# Learning Dynamic Graph Embeddings with Neural Controlled Differential Equations

Tiexin Qin, Benjamin Walker, Terry Lyons, Hong Yan, *Life Fellow, IEEE* and Haoliang Li

**Abstract**—This paper focuses on representation learning for dynamic graphs with temporal interactions. A fundamental issue is that both the graph structure and the nodes own their own dynamics, and their blending induces intractable complexity in the temporal evolution over graphs. Drawing inspiration from the recent progress of physical dynamic models in deep neural networks, we propose *Graph Neural Controlled Differential Equations* (GN-CDEs), a continuous-time framework that jointly models node embeddings and structural dynamics by incorporating a graph enhanced neural network vector field with a time-varying graph path as the control signal. Our framework exhibits several desirable characteristics, including the ability to express dynamics on evolving graphs without piecewise integration, the capability to calibrate trajectories with subsequent data, and robustness to missing observations. Empirical evaluation on a range of dynamic graph representation learning tasks demonstrates the effectiveness of our proposed approach in capturing the complex dynamics of dynamic graphs. Our code is available at https://github.com/WonderSeven/graph-neural-cdes.

**Index Terms**—Dynamic graph, embedding learning, graph neural network, controlled differential equations

## 1 Introduction

GRAPH representation learning analyzes complex structured data by representing node attributes and relationships in a low-dimensional vector space. In recent years, it has attracted increasing attention owing to the prevalence of graph-structured data. The use of deep neural networks, particularly graph neural networks (GNNs), has further facilitated the ability of graph representation learning to represent nodes. For example, GNNs have been used to study social media networks [1, 2], protein interactions [3], traffic flow forecasting [4], and neuroscience [5].

Many applications of graph representation learning involve temporal interactions, yet most existing methods do not consider such dynamics. As [6] pointed out, ignoring the temporal evolution in dynamic graphs can result in suboptimal performance. In certain scenarios, the dynamic structure holds key insights into the system. For example, when using a pandemic model to predict the spread of infection, the evolution of social relationships due to human events (*e.g.*, immigration, travel, education) must be taken into account [7]. In another example, malignant cells within tumors secrete proteins that influence neighboring stromal cells and create an environment conducive to their growth and metastasis [8]. For more examples, see [9].

In this article, we focus upon the realm of dynamic graph representation learning, where explicitly modeling both the time- and node-dependent interactions is generally required to better represent temporal evolution and the dynamic nature of the data. Despite the importance, it can be rather challenging to capture both of these dynamics effectively, especially when the changes are continuous-time and non-linear. Early work approached this by treating a dynamic graph as a sequence of snapshots [10, 11, 12], adapting existing GNNs through temporal batching. However, such methods would introduce artificial temporal boundaries. Moreover, temporal graph networks have emerged which treat the graph as an event stream and incorporate time encoding into message passing for continuous-time modeling [6, 13, 14, 15]. While these methods improve dynamic modeling, they are often limited to providing node temporal representations at discrete interaction timestamps and suffer from the staleness problem [9]. Recently, neural differential equations have been introduced to model node and edge representations that evolve over time [16, 17, 18, 19]. Nevertheless, these approaches still rely on a static graph structure or segments of static graphs for dynamic inference.

To address these limitations, we propose a novel and unified framework for dynamic graph representation learning that handles both the *structural dynamics* and *intrinsic node dynamics* simultaneously, wherein node embeddings are considered to undergo a continuous evolution over time, grounded in Neural Controlled Differential Equations (Neural CDEs) [20] built for continuous time-series modeling. Neural CDEs are a powerful concept that have the desired calibration ability with subsequent data, robustness to missing values, and memory-efficient training from adjoint-based backpropagation. We extend this concept to dynamic graphs, which we refer to as Graph Neural Controlled Differential Equations (GN-CDEs). Specifically, we construct a continuous-time path for the evolving graph topology, which serves as the control signal driving our neural controlled differential equation. After that, we incorporate the graph structure directly into the vector field to bias the learned dynamics towards solutions that are conditioned on the current graph structure. Based on these, our model enhances its ability to capture the causal effects inherent

- *Tiexin Qin, Hong Yan and Haoliang Li are with the Department of Electrical and Engineering, City University of Hong Kong, Hong Kong. Email: tiexinqin2-c@my.cityu.edu.hk, ityan@cityu.edu.hk, haoliang.li@cityu.edu.hk*
- *Benjamin Walker and Terry Lyons are with the Mathematical Institute, University of Oxford, Oxford, United Kingdom. Email: benjamin.walker2@balliol.ox.ac.uk, tlyons@maths.ox.ac.uk.*

*Manuscript received April 19, 2005; revised August 26, 2015.*
*(Corresponding author: Haoliang Li.)*

in dynamic graph structures. Excitingly, the capability of adjusting the predicted trajectories with incoming partially observed and irregularly sampled data from Neural CDEs still holds for our model, making it promising for practical usage. It is worth noting that, GN-CDEs are a flexible framework that we can leverage to tackle node attribute prediction, dynamic node classification, and temporal link prediction tasks with minor modifications. Besides, it can easily be extended to more complex graph structures (such as directed graphs and knowledge graphs). To further demonstrate the superiority of our method, we experimentally evaluate it on node attribute prediction and link affinity prediction tasks under evolving graph structures, where our method achieves favorable results across different setups. The contributions of this work are summarized below.

- We introduce GN-CDEs, a generic framework for modelling dynamic graphs by encoding their structure as paths within a controlled differential equation, allowing for continuous-time representation and evolution of node embeddings.
- We provide a theoretical analysis of our model, including guarantees on the existence and uniqueness of solutions, and establish formal equivalences and differences between GN-CDE and related Neural ODE- and CDE-based approaches.
- To ensure scalability, we propose a simplified version of GN-CDE that retains the key multiplicative interaction between the hidden state and control path while enabling efficient computation on large graphs.
- Experimental results verify that our proposed method achieves superior performance compared to a range of baseline methods across different graph tasks.

### 1.1 Related Works

**Graph Embedding Learning.** Graph representation learning, which aims to derive meaningful embeddings from the topological structure and node attributes of a graph, has been extensively studied for relational data analysis. Early works include graph factorization techniques [21] and random walk-based methods [22]. In line with the success of deep learning, Graph Neural Networks have emerged as powerful tools that learn node representations by aggregating information from neighboring nodes across multiple layers, achieving outstanding performance in various tasks [23, 24]. However, all these aforementioned models are designed for representation learning on static graphs, without temporal information.

In the real world, graphs are inherently dynamic rather than static. To name but a few, the interactions of users change in time on e-commerce and social platforms. However, due to the evolutionary complexity and dynamics of time-varying graphs, it is challenging to develop useful tools for dynamic graph representation learning. To address this, considerable research attention has been devoted to dynamic graph neural networks [25]. For example, TGN [13] introduces an event-based message passing framework for generating temporal embeddings, which is further enhanced by [26] through incorporating an identification between source and target nodes. Other representative methods, such as DyRep [14] and TGAT [6], leverage self-attention mechanisms to aggregate temporal-topological neighbors. Meanwhile, approaches like CAWN [27] and GraphMixer [15] focus on learning edge embeddings for temporal link prediction tasks. Despite their successes, these event-driven models are often limited to providing node temporal representations at discrete interaction timestamps and are prone to the staleness problem [9], where embeddings become outdated as the graph continues to evolve.

**Neural Differential Equations.** Neural differential equations offer a continuous formulation for modeling temporal dynamics on hidden representations via neural network parameterized vector fields [28]. One direction that bridges this framework with GNNs is continuous-depth GNNs, which utilize the integration procedure to simulate the continuous message-passing flow over graphs [29, 30]. Beyond this, graphs have also been used to encode spatial dependencies in spatial-temporal forecasting tasks, where differential equation models capture temporal dynamics and GNNs model spatial relations [18, 31]. Additionally, GNNs have been explored as learned discretizations of partial differential equation operators on unstructured meshes for simulating complex physics [32, 33, 34]. Notably, most existing works assume a fixed graph topology over time, leaving the incorporation of structural dynamics an unresolved challenge.

### 1.2 Paper organization and notations

The rest of this paper is structured as follows. In Section 2, we briefly introduce the background knowledge. In Section 3, we describe our main model, compare it with the competitors, and provide an approximation for efficient computation. The application in several representative graph representation learning tasks is also presented. In Section 4, we report the empirical performance of our model in node attribute prediction task. Finally, we conclude in Section 5. Complete proofs of our theoretical results follow thereafter in the appendix.

Before continuing, we introduce several notations used throughout the paper. First of all, we use lower-case letters to denote scalars, bold lower-case letters to denote vectors, and bold upper-case letters to denote matrices. For a matrix $\mathbf{X}$, we represent the $i$-th row of $\mathbf{X}$ as $\mathbf{X}^{(i)}$, and the element at the $i$-th row and $j$-th column as $\mathbf{X}^{(i,j)}$.

## 2 Preliminary

This section briefly reviews the basic definitions and common manners to learn graph embeddings, and then presents two typical neural differential equations.

