Signed Graph Neural Ordinary Differential Equation for Modeling Continuous-Time Dynamics

Lanlan Chen[1], **Kai Wu**[2*], **Jian Lou**[3], **Jing Liu**[1]
[1]Guangzhou Institute of Technology, Xidian University
[2]School of Artificial Intelligence, Xidian University
[3]Zhejiang University
21181214030@stu.xidian.edu.cn, kwu@xidian.edu.cn, jian.lou@hoiying.net, neouma@163.com

Abstract

Modeling continuous-time dynamics constitutes a foundational challenge, and uncovering inter-component correlations within complex systems holds promise for enhancing the efficacy of dynamic modeling. The prevailing approach of integrating graph neural networks with ordinary differential equations has demonstrated promising performance. However, they disregard the crucial signed information potential on graphs, impeding their capacity to accurately capture real-world phenomena and leading to subpar outcomes. In response, we introduce a novel approach: a signed graph neural ordinary differential equation, adeptly addressing the limitations of miscapturing signed information. Our proposed solution boasts both flexibility and efficiency. To substantiate its effectiveness, we seamlessly integrate our devised strategies into three preeminent graph-based dynamic modeling frameworks: graph neural ordinary differential equations, graph neural controlled differential equations, and graph recurrent neural networks. Rigorous assessments encompass three intricate dynamic scenarios from physics and biology, as well as scrutiny across four authentic real-world traffic datasets. Remarkably outperforming the trio of baselines, empirical results underscore the substantial performance enhancements facilitated by our proposed approach. Our code can be found at https://github.com/beautyonce/SGODE.

Introduction

Complex systems prevalent in the real world, such as gene regulation (Marbach et al. 2012), social networks (Wasserman, Faust et al. 1994), climate models (Hwang et al. 2021), and traffic systems (Zhao et al. 2019), often find representation as complex networks governed by nonlinear dynamics (Lieberman, Hauert, and Nowak 2005). In contrast to deterministic and easily obtainable graphs (Wu et al. 2020a; Feng et al. 2023), the graph structures of these complex networks are challenging to explicitly articulate. Despite the extensive exploration of nonlinear dynamical systems, a significant number of complex networks continue to evade a clear understanding of their underlying dynamics.

In recent years, a noteworthy trend has emerged, involving the fusion of ordinary differential equations (ODEs) with neural networks to acquire insights into continuous-time dynamics (Chen et al. 2018; Rubanova, Chen, and Duvenaud 2019; Kidger et al. 2020; Jhin et al. 2021a; Hwang et al. 2021; Huang, Sun, and Wang 2021; Fang et al. 2021; Choi et al. 2022; Jhin et al. 2022).These hybrids, encompassing both ODEs and graph neural networks (GNNs), have demonstrated promising performance across a variety of tasks. This includes climate modeling (Hwang et al. 2021; Jhin et al. 2021a), traffic flow prediction (Poli et al. 2019; Fang et al. 2021; Choi et al. 2022), node classification (Zang and Wang 2020; Xhonneux, Qu, and Tang 2020), and dynamic interactions (Huang, Sun, and Wang 2021). However, it remains pertinent to note that a substantial portion of complex networks retains enigmatic dynamics yet to be fully unraveled.

The majority of existing methodologies predominantly focus on either inferring or utilizing unsigned graphs, wherein only the presence or absence of dependencies between nodes is taken into account, while the type of edges are disregarded. As shown in Table 1, the current GNN-ODE methods are unable to capture and use signed information of dynamics, thus rendering this problem challenging. It is noteworthy that within unsigned graphs, nodes face challenges in shifting towards the opposing trend (rise or decline) when they and their neighboring nodes align in a similar variation trend (decline or rise). This constraint inherently restricts the capacity to represent dynamic processes effectively. A remedy for this limitation is the introduction of signed connections, which can markedly enhance the scenario. Intriguingly, signed graphs find applicability across a multitude of complex systems (Shi, Altafini, and Baras 2019). Notable instances include predation and prey relationships within ecosystems, activation and repression dynamics in gene regulation networks (Karlebach and Shamir 2008), as well as cooperation and antagonism dynamics observed within social and economic networks (Derr, Ma, and Tang 2018). To offer a concrete illustration, we examine the context of a three-way intersection within traffic systems, thereby illustrating the presence of signed graphs (refer to Appendix: https://arxiv.org/abs/2312.11198).

In the context of an unknown underlying graph, one approach involves inferring the graph structure, which is inherently tied to the realm of graph structure learning. However, current graph structure learning methods fall short in handling signed information. To address this challenge, various strategies have been explored. One approach entails captur-

*Corresponding author

Method	Formula	Graph Information	Signed ?
NDCN (Zang and Wang 2020)	$\frac{d\boldsymbol{H}(t)}{dt} = f\left(\widetilde{\boldsymbol{A}}, \boldsymbol{H}(t), \boldsymbol{\theta}\right)$	$\widetilde{\boldsymbol{A}} = \boldsymbol{D}^{-\frac{1}{2}}(\boldsymbol{D} - \boldsymbol{A})\boldsymbol{D}^{-\frac{1}{2}}$	✗
STGODE (Fang et al. 2021)	$\frac{d\boldsymbol{H}(t)}{dt} = f(\widetilde{\boldsymbol{A}}, \boldsymbol{H}(t), \boldsymbol{H}(0), \boldsymbol{\theta})$	$\widetilde{\boldsymbol{A}} = \frac{\alpha}{2}(\boldsymbol{I} + \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}})$	✗
STG-NCDE (Choi et al. 2022)	$\frac{d\boldsymbol{H}(t)}{dt} = f(\widetilde{\boldsymbol{A}}, \boldsymbol{H}(t), \boldsymbol{\theta})$	$\widetilde{\boldsymbol{A}} = \boldsymbol{I} + \phi\left(\sigma\left(\boldsymbol{E}\boldsymbol{E}^T\right)\right)$	✗
SGODE (This Work)	$\frac{d\boldsymbol{H}(t)}{dt} = f\left(\boldsymbol{K}, \boldsymbol{H}(t), \boldsymbol{B}(t), \boldsymbol{\theta}\right)$	$\boldsymbol{K} = \sigma(\boldsymbol{E}_1\boldsymbol{E}_2^T) + (-\sigma(\boldsymbol{E}_3\boldsymbol{E}_4^T))$	✓

Table 1: ODE-GNN approaches. $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is the adjacency matrix of the network, and $\boldsymbol{D} \in \mathbb{R}^{n\times n}$ is the corresponding degree matrix. ϕ is a softmax activation, and σ is a rectified linear unit. $\boldsymbol{I} \in \mathbb{R}^{n\times n}$ is the identity matrix. $\boldsymbol{E} \in \mathbb{R}^{n\times d}$ represents the node embedding matrix, where d is the embedding dimension, much smaller than n. $\boldsymbol{H}(t) \in \mathbb{R}^{n\times h}$ represents the state feature in the hidden space at time t with dimension h. In this context, $\boldsymbol{\theta}$ refers to the other parameters. While other methods utilize the normalized adjacency matrix $\widetilde{\boldsymbol{A}} \in \mathbb{R}^{n\times n}$ of a homogeneous graph, this study constructs a signed coefficient matrix $\boldsymbol{K} \in \mathbb{R}^{n\times n}$ composed of positive and negative coefficient matrices, which is crucial for capturing different types of information.

ing the edge distribution (Kipf et al. 2018; Franceschi et al. 2019; Shang, Chen, and Bi 2021), followed by sampling the graph's adjacency matrix from this distribution. Yet, the complexity arises when assuming a bipartite adjacency matrix and attempting separate learning of its positive and negative edges. This introduces intricate optimization issues due to the inherent uncertainty tied to the sampled graph. Additionally, distinct optimization processes for positive and negative graphs exacerbate uncertainties, impeding the achievement of convergence. Moreover, approximating the edge distribution through neural networks imposes substantial computational and storage demands on a high-precision ODE solver. Alternatively, an approach directly embeds the graph into a matrix composed of learnable parameters within a GNN (Wu et al. 2019; Bai et al. 2020; Wu et al. 2020b; Deng and Hooi 2021; Choi et al. 2022; Jin et al. 2023; Jiang et al. 2023), enabling simultaneous optimization alongside other pertinent parameters. While methods employing learnable parameter matrices are simple to optimize, they often neglect signed information and remain susceptible to overfitting. Hence, there is a compelling need to devise a GNN-ODE method that strikes a balance between ease of optimization, consideration of signed information, preservation of stability, and mitigation of overfitting.

