l)$ are temporal and graph convolution layers at the l-th layer to capture temporal and spatial dependencies, parameterized by $\mathbf{\Theta}_l$ and $\mathbf{\Phi}_l$ respectively.

By iteratively performing spatial and temporal convolutions, the above formulation gives a *discrete* solution of modeling the spatiotemporal information, where a sequence of individually parameterized spatial and temporal transformations are stacked within a model. This leads to several important limitations. **Firstly**, learning the long-term temporal dynamics by combining a sequence of temporal convolutions results in the discontinuous trajectories of latent representations, thus making the model prone to numerical errors. **Secondly**, most of the existing methods, including MTGNN, resort to modeling the discrete dynamics of shallow graph propagation, limiting the model's ability to accurately capture long-range spatial dependencies between time series. **Thirdly**, a layer-wise parametrization in Eq. 1 makes discrete methods not only computational inefficient but also hard to converge. The dedicated designs, such as residual and skip connections, while being important, also lead to complex neural architectures and redundant trainable parameters.

In this paper, we address the above limitations by proposing a new paradigm named `DyG-ODE` to learn the *continuous* spatiotemporal dynamics of multivariate time series. Specifically, inspired by neural ordinary differential equations (NODEs) (Chen et al., 2018), we generalize the discrete formulation in Eq. 1 with an explicit ODE to parameterize the derivation of latent spatial and temporal representations. In such a way, it not only enables the model to characterize the continuous long-range spatiotemporal dynamics with infinite spatial and temporal receptive fields but also simplifies neural architectures and reduces the number of trainable parameters. In practice, we solve a simple initial value problem of ODE to obtain the learned time series representations:

$$\begin{cases} \mathbf{H}_{l+1} = \widetilde{\mathbf{H}}_l + f(\mathbf{H}_l, \mathbf{\Theta}_l, \mathbf{\Phi}_l), \\ \mathbf{H}_{out} = \mathbf{H}_L. \end{cases} \quad \Rightarrow \quad \begin{cases} \dfrac{\mathrm{d}\mathbf{H}(t)}{\mathrm{d}t} = f\big(\mathbf{H}(t), \mathbf{\Theta}, \mathbf{\Phi}\big), \\ \mathbf{H}_{out} = \mathrm{ODESolve}\left(\mathbf{H}(0), f, t_0, t_1, \mathbf{\Theta}, \mathbf{\Phi}\right). \end{cases} \tag{2}$$

In fact, our formulation in Eq. 2 consists of two coupled ODE functions wrapped as $f(\mathbf{H}(t), \mathbf{\Theta}, \mathbf{\Phi})$, defining the continuous exterior temporal aggregation and interior graph propagation processes. To achieve this, we have further answered several unresolved technical questions: (**Q1**) How to characterize a continuous graph propagation process on arbitrary multivariate time series data with dynamically learnable graph structures? (**Q2**) How to model a temporal aggregation process by an ODE with dynamic hidden state dimensions? (**Q3**) How these two processes can be intersected with each other to capture rich spatiotemporal dynamics within a single ODE? By addressing these challenges (in Section 3.1-3.3), our method not only generalizes the existing convolution-based designs to a continuous paradigm but also alleviates the reliance on graph priors, providing a simpler way to model multivariate time series that is both more effective and more efficient, and thus solving the three limitations of existing discrete solutions. We summarize our contributions as follows:

- We propose a novel and potent ODE-based method to learn the spatiotemporal representations on multivariate time series data by unifying two ODEs without complicating the model design and optimization.
- We design a spatial ODE together with a graph learning module to capture the long-range spatial dependencies between time series, which alleviates the over-smoothing problem and the reliance on static graph priors.
- We propose a temporal ODE by generalizing the canonical temporal convolutions to learn the continuous temporal dynamics of time series, which is more powerful and efficient than its discrete counterparts.
- We conduct extensive experiments to demonstrate the effectiveness and efficiency of the proposed method, showing great application prospects in the real world.

2 Related Work

Spatiotemporal Representation Learning. This topic has been explored in multiple domains, such as video representation learning (Hou et al., 2021) in computer vision and multivariate time series modeling in data mining. For the latter one, the conventional approaches mainly rely on convolution-based (Lai et al., 2018; Shih et al., 2019) or attentive (Huang et al., 2019) networks to capture both spatial and temporal correlations. Recently, GNN-based methods have been proposed for traffic forecasting (Yu et al., 2018; Chen et al., 2020; Zheng et al., 2020) on predefined graph structures. As a new family of neural networks to encode both attributive and structural information, different GNNs are proposed to learn on static (Kipf & Welling, 2017; Veličković et al., 2018;

Hamilton et al., 2017) and dynamic graphs (Xu et al., 2020; Rossi et al., 2020; Wang et al., 2020). In traffic forecasting, the concept of spatiotemporal graphs (a special type of dynamic graphs) has been recently applied where node attributes are dynamic while the underlying graph topology (connectivity) is fixed. In this paper, inspired by graph structure learning (Shang et al., 2020; Wu et al., 2020), we model multivariate time series as more general dynamic graphs, in which both node attributes and underlying connectivity evolve over time. This allows us to design a more dedicated model to capture both spatial and temporal information, thereby achieving better performance. On this basis, we further generalize the existing discrete modeling approaches and propose a more promising plug-and-play continuous spatiotemporal model on arbitrary multivariate time series data.

Neural Ordinary Differential Equations (NODEs). Chen et al. (2018) introduced a new paradigm of continuous-time model by generalizing existing discrete deep neural networks. Taking a residual network for example, the basic concept of NODEs is coherent with Eq. 2. Recently, NODEs have been adopted in some research fields, such as reinforcement learning (Du et al., 2020), graph neural networks (Xhonneux et al., 2020; Wang et al., 2021), and traffic forecasting (Fang et al., 2021). Specifically, as the only ODE-based method for spatiotemporal representation learning, STGODE (Fang et al., 2021) merely considers the continuous graph propagation on predefined static graph structures, without modeling the continuous temporal dynamics. Our approach distinguishes from it in two important aspects. Firstly, we couple the continuous temporal aggregation and graph propagation. Secondly, our method eliminates the reliance on predefined graph structures. As a result, our method can be applied to broader applications and gain more competitive performance.

3 METHODOLOGY

In this section, we present the proposed `DyG-ODE` method to learn the continuous spatiotemporal representations of multivariate time series. Our method consists of two main parts named *Continuous Graph Propagation* (CGP) and *Continuous Temporal Aggregation* (CTA), as shown in Figure 1. We also describe the overall architecture together with the model training and highlight the differences from existing discrete approaches by theoretically analyzing the properties of our method.

Problem formulation. Let $\mathbf{X} \in \mathbb{R}^{N\times D\times S}$ denote a multivariate time series with N variables , D feature dimensions, and S time steps in total for training. Specifically, we define $\mathbf{X}^i \in \mathbb{R}^{D\times S}$ as the i-th time series for all features and time steps, and $\mathbf{X}_t \in \mathbb{R}^{N\times D}$ as the t-th time step for all series and features. Given a sequence of T historical observations $\mathbf{X}_{t+1:t+T} \in \mathbb{R}^{N\times D\times T}$, our objective is to learn a spatiotemporal encoder $f(\cdot) : \mathbb{R}^{N\times D\times T} \to \mathbb{R}^{N\times D'}$, where the learned representation $\mathbf{H}_{out} = f(\mathbf{X}_{t+1:t+T})$ can then be used in various downstream tasks, such as single-step and multi-step forecasting of future observations as we demonstrate in Section 4.2. Formally, given a loss function $\ell(\cdot)$ and for each valid time step t, the problem can be formulated as follows:

$$f^*, g^* = \arg\min_{f,g} \sum_t \ell\Big(g\big(f(\mathbf{X}_{t+1:t+T})\big), \mathbf{Y}\Big), \tag{3}$$

where $f^*(\cdot)$ and $g^*(\cdot)$ represent the encoder and decoder with learned optimal parameters. Specifically, we let $\mathbf{Y} = \mathbf{X}_{t+T+H} \in \mathbb{R}^{N\times D\times 1}$ for single-step forecasting, and $\mathbf{Y} = \mathbf{X}_{t+T+1:t+T+H} \in \mathbb{R}^{N\times D\times H}$ for multi-step forecasting. H represents the specific forecasting horizon.