### 2.1 Graph Embedding Learning

**Static graph.** A static graph only contains a fixed topological structure. Let a static graph represented as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ where $\mathcal{V}$ is the set of nodes, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges. Let $v_i \in \mathcal{V}$ denote a node and $e_{ij} \in \mathcal{E}$ denote an edge between node $v_i$ and $v_j$, $i, j \in \{1, \ldots, |\mathcal{V}|\}$. Then the topology of the graph can be represented by an adjacency matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ where $\mathbf{A}^{(i,j)} = 1$ if $e_{ij} \in \mathcal{E}$ otherwise 0. In most complex scenarios, the graph is equipped with a node attribute matrix $\mathbf{F} = \{\mathbf{F}^{(i)}\}_{i=1}^{|\mathcal{V}|}, \mathbf{F}^{(i)} \in \mathbb{R}^m$ and edge feature matrix $\mathbf{E} = \{\mathbf{E}^{(i,j)}\}_{i,j=1}^{|\mathcal{V}|}, \mathbf{E}^{(i,j)} \in \mathbb{R}^w$. Graph embedding learning for static graphs is to create an embedding $\boldsymbol{z}(v_i)$ for each node $v_i$ following a specified aggregation rule such that the specific local topology and node intrinsic information can be captured, formally

$$\boldsymbol{z}(v_i) = \sum_{j,\ \mathbf{A}^{(i,j)}=1} h(\mathrm{msg}(\mathbf{F}^{(i)}, \mathbf{F}^{(j)}, \mathbf{E}^{(i,j)}), \mathbf{F}^{(i)}),$$

where msg and $h$ are predefined or learnable functions working for message passing and aggregation, respectively.

**Dynamic graphs.** Dynamic graphs can be broadly categorized into *continuous-time dynamic graphs* and *discrete-time dynamic graphs* [9]. A continuous-time dynamic graph $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t)$ consists of an evolving set of nodes $\mathcal{V}_t$, edges $\mathcal{E}_t$, and potentially node and edge attributes $\mathbf{F}_t$ and $\mathbf{E}_t$. A discrete-time dynamic graph is a sequence of observations from a continuous-time dynamic graph, $\{\mathcal{G}_{t_i}\}_{i=0}^{N}$. Dynamic graphs exhibit structural dynamics arising from edge addition or deletion, node addition or deletion events, and node intrinsic dynamics due to transformations in node or edge features over time. As a result, the adjacency matrix $\mathbf{A}_t$ is also time varying. In this work, we embark on representation learning for continuous-time dynamic graphs with irregularly sampled observations. We start with undirected graphs ($\mathbf{A}_t$ is symmetric) without time-varying node attributes and edge features, then we discuss the extensions to more subtle graph structures. It is noteworthy that our proposed method can also handle tasks on discrete-time dynamic graphs in a seamless manner.

### 2.2 Neural Differential Equations

**Neural ordinary differential equations (Neural ODEs).** Neural ordinary differential equations [28] are the continuous-depth analogue to residual neural networks. Let $f_\theta : \boldsymbol{x} \to \boldsymbol{y}$ be a function mapping with some learnable parameters $\theta$, and $\zeta_\theta$ and $\ell_\theta$ are two linear maps. Neural ODEs are defined as

$$\boldsymbol{z}_t = \boldsymbol{z}_0 + \int_0^t f_\theta(\boldsymbol{z}_s)\mathrm{d}s \quad \text{and} \quad \boldsymbol{z}_0 = \zeta_\theta(\boldsymbol{x}), \tag{1}$$

where $\boldsymbol{y} \approx \ell_\theta(\boldsymbol{z}_T)$ can be utilized to approximate the desired output. In this formula, the solution $\boldsymbol{z}_t$ is determined by the initial condition on $\boldsymbol{z}_0$ when $\theta$ has been learned. There exists no direct way to modify the trajectory given subsequent observations, let alone tackle structural dynamics in the data generation procedure, making the plain Neural ODEs not suitable for dynamic graph setups.

**Neural controlled differential equations (Neural CDEs).** Neural controlled differential equations [20] are the continuous-time analogue to recurrent neural networks and provide a natural method for modeling temporal dynamics with neural networks.

Provide an irregularly sampled time series $\boldsymbol{x} = ((t_0, \boldsymbol{x}_{t_0}), (t_1, \boldsymbol{x}_{t_1}), \ldots, (t_N, \boldsymbol{x}_{t_N}))$, with each $t_k \in \mathbb{R}$ the time stamp of the observation $\boldsymbol{x}_{t_k} \in \mathbb{R}^v$ and $t_0 < \cdots < t_N$. Let $X : [t_0, t_N] \to \mathbb{R}^{v+1}$ be a continuous function of bounded variation with knots at $t_0, \ldots, t_N$ such that $X_{t_k} = (t_k, \boldsymbol{x}_{t_k})$. Let $f_\theta : \mathbb{R}^w \to \mathbb{R}^{w\times(v+1)}$ and $\zeta_\theta : \mathbb{R}^{v+1} \to \mathbb{R}^w$ be neural networks depending on their own learnable parameters $\theta$. Then Neural CDEs are defined as

$$\boldsymbol{z}_t = \boldsymbol{z}_{t_0} + \int_{t_0}^t f_\theta(\boldsymbol{z}_s)\mathrm{d}X_s \quad \text{for } t \in (t_0, t_N], \tag{2}$$

where $\boldsymbol{z}_{t_0} = \zeta_\theta(t_0, \boldsymbol{x}_{t_0})$ and $\boldsymbol{z}_t$ is the solution of the CDE. A key difference from Neural ODEs is the interpolation of observations to form a continuous path $X_s$. This allows for the natural inclusion of time-varying data into the integration process, thereby enabling the trajectory of the system to adapt according to subsequent observations.

## 3 Methodology

In this section, we first present the embedding learning problem under evolving graphs. Then we introduce our proposed differential model. After that, we provide the applications to several representative graph-related tasks (*e.g.*, node attributes prediction, dynamic node classification, temporal link prediction).

### 3.1 Problem Setup

Consider a dynamic graph generated following an underlying continuous procedure that we only observe a sequence of irregularly sampled graph snapshots $\mathcal{G} = \{(t_0, G_{t_0}), \ldots, (t_N, G_{t_N})\}$, with each $t_k \in \mathbb{R}$ the time stamp of the observed graph $G_{t_k}$ and $t_0 < \cdots < t_N$. Among these observations, a graph snapshot $G_{t_k} = \{\mathcal{V}, \mathcal{E}\}$ is comprised of nodes $\mathcal{V} = \{v_1, \ldots, v_{|\mathcal{V}|}\}$ and edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ (we assume all snapshots share a common node set and edge set, and omit the subscript for simplicity). Commonly, we can represent the graph topological information for graph $G_{t_k}$ via a time-specified adjacency matrix $\mathbf{A}_{t_k} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ that each interaction $e_{ij} \in \mathcal{E}$ is valued in $\mathbf{A}_{t_k}$ where $i, j \in \{1, \ldots, |\mathcal{V}|\}$. Our goal is to learn a non-linear dynamical system on the dynamic graph $\mathcal{G}$ based on the observations, formally the dynamics follow the form:

$$\mathbf{Z}_t = \mathbf{Z}_{t_0} + \int_{t_0}^t f(\mathbf{Z}_s)\mathrm{d}\mathbf{X}_s \quad \text{for } t \in (t_0, t_N], \tag{3}$$

where $\mathbf{Z}_t = \{\boldsymbol{z}_t(v_i)\}_{i=1}^{|\mathcal{V}|}, \mathbf{Z}_t \in \mathbb{R}^{|\mathcal{V}|\times d}$ is output node embedding matrix, $\mathbf{X}$ is an input signal path defined on $[t_0, t_N]$ which comprises the evolving topology of $\mathcal{G}$. The subscript notation here refers to function evaluation over time. Combining the model with an assigning function $\tau : \mathbb{R}^d \to \mathbb{R}^c$ on $\mathcal{G}$, such that $\mathbf{Y}_t = \tau(\mathbf{Z}_t), \mathbf{Y}_t \in \mathbb{R}^{|\mathcal{V}|\times c}$, allows applications in several machine learning tasks, such as node attribute prediction [35], dynamic node classification [11], temporal link prediction [15] *etc.* Very recently, some works incorporate graph convolutional networks with ODEs for the continuous inference [16, 17, 18], however, they degrade to an oversimplified setup where the neighborhood for nodes remains unchanged over time, making the proposed methods impractical for usage since the structural change could yield an unignorable effect on node embeddings. To make this problem solvable and the designed methods practical, we make the following assumption:

**Assumption 1** (Continuity)**.** The evolving path $\mathbf{z} : [t_0, t_N] \to \mathbb{R}^d$ of each node embedding is absolutely continuous.

This continuity assumption is standard for enabling differential equations [20, 28] and widely used by current dynamic graph approaches [16, 17, 18, 36].