We propose a **Signed Graph Neural Ordinary Differential Equation**, called SGODE, to capture and use positive and negative information of nodes during continuous dynamics. Our contributions are summarized as follows:

1) The significance of signed information in real-world scenarios is undeniable. In response, we propose a straightforward yet impactful solution that effectively addresses the limitations of the current GNN-ODE methodologies by capturing and leveraging this crucial information.

2) SGODE offers a high degree of flexibility and seamlessly integrates into various frameworks for graph-based dynamic modeling, including graph neural ODEs, graph neural controlled differential equations (NCDEs), and graph recurrent neural networks (RNNs). Empirical results across multiple dynamic modeling datasets and traffic flow datasets substantiate the effectiveness of SGODE.

Background

Notations. We denote the training data containing n time series as $\boldsymbol{X}$. $\boldsymbol{X}^i$ represents the features of the i-th time series, and $\boldsymbol{X}(t)$ encompasses the features of all nodes at time t. The remaining notations are described in Table 1. Specifically, if the relationship between node i and node j is positive, we have $\boldsymbol{K}_{ij} > 0$, while a negative relationship corresponds to $\boldsymbol{K}_{ij} < 0$. An edge is absent when $\boldsymbol{K}_{ij} = 0$.

For synthetic dynamics, there are a total of S time steps for training, written as $\boldsymbol{X}_S = \{\boldsymbol{X}_{s1}, \boldsymbol{X}_{s_2}, \ldots, \boldsymbol{X}_{s_s}\}$, and P steps for forecasting, written as $\boldsymbol{X}_P = \{\boldsymbol{X}_{p_1}, \boldsymbol{X}_{p_2}, \ldots, \boldsymbol{X}_{p_p}\}$. If the time step intervals in S and P are fixed, the sampling is considered as equal intervals sampling; otherwise, it is referred to as irregular sampling. Given S, $\boldsymbol{X}_S$ and a set of times P, the model needs to predict $\boldsymbol{X}_P$. By using $\boldsymbol{\theta}$ to represent other parameters in the network except $\boldsymbol{K}$, the model is denoted as $\widehat{\boldsymbol{X}}_P = M(\boldsymbol{K}, \boldsymbol{\theta}, \boldsymbol{X}_S)$. For traffic flow data, we utilize a T-step window approach to forecast the subsequent τ steps. If $t+1$ represents the initial time step of a certain window, the model can be denoted as $\widehat{\boldsymbol{X}}_{t+T+1:t+T+\tau} = M(\boldsymbol{K}, \boldsymbol{\theta}, \boldsymbol{X}_{t+1:t+T})$. Take $\mathcal{L}$ to represent the loss function of the predicted value and ground truth. Then our goal is $\arg\min_{\boldsymbol{K},\boldsymbol{\theta}} \sum_t \mathcal{L}(M(\boldsymbol{K}, \boldsymbol{\theta}, \boldsymbol{X}_{z_1}), \boldsymbol{X}_{z_2})$ For synthetic dynamics, we set $z_1 = S$, $z_2 = P$, and for traffic flow prediction, we set $z_1 = t+1 : t+T$, $z_2 = t+T+1 : t+T+\tau$ for all the training examples that are partitioned by window. In the following, we review of the relevant models, including NDCN (Zang and Wang 2020), STG-NCDE (Choi et al. 2022), and DCRNN (Li et al. 2018).

NDCN. NDCN (Zang and Wang 2020) can be considered as either a continuous-time GNN or a graph neural ODE model. NDCN utilizes an encoding function to transform $\boldsymbol{X}(t)$ into the hidden space, employs a continuous model to regulate the dynamics on the graph within the hidden space, and subsequently decodes the hidden state back into the original space. For irregular and equal intervals sampling tasks, NDCN uses L_1 loss. The ODE layer is

$$\frac{d\boldsymbol{H}(t)}{dt} = \sigma\left(\widetilde{\boldsymbol{A}}\boldsymbol{H}(t)\boldsymbol{W}_h + \boldsymbol{b}_h\right), \tag{1}$$

where $\boldsymbol{W}_h \in \mathbb{R}^{n\times h}$ and $\boldsymbol{b}_h \in \mathbb{R}^h$ are the parameters of fully connected layer FC. NDCN adopts a linear diffusion operator $\widetilde{\boldsymbol{A}}$ (shown in Table 1).

STG-NCDE. STG-NCDE (Choi et al. 2022) STG-NCDE (Choi et al. 2022) combines the advantages of graph convolutional network and NCDE to design a unified space-time NCDE framework, which encompasses two NCDEs, a CDE that generates a continuous path for each node, denoted as f, and another CDE applies a method for spatial and temporal processing jointly, denoted as g. Denote the hidden states of f and g as $\boldsymbol{H}(t)$ and $\boldsymbol{Z}(t)$ respectively, then we have

$$\frac{d}{dt}\begin{bmatrix}\boldsymbol{Z}(t)\\ \boldsymbol{H}(t)\end{bmatrix} = \begin{bmatrix} g(\boldsymbol{Z}(t);\boldsymbol{\theta}_g) f(\boldsymbol{H}(t);\boldsymbol{\theta}_f)\frac{d\boldsymbol{X}(t)}{dt} \\ f(\boldsymbol{H}(t);\boldsymbol{\theta}_f)\frac{d\boldsymbol{X}(t)}{dt}\end{bmatrix}. \quad (2)$$

The initial value is determined by the fully connected layer: $\boldsymbol{H}(0) = \mathrm{FC}_{dim(\boldsymbol{X}^i)\to dim(\boldsymbol{H}^i)}(\boldsymbol{X}(t_0))$, $\boldsymbol{Z}(0) = \mathrm{FC}_{dim(\boldsymbol{H}^i)\to dim(\boldsymbol{Z}^i)}(\boldsymbol{H}(0))$. $\boldsymbol{\theta}_f$ is the parameters of the CDE function f. $\mathrm{FC}_{input\_size\to output\_size}$ means a fully-connected layer whose input size is $input\_size$ and output size is $output\_size$. f is composed of l layers of MLPs, with the activation function in the final layer being ψ (hyperbolic tangent). The activation functions in the other layers are σ. g and f share similar structures, the only difference being that g incorporates adaptive graph information in the middle layer. Let $\boldsymbol{Z}_{B_0}(t)$ and $\boldsymbol{Z}_{B_1}(t)$ be the input and output of this layer. The equation is given by

$$\boldsymbol{Z}_{B_1}(t) = (\boldsymbol{I} + \phi(\sigma(\boldsymbol{E}\boldsymbol{E}^T)))\boldsymbol{Z}_{B_0}(t)\boldsymbol{W}_s, \quad (3)$$

where $\boldsymbol{I}$ is the $n \times n$ identity matrix. $\boldsymbol{E}$ is a trainable node embedding matrix, $\boldsymbol{E}^T$ is its transpose. Eq. 3 has two important constituent components $\boldsymbol{I} + \phi(\sigma(\boldsymbol{E}\boldsymbol{E}^T))$ and $\boldsymbol{W}_s$ (Bai et al. 2020), the former adaptively learns spatial dependencies, and the latter extracts the specific pattern of each node through $\boldsymbol{W}_s = \boldsymbol{E}\boldsymbol{W}_{pool}$, where $\boldsymbol{E} \in \mathbb{R}^{n\times d}$ and $\boldsymbol{W}_{pool} \in \mathbb{R}^{d\times h\times h}$ is weight pool for $\boldsymbol{E}$.