3.1 CONTINUOUS GRAPH PROPAGATION

In `DyG-ODE`, we integrate the temporal aggregation and graph propagation processes to capture rich spatiotemporal patterns from historical observations. At each aggregation step and for simplicity, the spatial dependencies between time series can be characterised by the combination of a feature propagation and a linear transformation on a specific graph snapshot. Specifically, given an adjacency matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$ and an initial state $\mathbf{H}_0^G \in \mathbb{R}^{N\times D'\times Q}$ acquired from the temporal aggregation process, a discrete formulation of the K-hop graph propagation is defined as (Wu et al., 2019a):

$$\begin{cases} \mathbf{H}_{k+1}^G = \widehat{\mathbf{A}}\ \mathbf{H}_k^G,\ k \in \{0, \cdots, K\}, \\ \mathbf{H}_{out}^G = \mathbf{H}_K^G\ \mathbf{\Phi}, \end{cases} \tag{4}$$

where $\widehat{\mathbf{A}}$ denotes the normalized adjacency matrix, $\mathbf{H}_{out}^G \in \mathbb{R}^{N\times D'\times Q}$ is the output representation, and $\mathbf{\Phi} \in \mathbb{R}^{D'\times D'}$ is a trainable parameter matrix. In practice, we define the tensor multiplication

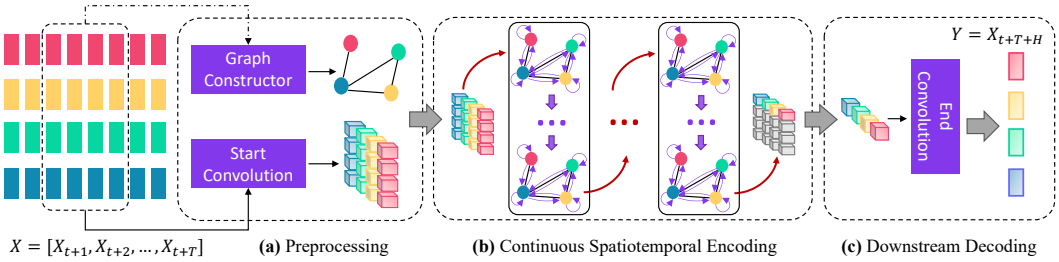

Figure 1: The overall framework of DyG-ODE. Given a sequence of historical observations, we first map them to the latent space and learn an associated graph structure. Then, the continuous dynamics of spatial and temporal clues are modeled by coupling two ODEs from different perspectives. Finally, the learned representations can be used in various tasks, such as the single-step forecasting.

with the Einstein summation in the above equation to sum the element products along specific dimensions. This is because the graph propagation only operates on the first two dimensions of hidden states without aggregating information along the time axis (with sequence length Q).

Compared with GCN (Kipf & Welling, 2017), Eq. 4 eliminates the redundant nonlinearities and further decouples the feature propagation and transformation steps, resulting in a simpler and more efficient model while maintaining comparable accuracy. However, this discrete formulation is error-prone and vulnerable to over-smoothing when performing a deep graph propagation. The underlying cause of these two problems in Eq. 4 can be uncovered by decomposing the propagation depth K into the combination of integration time T_{cgp} and interval Δt_{cgp}, i.e., $K = T_{cgp}/\Delta t_{cgp}$. From the perspective of a continuous process, a selected T_{cgp} and Δt_{cgp} control the number of function evaluations, which is equivalent to describe how many times feature propagation is executed, a.k.a. the propagation depth K in the discrete formulation. Therefore, considering a case where a fixed integration time and smaller intervals are applied, we can naturally have the following transformation with propagation steps $k \in \{0, \cdots, K\}$ being replaced by a continuous variable $t \in \mathbb{R}_0^+$:

$$\begin{aligned} \mathbf{H}^G(t + \Delta t_{cgp}) &= \mathbf{H}^G(t) + \Delta t_{cgp}(\widehat{\mathbf{A}} - \mathbf{I}_N)\mathbf{H}^G(t) \\ &= \left[(1 - \Delta t_{cgp})\mathbf{I}_N + \Delta t_{cgp}\widehat{\mathbf{A}}\right]\mathbf{H}^G(t). \end{aligned} \tag{5}$$

Based on this analysis, we can find that Eq. 4 rigidly couples the propagation depth and integration time by enforcing $\Delta t_{cgp} = 1$ (i.e., the above equation degrades to Eq. 4 when interval $\Delta t_{cgp} = 1$). If so, letting $K = T_{cgp} \to \infty$ not only makes the graph Laplacian eigenvalues in a discrete propagation tend to zeros (Appendix A) but also leads to infinite numerical errors (Appendix B), which prevents the model from accurately capturing long-range spatial dependencies. In this work, inspired by Wang et al. (2021), we disentangle the coupling between K and T_{cgp}, which alleviates the aforementioned problems by avoiding $T_{cgp} \to \infty$. We provide detailed theoretical justifications in Section 3.4. Specifically, in DyG-ODE, we generalize Eq. 4 with its continuous formulation in the following proposition based on Eq. 5, which allows the model to capture fine-grained and long-range spatial dependencies between time series.

Proposition 1. *The continuous dynamics of simplified graph propagation described in Eq. 4 admits the following ODE:*

$$\frac{\mathrm{d}\mathbf{H}^G(t)}{\mathrm{d}t} = (\widehat{\mathbf{A}} - \mathbf{I}_N)\,\mathbf{H}^G(t), \tag{6}$$

with the initial state $\mathbf{H}^G(0) = \mathbf{H}_0^G$, where $\mathbf{H}_0^G$ is an intermediate state of the continuous temporal aggregation process described in Section 3.2.

To further reduce numerical errors, we propose an attentive transformation to replace the linear mapping in Eq. 4, which integrates not only the final but also the initial and some intermediate states as the output of graph propagation:

$$\begin{cases} \mathbf{H}^G(0), \mathbf{H}^G(t_i), \cdots, \mathbf{H}^G(T_{cgp}) = \mathrm{ODESolve}(\mathbf{H}^G(0), \dfrac{\mathrm{d}\mathbf{H}^G(t)}{\mathrm{d}t}, 0, t_i, \cdots, T_{cgp}), \\ \mathbf{H}_{out}^G = \displaystyle\sum_{t_i} \mathbf{H}^G(t_i)\,\mathbf{\Phi}_{\mathbf{t_i}}, \quad t_i \in [0, T_{cgp}], \end{cases} \tag{7}$$

where $\mathbf{H}^G(t_i)$ denotes the selected intermediate states, and we only take t_i that is divisible by Δt_{cgp} for simplicity in practice.

Dynamic graph structure learning. In Eq. 6, it remains unknown how the graph adjacency matrix $\mathbf{A}$ is constructed. To address the first technical question (**Q1**) and handle time series without graph priors (e.g., unknown $\mathbf{A}$), we adopt a direct optimization approach to learn dynamic graph structures together with the entire model, where node connections evolve with model training. Specifically, for a sequence of historical observations, the underlying adjacency matrix $\mathbf{A}$ is dynamically optimized as training progresses, which is precisely defined in Appendix C.

3.2 Continuous Temporal Aggregation

Solving the spatial ODE described in Eq. 7 only allows the model to capture the spatial dependencies between time series at a certain time step. To complete the missing temporal information, we couple it with an exterior temporal ODE, which allows `DyG-ODE` to learn the continuous dynamics of multivariate time series from both spatial and temporal perspectives.

We first introduce the composition of temporal ODE to characterize the long-term temporal dependencies. In view of the shortcomings of recurrent neural networks (RNNs), such as time-consuming iteration and gradient explosion (Wu et al., 2019b), we resort to stacking multiple residual temporal convolution blocks to capture and aggregate temporal patterns in a non-recursive manner:

$$\mathbf{H}_{l+1}^T = \mathcal{T}(\mathbf{H}_l^T, Q_{l+1}) + \mathrm{TCN}(\mathbf{H}_l^T, \mathbf{\Theta}_l),\ l \in \{0, \cdots, L\}, \tag{8}$$

where $\mathrm{TCN}(\cdot, \mathbf{\Theta}_l)$ is an individually parameterized temporal convolution, $\mathcal{T}(\cdot)$ denotes the truncate function, and $\mathbf{H}_l^T \in \mathbb{R}^{N \times D' \times Q_l}$ is the output of the l-th layer with sequence length Q_l. In this formulation, the last dimension of the residual input $\mathbf{H}_l^T$ has to be truncated to Q_{l+1} before adding to its transformation because the length of latent representations shrink gradually after each aggregation step, i.e., $Q_{l+1} = Q_l - r^l \times (k-1)$ and $Q_1 = R - k + 1$. Specifically, we define $\mathbf{H}_0^T \in \mathbb{R}^{N \times D' \times R}$ as the initial state, r, k and R are dilation factor, kernel size, and model receptive field. In practice, we assure $R > T$ to losslessly encode all historical observations, where $R = L(k-1) + 1$ when $r = 1$, and $R = 1 + (k-1)(r^L - 1)/(r-1)$ when $r > 1$.

However, the discrete formulation in Eq. 8 suffers from two main limitations: (1) It fails to learn the fine-grained and accurate long-term temporal dynamics with a fixed large integration interval (i.e., $\Delta t_{cta} = 1$), which breaks the continuity of the latent representation trajectories; (2) It has a large number of trainable parameters and requires additional designs to avoid the gradient vanishing issue and ensure model convergence, resulting in high computational overhead. This also hinders building a deep network to obtain a large receptive field without losing fine-grained information. Thus, we apply a similar idea to disentangle the coupling between the aggregation depth L and the integration time T_{cta} by letting $L = T_{cta}/\Delta t_{cta}$. As such, given a desired terminate time $T_{cta} = t_1$ and initial state $\mathbf{H}_0^T$, we characterize the entire continuous temporal aggregation process with a single set of parameters $\mathbf{\Theta}$ by making $\Delta t_{cta} \to 0$, which is more efficient and allows the model to obtain a theoretically infinite receptive field by constructing a "deeper" network:

$$\begin{cases} \mathbf{H}_{t_1}^T = \mathbf{H}_0^T + \displaystyle\int_{t_0}^{t_1} \mathcal{P}\big(\mathrm{TCN}(\mathbf{H}_t^T, \mathbf{\Theta}), R\big)\,\mathrm{d}t, \\ \mathbf{H}_{out}^T = \mathbf{H}_{t_1}^T[\cdots, -1]. \end{cases} \tag{9}$$

To achieve this and address the second technical question (**Q2**), we design a simple zero-padding trick to ensure the invariance of hidden state dimensions, where the length of latent representations is padded to R with a padding function $\mathcal{P}(\cdot)$ after each aggregation step. In such a way, we have the second proposition defined as follows:

Proposition 2 *The temporal aggregation process described in Eq. 8 is a discretization of the following ODE:*

$$\frac{\mathrm{d}\mathbf{H}^T(t)}{\mathrm{d}t} = \mathcal{P}\big(\mathrm{TCN}(\mathbf{H}^T(t), t, \mathbf{\Theta}), R\big), \tag{10}$$

with the initial state $\mathbf{H}^T(0) = \mathbf{H}_0^T$, which is obtained by mapping the input signals to the latent space with a separate convolution layer parameterized by $\mathbf{\Gamma}_{sc}$, i.e., $\mathbf{H}_0^T = \mathrm{Conv}_{1\times1}(\mathbf{X}, \mathbf{\Gamma}_{sc})$.