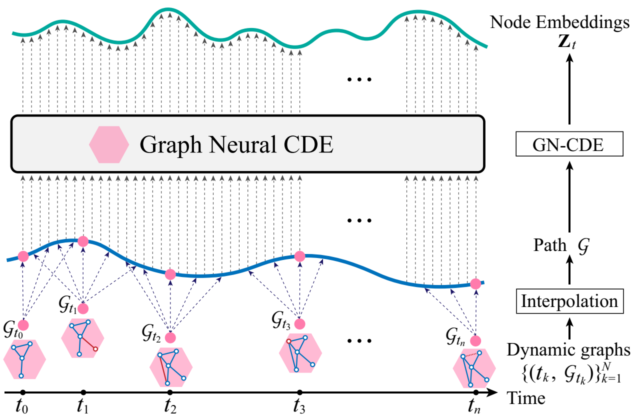



Fig. 1: An overview of our proposed GN-CDE. In the canonical setting, dynamic graphs are characterized by an evolving adjacency matrix but can be extended to more sophisticated scenarios as discussed in Section 3.6.

### 3.2 GN-CDEs: Graph Neural Controlled Differential Equations

An illustration of the proposed GN-CDE framework can be found in Fig. 1. Before introducing our differential equations, we need to prepare a continuously evolving path for graph structure first. Specifically, for a dynamic graph $G_{t_k}$ collected at time stamp $t_k$ endowed with adjacency matrix $\mathbf{A}_{t_k}$, we augment each interaction $e_{ij}$ in $\mathbf{A}_{t_k}$ by time stamp as $\hat{e}_{ij} = (t_k, e_{ij}) \in \mathbb{R}^2$ where $i, j \in \{1, \ldots, |\mathcal{V}|\}$ such that all these processed interactions can be represented by a time augmented adjacency matrix as $\hat{\mathbf{A}}_{t_k} = (t_k, \mathbf{A}_{t_k}) \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|\times 2}$. After that, we interpolate each possible interaction among two nodes independently utilizing the discrete observations to form a continuous path, this can be represented as $\hat{\mathbf{A}} : [t_0, t_N] \to \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|\times 2}$ such that $\hat{\mathbf{A}}_{t_k} = (t_k, \mathbf{A}_{t_k})$. In this paper, we assume $\hat{\mathbf{A}}$ to be piecewise continuously differentiable with bounded second derivative, thus many interpolation schemes can be employed [37].

Then, let $\zeta_\theta : \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|\times 2} \to \mathbb{R}^{|\mathcal{V}|\times d}$ and $f_\theta : \mathbb{R}^{|\mathcal{V}|\times d} \times \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|} \to \mathbb{R}^{|\mathcal{V}|\times d} \times \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|\times 2}$ be two graph neural networks. We define our controlled differential equation for dynamic graphs as follows

$$\mathbf{Z}_t = \mathbf{Z}_{t_0} + \int_{t_0}^{t} f_\theta(\mathbf{Z}_s, \mathbf{A}_s)\mathrm{d}\hat{\mathbf{A}}_s \quad \text{for } t \in (t_0, t_N], \tag{4}$$

where $\mathbf{Z}_{t_0} = \zeta_\theta(t_0, \mathbf{A}_{t_0})$ is treated as the initial value to avoid translational invariance. One can utilize another linear function $\ell_\theta$ to acquire the final prediction as $\tilde{\mathbf{Y}}_t = \ell_\theta(\mathbf{Z}_t), \tilde{\mathbf{Y}}_t \in \mathbb{R}^{|\mathcal{V}|\times c}$. The notation "$f_\theta(\mathbf{Z}_s, \mathbf{A}_s)\mathrm{d}\hat{\mathbf{A}}_s$" in Eq. (4) represents a tensor contraction between the final three axes of $f_\theta(\mathbf{Z}_s, \mathbf{A}_s)$ and all the axes of $\mathrm{d}\hat{\mathbf{A}}_s$. Our formula differs from the standard Neural CDE model [20] by explicitly capturing the causal effect of structural dynamics through the constructed graph path $\hat{\mathbf{A}}$, making it more suitable for dynamic graph scenarios, while Neural CDEs primarily emphasize sequential dependencies among observations rather than topology-driven effects. Under this modification, $\mathbf{A}_s$ in $f_\theta(\mathbf{Z}_s, \mathbf{A}_s)$ can bias the learned dynamics towards solutions conditioned on the current graph structure, and the derivative $\mathrm{d}\hat{\mathbf{A}}_s$ can indicate the magnitude and direction of instantaneous structural changes.

**Algorithm 1** Continuous inference of GN-CDE algorithm

**Input:** Sequentially observed topological structures of a dynamic graph $\{(t_0, \mathbf{A}_{t_0}), \ldots, (t_N, \mathbf{A}_{t_N})\}$, initial function $\zeta_\theta$, vector field $f_\theta$ and decoder $\ell_\theta$.
**Initializing:**
▷ $\hat{\mathbf{A}}$: Interpolate the time-augmented adjacency matrix;
▷ $\mathbf{Z}_{t_0} \leftarrow \zeta_\theta(t_0, \mathbf{A}_{t_0})$;
**Continuously inferring:**
▷ $\mathbf{Z}_t \leftarrow \text{ODESolve}(\mathbf{Z}_{t_0}, \hat{\mathbf{A}}, t_0, t_N, f_\theta)$ following Eq. (5);
▷ $\tilde{\mathbf{Y}}_t \leftarrow \ell_\theta(\mathbf{Z}_t)$;
**Return:** $\mathbf{Z}_t$, $\tilde{\mathbf{Y}}_t$

**Theorem 2.** *If $f_\theta$ is globally Lipschitz continuous and $\hat{\mathbf{A}}_t$ is a piecewise continuously differentiable interpolation, then the solution $\mathbf{Z}_t$ of 4 exhibits global existence and uniqueness.*

*Proof.* The proof is a straightforward application of the Picard–Lindelöf theorem [38]. □

Let $f_\theta$ be implemented as a Graph Neural Network of the form $f_\theta(\mathbf{Z}_s, \mathbf{A}_s) = \sigma(\mathbf{A}_s\mathbf{Z}_s^{(l)}\mathbf{W}^{(l)})$, where $\mathbf{W}^{(l)}$ denotes the parameters for $l$-th GCN layer and $\sigma$ is an activation function with Lipschitz constant 1 (*e.g.*, ReLU). Since each operation in this composition is Lipschitz continuous, it follows that $f_\theta$ is globally Lipschitz continuous. We elaborate on theoretical comparisons of different interpolation schemes for GN-CDE in Appendix C. In practice, we can leverage a regularized adjacency matrix of $\mathbf{A}_s$ to stabilize the algorithm learning [23].

**Evaluating.** Provided $\hat{\mathbf{A}}$ as piecewise continuously differentiable, Eq. (4) can be rewritten as

$$\mathbf{Z}_t = \mathbf{Z}_{t_0} + \int_{t_0}^{t} f_\theta(\mathbf{Z}_s, \mathbf{A}_s)\frac{\mathrm{d}\hat{\mathbf{A}}_s}{\mathrm{d}s}\mathrm{d}s \quad \text{for } t \in (t_0, t_N], \tag{5}$$

where $\mathbf{Z}_{t_0} = \zeta_\theta(t_0, \mathbf{A}_{t_0})$. This model can be interpreted as an ordinary differential equation by taking "$f_\theta(\mathbf{Z}_s, \mathbf{A}_s)\frac{\mathrm{d}\hat{\mathbf{A}}_s}{\mathrm{d}s}$" as a whole, and one can solve it using the same techniques for Neural ODEs [20]. The continuous inference procedure of GN-CDE using an ODE solver is depicted in Algorithm 1.

### 3.3 Properties

*Robustness to missing values.* GN-CDEs are capable of processing partially observed data. This is because each channel of input is independently interpolated between observations, allowing $\hat{\boldsymbol{A}}_s$ to be constructed in exactly the same manner as before.
*Calibration.* The model supports calibration through the incorporation of additional observations at intermediate time points. This property is inherited from Neural CDEs [20], which naturally allow for trajectory refinement as new data becomes available.
*Decoupling of forward passes from number of observations.*: In contrast to recurrent neural networks, the number of forward passes through the vector field of a GN-CDE is not determined by the number of observations in the time series, but by the choice of differential equation solver. This makes our model robust to oversampled data.

*Memory-efficient adjoint back-propagation.* Our model can continuously incorporate incoming data without interrupting the differential equation. As a result, memory-efficient adjoint backpropagation techniques may be employed for model training.

### 3.4 Comparison to Alternative Models

In this section, we compare and discuss our framework with two alternatives that also combine dynamic graph structure with differential equations.

① Neural CDE

An alternative of Eq. (4) is implementing the vector field without $\mathbf{A}_s$ as input, which follows the standard Neural CDE presented in Eq. (2). We formulize this as

$$\mathbf{Z}_t = \mathbf{Z}_{t_0} + \int_{t_0}^{t} f_\theta(\mathbf{Z}_s)\mathrm{d}\hat{\mathbf{A}}_s \quad \text{for } t \in (t_0, t_N], \tag{6}$$

where $\mathbf{Z}_{t_0} = \zeta_\theta(t_0, \mathbf{A}_{t_0})$. This model emphasizes the linear dependency on $\mathrm{d}\hat{\mathbf{A}}_s$.