DCRNN. DCRNN (Li et al. 2018) introduces diffusion graph convolutions to capture spatial dependencies, facilitating the modeling of traffic flow dynamics from a spatio-temporal perspective. Within DCGRU, GRU (Chung et al. 2014) serves as a recurrent neural network to simulate time dependence, with matrix multiplication being replaced by diffusion convolution,

$$\begin{aligned}
\boldsymbol{R}(t) &= \mathrm{sigmoid}\left(\boldsymbol{W}_{\boldsymbol{R}} \star_A [\boldsymbol{X}(t) \| \boldsymbol{H}(t-1)] + b_R\right),\\
\boldsymbol{C}(t) &= \tanh\left(\boldsymbol{W}_{\boldsymbol{C}} \star_{\boldsymbol{A}} [X(t) \| (\boldsymbol{R}(t) \odot \boldsymbol{H}(t-1)] + b_C\right),\\
\boldsymbol{U}(t) &= \mathrm{sigmoid}\left(\boldsymbol{W}_U \star_A [\boldsymbol{X}(t) \| \boldsymbol{H}(t-1)] + b_U\right),\\
\boldsymbol{H}(t) &= \boldsymbol{U}(t) \odot \boldsymbol{H}(t-1) + (1 - \boldsymbol{U}(t)) \odot \boldsymbol{C}(t),
\end{aligned} \quad (4)$$

where diffusion convolution $\star_{\boldsymbol{A}}$ is defined as,

$$\boldsymbol{W}_Q \star_{\boldsymbol{A}} \boldsymbol{H}(t) = \sum_m \left(w^Q_{m_1}\left(\boldsymbol{D}_O^{-1}\boldsymbol{A}\right)^m + w^Q_{m_2}\left(\boldsymbol{D}_I^{-1}\boldsymbol{A}^T\right)^m\right)\boldsymbol{H}(t).$$

$\boldsymbol{D}_O$ and $\boldsymbol{D}_I$ are out-degree and in-degree matrix, $||$ is connected along the feature dimension, and $\odot$ denotes element-wise product. The diffusion step m is a hyperparameter, $w^Q_{m_1}, w^Q_{m_2}, b_Q$ for $Q = R, U, C$ are all model parameters. For simplicity, our expression here follows the conventions of GTS (Shang, Chen, and Bi 2021).

Methods

Signed Graph Ordinary Differential Equation. Learning the distribution of edges in a signed graph is a challenging optimization task, particularly when the edge distribution must be learned through a large neural network. When the number of nodes expands to hundreds, high-precision ODE solvers encounter difficulties in calculation and storage. Therefore, we opt for direct initialization of the corresponding node vector for each node. We consider two forms of coefficient matrix $\boldsymbol{K}$, namely $\boldsymbol{K} = \boldsymbol{E}_1\boldsymbol{E}_2^T$ and $\boldsymbol{K} = \boldsymbol{K}_{pos} + \boldsymbol{K}_{neg}$, where $\boldsymbol{K}_{pos} = \sigma(\boldsymbol{E}_1\boldsymbol{E}_2^T), \boldsymbol{K}_{neg} = -\sigma(\boldsymbol{E}_3\boldsymbol{E}_4^T), \boldsymbol{E}_* \in \mathbb{R}^{n\times d}$. The first form indicates learning the positive and negative relations of each node pair. The second form indicates that only the positive and negative relations of some node pairs are learned, and the node pairs with low dependency are ignored. We do not use any other constraints to ensure that $\boldsymbol{K}$ learns the connection weights.

Without constraints, the evolution of unkonwn signed graph ODEs is unstable due to the extensive parameter space and the random initialization of embedding matrices. Motivated by the integration of initial value-related terms into the ODE for neighborhood information learning while retaining original features (Xhonneux, Qu, and Tang 2020), we introduce self-trend features $\boldsymbol{B}^i$ associated with the state of the ith node, augmenting the stability of the dynamics learning process. The inclusion of $\boldsymbol{B}(t)$ implies that the instantaneous rate of change in node features is influenced not just by interactions with other nodes, but also by its inherent changes. Learning self-evolving features is comparably straightforward compared to graph-related learning processes, allowing for representation of temporal variations. We posit that $\boldsymbol{B}(t)$ primarily encompasses three factors: the constant term, initial features, and the features at the current time step. Formally, the interaction of information on signed graphs is defined as follows:

$$\frac{\mathrm{d}\boldsymbol{H}(t)}{\mathrm{d}t} = f(\boldsymbol{K}\boldsymbol{H}(t)\boldsymbol{W}_h + \boldsymbol{B}(t)), \quad (5)$$

$$\boldsymbol{B}(t) = \lambda_1 g_1(\boldsymbol{H}(t)) + \lambda_2 g_2(\boldsymbol{H}(0)) + \lambda_3 \boldsymbol{B}_0, \quad (6)$$

where both f and g simply refer to functions, and $\boldsymbol{B}_t$ describes the information of the state trend at time t. $\lambda_1, \lambda_2, \lambda_3 \in [0, 1]$ indicates whether the item is taken. We emphasize the significance of focusing on current time-step features, whether for longer-term predictions or shorter periods within the prediction window (e.g., $\lambda_1 = 1$). Even incorporating basic linear relationships can effectively enhance the performance of the ODE model.

Embedding SGODE into NDCN. We first embedding SGODE into NDCN (Zang and Wang 2020) to show its effectiveness of modeling continuous-time dynamics. We replace the intermediate ODE layer (Eq. 1) with our proposed SGODE layer, as described in Eq. 5,

$$\frac{d\boldsymbol{H}(t)}{dt} = \sigma\left(\boldsymbol{K}\boldsymbol{H}(t)\boldsymbol{W}_h + \boldsymbol{B}(t) + \boldsymbol{b}_h\right). \quad (7)$$

We consider three forms of $\boldsymbol{K}$. The first form directly represents $\boldsymbol{K}$ by $n \times n$ learnable parameters, called **SGODEv1**. The second is to express $\boldsymbol{K} = \boldsymbol{E}_1\boldsymbol{E}_2^T$ using the similarity calculated by two node embedding matrices, denoted as

SGODEv2. The third is to utilize two node embedding matrices to calculate the positive relationships, and employ two node embedding matrices to calculate the negative relationships, $\boldsymbol{K} = \sigma(\boldsymbol{E}_1\boldsymbol{E}_2^T) - \sigma(\boldsymbol{E}_3\boldsymbol{E}_4^T)$, denoted as **SGODEv3**. Given the simplicity of the synthetic dynamics, in **SGODEv3**, we establish a straightforward linear scaling relationship between $\boldsymbol{B}(t)$ and $\boldsymbol{H}(t)$, as follows: $\boldsymbol{B}(t) = \boldsymbol{b}\boldsymbol{H}(t)\boldsymbol{W}_h$, where $\boldsymbol{b} \in \mathbb{R}^n$. In practice, these learnable parameters are trained jointly with other network parameters. Further details can be found in Appendix.