We illustrate the detailed design of $\mathrm{TCN}(\cdot, \mathbf{\Theta})$ in Appendix D.

3.3 Overall Architecture and Model Training

Overall architecture. We have the proposed DyG−ODE method defined below by unifying the aforementioned two ODEs. To address the third technical question (**Q3**), it is worth noting that instead of simply concatenate them end-to-end, we take each intermediate state of the temporal ODE as the initial state of the spatial ODE, thereby allowing the model to intersect individual continuous graph propagation and temporal aggregation processes simultaneously. Given two black-box ODE solvers, the learned spatiotemporal representations can be obtained by integrating the following ODE:

$$\mathbf{H}_{out} = \text{ODESolve}^1\left(\mathbf{H}(0), \frac{\mathrm{d}\mathbf{H}(t)}{\mathrm{d}t}, T_{cta}\right), \tag{11}$$

$$\frac{\mathrm{d}\mathbf{H}(t)}{\mathrm{d}t} = \mathcal{P}\Bigg(\mathcal{A}\Big(\text{ODESolve}^2\left(\text{TCN}(\mathbf{H}(t), t, \boldsymbol{\Theta}), \frac{\mathrm{d}\mathbf{H}^G(\tau)}{\mathrm{d}\tau}, 0, \tau_i, \cdots, T_{cgp}\right), \boldsymbol{\Phi}\Big), R\Bigg). \tag{12}$$

In the above equations, the interior ODE solving and the attentive transformation, i.e., $\mathcal{A}(\cdot, \boldsymbol{\Phi})$, are given by Eq. 7 by letting $\mathbf{H}^G(0) = \text{TCN}(\mathbf{H}(t), t, \boldsymbol{\Theta})$. In particular, we let the initial state $\mathbf{H}(0) = \mathbf{H}^T(0)$ in Eq. 10, and further define ODESolve^1 and ODESolve^2 as the Euler or Runge-Kutta 3/8 method with different integration time and intervals for simplicity.

Model training. The overall framework of DyG−ODE is shown in Figure 1. Given a sequence of historical observations $\mathbf{X}_{t+1:t+T}$, we first learn its representation $\mathbf{H}_{out} \in \mathbb{R}^{N \times D'}$ and then make predictions based on it with the decoder $g(\cdot, \boldsymbol{\Gamma}_{dc})$. Thus, our training objective described in Eq. 3 can be reformulated as follows:

$$f^*, g^* = \arg\min_{f,g} \sum_t \ell\Big(g\big(f(\mathbf{X}_{t+1:t+T}; \boldsymbol{\Gamma}_{sc}, \boldsymbol{\Theta}, \boldsymbol{\Gamma}_{gc}, \boldsymbol{\Phi}), \boldsymbol{\Gamma}_{dc}\big), \mathbf{Y}\Big), \tag{13}$$

where $\ell(\cdot)$ denotes the mean absolute error (MAE). For simplicity, we optimize model parameters via the standard reverse-mode differentiation, which can be replaced by the adjoint sensitivity method as suggested by Chen et al. (2018) for a better memory efficiency. The algorithm of DyG−ODE is in Appendix I.

3.4 Comparison with Discrete Variants

Compared with the existing GNN-based methods (Yu et al., 2018; Wu et al., 2019b; 2020), our approach is free from the over-smoothing issue (e.g., the model performance drops when the number of layers increases). This allows our model to capture long-range spatial dependencies by disengaging the tie between the graph propagation depth and integration time. Thus, our method can perform a deeper feature propagation and obtain a larger spatial receptive field by aggregating farther neighboring information. Formally, our method possesses the following properties:

Property 1 *The discrete formulation in Eq. 4 is subject to over-smoothing and thus hinders the model from capturing long-range spatial dependencies by performing deep graph propagation. In contrast, for a fixed integration time T_{cgp}, DyG−ODE ensures the convergence of learned spatial representations by defining $K = T_{cgp}/\Delta t_{cgp}$ and letting $K \to \infty$.*

Proof. See Appendix A.

Property 2 *Increasing the propagation depth in Eq. 4 leads to large numerical errors. In DyG−ODE, letting $K = T_{cgp}/\Delta t_{cgp} \to \infty$ makes the numerical errors approaching to zero with a fixed integration time T_{cgp}.*

Proof. See Appendix B.

The above properties are further empirically validated in Figure 2 in Section 4.2. Also, in the proposed temporal ODE, we parameterize the derivation of latent representations to characterize the nature of a temporal aggregation process. As such, we break the tie between the aggregation depth and the integration time, which allows constructing an infinitely deep temporal aggregation with a single set of parameters. Thus, DyG−ODE is not only simpler and easier to converge but also more efficient and effective to model multivariate time series compared with the existing discrete methods.

4 EXPERIMENTS

In this section, we conduct extensive experiments on five benchmark datasets to show the performance of DyG−ODE. We also demonstrate the potency and efficiency of the two proposed continuous regimes, showing superior properties compared with common discrete neural architectures.

4.1 EXPERIMENT SETUP

Datasets. We experiment on five benchmark datasets, three of which are conventional time series datasets (Lai et al., 2018), i.e., Electricity, Solar-Energy, and Traffic, without predefined graph structures, and the rest two are traffic datasets (Li et al., 2018), i.e., METR-LA and PEMS-BAY, with predefined sensor maps. We summarize the dataset statistics and specific splits in Appendix E.

Baselines. We evaluate and compare DyG−ODE with representative and state-of-the-art time series baselines, including LSTNet (Lai et al., 2018), MTGNN (Wu et al., 2020), and HyDCNN (Li et al., 2021) on three time series datasets for *single-step forecasting*. We further compare our algorithm with strong GNN-based baselines on two traffic datasets for *multi-step forecasting*, including DCRNN (Li et al., 2018), STGCN (Yu et al., 2018), Graph WaveNet (Wu et al., 2019b), MRA-BCGN (Chen et al., 2020), GMAN (Zheng et al., 2020) and STGODE (Fang et al., 2021). Note that MTGNN is an algorithm which does not rely on a predefined graph structure, thus it is applicable and compared in both single/multi-step forecasting settings. See Appendix E for more details.

Evaluation metrics and parameter settings. Following Wu et al. (2020), we adopt Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and Mean Absolute Percentage Error (MAPE) as our evaluation metrics for multi-step forecasting; we choose Root Relative Squared Error (RSE) and Empirical Correlation Coefficient (CORR) as the evaluation metrics for single-step forecasting, where better performance is indicated by higher CORR and lower RSE values. All experiments are independently repeated five times and the averaged performances are reported. For more details on the hyperparameter settings, see Appendix E.

4.2 RESULTS

Table 1: Single-step forecasting results on three benchmark time series datasets. **Red** denotes the best performances, and underline highlights the second best results.

Dataset			Metric	AR	VARMLP	GRU	LSTNet	TPA-LSTM	MTGNN	HyDCNN	DyG-ODE
Electricity	Horizon	3	RSE↓	0.0995	0.1393	0.1102	0.0864	0.0823	0.0745	0.0832	**0.0736**
			CORR↑	0.8845	0.8708	0.8597	0.9283	0.9439	**0.9474**	0.9354	0.9430
		6	RSE↓	0.1035	0.1620	0.1144	0.0931	0.0916	0.0878	0.0898	**0.0809**
			CORR↑	0.8632	0.8389	0.8623	0.9135	0.9337	0.9316	0.9329	**0.9340**
		12	RSE↓	0.1050	0.1557	0.1183	0.1007	0.0964	0.0916	0.0921	**0.0891**
			CORR↑	0.8591	0.8192	0.8472	0.9077	0.9250	0.9278	**0.9285**	0.9279
Traffic	Horizon	3	RSE↓	0.5991	0.5582	0.5358	0.4777	0.4487	0.4162	0.4198	**0.4127**
			CORR↑	0.7752	0.8245	0.8511	0.8721	0.8812	0.8963	0.8915	**0.9020**
		6	RSE↓	0.6218	0.6579	0.5522	0.4893	0.4658	0.4754	0.4290	**0.4259**
			CORR↑	0.7568	0.7695	0.8405	0.8690	0.8717	0.8667	0.8855	**0.8945**
		12	RSE↓	0.6252	0.6023	0.5562	0.4950	0.4641	0.4461	0.4352	**0.4329**
			CORR↑	0.7544	0.7929	0.8345	0.8614	0.8717	0.8794	0.8858	**0.8899**
Solar	Horizon	3	RSE↓	0.2435	0.1922	0.1932	0.1843	0.1803	0.1778	0.1806	**0.1693**
			CORR↑	0.9710	0.9829	0.9823	0.9843	0.9850	0.9852	0.9865	**0.9868**
		6	RSE↓	0.3790	0.2679	0.2628	0.2559	0.2347	0.2348	0.2335	**0.2171**
			CORR↑	0.9263	0.9655	0.9675	0.9690	0.9742	0.9726	0.9747	**0.9771**
		12	RSE↓	0.5911	0.4244	0.4163	0.3254	0.3234	0.3109	0.3094	**0.2901**
			CORR↑	0.8107	0.9058	0.9150	0.9467	0.9487	0.9509	0.9515	**0.9577**