**Theorem 3.** *Any equation of the form $\mathbf{Z}_t = \mathbf{Z}_{t_0} + \int_{t_0}^{t} f_\theta(\mathbf{Z}_s)\mathrm{d}\hat{\mathbf{A}}_s$ can be represented exactly by a Graph Neural Controlled Differential Equation of the form $\mathbf{Z}_t = \mathbf{Z}_{t_0} + \int_{t_0}^{t} f_\theta(\mathbf{Z}_s, \mathbf{A}_s)\mathrm{d}\hat{\mathbf{A}}_s$ and vice versa.*

*Proof.* Here we sketch the proof idea. Since the vector field $f_\theta$ in $\mathbf{Z}_t = \mathbf{Z}_{t_0} + \int_{t_0}^{t} f_\theta(\mathbf{Z}_s, \mathbf{A}_s)\mathrm{d}\hat{\mathbf{A}}_s$ takes both the $\mathbf{Z}_s$ and $\mathbf{A}_s$ as input, we can construct a stacked input $\boldsymbol{\beta}_s = \begin{bmatrix}\mathbf{Z}_s \\ \mathbf{A}_s\end{bmatrix}$ for alternative such that the newly built controlled differential equation on $\boldsymbol{\beta}_s$ is equivalent to the Neural CDE formulation presented in Eq. (6). The full proof of Theorem 3 is detailed in Appendix A.1. □

Although the dynamic graph can be learned by these two forms of CDE according to Theorem 3, they own different preferences during model learning stage. In the experimental part, we find that GN-CDE with non-linear dependency (as shown in Eq. (4)) performs better compared to the linear dependency variant (Eq. (6)). We conjecture that this is due to the fact that explicitly incorporating dynamic graph structure into the vector field allows for more precise control of information flow among nodes over time.

① Neural ODE

Another alternative to Eq. (4) could be directly incorporating the graph structure into the vector field of Neural ODEs presented in Eq. (1) and defining the graph neural ODE model as

$$\mathbf{Z}_t = \mathbf{Z}_{t_0} + \int_{t_0}^{t} f_\theta(\mathbf{Z}_s, \mathbf{A}_{\lfloor s \rfloor})\mathrm{d}s \quad \text{for } t \in (t_0, t_N], \tag{7}$$

where $\mathbf{Z}_{t_0} = \zeta_\theta(t_0, \mathbf{A}_{t_0})$, $\mathbf{A}_{\lfloor s \rfloor} = \mathbf{A}_{t_k}$ if $t_k \le s < t_{k+1}$.

**Theorem 4.** *Any equation of the form $\mathbf{Z}_t = \mathbf{Z}_{t_0} + \int_{t_0}^{t} f_\theta(\mathbf{Z}_s, \mathbf{A}_{\lfloor s \rfloor})\mathrm{d}s$ may be represented exactly by a Graph Neural Controlled Differential Equation of the form $\mathbf{Z}_t = \mathbf{Z}_{t_0} + \int_{t_0}^{t} f_\theta(\mathbf{Z}_s, \mathbf{A}_s)\mathrm{d}\hat{\mathbf{A}}_s$. However, the converse statement is not true.*

*Proof.* Theorem 3 allows us to identify any equation of the form $\mathbf{Z}_t = \mathbf{Z}_{t_0} + \int_{t_0}^{t} f_\theta(\mathbf{Z}_s, \mathbf{A}_s)\mathrm{d}\hat{\mathbf{A}}_s$ with an equivalent equation of the form $\mathbf{Z}_t = \mathbf{Z}_{t_0} + \int_{t_0}^{t} f_\theta(\mathbf{Z}_s)\mathrm{d}\hat{\mathbf{A}}_s$. The result then follows as a consequence of Theorem C.1 in [20]. □

According to Theorem 4, although the Neural ODE model can also take the dynamic graph structure into the vector field computational procedure, its representation ability is inferior to our proposed GN-CDE model.

### 3.5 Approximation of GN-CDE

Directly implementing Eq. (5) by following [20] will incur an undesirable computational burden due to the high dimensional tensor contraction. To ensure the scalability of our model to large neural networks, we incorporate approximations which reduce the computational complexity by leveraging a message passing mechanism. First, assuming a global time shared by all nodes, the time-augmented adjacency matrix can be simplified to $\mathrm{d}\hat{\mathbf{A}}_s = [\mathrm{d}\mathbf{A}_s, \mathrm{d}s]$. Second, we propose to fuse $\mathrm{d}\mathbf{A}_s$ together with $\mathbf{A}_s$ into the vector field using a transformation matrix to produce a new adjacency matrix $\tilde{\mathbf{A}}_s$, that contains both the current graph structure and the instantaneous structural change. This yields two advantages: 1) the information diffusion procedure by learnable parameters can be approximated via our adjusted adjacency matrix $\tilde{\mathbf{A}}_s$; 2) the dimension of output for $f_\theta$ can be significantly reduced, from $\mathbb{R}^{|\mathcal{V}|\times d} \times \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|+1}$ to $\mathbb{R}^{|\mathcal{V}|\times d}$. Formulaically, for a vector field parameterized by a $L$-layers graph neural network, the approximated equation that can be implemented much more efficiently as follows

$$\mathbf{Z}_t = \mathbf{Z}_{t_0} + \int_{t_0}^{t} \sigma\Big(\tilde{\mathbf{A}}_s\mathbf{Z}_s^{(L)}\mathbf{W}^{(L)}\Big)\mathrm{d}s \quad \text{for } t \in (t_0, t_N], \tag{8}$$

where $\tilde{\mathbf{A}}_s = \mathbf{W}^{(DR1)}\begin{bmatrix}\hat{\mathbf{A}}_s \\ \frac{\mathrm{d}\hat{\mathbf{A}}_s}{\mathrm{d}s}\end{bmatrix}\mathbf{W}^{(DR2)}$, and $\mathbf{W}^{(DR1)} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ and $\mathbf{W}^{(DR2)} \in \mathbb{R}^{2|\mathcal{V}|\times|\mathcal{V}|}$ are transformation matrices for the fusion. Additionally, $\mathbf{Z}_s^{(L)}$ is acquired iteratively by following the rule: $\mathbf{Z}_s^{(l)} = \sigma\big(\tilde{\mathbf{A}}_s\mathbf{Z}_s^{(l-1)}\mathbf{W}^{(l-1)}\big)$ for $l \in \{1, ..., L\}$, and $\sigma$ is ReLU activation function.

**Theorem 5.** *Given a sufficient number of copies, (8) has the same expressivity as a Graph Neural Controlled Differential Equation model with the form $\mathbf{Z}_t = \mathbf{Z}_{t_0} + \int_{t_0}^{t} f_\theta(\mathbf{Z}_s, \mathbf{A}_s)\mathrm{d}\hat{\mathbf{A}}_s$.*

*Proof.* The full proof of Theorem 5 is deferred to Appendix B; here we present an overview. Similarly to the proof of the universality of NCDEs [20, Theorem B.14], the proof of Theorem 5 proceeds by showing that you can approximate the truncated signature of the input path. Since linear maps on the signature can approximate any real-valued continuous function defined on compact sets of paths to arbitrary precision, this is sufficient to prove universality. Unlike the proof of NCDE universality, the proof of Theorem 5 proceeds by a direct construction of the weights necessary to recreate the truncated signature, with each copy calculating one signature term. □

In addition to the theoretical analysis of Theorem 5, we also provide an empirical comparison of this simplified version in Section 4 to evaluate its performance.