Embedding SGODE into STG-NCDE. STG-SGCDE refer to the STG-NCDE variant by modify the adaptive graph convolution layer of Eq.3 with the proposed SGODE. We observe that $\boldsymbol{E}$ for constructing the adaptive adjacency matrix is closely related to $\boldsymbol{W}_S$ for extracting patterns of each node. In order to establish a relevant connection between $\boldsymbol{K}$ and $\boldsymbol{W}_{S_1}$, we adopt a node embedding matrix $\boldsymbol{E}_1$ that controls both positive and negtive relations and get weight pool of each node with $\boldsymbol{W}_{S_1}$. To enhance the sparsity of the graph, we employ an adaptive mask matrix (Jhin et al. 2021b). Formally,

$$\begin{aligned} \boldsymbol{K}_0 &= \sigma(\boldsymbol{E}_1\boldsymbol{E}_2^T) - \sigma(\boldsymbol{E}_1\boldsymbol{E}_3^T), \\ \boldsymbol{W}_{s_1} = \boldsymbol{E}_1\boldsymbol{W}_{pool_1}, &\boldsymbol{W}_{s_2} = \boldsymbol{E}_1\boldsymbol{W}_{pool_2}, \end{aligned} \tag{8}$$

and construction of the adaptive mask matrix follows these steps: initially, a learnable matrix is set as $\boldsymbol{M} = \boldsymbol{E}_{M1}\boldsymbol{E}_{M2}^T$. Subsequently, a hard sigmoid function is employed for rigorous classification, and the result is rounded up to yield the final mask matrix, denoted as $\boldsymbol{M}$. The detailed training approach is outlined below:

$$\varphi(x) = round(\text{hardsigmoid}(\alpha x)), \tag{9}$$

for the forward path, and

$$\nabla\varphi(x) = \nabla\,\text{hardsigmoid}(\alpha x), \tag{10}$$

for the backward path, where the temperature $\alpha \geq 1.0$ is a hyperparameter to control the slope of the sigmoid function. Given the presence of $\boldsymbol{W}_{s_1}(t)$ for extracting interaction information $\boldsymbol{K}\boldsymbol{Z}_{B_0}(t)$. We then define $g_1(\boldsymbol{Z}_{B_0}(t)) = \boldsymbol{Z}_{B_0}(t)\boldsymbol{W}_{s_2} = \boldsymbol{Z}_{B_0}(t)\boldsymbol{E}_1\boldsymbol{W}_{pool_2}$. Finally, we have $\boldsymbol{K} = \varphi(\boldsymbol{M}) \odot \boldsymbol{K}_0$, and $\boldsymbol{B}(t) = \boldsymbol{Z}_{B_0}(t)\boldsymbol{W}_{s_2} + \boldsymbol{B}_0$. i.e. we substitute this equation for Eq. 3.

Embedding SGODE into DCRNN. SGODE-RNN refers to the DCRNN variant. Our continuous graph diffusion method,

$$\boldsymbol{W}_{\boldsymbol{Q}} \star_{\boldsymbol{K}} \boldsymbol{H}(t) = \sum_m \boldsymbol{w}_{t_m}^{Q}\boldsymbol{H}(t_m), \tag{11}$$

$$\frac{\mathrm{d}\boldsymbol{H}(t_m)}{\mathrm{d}t} = \boldsymbol{K}\boldsymbol{H}(t_m)\boldsymbol{W}_h + \boldsymbol{B}(t_m), \tag{12}$$

depicts the continuous dynamics of hidden states on a signed graph during state transitions and the extraction of information on continuous dynamics. We establish a two-layer FC network as $g_2(\boldsymbol{H}(0)) = \text{FC}_{\dim(\boldsymbol{H}^i)\to\dim(\boldsymbol{H}^i)}(\sigma(\text{FC}_{\dim(\boldsymbol{H}^i)\to\dim(\boldsymbol{H}^i)}(\boldsymbol{H}(t))))$, and $g_1(\boldsymbol{H}(t_m)) = \boldsymbol{b}\boldsymbol{H}(t)\boldsymbol{W}_h$. Here, $\boldsymbol{K} = \sigma(\boldsymbol{E}_1\boldsymbol{E}_2^T) - \sigma(\boldsymbol{E}_3\boldsymbol{E}_4^T)$. The ODE within the interval $[t, t+1]$ can be perceived as a normalization of an ODE of arbitrary length. To enhance monitoring of the internal evolution within the ODE solver and mitigate error accumulation, we extract features from a set of m equidistantly sampled states within the range of $[0, 1]$. If $m = 1$, we extract only the starting and ending information of ODE. This strategy enables an increased focus on the inherent trend of self-evolution, while also taking into account node interactions and self-evolution features, thereby enhancing the model's fitting capabilities.

Training Loss

The training loss function in our model is shown as follows,

$$\mathcal{L} = \frac{1}{|P|}\frac{1}{|\tau|}\sum_{p\in P}\sum_{\tau}|\widehat{\boldsymbol{X}}(P(t)) - \boldsymbol{X}(P(t))|, \tag{13}$$

p refers to each training example in the training set P, τ refers to the time step to be predicted in each training example, $|P|$ is the number of training examples, and $|\tau|$ is the prediction step number.

Experiments

Experimental Setup

Dataset *Three Continuous-time Dynamics*. We follow the dataset settings in NDCN (Zang and Wang 2020). We generate three continuous-time dynamics: heat diffusion, mutualistic interaction, and gene regulation dynamics on five graphs, including 1) Grid network, where each node is connected with 8 neighbors; 2) Random network (Erdős, Rényi et al. 1960); 3) Power-law network (Barabási and Albert 1999); 4) Small-world network (Watts and Strogatz 1998); 5) Community network (Fortunato 2010). We categorize irregular sampling into two types: interpolated value prediction and extrapolated value prediction. The training, interpolation prediction, and extrapolation prediction are allocated in a ratio of 80/20/20, respectively. The details of three continuous-time dynamics and data generation are shown in Appendix.

Traffic Prediction Dataset. Four publicly available real-world traffic datasets are employed: (1) METR-LA and PEMS-BAY (Li et al. 2018). METR-LA contains the traffic information on the highways of Los Angeles County and consists of 207 sensors, collecting data over a span of 4 months. PEMS-BAY comprises 325 sensors in the Bay Area, with data collected over a period of 6 months. (2) PeMS traffic datasets (Chen et al. 2001), previously employed in other works (Fang et al. 2021; Choi et al. 2022), include PeMSD4 and PeMSD8 with 307 and 170 nodes, respectively. The frequency of data in these four datasets is uniformly set to 5 minutes. For METR-LA and PEMS-BAY, we adopt the train/valid/test split of 70%/10%/20% as suggested by GTS (Shang, Chen, and Bi 2021), while for PeMSD4 and PeMSD8, we follow STG-NCDE (Choi et al. 2022) and use a split of 60%/20%/20%.