Overall comparisons. We first report the results of different algorithms on different horizons for single-step forecasting in Table 1. We have two important observations: In general, our method achieves the best performance on three time series datasets, even when compared with HyDCNN, indicating the effectiveness in modeling multivariate time series with dynamic graph neural ODEs; (2) Our method significantly surpasses MTGNN in most cases with the same graph constructor, especially for long-term forecasting (i.e., horizon 6 and 12), demonstrating the superiority of our continuous regimes in capturing fine-grained and long-range spatial and temporal dependencies.

Table 2: Results of multi-step forecasting on two benchmark traffic datasets, where **red** and underline denote the best and the second best results.

Dataset		Metric	DCRNN	STGCN	GraphWaveNet	GMAN	MRA-BCGN	MTGNN	STGODE	DyG-ODE
METR-LA	15 min	MAE	2.77	2.88	2.69	2.77	2.67	2.69	3.47	**2.66**
		RMSE	5.38	5.74	5.15	5.48	5.12	5.18	6.76	**5.10**
		MAPE (%)	7.30	7.62	6.90	7.25	**6.80**	6.90	8.76	6.87
	30 min	MAE	3.15	3.47	3.07	3.07	3.06	3.05	4.36	**3.00**
		RMSE	6.45	7.24	6.22	6.34	6.17	6.18	8.47	**6.05**
		MAPE (%)	8.80	9.57	8.37	8.35	8.30	8.21	11.14	**8.19**
	60 min	MAE	3.60	4.59	3.53	3.40	3.49	3.50	5.50	**3.39**
		RMSE	7.60	9.40	7.37	7.21	7.30	7.25	10.33	**7.05**
		MAPE (%)	10.5	12.7	10.01	**9.72**	10.00	9.90	14.32	9.80
PEMS-BAY	15 min	MAE	1.38	1.36	1.30	1.34	**1.29**	1.34	1.43	1.31
		RMSE	2.95	2.96	2.74	2.82	**2.72**	2.81	2.88	2.76
		MAPE (%)	2.90	2.90	**2.73**	2.81	2.90	2.82	2.99	2.75
	30 min	MAE	1.74	1.81	1.63	1.62	1.61	1.66	1.84	**1.61**
		RMSE	3.97	4.27	3.70	3.72	3.67	3.74	3.90	**3.66**
		MAPE (%)	3.90	4.17	3.67	3.63	3.80	3.72	3.84	**3.62**
	60 min	MAE	2.07	2.49	1.95	**1.86**	1.91	1.94	2.30	1.88
		RMSE	4.74	5.69	4.52	4.32	4.46	4.48	4.89	**4.31**
		MAPE (%)	4.90	5.79	4.63	**4.31**	4.60	4.58	4.61	4.39

To further demonstrate the advantage of DyG-ODE, we compare it with competitive GNN-based methods on two benchmark traffic datasets under the setting of multi-step forecasting, where all baselines use predefined graph structures only except for MTGNN and our method. We summarize the results in Table 2, from which we have the following observations: (1) Similar to single-step forecasting, our method consistently outperforms MTGNN under this setting with the same graph constructor, which further confirms the effectiveness of DyG-ODE in modeling multivariate time series data; (2) Our method demonstrates better performance compared with STGODE. We conjecture that it is attributed to two reasons. Firstly, the proposed temporal ODE enables our approach to capture better fine-grained and long-term temporal dynamics. Secondly, our graph module is not only more expressive with the attentive transformation but also free from graph priors thus more robust to dataset biases. (3) DyG-ODE surpasses DCRNN, STGCN, and Graph WaveNet significantly without relying on graph priors. When compared to MRA-BCGN and GMAN, our method achieves the best or on-par performance, demonstrating its competitiveness.

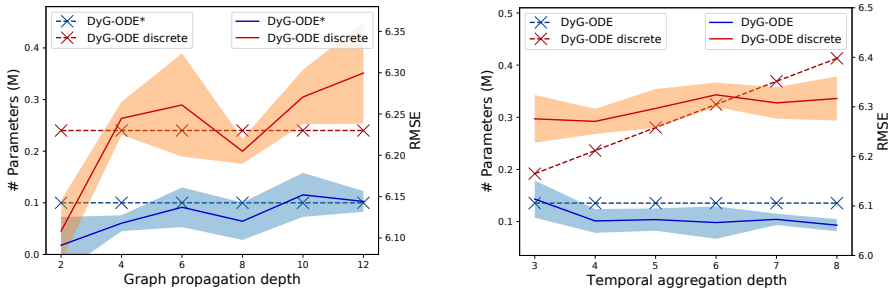

Figure 2: Model parameters and averaged performances w.r.t. graph propagation and temporal aggregation depths on the METR-LA dataset. DyG-ODE* is a variant of our method with the attentive transformation disabled in Eq. 7. DyG-ODE discrete denotes the discrete variant of our method by combining Eq. 4 and the padding version of Eq. 8.

Case studies and parameter studies. To further evaluate the forecasting quality, we provide two case studies in Appendix F to visually compare our method with some discrete baselines. Besides, we also provide detailed parameter studies in Appendix H together with Figure 2 to assess our method from various perspectives.

Investigating the two continuous regimes. To empirically validate our justifications in Section 3.4 and study the behavioral differences between our method and its discrete variant, we dissect DyG-ODE by comparing the model performance and the number of parameters with different graph propagation and temporal aggregation depths. Firstly, the left chart in Figure 2 compares the proposed continuous graph propagation with its discrete implementation (Eq. 4). In particular, we disable the attentive transformation in this experiment to expose the essence of our proposed spatial

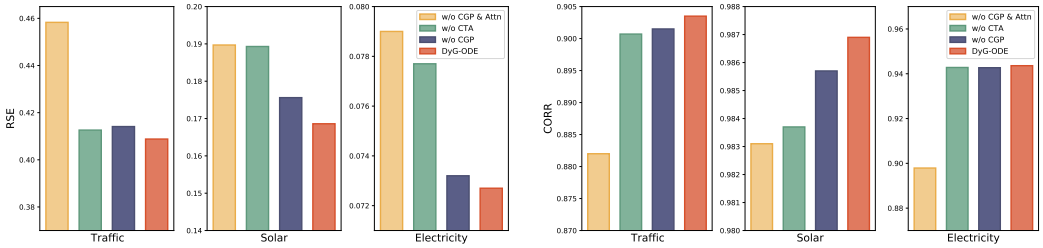

Figure 3: The ablation study on three time series datasets. We replace the continuous temporal aggregation and graph propagation in `DyG-ODE` with their discrete implementations, denoted as `w/o CTA` and `w/o CGP`. We further remove the attentive and continuous regime in CGP to construct the `w/o CGP & Attn`. A lower RSE and a higher CORR are expected.

ODE in Eq. 6. Compared with `DyG-ODE discrete` (solid red curve), our method (solid blue line) is more robust (in terms of RMSE) to the over-smoothing problem with increased propagation depths, where the gradually flattened performance curve and shrunk standard deviations indicate that our method allows the learned spatial representations to converge to a sweet spot by exploiting the long-range spatial dependencies, bringing significantly lower numerical errors (in terms of RMSE) and better stability (w.r.t. standard deviations). Specifically, in our method, we find that larger propagation depths benefit to learn more stable representations with the on-par performance on downstream tasks compared with the shallow propagation, e.g., the MAE and RMSE for propagation depth 2 and 8 are $3.038 \pm \mathbf{0.0131}$ vs. $3.037 \pm \mathbf{0.0086}$ and $6.091 \pm \mathbf{0.0342}$ vs. $6.120 \pm \mathbf{0.0227}$. At the same time, our method also demonstrates a better parameter efficiency (in terms of # of parameters) compared with `DyG-ODE discrete`. It is worth noting that in this experiment, `DyG-ODE`* and `DyG-ODE discrete` have constant parameters as feature propagation is parameterless.

The second chart in Figure 2 compares our method with its discrete variant by varying the temporal aggregation depth. We can observe that with the increase in aggregation depth, `DyG-ODE` converges with decreased numerical errors. We can also observe that for `DyG-ODE discrete`, an increase in aggregation depth requires an increase in model parameters, hence its complexity. In contrast, `DyG-ODE` breaks this tie and thus allows a deeper and easier-optimized temporal aggregation to model the long-term and fine-grained continuous temporal dynamics of time series. In Appendix G, we further demonstrate that `DyG-ODE` is also more **computational efficient** than discrete methods.