TABLE 1: Quantitative evaluation of prediction accuracy between GN-CDE and other baselines on node attribute prediction tasks. Here, the sum of extrapolation and interpolation results is presented for evaluating the prediction performance throughout the whole dynamic procedure. Best results are printed in boldface.

| Model | Algorithms | Grid | Random | Power Law | Small World | Community |
|---|---|---|---|---|---|---|
| Heat Diffusion | Neural ODE | 1.091 ± 0.344 | 0.629 ± 0.116 | 1.154 ± 0.189 | 1.093 ± 0.123 | 1.408 ± 0.091 |
| | Neural CDE | 0.962 ± 0.306 | 1.601 ± 0.471 | 1.642 ± 0.313 | 1.201 ± 0.179 | 1.857 ± 0.312 |
| | STG-NCDE | 0.861 ± 0.541 | 1.676 ± 0.260 | 2.311 ± 0.450 | 2.491 ± 0.313 | 1.922 ± 0.216 |
| | GN-CDE | **0.369 ± 0.134** | **0.521 ± 0.202** | **0.630 ± 0.135** | **0.484 ± 0.127** | **0.457 ± 0.112** |
| Gene Regulation | Neural ODE | 3.153 ± 0.562 | 3.732 ± 1.066 | 2.549 ± 0.226 | 2.252 ± 0.430 | 4.685 ± 0.759 |
| | Neural CDE | 2.967 ± 0.245 | 6.107 ± 3.202 | 2.764 ± 0.162 | 2.302 ± 0.591 | 5.325 ± 0.500 |
| | STG-NCDE | 6.554 ± 0.621 | 9.285 ± 0.808 | 3.917 ± 1.010 | 4.920 ± 1.247 | 8.278 ± 2.597 |
| | GN-CDE | **1.388 ± 0.262** | **2.193** ± 0.550 | **0.886 ± 0.072** | **1.331 ± 0.323** | **1.737 ± 0.260** |

### 3.6 Applications

**Node attributes prediction.** For the node attributes prediction task, we are equipped with a node attributes matrix $\mathbf{F}_{t_k} \in \mathbb{R}^{|\mathcal{V}|\times m}$ that contains $m$-dimensional attributes of the nodes for model training. As new edges and nodes emerge at time stamp $t_k$, both $\mathbf{A}_{t_k}$ and $\mathbf{F}_{t_k}$ evolve accordingly under the effects of graph structural dynamics and intrinsic dynamics of nodes. Our goal is to predict the node attributes, *i.e.*, to predict $\mathbf{F}_t$ at unseen time $t$ based on previous observations. To achieve this, we need to learn informative node representations $\mathbf{Z}_t \in \mathbb{R}^{|\mathcal{V}|\times d}$ that can be used for the prediction of the nodes attributes $\mathbf{F}_t$. The objective is to minimize the following expected loss

$$\min_{f_\theta,\ell_\theta} \mathbb{E}_t[\mathrm{Loss}(\mathbf{F}_t, \tilde{\mathbf{Y}}_t)] \quad \text{for } t \in (t_0, T], \tag{9}$$

where $\tilde{\mathbf{Y}}_t = \ell_\theta(\mathbf{Z}_t)$ is the prediction based on $\mathbf{Z}_t$ which is inferred by Eq. (4). We can use the (mean) absolute error or squared error to measure the mismatch between $\mathbf{F}_t$ and $\tilde{\mathbf{Y}}_t$. Here, $T$ can be either $T \le t_N$ which corresponds to interpolation prediction or $T > t_N$ which corresponds to extrapolation prediction.

**Dynamic node classification.** The task of node classification is leveraging the collected information at time stamps $\{t_0, \ldots, t_N\}$ to predict the label of nodes $\tilde{\mathbf{Y}}_t$ at time stamp $t, t > t_N$. The objective function for minimization is

$$\min_{f_\theta,\ell_\theta} \mathbb{E}_t[\mathrm{Loss}(\mathbf{Y}_t, \tilde{\mathbf{Y}}_t)] \quad \text{for } t \in (t_N, T], \tag{10}$$

where $\tilde{\mathbf{Y}}_t = \ell_\theta(\mathbf{Z}_t)$, $\ell_\theta$ is a MLP followed by a softmax activation function to obtain the class probability, and we can employ the cross-entropy loss for the measurement of prediction error. Commonly, this works on attributed graphs, which means the node attribute matrix $\mathbf{F}_{t_k}$ and edge feature matrix $\mathbf{E}$ are provided in advance. Thus, the controlled differential equation can be written as

$$\mathbf{Z}_t = \mathbf{Z}_{t_0} + \int_{t_0}^{t} f_\theta\Big(\mathbf{Z}_s, \mathbf{A}_s, \mathbf{F}_s, \mathbf{E}_s\Big)\mathrm{d}\hat{\mathbf{A}}_s \quad \text{for } t \in (t_0, t_N], \tag{11}$$

where $\mathbf{Z}_{t_0} = \zeta_\theta(t_0, \mathbf{A}_{t_0}, \mathbf{F}_{t_0}, \mathbf{E}_{t_0})$. In this equation, we need to interpolate the time augmented adjacency matrix as before and the node attributes $\mathbf{F}_{t_k}$ and edge attributes $\mathbf{E}_{t_k}$ to conduct the integral. Moreover, when $\mathbf{F}_{t_k}$ or $\mathbf{E}_{t_k}$ is not given, we can disable these terms in Eq. (11) accordingly.

**Temporal link prediction.** This task is to predict the existence of an edge $e_{ij}$ at unseen future time $t, t > t_N$. The node attribute matrix $\mathbf{F}_{t_k}$ and edge feature matrix $\mathbf{E}$ are also provided, thus we can follow the same setup designed for dynamic node classification tasks. To acquire the link probability among two nodes, we apply a MLP over the concatenation of the corresponding nodes' embeddings.

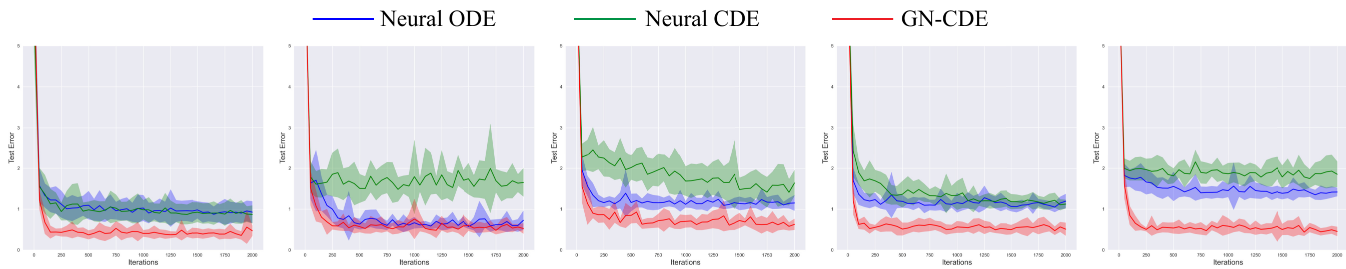
## 4 Experiments

In this section, we conduct a comprehensive set of experiments on node attribute prediction tasks to validate the effectiveness of our proposed GN-CDE model. More details of dataset construction and experimental setup can be found in the supplementary material.

### 4.1 Dynamic Node Property Prediction

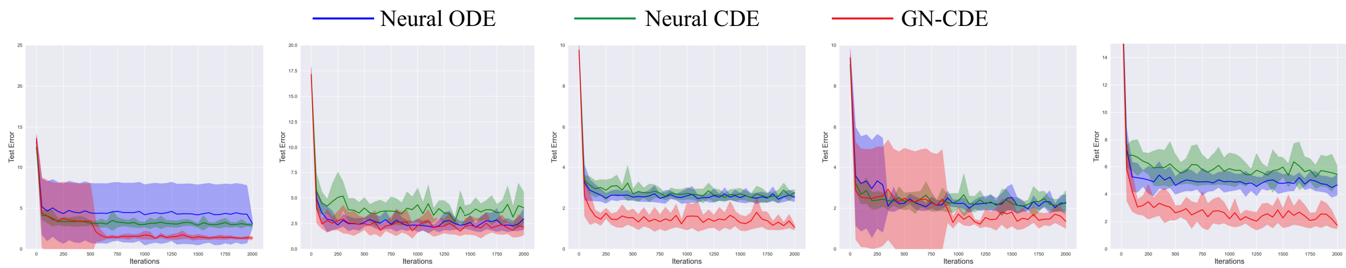
**Experimental Setup.** We consider two representative dynamic models: heat diffusion dynamics and gene regulatory dynamics. The underlying networks own 400 nodes and are initialized as Grid network, Random network, Power-law network, Small world network, and Community network, respectively. Next, some edges are randomly dropped or added occasionally to simulate dynamic environments. We irregularly sample 120 snapshots from the continuous-time dynamics to form the entire observations. The standard data splits presented in [16] are utilized, with 80 snapshots used for training, 20 snapshots for testing the interpolation prediction task, and 20 snapshots for testing the extrapolation prediction task.

The methods for comparison include Neural ODE (presented in Eq. (7)), Neural CDE (presented in Eq. (6)) and STG-NCDE [18]. We implement our method following Eq. (8) based on torchcde [20], which supports differentiable CDE solvers on GPUs. A natural cubic spline is employed to interpolate the observed graph structures continuously over time. We take $\tilde{\mathbf{A}}_s$ to be the sum of $\mathbf{A}_s$ and $\frac{\mathrm{d}\mathbf{A}_s}{\mathrm{d}s}$ for a tiny implementation. The vector field is parameterized using a single GCN layer followed by a ReLU activation function with the dimension of output node embeddings set as $d = 20$. To enable node property prediction, we further attach one linear layer to acquire the final node states. Model parameters are optimized by minimizing the F1 loss between the predicted and ground truth dynamics




(a) Grid (b) Random (c) Power Law (d) Small World (e) Community

Fig. 2: **Heat Diffusion:** The test errors of Neural ODE, Neural CDE and our GN-CDE models with respect to the optimization iteration count under five different graph structures: (a) grid, (b) Random, (c) power law, (d) small world and (e) community.