Baselines To show the effectiveness of SGODE on modeling continuous dynamics, the following basedlines are employed. (1) **NDCN** (Zang and Wang 2020). It is a representative algorithm for ODE-GNN methods using real graphs. (2) **No-graph**. We remove graph structure in NDCN as the benchmark algorithm directly. (3) **Adp-NDCN**. For a fair

comparison, we replace the linear map in NDCN with a learnable matrix $\phi(\sigma(\boldsymbol{E}_1\boldsymbol{E}_2^T))$ (Bai et al. 2020; Choi et al. 2022; Wu et al. 2019, 2020b). (4) **GTS-NDCN**. For a fair comparison, we replace the linear map in NDCN with a learnable matrix proposed in (Shang, Chen, and Bi 2021). The details of **SGODEv1**, **SGODEv2**, and **SGODEv3** are shown in the **Methods** section.

For traffic prediction problems, We compare with widely used time series regression models, including (1) **HA**: Predicting based on the historical average; (2) **ARIMA**: Forecasting based on the statistical characteristics of stationary time series. (3) **VAR**: Vector Auto-Regression. (4) **SVR**: Support Vector Regression which uses linear support vector machine for the regression task; The following deep neural network based approaches are also included: (5) **FNN**: Feed-forward Neural Network ; (6) **LSTM**: Recurrent Neural Network with fully connected LSTM hidden units (Sutskever, Vinyals, and Le 2014); (7) **DCRNN** (Li et al. 2018): a deep learning framework for traffic forecasting that incorporates both spatial and temporal dependency in the traffic flow; (8) **LDS** (Franceschi et al. 2019): the model considers graphs as hyperparameters within a two-layer optimization framework, where it learns parameterized element-wise Bernoulli distributions; (9) **GWNet** (Wu et al. 2019): the most representative deep models for traffic forecasting; (10) **MTGNN** (Wu et al. 2020b) is an extended version of GWNet that extends the adaptive graph leaning part; (11) **GTSv** and **GTS** (Shang, Chen, and Bi 2021): they are variants that apply inference graphs to T-GCN (Zhao et al. 2019) and DCRNN models respectively. (12) **STEP** (Shao et al. 2022) is an extension of spatial-temporal GNN enhanced by a scalable time series Pre-training model; (13) **MegaCRN** (Jiang et al. 2023): they designed meta-graph learner for spatio-temporal graph learning, to explicitly disentangles the heterogeneity in space and time. To additionally demonstrate the performance of SGODE embedded in the existing NCDE framework, we compare it with the advanced STGODE (Fang et al. 2021) and STG-NCDE (Choi et al. 2022).

Hyperparameters. we set the learning rate to 0.005 for SGODE-RNN and to 0.001 for STG-SGCDE. We have reproduced the results for four methods, i.e., NDCN, GTS, STG-NCDE, and MegaCRN. If the original paper provides hyperparameters (e.g., STG-NCDE), we adhere to the same settings there. If not (e.g., GTS, MegaCRN), we conduct experiments within their recommended hyperparameter ranges. Specifically, for GTS, we perform a grid search on hyperparameters including learning rate, number of clusters, and regularization weight. For MegaCRN, we set the number of meta-nodes to 20. Detailed parameter settings for SGODE and baselines are available in Appendix.

These methods are evaluated based on three commonly used metrics, including (1) Mean Absolute Error (MAE), (2) Mean Absolute Percentage Error (MAPE), and (3) Root Mean Squared Error (RMSE). To maintain consistency with the referenced algorithms, we refer to DCRNN (Li et al. 2018) for the evaluation index calculation method on METR-LA and PEMS-BAY, and refer to STG-NCDE (Choi et al. 2022) on PeMS04 and PeMSD8.

	Methods	Grid	Rand.	Power	Small	Com.
I	No-graph	41.1	10.1	20.7	21.2	24.2
	NDCN	3.3	3.4	6.0	3.6	3.7
	Adp-NDCN	7.9	8.3	12.6	9.8	9.3
	GTS-NDCN	9.9	5.0	7.4	8.5	10.0
	SGODEv1	4.5	**1.5**	3.2	4.7	2.1
	SGODEv2	**2.5**	2.0	**1.7**	**1.7**	**1.7**
	SGODEv3	3.4	2.0	2.8	2.9	3.8
II	No-graph	32.2	10.3	31.3	18.0	14.7
	NDCN	8.2	6.1	6.6	4.4	9.1
	Adp-NDCN	5.9	9.6	8.4	6.4	10.2
	GTS-NDCN	6.5	9.0	10.4	9.4	8.4
	SGODEv1	6.1	5.6	5.7	**3.0**	7.7
	SGODEv2	5.1	**4.2**	**5.2**	4.8	**6.2**
	SGODEv3	**4.7**	7.9	5.8	3.1	9.4
III	No-graph	26.9	11.5	25.2	15.7	18.2
	NDCN	5.9	3.3	3.1	4.0	**1.9**
	Adp-NDCN	4.5	1.6	3.1	2.6	2.8
	GTS-NDCN	4.9	2.7	6.1	6.2	3.4
	SGODEv1	3.3	2.3	**2.0**	2.8	2.6
	SGODEv2	**2.4**	**1.5**	2.8	**2.3**	4.6
	SGODEv3	2.5	2.5	2.0	2.4	2.7

Table 2: MAPE of interpolation of continuous-time network dynamics. I: heat diffusion dynamics; II: mutualistic interaction dynamics; III: gene regulatory dynamics; Rand.: Random, Com.: Community. Those in black font indicate the best performance. The underline corresponds to the second-ranked value.

Results

Results on Three Dynamics The purpose of synthetic data experiment is to test the performance of different graph strategies (GTS: graph distribution inference, Adaptive: adaptive graph without negative information, NDCN: real graph) based on a unified base model. We conducted experiments involving both internal and external interpolation within continuous-time dynamics. The outcomes of our internal interpolation experiment are detailed in Table 2, while those from the external interpolation experiment are available in the Appendix. SGODE adeptly captures the evolution of various nodes across time and exhibits strong performance across diverse graph dynamics. Notably, methods employing learnable graphs are at least on par with NDCN when using the actual graph and, importantly, outperform approaches that lack a graph structure.

In the interpolation experiment, our method demonstrates superior accuracy compared to other techniques, particularly the graph inference methods Adp-NDCN and GTS-NDCN. Conversely, the NDCN variant lacking a graph exhibits a significant loss value, indicating its incapability to learn dynamic models and underscoring the importance of aggregating neighborhood information. In extrapolation experiments, SGODE exhibits substantial superiority over No-graph, Adaptive, GTS, and NDCN in heat diffusion dynamics. Our method also consistently achieves top-tier outcomes in most extrapolation cases. The utilization of node embedding