Ablation study. We construct three variants of our method to study the effectiveness of core components. Specifically, `DyG-ODE w/o CTA` and `DyG-ODE w/o CGP` replace the temporal and spatial ODEs with their discrete implementations to study the potency of two continuous regimes. `DyG-ODE w/o CGP & Attn` further removes the attentive transformation in `DyG-ODE w/o CGP` to investigate the effectiveness of graph attentive transformation. In particular, the `DyG-ODE discrete` in Figure 2 is equivalent to `DyG-ODE w/o CTA & CGP & Attn`, which has been investigated before so we omit this variant in ablation study. The experimental results are in Figure 3, where our method equipped with all components has the best performance across all datasets. In particular, we observe that our spatial ODE with the attentive transformation benefits the model best to learn effective representations. Besides, even though we adopt a relatively shallow propagation depth (e.g., two steps on the Electricity and Traffic datasets, and four steps on the Solar dataset) in ablation studies for fair comparisons with semi-discrete variants, the performance gains obtained by the CGP itself are also notable in most cases. A similar observation can also be made for CTA, where replacing it with (the padding version of) Eq. 8 degrades the performance sharply.

5 Conclusion

In view of the shortcomings of existing discrete spatiotemporal modeling approaches, we investigate the use of neural ordinary differential equations and dynamic graph structure learning to model the continuous spatiotemporal dynamics of multivariate time series by intersecting the continuous graph propagation and temporal aggregation processes characterized by two ODEs, which allows the model to obtain more expressive representations without relying on graph priors, showing strong potential in real-world applications in which such information is not available. We also theoretically analyze the main properties of the proposed method and further demonstrate that it is not only more effective but also more efficient compared with the existing discrete methods.

REFERENCES

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 6572–6583, 2018.

Weiqi Chen, Ling Chen, Yu Xie, Wei Cao, Yusong Gao, and Xiaojie Feng. Multi-range attentive bicomponent graph convolutional network for traffic forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 3529–3536, 2020.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*, 2014.

Fan RK Chung and Fan Chung Graham. *Spectral graph theory*. Number 92. American Mathematical Soc., 1997.

Jianzhun Du, Joseph Futoma, and Finale Doshi-Velez. Model-based reinforcement learning for semi-markov decision processes with neural odes. *arXiv preprint arXiv:2006.16210*, 2020.

Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented neural odes. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pp. 3140–3150, 2019.

Zheng Fang, Qingqing Long, Guojie Song, and Kunqing Xie. Spatial-temporal graph ode networks for traffic flow forecasting. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 364–373, 2021.

William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 1025–1035, 2017.

Ruibing Hou, Hong Chang, Bingpeng Ma, Rui Huang, and Shiguang Shan. Bicnet-tks: Learning efficient spatial-temporal representation for video person re-identification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2014–2023, 2021.

Siteng Huang, Donglin Wang, Xuehan Wu, and Ao Tang. Dsanet: Dual self-attention network for multivariate time series forecasting. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pp. 2129–2132, 2019.

Weiwei Jiang and Jiayun Luo. Graph neural network for traffic forecasting: A survey. *arXiv preprint arXiv:2101.11174*, 2021.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

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

Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations*, 2018.

Yangfan Li, Kenli Li, Cen Chen, Xu Zhou, Zeng Zeng, and Keqin Li. Modeling temporal patterns with dilated convolutions for time-series forecasting. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 16(1):1–22, 2021.

Lingbo Liu, Jiajie Zhen, Guanbin Li, Geng Zhan, Zhaocheng He, Bowen Du, and Liang Lin. Dynamic spatial-temporal representation learning for traffic flow prediction. *IEEE Transactions on Intelligent Transportation Systems*, 2020.

Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal graph networks for deep learning on dynamic graphs. In *ICML 2020 Workshop on Graph Representation Learning*, 2020.

Chao Shang, Jie Chen, and Jinbo Bi. Discrete graph structure learning for forecasting multiple time series. In *International Conference on Learning Representations*, 2020.

Shun-Yao Shih, Fan-Keng Sun, and Hung-yi Lee. Temporal pattern attention for multivariate time series forecasting. *Machine Learning*, 108(8):1421–1441, 2019.

Joakim Skardinga, Bogdan Gabrys, and Katarzyna Musial. Foundations and modelling of dynamic networks using dynamic graph neural networks: A survey. *IEEE Access*, 2021.

Weiping Song, Zhiping Xiao, Yifan Wang, Laurent Charlin, Ming Zhang, and Jian Tang. Session-based social recommendation via dynamic graph attention networks. In *Proceedings of the Twelfth ACM international conference on web search and data mining*, pp. 555–563, 2019.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Yanbang Wang, Yen-Yu Chang, Yunyu Liu, Jure Leskovec, and Pan Li. Inductive representation learning in temporal networks via causal anonymous walks. In *International Conference on Learning Representations*, 2020.

Yifei Wang, Yisen Wang, Jiansheng Yang, and Zhouchen Lin. Dissecting the diffusion process in linear graph convolutional networks. *arXiv preprint arXiv:2102.10739*, 2021.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pp. 6861–6871. PMLR, 2019a.

Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. Graph wavenet for deep spatial-temporal graph modeling. In *The 28th International Joint Conference on Artificial Intelligence (IJCAI)*, 2019b.

Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. 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*, pp. 753–763, 2020.

Louis-Pascal Xhonneux, Meng Qu, and Jian Tang. Continuous graph neural networks. In *International Conference on Machine Learning*, pp. 10432–10441. PMLR, 2020.

Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Inductive representation learning on temporal graphs. *arXiv preprint arXiv:2002.07962*, 2020.

Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *IJCAI*, 2018.

G Peter Zhang. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50:159–175, 2003.

Mengqi Zhang, Shu Wu, Xueli Yu, and Liang Wang. Dynamic graph neural networks for sequential recommendation. *arXiv preprint arXiv:2104.07368*, 2021.

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

Appendices

A PROOF OF PROPERTY 1

Given a simplified graph feature propagation in Eq. 4, we give a proof that it is characterized by the following ODE.

$$\mathbf{H}^G_{k+1} = \widehat{\mathbf{A}}\,\mathbf{H}^G_k,\ k \in \{0, \cdots, K\} \quad \Rightarrow \quad \frac{\mathrm{d}\mathbf{H}^G(t)}{\mathrm{d}t} = (\widehat{\mathbf{A}} - \mathbf{I}_N)\,\mathbf{H}^G(t), \\ = -\mathbf{L}\mathbf{H}^G(t),\ t \in \mathbb{R}^+_0,$$

where $\mathbf{L} = \mathbf{I}_N - \widehat{\mathbf{A}}$ denotes the normalized graph Laplacian. Regarding the above ODE, it can be naturally viewed as a general graph heat diffusion process with the Laplacian $\mathbf{L}$ (Wang et al., 2021; Chung & Graham, 1997), where the closed-form solution is:

$$\frac{\mathrm{d}\mathbf{H}^G(t)}{\mathrm{d}t} = -\mathbf{L}\mathbf{H}^G(t) \quad \Rightarrow \quad \mathbf{H}^G(t) = e^{-t\mathbf{L}}\mathbf{H}^G(0). \tag{14}$$

In the above equation, $e^{-t\mathbf{L}}$ is known as the heat kernel. For the graph Laplacian $\mathbf{L} = \mathbf{I}_N - \widehat{\mathbf{A}}$, if $\widehat{\mathbf{A}}$ is symmetrically normalized, we have $\mathbf{L} = \mathbf{I}_N - \widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}} = \widetilde{\mathbf{D}}^{-\frac{1}{2}}(\widetilde{\mathbf{D}} - \widetilde{\mathbf{A}})\widetilde{\mathbf{D}}^{-\frac{1}{2}}$, which is symmetric and positive semi-definite. Thus, the eigen-decomposition of $\mathbf{L}$ can be defined as follows:

$$\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top, \tag{15}$$

where $\mathbf{U}$ is an orthogonal matrix of eigenvectors, and $\mathbf{\Lambda}$ is a diagonal matrix that consists of eigenvalues $\lambda_i \geq 0$. Based on this, the heat kernel can be decomposed as follows based on the Taylor expansion:

$$e^{-t\mathbf{L}} = \sum_{k=0}^{\infty}\frac{1}{k!}(-t\mathbf{L})^k = \sum_{k=0}^{\infty}\frac{t^k}{k!}\big[\mathbf{U}(-\mathbf{\Lambda})\mathbf{U}^\top\big]^k = \mathbf{U}\big[\sum_{k=0}^{\infty}\frac{t^k}{k!}(-\mathbf{\Lambda})^k\big]\mathbf{U}^\top = \mathbf{U}e^{-t\mathbf{\Lambda}}\mathbf{U}^\top. \tag{16}$$

Thus, the eigen-decomposition of the heat kernel can be easily obtained:

$$e^{-t\mathbf{L}} = \mathbf{U}\begin{pmatrix} e^{-t\lambda_1} & 0 & \cdots & 0 \\ 0 & e^{-t\lambda_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & e^{-t\lambda_N} \end{pmatrix}\mathbf{U}^\top, \tag{17}$$

where for each of eigenvalues $e^{-t\lambda_i}$, they satisfy the following property when $t \to \infty$:

$$\lim_{t\to\infty} e^{-t\lambda_i} = \begin{cases} 0, & \text{if } \lambda_i > 0 \\ 1, & \text{if } \lambda_i = 0. \end{cases} \tag{18}$$

As such, given a graph that $\mathbf{A} \neq \mathbf{I}_N$, increasing the propagation depth in Eq. 4 will inevitably lead to the over-smoothing problem, where the eigenvalues are zeroed with $K = T \to \infty$:

$$\lim_{T\to\infty}\mathbf{H}^G(t) = e^{-T\mathbf{L}}\mathbf{H}^G(0) = \mathbf{0}. \tag{19}$$

Conversely, in the proposed spatial ODE, we disengage the coupling between the propagation depth K and terminal (integration) time T_{cgp} by making $K = T_{cgp}/\Delta t_{cgp}$. Thus, it is possible to increase the propagation depth without letting $T_{cgp} \to \infty$, which ensures the convergence of the learned spatial representations.