(a) Grid (b) Random (c) Power Law (d) Small World (e) Community

Fig. 3: **Gene Regulation:** The test errors of Neural ODE, Neural CDE and our GN-CDE models with respect to the optimization iteration count under five different graph structures: (a) grid, (b) Random, (c) power law, (d) small world and (e) community.

using the Adam optimizer for 2,000 iterations with an initial learning rate of 0.01. For a fair comparison, we keep the neural network architecture of the vector field the same for all baselines. We report the mean absolute error averaged over 10 runs with the standard deviation shown aside.

**Quantitative Results.** The results of our proposed GN-CDE model and other baseline methods in node attribute prediction tasks are summarized in Table 1. We can observe that, our GN-CDE model consistently achieves the lowest prediction error in both heat diffusion prediction tasks and gene regulation prediction tasks across diverse dynamic networks. The improvements over the Neural ODE and Neural CDE methods are marginal but consistent. Additionally, in terms of gene regulation prediction, all approaches exhibit higher prediction errors compared to heat diffusion prediction. This discrepancy can be attributed to the inherent complexity of gene regulation dynamics, as discussed in detail in Appendix E. Despite this, our proposed approach still outperforms the baseline methods by a significant margin. This demonstrates the effectiveness of our model in accurately modeling the whole dynamics under evolving graph structures, no matter for interpolation or for extrapolation.

**Convergence Analysis.** We then investigate the convergence of our model by reporting the prediction errors with respect to the number of iteration steps. The results for heat diffusion and gene regulation are depicted in Fig. 2 and Fig. 3, respectively. It can be seen that our GN-CDE model exhibits efficient convergence, commonly reaching a stable test error in less than 200 iterations. The convergence speed is comparable to that of the Neural ODE and Neural CDE methods, while demonstrating significantly lower prediction errors. This confirms the strong convergence capability of our proposed method. However, we also observe a noticeable oscillation for GN-CDE in gene regulation involving a small world graph structure. This arises from a single random seed, for which almost 50% of the structural changes happen during the final 20% of the time span. We conjecture that the high frequency of structural changes within a short duration renders the synthesized data difficult to fit.

TABLE 2: Quantitative evaluation of different interpolation schemes on Heat Diffusion dynamics over the grid and random networks.

| Control | Grid | Random |
|---|---|---|
| Linear | $0.453 \pm 0.152$ | $0.554 \pm 0.119$ |
| Rectilinear | $0.873 \pm 0.272$ | $0.777 \pm 0.148$ |
| Cubic Hermite | $0.491 \pm 0.075$ | $0.523 \pm 0.072$ |
| Natural Cubic | $\mathbf{0.369 \pm 0.134}$ | $\mathbf{0.521 \pm 0.202}$ |

**Parameter Sensitivity Analysis.** Different interpolation schemes directly affect the smoothness of the interpolated graph path for message passing in GN-CDE. To empirically evaluate their impact on model performance, we compare GN-CDE with linear, rectilinear, natural cubic, and cubic hermite interpolation strategies. The experimental results on heat diffusion dynamics over both grid and random networks are presented in Table 2. As seen, natural cubic interpolation consistently achieves the best performance, yielding the lowest error on both graph types (0.369 on grid and 0.521 on random networks). This suggests that smoother and higher-order continuous interpolation can better capture the underlying temporal evolution of node embeddings in dynamic graphs. In contrast, less smooth

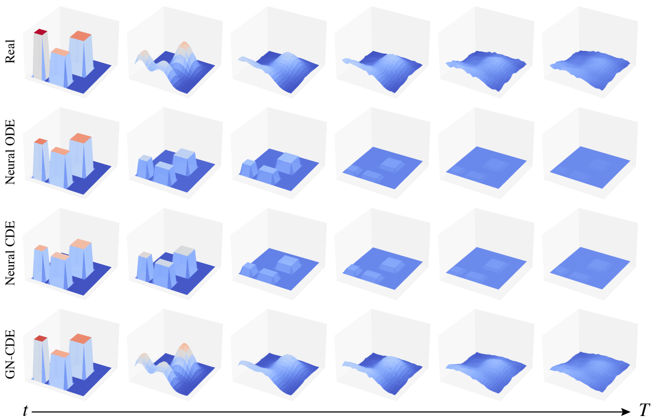



Fig. 4: Visualization of learned dynamics for heat diffusion over dynamic graphs. Our GN-CDE model fits the dynamics for the whole progress accurately. All experiments have been averaged over 10 random seeds with the stand deviation reported.

TABLE 3: Experimental results for dynamic node affinity prediction task on the tgbn-trade and tgbn-genre datasets. Best results are in bold, and second-best are underlined.

| Algorithms | tgbn-trade | tgbn-genre |
|---|---|---|
| JODIE [11] | 0.374 ± 0.09 | 0.350 ± 0.04 |
| TGAT [6] | 0.375 ± 0.07 | 0.352 ± 0.03 |
| CAWN [39] | 0.374 ± 0.09 | - |
| TCL [40] | 0.375 ± 0.09 | 0.354 ± 0.02 |
| GraphMixer [15] | 0.375 ± 0.11 | 0.352 ± 0.03 |
| DyGFormer [27] | 0.388 ± 0.64 | 0.365 ± 0.20 |
| DyRep [14] | 0.374 ± 0.001 | 0.351 ± 0.001 |
| TGN [13] | 0.374 ± 0.001 | 0.367 ± 0.058 |
| TGNv2 [26] | **0.736 ± 0.006** | 0.469 ± 0.002 |
| Neural ODE | 0.725 ± 0.022 | 0.465 ± 0.008 |
| Neural CDE | 0.721 ± 0.018 | <u>0.478 ± 0.022</u> |
| STG-NCDE [18] | 0.625 ± 0.021 | 0.438 ± 0.038 |
| GN-CDE | <u>0.731 ± 0.017</u> | **0.529 ± 0.015** |

schemes such as rectilinear interpolation result in higher errors, indicating poor capability in modeling the latent dynamics.

**Analysis on Instantaneous Structural Changes.** To assess the effectiveness of our approach in dealing with instantaneous structural changes, we compare the ground truth dynamics with the learned dynamics from our method on the heat diffusion task over a time-varying grid network. The results are illustrated in Fig. 4. Due to the instantaneous structural changes during the process of energy diffusion towards neighboring nodes, the node attribute surface of the ground truth displays several irregularities (as shown in the first panel of Fig. 4). Both the Neural ODE and Neural CDE methods fail to precisely fit the surface. This is due to their poor capability in handling structural changes. On the contrary, our GN-CDE model demonstrates favorable performance, accurately fitting the dynamics throughout the entire time span. This confirms the superiority of incorporating structural dynamics into both the vector field and differential term, particularly in handling instantaneous structural changes.

### 4.2 Dynamic Link Affinity Prediction

**Experimental Setup.** We evaluate our framework's performance on two real-world datasets, tgbn-trade and tgbn-genre from the Temporal Graph Benchmark (TGB) [41]. The task is to predict a node's future affinity towards other nodes based on the observed historical evolution of a temporal graph, which is particularly relevant in applications such as recommendation systems. Each dataset is split chronologically with the ratio of 70%, 15%, and 15% for training, validation, and testing, respectively.

In addition to the differential equation-based Neural ODE/CDE and STG-NCDE baselines, we further incorporate state-of-the-art temporal graph neural network-based dynamic graph models for comparison. These approaches include (1) JODIE [11]; (2) TGAT [6]; (3) CAWN [39]; (4) TCL [40]; (5) GraphMixer [15]; (6) DyGFormer [27]; (7) DyRep [14]; (8) TGN [13]; (9) TGNv2 [26]. We report the averaged NDCE@10 metric over 10 random seeds for all experiments.

**Results.** The results are summarized in Table 3. In general, neural differential equation-based methods outperform temporal graph neural network-based approaches by a large margin with one exception of TGN-V2, which is specifically tailored for this task via explicitly incorporating source-target identification into its message-passing process. Nevertheless, our proposed GN-CDE achieves a score of 0.731 on the tgbn-trade dataset, second only to TGN-V2, and the best performance of 0.529 on the tgbn-genre dataset, demonstrating its great effectiveness in capturing complex temporal dependencies and dynamic patterns in real-world scenarios.

## 5 Conclusion

In this paper, we propose a novel differential equations-based framework called GN-CDE for representation learning on continuous-time dynamic graphs. Specifically, GN-CDE creates graph paths within the controlled differential equation such that the graph structural dynamics can be naturally incorporated when conducting integration. This is a generic framework that we can apply it to solve various dynamic graph tasks and extend it to different graph types with minor modifications. Experimental results on node attribute prediction tasks across different underlying graph structures demonstrate the superiority of our proposed method compared to other baselines. In our future work, we will explore more challenging applications of dynamic graphs in real-word scenarios.

## Acknowledgments

This work is supported by the Hong Kong Innovation and Technology Commission (InnoHK Project CIMDA), the Hong Kong Research Grants Council (Project 11201825), and the Institute of Digital Medicine, City University of Hong Kong (Projects 9229503 and 9610460), in part by the Hong Kong Research Grants Council under Projects 21200522, 11200323 and 11203220, in part by the Hong

Kong Innovation and Technology Commission (Project GHP/044/21SZ), and in part by City University of Hong Kong 11207523, Sichuan Science and Technology Fund and 2025ZNSFSC0511.