METR-LA	15min			30min			60min		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
HA	4.16	7.80	13.0%	4.16	7.80	13.0%	4.16	7.80	13.0%
ARIMA	3.99	8.21	9.6%	5.15	10.45	12.7%	6.90	13.23	17.4%
VAR	4.42	7.89	10.2%	5.41	9.13	12.7%	6.52	10.11	15.8%
SVR	3.99	8.45	9.3%	5.05	10.87	12.1%	6.72	13.76	16.7%
FNN	3.99	7.94	9.9%	4.23	8.17	12.9%	4.49	8.69	14.0%
LSTM	3.44	6.30	9.6%	3.77	7.23	10.9%	4.37	8.69	13.2%
DCRNN	2.77	5.38	7.3%	3.15	6.45	8.8%	3.60	7.59	10.5%
GWNet	2.69	5.15	6.9%	3.07	6.22	8.4%	3.53	7.37	10.0%
LDS	2.75	5.35	7.1%	3.14	6.45	8.6%	3.63	7.67	10.3%
MTGNN	2.69	5.18	6.9%	3.05	6.17	8.2%	3.49	7.23	9.9%
GTSv	2.74	5.09	7.3%	3.11	6.02	8.7%	3.53	6.84	10.3%
GTS	2.64	4.95	6.8%	3.01	5.85	8.2%	3.41	6.74	9.9%
STEP	2.61	4.98	6.6%	2.96	5.97	8.0%	3.37	6.99	9.6%
MegaCRN	2.60	4.81	6.4%	3.00	5.74	7.8%	3.41	6.74	9.4%
SGODE-RNN	**2.54**	**4.74**	**6.3%**	**2.93**	**5.66**	**7.7%**	**3.33**	**6.58**	**9.3%**
PEMS-BAY	15min			30min			60min		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
HA	2.88	5.59	6.8%	2.88	5.59	6.8%	2.88	5.59	6.8%
ARIMA	1.62	3.30	3.5%	2.33	4.76	5.4%	3.38	6.50	8.3%
VAR	1.74	3.16	3.6%	2.32	4.25	5.0%	2.93	5.44	6.5%
SVR	1.85	3.59	3.8%	2.48	5.18	5.5%	3.28	7.08	8.0%
FNN	2.20	4.42	5.2%	2.30	4.63	5.4%	2.46	4.98	5.9%
LSTM	2.05	4.19	4.8%	2.20	4.55	5.2%	2.37	4.96	5.7%
DCRNN	1.39	2.95	2.9%	1.74	3.97	3.9%	2.07	4.74	4.9%
GWNet	1.30	2.74	2.7%	1.63	3.70	3.7%	1.95	4.52	4.6%
LDS	1. 33	2.81	2.8%	1.67	3.80	3.8%	1.99	4.59	4.8%
MTGNN	1.32	2.79	2.8%	1.65	3.74	3.7%	1.94	4.49	4.5%
GTSv	1.35	2.64	2.9%	1. 69	3.45	3.9%	1.99	4.05	4.7%
GTS	1.32	2.62	2.8%	1.64	3.41	3.6%	1.91	3.97	4.4%
STEP	1.36	2.73	2.8%	1.67	3.58	3.6%	1.99	4.20	4.6%
MegaCRN	1.33	2.59	2.8%	1.65	3.40	3.6%	1.92	4.04	4.5%
SGODE-RNN	**1.29**	**2.55**	**2.7%**	**1.61**	**3.35**	**3.6%**	**1.90**	**3.96**	**4.4%**

Table 3: Forecasting error on METR-LA and PEMS-BAY.

matrices to approximate the coefficient matrix, as opposed to directly learning the entire matrix, maintains accuracy without compromise. The marginal underperformance of our approach relative to NDCN in certain network dynamics may arise from the fact that these dynamics are generated from ground-truth graphs. By incorporating SGODE, we enhance the precision of the ODE-GNN model in predictive dynamics, especially in the context of interpolation prediction.

Results on Traffic Datasets *Forecasting quality.* For prediction accuracy, we compare SGODE-RNN with the previously mentioned methods. Following the conventions of DCRNN (Li et al. 2018), GTS (Shang, Chen, and Bi 2021), and STG-NCDE (Choi et al. 2022), we present results for 15min, 30min, and 60min on METR-LA and PEMS-BAY datasets in Table 3, while average outcomes on PeMSD4 and PeMSD8 are reported in Table 4.

We have observed the following phenomena: 1) We examine the learned $\boldsymbol{K}$ and find that both positive and negative relationships exist. The results demonstrates that our proposed approach is capable of capturing signed information (Details in the Appendix). 2) On the METR-LA and PEMS-BAY datasets, SGODE significantly outperforms DCRNN, and it also surpasses state-of-the-art traffic flow prediction methods. This showcases the effectiveness of our proposed strategy in leveraging signed information and the positive benefits of embedding such information.

Since our approach relies on ODEs, we compare our experimental results with STNODE and STG-NCDE, as shown in Table 4. The results of our experiments convincingly showcase the efficacy of our method, while our variants further unlock the potential of the STG-NCDE model. Moreover, our proposed SGODE-RNN demonstrates remarkable competitiveness. Notably, it is observed that the MAPE and MSE metrics of GTS and SGODE-RNN do not align with the MAE metric on the PeMSD4 dataset, possibly due to the presence of some minima close to zero in this dataset.

Irregular traffic forecasting. In practice, data recording and storage errors can lead to the unavailability of specific data (Choi et al. 2022). We compare STG-SGCDE and STG-NCDE on PeMSD4 and PeMSD8 datasets with 10%, 30%,

Model	PeMSD4			PeMSD8		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE
HA	38.03	59.24	27.88%	34.86	59.24	27.88%
ARIMA	33.73	48.80	24.18%	31.09	44.32	22.73%
VAR	24.54	38.61	17.24%	19.19	29.81	13.10%
DCRNN	21.22	33.44	14.17%	16.82	26.36	10.92%
GWNet	24.89	39.66	17.29%	18.28	30.05	12.15%
STGODE	20.84	32.82	13.77%	16.81	25.97	10.62%
STG-NCDE	19.21	31.09	12.76%	15.45	24.81	9.92%
GTS	19.36	32.99	13.54%	14.82	**23.80**	9.52%
MegaCRN	18.92	31.90	12.89%	14.91	23.97	9.60%
SGCDE*	19.06	**30.96**	**12.65%**	15.34	24.44	9.92%
SGODE*	**18.81**	31.57	12.87%	**14.55**	23.85	**9.30%**

Table 4: Forecasting error on PeMSD4 and PeMSD8. SGCDE*: STG-SGCDE, SGODE*: SGODE-RNN.

Missing rate		PeMSD4		PeMSD8	
		NCDE*	SGCDE*	NCDE*	SGCDE*
10%	MAE	19.61	**19.15**	17.21	**15.67**
	RMSE	31.55	**31.05**	26.96	**24.95**
	MAPE	13.02%	**12.96%**	10.68%	**10.05%**
30%	MAE	**19.37**	19.41	17.42	**16.16**
	RMSE	**31.27**	31.29	27.48	**25.15**
	MAPE	13.12%	**12.86%**	11.03%	**10.86%**
50%	MAE	20.35	**19.70**	17.21	**15.88**
	RMSE	32.41	**31.86**	26.93	**25.11**
	MAPE	13.50%	**13.07%**	11.02%	**10.33%**

Table 5: Forecasting error on irregular PeMSD4 and PeMSD8. NCDE*: STG-NCDE, SGCDE*: STG-SGCDE.

and 50% data corruption, and the results are presented in Table 5. Our experimental setup is based on STG-NCDE. The results indicate that our approach exhibits greater robustness to irregular datasets compared to STG-NCDE. This underscores the accuracy of our method in fitting continuous dynamics.

Error for each horizon. We present the error analysis of each horizon prediction in Fig.1, where we predict a total of 12 horizons. It is evident that the level of error is highly correlated with forecast time. Across all horizons, SGODE-RNN outperforms both baseline models GTS and STG-NCDE in terms of lower error rates. While our proposed methods STG-SGCDE and SGODE-RNN exhibit superior performance in the initial few horizons, their advantage over STG-NCDE and GTS gradually diminishes as forecasting progresses.

Ablation Study

We verify the validity of our proposed two key components, the designed $\boldsymbol{K}$ and $\boldsymbol{B}$. We denote the SGODE-RNN with negative links removed and without its trend $\boldsymbol{B}$ as *Positive1* and *without-*$\boldsymbol{B}$, respectively. *Only FF* performs feature extraction solely on the final ODE result, without extracting features from the initial state and intermediate states. *Positive2* refers to $\boldsymbol{K} = \boldsymbol{K}_{pos_1} + \boldsymbol{K}_{pos_2}$. We show the results on four datasets in Table 6.