B PROOF OF PROPERTY 2

Similar to the proof in Appendix A, given a terminal (integration) time T_{cgp}, the proposed spatial ODE can be viewed as a general graph heat diffusion process with the Laplacian $\mathbf{L}$, where the closed-form solution is given by:

$$\mathbf{H}^G(T_{cgp}) = e^{-T_{cgp}\mathbf{L}}\,\mathbf{H}^G(0). \tag{20}$$

For the heat kernel $e^{-T_{cgp}\mathbf{L}}$, its can be expanded in a Taylor series:

$$e^{-T_{cgp}\mathbf{L}} = \sum_{k=0}^{\infty} \frac{T_{cgp}^k}{k!}(-\mathbf{L})^k. \tag{21}$$

Accordingly, Eq. 20 can be reformulated as follows:

$$\mathbf{H}^G(T_{cgp}) = \Big[\sum_{k=0}^{\infty} \frac{T_{cgp}^k}{k!}(-\mathbf{L})^k\Big]\,\mathbf{H}^G(0). \tag{22}$$

Considering an Euler solver is applied, the numerical solution of the above equation after K propagation steps is:

$$\widehat{\mathbf{H}}^G(T_{cgp}) = (\mathbf{I}_N - \frac{T_{cgp}}{K}\mathbf{L})^k\,\mathbf{H}^G(0). \tag{23}$$

Thus, the numerical errors between the analytical and solved numerical solutions (i.e., Eq. 22 and 23) can be simply defined as follows:

$$\mathbf{E}_{T_{cgp}}^{(K)} = \mathbf{H}^G(T_{cgp}) - \widehat{\mathbf{H}}^G(T_{cgp}). \tag{24}$$

According to Wang et al. (2021), we have $\mathbf{E}_{T_{cgp}}^{(K)}$ to be upper bounded by the following inequation:

$$\Big\|\mathbf{E}_{T_{cgp}}^{(K)}\Big\| = \frac{T_{cgp}\,\|\mathbf{L}\|\,\big\|\mathbf{H}^G(0)\big\|}{2K}(e^{T_{cgp}\|\mathbf{L}\|} - 1) \tag{25}$$

Thus, for a fixed terminal time T_{cgp}, we can easily find that $\mathbf{E}_{T_{cgp}}^{(K)} \to 0$ by letting the propagation depth $K \to \infty$. Conversely, let $T = K \to \infty$ in Eq. 4 will lead $\mathbf{E}_{T_{cgp}}^{(K)} \to \infty$.

C ADDITIONAL DETAILS OF GRAPH STRUCTURE LEARNING

The adjacency matrix $\mathbf{A}$ used in Eq. 6 is dynamically optimized as training evolves, which is given by:

$$\begin{cases} \mathbf{M}^k = \tanh(\beta\,\mathbf{E}^k\mathbf{\Gamma}_{gc}^k),\ k \in \{1,2\}, \\ \mathbf{A}_{ij} = \mathrm{ReLU}\Big(\tanh\big(\beta(\mathbf{M}_{ij}^1\mathbf{M}_{ij}^{2\,\mathsf{T}} - \mathbf{M}_{ij}^2\mathbf{M}_{ij}^{1\,\mathsf{T}})\big)\Big), \end{cases} \tag{26}$$

where $\mathbf{M}^1, \mathbf{M}^2 \in \mathbb{R}^{N\times d}$ are characterised by two neural networks with randomly initialized embedding matrices $\mathbf{E}^1, \mathbf{E}^2 \in \mathbb{R}^{N\times d}$ and trainable parameters $\mathbf{\Gamma}_{gc}^1, \mathbf{\Gamma}_{gc}^2 \in \mathbb{R}^{d\times d}$. β is a hyperparameter to adjust the saturation rate of activation functions. In particular, the learned graph structure is made sparse to reduce the computational cost and is supposed to be uni-directional because changes in a time series are likely to unidirectionally lead to fluctuations in other series (Wu et al., 2020).

D ADDITIONAL DETAILS OF CONTINUOUS TEMPORAL AGGREGATION

For the design of $\mathrm{TCN}(\cdot, \mathbf{\Theta})$ within the temporal ODE function (Eq. 10), we adopt a gating mechanism to control the amount of information flows at each aggregation step:

$$\mathrm{TCN}(\mathbf{H}^T(t), t, \mathbf{\Theta}) = f_{\mathcal{C}}(\mathbf{H}^T(t), t, \mathbf{\Theta}_c) \odot f_{\mathcal{G}}(\mathbf{H}^T(t), t, \mathbf{\Theta}_g), \tag{27}$$

where $\odot$ denotes the element-wise product, $f_{\mathcal{C}}(\cdot, \mathbf{\Theta}_c)$ and $f_{\mathcal{G}}(\cdot, \mathbf{\Theta}_g)$ are filtering and gating convolutions that share the identical network structure but with different parameters. Formally, we define them as follows:

$$\begin{cases} f_{\mathcal{C}}(\mathbf{H}^T(t), t, \mathbf{\Theta}_c) = \tanh\Big(\mathrm{Conv}_{1\times m}\left(\mathbf{H}^T(t), \mathbf{\Theta}_c^{1\times m}\right)\Big) = \tanh\left(\mathbf{W}_{\mathbf{\Theta}_c^{1\times m}} \star_\delta \mathbf{H}^T(t) + \mathbf{b}_{\mathbf{\Theta}_c^{1\times m}}\right), \\ f_{\mathcal{G}}(\mathbf{H}^T(t), t, \mathbf{\Theta}_g) = \sigma\Big(\mathrm{Conv}_{1\times m}\left(\mathbf{H}^T(t), \mathbf{\Theta}_g^{1\times m}\right)\Big) = \sigma\left(\mathbf{W}_{\mathbf{\Theta}_g^{1\times m}} \star_\delta \mathbf{H}^T(t) + \mathbf{b}_{\mathbf{\Theta}_g^{1\times m}}\right), \end{cases} \tag{28}$$

where $\sigma(\cdot)$ represents the sigmoid activation, and $\star_\delta$ denotes the convolution operation with an expandable dilation defined by $\delta = \lfloor r^{t/\Delta t_{cta}} \rfloor$. In practice, adopting a single kernel size is less effective to explore multi-granularity temporal patterns. Thus, inspired by Wu et al. (2020), we equip $f_{\mathcal{C}}(\cdot, \mathbf{\Theta}_c)$ and $f_{\mathcal{G}}(\cdot, \mathbf{\Theta}_g)$ with multiple convolutions with different kernel widths m. Since the most of time series data has inherent periods (e.g., 7, 14, 24, 28, and 30), letting kernel width in set $\{2, 3, 6, 7\}$ makes the aforementioned periods can be fully covered by integrating Eq. 10. To achieve this, we truncate and concatenate the outputs of $\mathrm{Conv}_{1\times m}(\cdot, \mathbf{\Theta}^{1\times m})$ with $m = \{2, 3, 6, 7\}$ before the nonlinear activation.

E ADDITIONAL DETAILS OF EXPERIMENT SETTING

Table 3: Dataset Statistics.

Datasets	# Samples	# Nodes	Sampling Rate	Input Length	Output Length
Electricity	26,304	321	1 hour	168	1
Solar-Energy	52,560	137	10 minutes	168	1
Traffic	17,544	862	1 hour	168	1
METR-LA	34,272	207	5 minutes	12	12
PEMS-BAY	52,116	325	5 minutes	12	12

E.1 SINGLE-STEP FORECASTING

We experiment on three benchmark multivariate time series datasets under this evaluation protocol:

- Electricity[1]: This dataset consists of the energy consumption records of 321 clients between 2012 and 2014 with the sampling rate set to 1 hour.
- Solar-Energy[2]: It contains the solar power production records of 137 PV plants in Alabama State in the year of 2006, where the sampling rate is 10 minutes.
- Traffic[3]: A collection of hourly road occupancy rates measured by 862 sensors in San Francisco Bay area between 2015 and 2016.

Following Lai et al. (2018) and Wu et al. (2020), we choose the input length 168 and split all above three datasets into training set (60%), validation set (20%), and testing set (20%) chronologically.