## REFERENCES


[1] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin, "Graph neural networks for social recommendation," in *The world wide web conference*, 2019, pp. 417–426.

[2] A. Sankar, Y. Liu, J. Yu, and N. Shah, "Graph neural networks for friend ranking in large-scale social platforms," in *Proceedings of the Web Conference 2021*, 2021, pp. 2535–2546.

[3] P. Gainza, F. Sverrisson, F. Monti, E. Rodola, D. Boscaini, M. Bronstein, and B. Correia, "Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning," *Nature Methods*, pp. 184–192, 2020.

[4] S. Lan, Y. Ma, W. Huang, W. Wang, H. Yang, and P. Li, "Dstagnn: Dynamic spatial-temporal aware graph neural network for traffic flow forecasting," in *International Conference on Machine Learning*, 2022, pp. 11 906–11 917.

[5] A. Bessadok, M. A. Mahjoub, and I. Rekik, "Graph neural networks in network neuroscience," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[6] D. Xu, C. Ruan, E. Korpeoglu, S. Kumar, and K. Achan, "Inductive representation learning on temporal graphs," in *International Conference on Learning Representations*, 2020.

[7] L. Zhong, "A dynamic pandemic model evaluating reopening strategies amid covid-19," *PloS one*, p. e0248302, 2021.

[8] O. L. Podhajcer, L. Benedetti, M. R. Girotti, F. Prada, E. Salvatierra, and A. S. Llera, "The role of the matricellular protein sparc in the dynamic interaction between the tumor and the host," *Cancer and Metastasis Reviews*, pp. 523–537, 2008.

[9] S. M. Kazemi, R. Goel, K. Jain, I. Kobyzev, A. Sethi, P. Forsyth, and P. Poupart, "Representation learning for dynamic graphs: A survey." *J. Mach. Learn. Res.*, pp. 1–73, 2020.

[10] P. Goyal, N. Kamra, X. He, and Y. Liu, "Dyngem: Deep embedding method for dynamic graphs," *arXiv preprint arXiv:1805.11273*, 2018.

[11] S. Kumar, X. Zhang, and J. Leskovec, "Predicting dynamic embedding trajectory in temporal interaction networks," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 1269–1278.

[12] A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, T. Schardl, and C. Leiserson, "Evolvegcn: Evolving graph convolutional networks for dynamic graphs," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020, pp. 5363–5370.

[13] E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, F. Monti, and M. Bronstein, "Temporal graph networks for deep learning on dynamic graphs," in *ICML 2020 Workshop on Graph Representation Learning*, 2020.

[14] R. Trivedi, M. Farajtabar, P. Biswal, and H. Zha, "Dyrep: Learning representations over dynamic graphs," in *International conference on learning representations*, 2019.

[15] W. Cong, S. Zhang, J. Kang, B. Yuan, H. Wu, X. Zhou, H. Tong, and M. Mahdavi, "Do we really need complicated model architectures for temporal networks?" in *International Conference on Learning Representations*, 2023.

[16] K. Zang and F. Wang, "Neural dynamics on complex networks," in *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 2020, pp. 1269–1278.

[17] Z. Wang, P. Yang, X. Fan, X. Yan, Z. Wu, S. Pan, L. Chen, Y. Zang, C. Wang, and R. Yu, "Contig: Continuous representation learning on temporal interaction graphs," *Neural Networks*, vol. 172, p. 106151, 2024.

[18] J. Choi, H. Choi, J. Hwang, and N. Park, "Graph neural controlled differential equations for traffic forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022, pp. 6367–6374.

[19] A. Liu and Y. Zhang, "Spatial–temporal dynamic graph convolutional network with interactive learning for traffic forecasting," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 7, pp. 7645–7660, 2024.

[20] P. Kidger, J. Morrill, J. Foster, and T. Lyons, "Neural controlled differential equations for irregular time series," *Advances in Neural Information Processing Systems*, pp. 6696–6707, 2020.

[21] A. Ahmed, N. Shervashidze, S. Narayanamurthy, V. Josifovski, and A. J. Smola, "Distributed large-scale natural graph factorization," in *Proceedings of the 22nd international conference on World Wide Web*, 2013, pp. 37–48.

[22] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th international conference on world wide web*, 2015, pp. 1067–1077.

[23] M. Welling and T. N. Kipf, "Semi-supervised classification with graph convolutional networks," in *J. International Conference on Learning Representations*, 2016.

[24] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.

[25] Z. Feng, R. Wang, T. Wang, M. Song, S. Wu, and S. He, "A comprehensive survey of dynamic graph neural networks: Models, frameworks, benchmarks, experiments and challenges," *arXiv preprint arXiv:2405.00476*, 2024.

[26] B. A. Tjandra, F. Barbero, and M. M. Bronstein, "Enhancing the expressivity of temporal graph networks through source-target identification," in *NeurIPS 2024 Workshop on Symmetry and Geometry in Neural Representations*, 2025.

[27] L. Yu, L. Sun, B. Du, and W. Lv, "Towards better dynamic graph learning: New architecture and unified library," *Advances in Neural Information Processing Systems*, vol. 36, pp. 67 686–67 700, 2023.

[28] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," *Advances in neural information processing systems*, vol. 31, 2018.

[29] M. Poli, S. Massaroli, C. M. Rabideau, J. Park, A. Yamashita, H. Asama, and J. Park, "Continuous-depth neural models for dynamic graph prediction," *arXiv preprint arXiv:2106.11581*, 2021.

[30] L.-P. Xhonneux, M. Qu, and J. Tang, "Continuous graph neural networks," in *International Conference on Machine Learning*, 2020, pp. 10 432–10 441.

[31] Z. Liu, X. Wang, B. Wang, Z. Huang, C. Yang, and W. Jin, "Graph odes and beyond: A comprehensive survey on integrating differential equations with graph neural networks," in *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, 2025, pp. 6118–6128.

[32] T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, and P. Battaglia, "Learning mesh-based simulation with graph networks," in *International conference on learning representations*, 2020.

[33] Z. Li, N. Kovachki, C. Choy, B. Li, J. Kossaifi, S. Otta, M. A. Nabian, M. Stadler, C. Hundt, K. Azizzadenesheli *et al.*, "Geometry-informed neural operator for large-scale 3d pdes," *Advances in Neural Information Processing Systems*, vol. 36, pp. 35 836–35 854, 2023.

[34] B. Zeng, Q. Wang, M. Yan, Y. Liu, R. Chengze, Y. Zhang, H. Liu, Z. Wang, and H. Sun, "Phympgn: Physics-encoded message passing graph network for spatiotemporal pde systems," in *International conference on learning representations*, 2025.

[35] J. Gao, B. Barzel, and A.-L. Barabási, "Universal resilience patterns in complex networks," *Nature*, vol. 530, no. 7590, pp. 307–312, 2016.

[36] J. Wang, G. Song, Y. Wu, and L. Wang, "Streaming graph neural networks via continual learning," in *Proceedings of the 29th ACM international conference on information & knowledge management*, 2020, pp. 1515–1524.

[37] J. Morrill, P. Kidger, L. Yang, and T. Lyons, "On the choice of interpolation scheme for neural CDEs," *Transactions of Machine Learning Research*, 2022.

[38] E. A. Coddington and N. Levinson, *Theory of Ordinary Differential Equations*. Tata McGrawHill Education, 1955.

[39] Y. Wang, Y.-Y. Chang, Y. Liu, J. Leskovec, and P. Li, "Inductive representation learning in temporal networks via causal anonymous walks," in *International Conference on Learning Representations*, 2021.

[40] L. Wang, X. Chang, S. Li, Y. Chu, H. Li, W. Zhang, X. He, L. Song, J. Zhou, and H. Yang, "Tcl: Transformer-based dynamic graph modelling via contrastive learning," *arXiv preprint arXiv:2105.07944*, 2021.

[41] S. Huang, F. Poursafaei, J. Danovitch, M. Fey, W. Hu, E. Rossi, J. Leskovec, M. Bronstein, G. Rabusseau, and R. Rabbany, "Temporal graph benchmark for machine learning on temporal graphs," *Advances in Neural Information Processing Systems*, vol. 36, pp. 2056–2073, 2023.

**Tiexin Qin** received the B.S. degree in electronic information science and technology from China University of Mining and Technology, Jiangsu, China, in 2018 and the M.S. degree in computer science from Nanjing University, Jiangsu, China, in 2021. He is currently pursuing the Ph.D. degree with Department of Electrical Engineering, City University of Hong Kong, Hong Kong. His research interests include machine learning for complex systems and transfer learning.

**Benjamin Walker** received his Masters in Mathematics and Physics from the University of Manchester, UK, in 2020 and his Masters in Mathematical Modeling and Scientific Computing from the University of Oxford, UK, in 2021. He is currently pursuing a DPhil in Mathematics with the Mathematical Institute, University of Oxford, UK. His research interests include time series modeling, neural differential equations, and rough path theory.