Model	PeMSD4			PeMSD8		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE
Only FF	19.32	33.03	13.05%	15.07	24.41	10.05%
Without-$\boldsymbol{B}$	19.01	31.61	14.13%	15.18	24.15	9.97%
Positivev1	19.24	31.91	13.16%	14.81	23.95	9.57%
Positivev2	19.56	32.69	13.79%	15.76	24.84	11.49%
SGODE	**18.81**	**31.57**	**12.87%**	**14.55**	**23.85**	**9.30%**
Model	METR-LA			PEMS-BAY		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE
Only FF	2.92	5.64	7.74%	1.57	3.19	3.53%
Without-$\boldsymbol{B}$	2.95	5.71	8.05%	1.58	3.23	3.51%
Positivev1	2.89	5.60	7.81%	1.57	3.19	3.54%
Positivev2	2.93	5.69	8.13%	1.57	3.22	3.58%
SGODE	**2.88**	**5.52**	**7.61%**	**1.55**	**3.15**	**3.50%**

Table 6: Ablation study of SGODE-RNN.

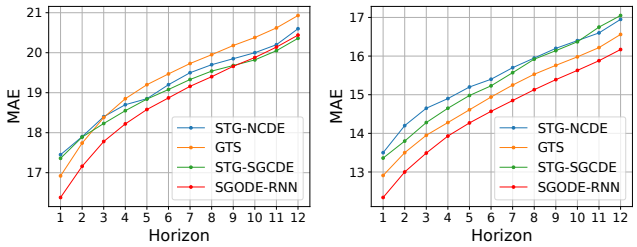
The experimental results demonstrate a significant performance decrease of SGODE when our strategies are not considered, except for the PEMS-BAY dataset, which exhibits a relatively minor improvement in performance related to signed relationships. In comparison to *Positive2*, we also eliminate the possibility of performance improvement due to the introduction of additional parameters.

Hyperparameters Analysis

We illustrate the impact of the diffusion steps m and the second dimension d of the node embedding matrix $\boldsymbol{E}_*$ in Fig. 2. Notably, higher diffusion steps (greater than 1) yield improved outcomes, suggesting the practicality of extracting intermediate states. Fig. 2b indicates that the model is sensitive to the node embedding dimension.

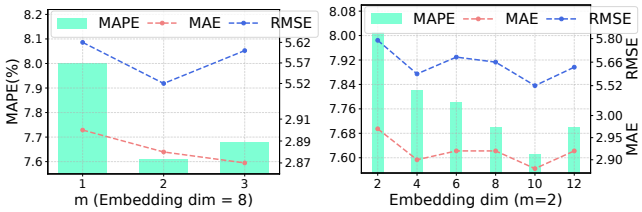
Efficiency Study

We assess the effectiveness of our approaches through comparisons with state-of-the-art methods. Fig. 3 showcases the efficiency of our methods in terms of parameters and runtime, compared to the state-of-the-art alternatives. Our methods (SGODE-RNN) achieve the lowest parameter count, meanwhile attain the minimum overall MAE. Additionally, our



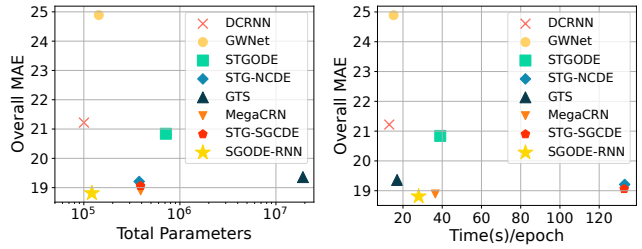
(a) MAE on PeMSD4 (b) MAE on PeMSD8

Figure 1: Prediction error at each horizon. More results in other datasets are in Appendix.



(a) Fix dim = 10 and vary m. (b) Fix m = 2 and vary dim.

Figure 2: Sensitivity analysis of m and d on METR-LA. More results in other datasets are shown in Appendix.



(a) Parametric efficiency (b) Runtime efficiency

Figure 3: Efficiency evaluation on PEMS04.

approach demonstrates reasonable runtime performance. Notably, methodologies incorporating ODE solvers result in extended training durations, while the supplementary runtime introduced by SGODE (STG-SGCDE) remains minimal.

Related Work

Lately, a noteworthy trend has emerged that involves the fusion of ODEs with graph neural networks to acquire insights into continuous-time dynamics. Furthermore, a concerted effort has been dedicated to incorporating graph structures into dynamic systems. Noteworthy endeavors have been made in this domain to extend NODE (Chen et al. 2018) to harness the rich graph structure (Hwang et al. 2021; Poli et al. 2019). GDE (Poli et al. 2019) extends the reach of GNNs into the continuous field by leveraging ODEs to learn input-output relationships. CGNN (Xhonneux, Qu, and Tang 2020) introduces a continuous message-passing layer, defining derivatives as combined representations of both the current and initial nodes. LG-ODE (Huang, Sun, and Wang 2020) employs neighborhood information to gather dynamic contextual cues, addressing scenarios where node states may not be observable over time. NDCN (Zang and Wang 2020) ingeniously combines ODEs and GNNs for modeling continuous-time dynamics. STGODE (Fang et al. 2021) captures spatio-temporal dynamics using tensor-based ODEs. Moreover, a distinct strand of research endeavors to extend ODEs to the realm of dynamic graphs. CG-ODE (Huang, Sun, and Wang 2021) integrates coupled ODEs to model dynamics based on edges and nodes, respectively. Jin *et al.* (Jin, Li, and Pan 2022) introduce explicit temporal dependencies through ODEs, showcasing their efficacy in graph-based modeling tasks and underlining the significance of graph representation. Nevertheless, the presumption of a known exact graph structure in advance is often challenging to maintain. STG-NCDE (Choi et al. 2022) introduces a dynamic approach with two NCDEs, adeptly handling spatial and temporal information using an adaptive normalized adjacency matrix. In a similar vein, MTGODE (Jin, Li, and Pan 2022) abstracted input sequences into dynamic graphs, where node features evolve over time alongside an unspecified graph structure.

Conclusions

We introduce a simple yet effective framework and extend its application to more intricate scenarios. Our investigation substantiates the inherent capability of the proposed methodology to effectively capture and leverage signed information. Furthermore, we underscore the method's adaptability, showcasing its effortless integration into a diverse range of cutting-edge dynamic modeling methodologies. Through comprehensive experimentation encompassing both synthetic and real-world datasets, we unveil the untapped potential of incorporating signed relations. This integration results in notable enhancements in performance, particularly in the context of short-term or interpolation prediction tasks. However, in cases where the graph size is substantial, challenges arise in accurately learning the signed graph, potentially leading to overfitting to local optima.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 62206205, 62206207, in part by the Young Talent Fund of Association for Science and Technology in Shaanxi, China under Grant 20230129, in part by the Guangdong High-level Innovation Research Institution Project under Grant 2021B0909050008, and in part by the Guangzhou Key Research and Development Program under Grant 202206030003.

References

Bai, L.; Yao, L.; Li, C.; Wang, X.; and Wang, C. 2020. Adaptive Graph Convolutional Recurrent Network for Traffic Forecasting. In *NeurIPS*, volume 33, 17804–17815.