[1] https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014

[2] http://www.nrel.gov/grid/solar-power-data.html

[3] http://pems.dot.ca.gov

We summarize the detailed dataset statistics in Table 3. In this setting, we evaluate the quality of learned spatiotemporal representations by comparing our method with the following time series forecasting baselines:

- AR: The auto-regressive model.
- VARMLP: The combination of the auto-regressive model and the multi-layer perception (Zhang, 2003).
- GRU: The recurrent neural network with the gated recurrent units (Cho et al., 2014).
- LSTNet: It combines the convolution and recurrent neural networks to capture the short-term and long-term multivariate temporal dependencies (Lai et al., 2018).
- TPA-LSTM: An attention-based recurrent neural network for multivariate time series forecasting (Shih et al., 2019).
- MTGNN: A time series forecasting model based on graph neural networks and dilated temporal convolutions (Wu et al., 2020).
- HyDCNN: It forecasts time series with a position-aware dilated temporal convolution neural network (Li et al., 2021).

We repeat our experiments five times and report the averaged results in Table 1. Specifically, the model is trained by the Adam optimizer with batch size 4, dropout rate 0.3, and gradient clip 5. The encoder and decoder hidden dimensions are fixed to 64 across all three datasets. For Electricity and Traffic, we train the model over 60 epochs with the base learning rate set to 0.001, where the learning decay is applied at epochs 20 and 40 with the ratio 0.5. For Solar-Energy, the model is trained 40 epochs with the learning rate 0.0001. For ODE-specific hyperparameters, we adopt the basic Euler solver in both spatial and temporal ODEs across all three datasets, where the integration times T_{cta} and T_{cgp} are set to 1.0 by default. For Electricity and Traffic, we let the integration intervals $\Delta t_{cta} = 0.2$ and $\Delta t_{cgp} = 0.5$. For Solar-Energy, we have $\Delta t_{cta} = 0.167$ and $\Delta t_{cgp} = 0.25$. For the graph constructor, we adopt the hyperparameters suggested by Wu et al. (2020).

E.2 MULTI-STEP FORECASTING

To further evaluate our method under the setting of multi-step forecasting, we conduct experiments on two benchmark traffic datasets:

- METR-LA[4]: It contains the traffic speed readings with 5 minutes sampling rate from the 207 loop detectors in Los Angeles County highways in the year of 2012.
- PEMS-BAY[4]: This dataset is provided by California Transportation Agencies Performance Measurement Systems, which consists of the traffic speed readings of 325 sensors in the Bay Area in the year of 2017, where the data sampling rate is same as in METR-LA.

Following (Li et al., 2018) and (Wu et al., 2020), we let the input and output lengths equal to 12 and further split two datasets into training set (70%), validation set (20%), and testing set (10%) chronologically. The detailed dataset statistics are provided in Table 3. Under this evaluation protocol, we select several GNN-based baselines:

- DCRNN: A graph diffusion-based convolutional recurrent neural network for traffic forecasting (Lai et al., 2018).
- STGCN: It stands for the spatiotemporal graph convolution network, which stacks graph and temporal convolutions to capture the spatial and temporal patterns simultaneously (Yu et al., 2018).
- Graph WaveNet: A spatiotemporal graph neural network that consists of graph and dilated 1D convolutions (Wu et al., 2019b).
- GMAN: A spatiotemporal graph neural network with the spatial and temporal attentions (Zheng et al., 2020).

[4]https://github.com/liyaguang/DCRNN

- MRA-BCGN: The multi-range attentive bicomponent graph convolution network for traffic forecasting (Chen et al., 2020).
- MTGNN: A time series forecasting model based on graph neural networks and dilated temporal convolutions (Wu et al., 2020).
- STGODE: An ODE-based spatiotemporal graph neural network for traffic forecasting (Fang et al., 2021).

Similar to the single-step forecasting, we repeat our experiments five times and report the averaged results in Table 2. For both datasets, we train the model 200 epochs by adopting the Adam optimizer with the base learning rate 0.001, gradient clip 5, and dropout rate 0.3. For METR-LA, the encoder and decoder hidden dimensions are 64 and 128. We adopt the Euler method to solve both spatial and temporal ODEs, where the integration times and intervals are set to 1.0 and 0.25. For PEMS-BAY, both encoder and decoder hidden dimensions are 128. On this dataset, we adopt the Runge-Kutta 3/8 method, where the integration times and intervals are both 1.0 for simplicity. All these experiments use the batch size 64 and the learning rate decay with the ratio 0.1 at the epoch 100. The hyperparameters of the graph constructor are same as the single-step forecasting.

F Additional Details of Forecasting Quality

We provide the forecasting visualizations of DyG−ODE on two datasets with different evaluation protocols. In Figure 4, we plot the forecasting results on 14 November 2006 of our model and two baseline methods on the Solar-Energy dataset, where the forecasting horizon is 30 minutes, i.e., $H = 3$. In this experiment, we randomly selected the readings of two PV plants as the ground truth, where we find that (1) DyG−ODE constantly gives more stable and accurate predictions compared with MTGNN, which can be regarded as a special discrete variant of our method; (2) Compared with conventional time series models (e.g., LSTNet), our approach provides more precise modeling of multivariate time series by leveraging the mechanisms of continuous spatial propagation, continuous temporal aggregation, and dynamic graph learning.

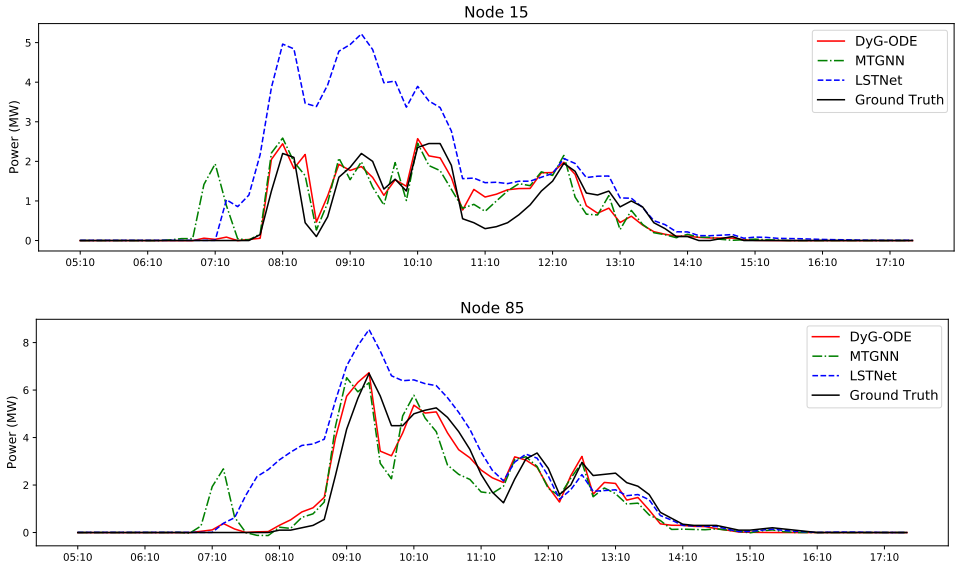

Figure 4: Visualization on two example time series from the Solar-Energy dataset.

We also visualize the multi-step forecasting results on the METR-LA dataset, as shown in Figure 5. Specifically, we randomly select two sensors (i.e., node 4 and 22) and plot the outputs of DyG−ODE and two baseline methods between 5 and 6 June in 2012, where the forecasting horizon is 15 minutes, i.e., $H = 3$. Similarly, we observed that (1) In general, our method can constantly and precisely track the trends, but DCRNN sometimes cannot do it, e.g., the blue dashed line between 7 and 11 am in the second chart of Figure 5; (2) DyG−ODE provides more accurate forecasting than MTGNN

by learning more expressive spatiotemporal representations, e.g., the red dashed line between 4.30 and 7.30 pm in both charts of Figure 5.

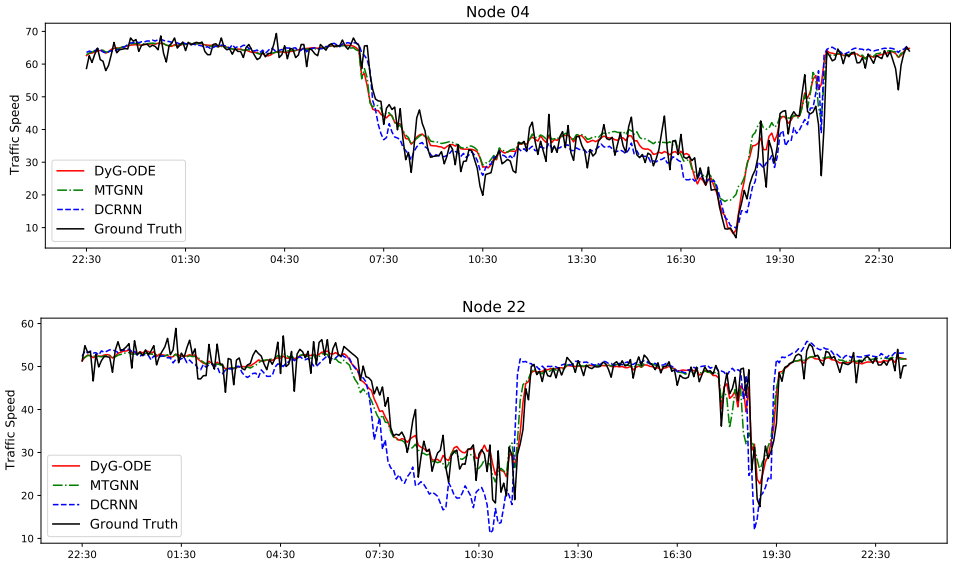

Figure 5: Visualization on two example time series from the METR-LA dataset.

G ADDITIONAL DETAILS OF MODEL COMPUTATIONAL EFFICIENCY

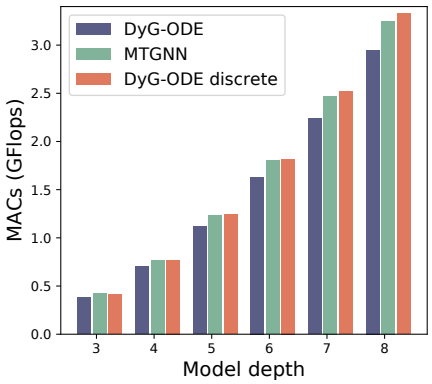

Figure 6: The computational overhead of `DyG-ODE` and two discrete methods w.r.t. the model (i.e., temporal aggregation) depth on the METR-LA dataset.

In Figure 6, we compare the required multiply-accumulate operations (MACs) of `DyG-ODE`, its discrete variant, and MTGNN (Wu et al., 2020). Specifically, our method constantly has a lower computational overhead than `DyG-ODE discrete` and MTGNN, especially for large model depths, demonstrating the computational efficiency of `DyG-ODE`. In comparison, discrete methods, e.g., MTGNN and our discrete variant, have more complex neural architectures, which inevitably introduce more intermediate operations and thus increase the computational overhead. In particular, we find that MTGNN is slightly more efficient than `DyG-ODE discrete` because the latter one adopts the padding version of Eq. 8, which inevitably involves more trainable parameters in the following $L - 1$ temporal convolution layers except for the first layer. Although our continuous version is also based on this padding trick, it is significantly more computationally efficient than MTGNN.

H ADDITIONAL DETAILS OF PARAMETER STUDY

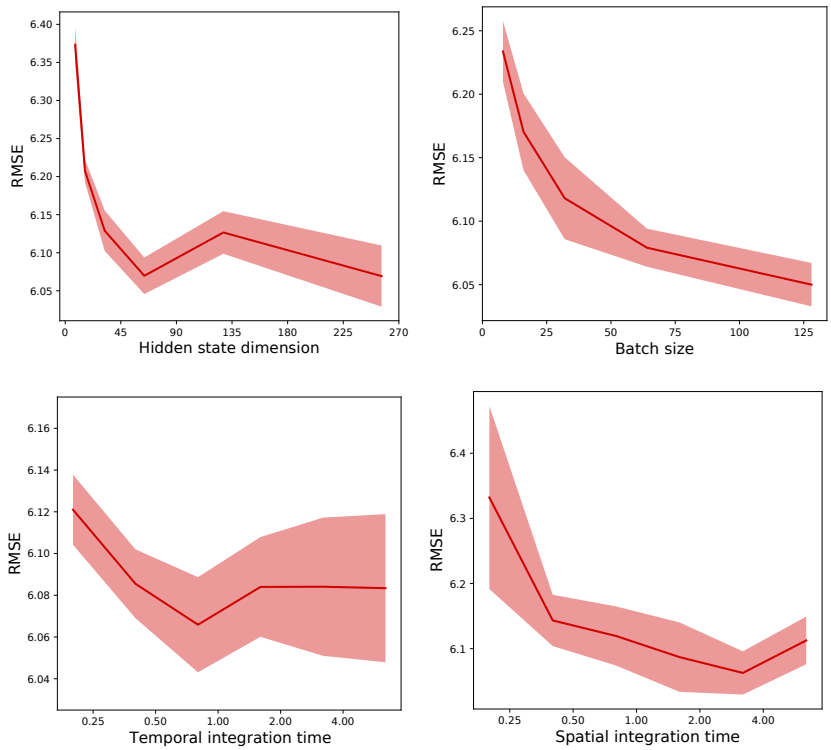

Figure 7: Parameter study on the METR-LA dataset.

Apart from the experiments on spatial and temporal propagation depths (a.k.a spatial and temporal integration intervals where the terminal times are fixed) in Figure 2, we also conduct experiments on other important hyperparameters in DyG−ODE, including temporal integration time T_{cta}, spatial integration time T_{cgp}, spatiotemporal encoder hidden dimension D', and batch size B, to investigate their impacts on our model, as shown in Figure 7. Specifically, we have the following observations: (1) Moderately increase the dimensions of the hidden states in DyG-ODE helps the model learning. We conjecture that this helps avoid the ODE trajectories intersecting with each other (Dupont et al., 2019), thus encourages our model to learn smoother ODE functions that can be easier solved; (2) For a specific spatial or temporal propagation depth, we can find a sweet spot when selecting the spatial or temporal integration time. It is possibly because a short terminal time hinders the convergence of the learned representations and a long time introduces relatively large numerical errors; (3) Within a reasonable range, e.g., from 32 to 128, moderately increasing the batch size improves the model performance. We hypothesize that a relatively large batch size in our method helps reduce the variances of mini-batch gradients, which reduces the impact of noise on the model training.

I ALGORITHM

Algorithm 1 The training algorithm of DyG-ODE.

Input: The training set $\mathbf{X}$, input lengh T, horizon H, batch size B, training epoch E, learning rate η, and the initialized DyG-ODE model $F(\cdot)$ with $\mathbf{\Theta}$, $\mathbf{\Phi}$, and $\mathbf{\Gamma}$

train_loader $\leftarrow \mathrm{DataLoader}(\mathbf{X}, T, H, B)$
for $i \in 1, 2, \cdots, E$ **do**
  train_loader.$\mathrm{shuffle}()$ ▷ Shuffle the training batches
  for $(\mathcal{X}, \mathcal{Y})$ in $\mathrm{enumerate}\left(\text{train\_loader.}\,\mathrm{get\_iterator}()\right)$ **do**
    $\widehat{\mathcal{Y}} \leftarrow F(\mathcal{X}; \mathbf{\Theta}, \mathbf{\Phi}, \mathbf{\Gamma})$
    $\mathcal{L} \leftarrow \mathrm{MAE}(\widehat{\mathcal{Y}}, \mathcal{Y})$
    Calculate the stochastic gradient of $\mathbf{\Theta}$, $\mathbf{\Phi}$, and $\mathbf{\Gamma}$ with respect to $\mathcal{L}$
    Update $\mathbf{\Theta}$, $\mathbf{\Phi}$, and $\mathbf{\Gamma}$ based on their gradients and η
  end for
  $\eta \leftarrow \mathrm{LRScheduler}(\eta, i)$ ▷ Update the learning rate w.r.t. training steps
end for

Output: The trained DyG-ODE model $F^*(\cdot)$

Algorithm 2 Solve the DyG-ODE $F(\cdot)$.

Input: Historical observations $\mathcal{X}$, temporal terminal time T_{cta}, spatial terminal time T_{cgp}, reception field R, and dilation factor r

def spatial_ode($\mathbf{H}_{in}, \mathbf{A}$)
  $\widehat{\mathbf{A}} \leftarrow \mathrm{Normalization}(\mathbf{A})$ ▷ Get the normalized adjacency matrix
  $\mathbf{H}_{out} \leftarrow \widehat{\mathbf{A}}\mathbf{H}_{in} - \mathbf{H}_{in}$ ▷ The spatial ODE function defined in Eq. 6
  return $\mathbf{H}_{out}$
end def

def temporal_ode($\mathbf{H}_{in}$)
  $\widetilde{\mathbf{H}} \leftarrow \mathrm{TCN}(\mathbf{H}_{in}, \mathbf{\Theta})$ ▷ The temporal convolution defined in Eq. 27
  $\mathbf{H}_0, \cdots, \mathbf{H}_{T_{cgp}} \leftarrow \mathrm{ODESolve}(\widetilde{\mathbf{H}}, spatial\_ode, 0, \cdots, T_{cgp})$ ▷ Solve the interior ODE
  $\mathbf{H} \leftarrow \mathrm{Attn}\left([\mathbf{H}_0, \cdots, \mathbf{H}_{T_{cgp}}], \mathbf{\Phi}\right)$ ▷ Attentive transformation defined in Eq. 7
  $\mathbf{H}_{out} \leftarrow \mathrm{zero\_padding}(\mathbf{H}, R)$ ▷ The padding trick in Eq. 12
  $\mathrm{TCN}(\cdot, \mathbf{\Theta}).\,\mathrm{update\_dilation}(r)$ ▷ Update the dilation of temporal convolution
  return $\mathbf{H}_{out}$
end def

$\mathbf{H}_0 \leftarrow \mathrm{Conv}(\mathcal{X}, \mathbf{\Gamma}_{\mathbf{sc}})$ ▷ Map the input to the latent space
$node\_idx \leftarrow \mathrm{range}(\mathcal{X}.shape[1])$ ▷ Get all time series indices in $\mathcal{X}$
$\mathbf{A} \leftarrow \mathrm{graph\_learner}\left(node\_idx, \mathbf{\Gamma}_{\mathbf{gc}}\right)$ ▷ Construct a graph for $\mathcal{X}$ at the i-th training step
$\mathbf{H} \leftarrow \mathrm{ODESolver}(\mathbf{H}_0, temporal\_ode, T_{cta})$ ▷ Solve the exterior ODE defined in Eq. 12
$\widehat{\mathcal{Y}} \leftarrow \mathrm{Conv}(\mathbf{H}, \mathbf{\Gamma}_{\mathbf{dc}})$ ▷ Forecast based on learned representations

Output: The forecasting results $\widehat{\mathcal{Y}}$