Barabási, A.-L.; and Albert, R. 1999. Emergence of scaling in random networks. *Science*, 286(5439): 509–512.

Chen, C.; Petty, K.; Skabardonis, A.; Varaiya, P.; and Jia, Z. 2001. Freeway performance measurement system: mining loop detector data. *Transportation Research Record*, 1748(1): 96–102.

Chen, R. T. Q.; Rubanova, Y.; Bettencourt, J.; and Duvenaud, D. K. 2018. Neural Ordinary Differential Equations. In *NeurIPS*, volume 31.

Choi, J.; Choi, H.; Hwang, J.; and Park, N. 2022. Graph neural controlled differential equations for traffic forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 6367–6374.

Chung, J.; Gulcehre, C.; Cho, K.; and Bengio, Y. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Deng, A.; and Hooi, B. 2021. Graph neural network-based anomaly detection in multivariate time series. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, 4027–4035.

Derr, T.; Ma, Y.; and Tang, J. 2018. Signed graph convolutional networks. In *2018 IEEE International Conference on Data Mining (ICDM)*, 929–934. IEEE.

Erdős, P.; Rényi, A.; et al. 1960. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1): 17–60.

Fang, Z.; Long, Q.; Song, G.; and Xie, K. 2021. Spatial-temporal graph ode networks for traffic flow forecasting. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, 364–373.

Feng, M.; Hou, H.; Zhang, L.; Guo, Y.; Yu, H.; Wang, Y.; and Mian, A. 2023. Exploring Hierarchical Spatial Layout Cues for 3D Point Cloud based Scene Graph Prediction. *IEEE Transactions on Multimedia*.

Fortunato, S. 2010. Community detection in graphs. *Physics Reports*, 486(3-5): 75–174.

Franceschi, L.; Niepert, M.; Pontil, M.; and He, X. 2019. Learning discrete structures for graph neural networks. In *International conference on machine learning*, 1972–1982. PMLR.

Huang, Z.; Sun, Y.; and Wang, W. 2020. Learning continuous system dynamics from irregularly-sampled partial observations. *Advances in Neural Information Processing Systems*, 33: 16177–16187.

Huang, Z.; Sun, Y.; and Wang, W. 2021. Coupled graph ode for learning interacting system dynamics. In *The 27th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*.

Hwang, J.; Choi, J.; Choi, H.; Lee, K.; Lee, D.; and Park, N. 2021. Climate Modeling with Neural Diffusion Equations. In *ICDM*, 230–239.

Jhin, S. Y.; Jo, M.; Kong, T.; Jeon, J.; and Park, N. 2021a. ACE-NODE: Attentive co-evolving neural ordinary differential equations. In *KDD*, 736–745.

Jhin, S. Y.; Lee, J.; Jo, M.; Kook, S.; Jeon, J.; Hyeong, J.; Kim, J.; and Park, N. 2022. Exit: Extrapolation and interpolation-based neural controlled differential equations for time-series classification and forecasting. In *Proceedings of the ACM Web Conference 2022*, 3102–3112.

Jhin, S. Y.; Shin, H.; Hong, S.; Jo, M.; Park, S.; Park, N.; Lee, S.; Maeng, H.; and Jeon, S. 2021b. Attentive Neural Controlled Differential Equations for Time-series Classification and Forecasting. In *ICDM*, 250–259.

Jiang, R.; Wang, Z.; Yong, J.; Jeph, P.; Chen, Q.; Kobayashi, Y.; Song, X.; Fukushima, S.; and Suzumura, T. 2023. Spatio-Temporal Meta-Graph Learning for Traffic Forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(7): 8078–8086.

Jin, M.; Li, Y.-F.; and Pan, S. 2022. Neural Temporal Walks: Motif-Aware Representation Learning on Continuous-Time Dynamic Graphs. In Oh, A. H.; Agarwal, A.; Belgrave, D.; and Cho, K., eds., *Advances in Neural Information Processing Systems*.

Jin, M.; Zheng, Y.; Li, Y.-F.; Chen, S.; Yang, B.; and Pan, S. 2023. Multivariate Time Series Forecasting With Dynamic Graph Neural ODEs. *IEEE Transactions on Knowledge and Data Engineering*, 35(9): 9168–9180.

Karlebach, G.; and Shamir, R. 2008. Modelling and analysis of gene regulatory networks. *Nature reviews Molecular cell biology*, 9(10): 770–780.

Kidger, P.; Morrill, J.; Foster, J.; and Lyons, T. 2020. Neural controlled differential equations for irregular time series. *Advances in Neural Information Processing Systems*, 33: 6696–6707.

Kipf, T.; Fetaya, E.; Wang, K.-C.; Welling, M.; and Zemel, R. 2018. Neural relational inference for interacting systems. In *International Conference on Machine Learning*, 2688–2697. PMLR.

Li, Y.; Yu, R.; Shahabi, C.; and Liu, Y. 2018. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In *International Conference on Learning Representations*.

Lieberman, E.; Hauert, C.; and Nowak, M. A. 2005. Evolutionary dynamics on graphs. *Nature*, 433(7023): 312–316.

Marbach, D.; Costello, J. C.; Küffner, R.; Vega, N. M.; Prill, R. J.; Camacho, D. M.; Allison, K. R.; Kellis, M.; Collins, J. J.; and Stolovitzky, G. 2012. Wisdom of crowds for robust gene network inference. *Nature Methods*, 9(8): 796–804.

Poli, M.; Massaroli, S.; Park, J.; Yamashita, A.; Asama, H.; and Park, J. 2019. Graph neural ordinary differential equations. *arXiv preprint arXiv:1911.07532*.

Rubanova, Y.; Chen, R. T. Q.; and Duvenaud, D. K. 2019. Latent Ordinary Differential Equations for Irregularly-Sampled Time Series. In *NeurIPS*, volume 32.

Shang, C.; Chen, J.; and Bi, J. 2021. Discrete graph structure learning for forecasting multiple time series. *arXiv preprint arXiv:2101.06861*.

Shao, Z.; Zhang, Z.; Wang, F.; and Xu, Y. 2022. Pre-training enhanced spatial-temporal graph neural network for multivariate time series forecasting. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 1567–1577.

Shi, G.; Altafini, C.; and Baras, J. S. 2019. Dynamics over signed networks. *SIAM Review*, 61(2): 229–257.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

Wasserman, S.; Faust, K.; et al. 1994. *Social network analysis: Methods and applications*. Cambridge university press.

Watts, D. J.; and Strogatz, S. H. 1998. Collective dynamics of 'small-world'networks. *Nature*, 393(6684): 440–442.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Philip, S. Y. 2020a. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1): 4–24.

Wu, Z.; Pan, S.; Long, G.; Jiang, J.; Chang, X.; and Zhang, C. 2020b. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *KDD*, 753–763.

Wu, Z.; Pan, S.; Long, G.; Jiang, J.; and Zhang, C. 2019. Graph WaveNet for Deep Spatial-Temporal Graph Modeling. In *IJCAI-19*, 1907–1913.

Xhonneux, L.-P.; Qu, M.; and Tang, J. 2020. Continuous graph neural networks. In *International Conference on Machine Learning*, 10432–10441. PMLR.

Zang, C.; and Wang, F. 2020. Neural dynamics on complex networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 892–902.

Zhao, L.; Song, Y.; Zhang, C.; Liu, Y.; Wang, P.; Lin, T.; Deng, M.; and Li, H. 2019. T-GCN: A temporal graph convolutional network for traffic prediction. *IEEE Transactions on Intelligent Transportation Systems*, 21(9): 3848–3858.