**Terry Lyons** was the Wallis Professor of Mathematics until 2022. He was a founding member and then Director (2011–2015) of the Oxford Man Institute of Quantitative Finance. He was the Director of the Wales Institute of Mathematical and Computational Sciences (WIMCS; 2008–2011). He went to Oxford in 2000 having previously been a Professor of Mathematics at Imperial College London (1993–2000), and before that he held the Colin Maclaurin Chair at Edinburgh (1985–1993).

**Hong Yan** received his PhD degree from Yale University. He was Professor of Imaging Science at the University of Sydney and currently is Wong Chun Hong Professor of Data Engineering and Chair Professor of Computer Engineering at City University of Hong Kong. Professor Yan's research interests include image processing, pattern recognition, and bioinformatics. He has over 600 journal and conference publications in these areas. Professor Yan is an IEEE Fellow and IAPR Fellow. He received the 2016 Norbert Wiener Award from the IEEE SMC Society for contributions to image and biomolecular pattern recognition techniques. He is a member of the European Academy of Sciences and Arts and a Fellow of the US National Academy of Inventors.

**Haoliang Li** received the B.S. degree in communication engineering from University of Electronic Science and Technology of China (UESTC) in 2013, and his Ph.D. degree from Nanyang Technological University (NTU), Singapore in 2018. He is currently an assistant professor in Department of Electrical Engineering, City University of Hong Kong. His research mainly focuses on AI security, multimedia forensics and transfer learning. His research works appear in international journals/conferences such as TPAMI, IJCV, TIFS, ICML, NeurIPS, and CVPR. He received the Wallenberg-NTU presidential postdoc fellowship in 2019, doctoral innovation award in 2019, and VCIP best paper award in 2020.

# Supplementary Materials

## Table of Contents



## Appendix A
## Comparison to alternative CDE models

### A.1 Proof of Theorem 3

Provide a dynamic graph comprising of a sequence of graph snapshots $\mathcal{G} = \{(t_0, G_{t_0}), ..., (t_N, G_{t_N})\}$, with each $t_k \in \mathbb{R}$ the time stamp of the observed graph $G_{t_k}$ and $t_0 < \cdots < t_N$. Besides, each graph $G_{t_k}$ is endowed with an adjacency matrix $\mathbf{A}_{t_k}$ representing the topological information for $G_{t_k}$. Further, let $\hat{\mathbf{A}}_s$ be some continuous interpolation of $\mathbf{A}_{t_k}$ such that $\hat{\mathbf{A}}_{t_k} = (t_k, \mathbf{A}_{t_k})$, then our GN-CDE model can be defined by

$$\mathbf{Z}_{t_0} = \zeta_\theta(t_0, \mathbf{A}_{t_0}), \quad \mathbf{Z}_t = \mathbf{Z}_{t_0} + \int_{t_0}^{t} f_\theta(\mathbf{Z}_s, \mathbf{A}_s)\, \mathrm{d}\hat{\mathbf{A}}_s \tag{12}$$

for $t \in (t_0, t_N]$. Let $\beta_s = \begin{bmatrix} \mathbf{Z}_s \\ \mathbf{A}_s \end{bmatrix}$, according to Eq. (12) we have

$$\beta_t = \begin{bmatrix} \mathbf{Z}_t \\ \mathbf{A}_t \end{bmatrix} = \begin{bmatrix} \mathbf{Z}_{t_0} \\ \mathbf{A}_{t_0} \end{bmatrix} + \int_{t_0}^{t} \begin{bmatrix} f_\theta(\mathbf{Z}_s, \mathbf{A}_s) \\ 1 \end{bmatrix} \mathrm{d}\hat{\mathbf{A}}_s \tag{13}$$

for $t \in (t_0, t_N]$. Then we can let $\tilde{f}_\theta(\beta_s) = \begin{bmatrix} f_\theta(\mathbf{Z}_s, \mathbf{A}_s) \\ 1 \end{bmatrix}$, the above equation can be rewritten as

$$\beta_t = \beta_{t_0} + \int_{t_0}^{t} \tilde{f}_\theta(\beta_s)\, \mathrm{d}\hat{\mathbf{A}}_s \tag{14}$$

for $t \in (t_0, t_N]$. This formulation is equivalent to the original Neural CDE formulation presented in Kidger et al. [20], then we accomplish the proof.

## Appendix B
## Feasibility of Our Proposed GN-CDE Approximation

### B.1 Maximal Expressivity and the Signature

**Definition 6** (Maximal expressivity [42])**.** Let $\mathcal{X}$ be a topological space, and let $\mathcal{F} = \{f_\theta : \mathcal{X} \to \mathbb{R} \mid \theta \in \Theta\}$ be a class of real-valued functions on $\mathcal{X}$, parametrised by some set $\Theta$. We say that $\mathcal{F}$ is maximally expressive (or universal) if, for every compact set $\mathcal{K} \subset \mathcal{X}$ and every real-valued continuous function $f : \mathcal{K} \to \mathbb{R}$, the following property holds:

$$\forall \epsilon > 0,\ \exists \theta \in \Theta \quad \text{s.t.} \quad \sup_{x \in \mathcal{K}} \left| f(x) - f_\theta(x) \right| < \epsilon. \tag{15}$$

Due to the universality of the signature, a continuous time series model is maximally expressive if, for all $N$ and $t$, there exists a model configuration such that the output path can be arbitrarily close to $S_t^N$ [43]. Given a path $X : [0, T] \to \mathbb{R}^e$, the truncated signature $S^N \in T^N(\mathbb{R}^e)$ of a path $X : [0, T] \to \mathbb{R}^e$ solves

$$\mathrm{d}S_s^N = (S_s^N \otimes \mathrm{d}X_s)^{\leq N}, \tag{16}$$

where $S_0^N = [1, 0, \ldots, 0]$. For a word $w = i_1 \cdots i_k$, let

$$\mathrm{index}(w) = 1 + \sum_{j=1}^{k} i_j e^{k-j}. \tag{17}$$

Then, letting $a = \mathrm{index}(w)$, an equivalent formulation of (16) is

$$\mathrm{d}S_{a,s}^N = A_{abc} S_{c,s}^N \mathrm{d}X_{b,s}, \tag{18}$$

where $A_{abc}$ is defined by

$$A_{abc} = \begin{cases} 1, & a = 1 + b + e(c-1), \\ 0, & \text{otherwise}, \end{cases} \tag{19}$$

for $b \in \{1, \ldots, e\}$ and $a, c \in \{1, \ldots, \kappa(e, N)\}$, where $\kappa(e, N)$ is the dimension of a depth$-N$ truncated signature of an $e-$dimensional path.

### B.2 Proof of Theorem 5

As shown in [20], NCDEs can approximate the truncated signature of their input path arbitrarily closely. Therefore, the GN-CDE,

$$\mathbf{Z}_t = \mathbf{Z}_{t_0} + \int_{t_0}^{t} f_\theta(\mathbf{Z}_s, \mathbf{A}_s) \mathrm{d}\hat{\mathbf{A}}_s, \tag{20}$$

is maximally expressive. We now show the same is true given a set of independent copies of the model

$$\mathbf{Z}_t = \mathbf{Z}_{t_0} + \int_{t_0}^{t} \mathbf{W}^{(1)} \mathrm{d}\hat{\mathbf{A}}_s \mathbf{W}^{(2)} \mathbf{Z}_s \mathbf{W}^{(L)}, \tag{21}$$

which is a specific version of our approximation (8), where $\sigma$ is the identity, $\mathbf{W}^{(DR1)} = \mathbf{W}^{(1)}$ and $\mathbf{W}^{(DR2)} = [0, \mathbf{W}^{(2)}]$.

In order to handle the independent copies as one system, we introduce an index $r \in \{1, \ldots, R\}$ and expand the driving path to $\bar{\mathbf{A}}_s = I_R \otimes \hat{\mathbf{A}}_s \in \mathbb{R}^{R \times R \times |\mathcal{V}| \times |\mathcal{V}|}$ and the hidden state to $\tilde{\mathbf{Z}}_s \in \mathbb{R}^{R \times R \times |\mathcal{V}| \times d}$. The sum of the independent GN-CDE copies is given by

$$\mathrm{d}Z_{ij,t} = \tilde{\mathbf{W}}_{rrik}^{(1)} \mathrm{d}\tilde{X}_{rrkl,t} \tilde{\mathbf{W}}_{rrlm}^{(2)} \tilde{Z}_{rrmo,t} \tilde{\mathbf{W}}_{rroj}^{(L)} \mathbf{I}_{rr}, \tag{22}$$

where $\mathbf{I} \in \mathbb{R}^{R \times R}$ is the identity matrix,

$$\tilde{\mathbf{W}}^{(1)} = \sum_{r=1}^{R} E_{rr} \otimes \mathbf{W}^{(1),r} \in \mathbb{R}^{R \times R \times |\mathcal{V}| \times |\mathcal{V}|} \tag{23}$$

$$\tilde{\mathbf{W}}^{(2)} = \sum_{r=1}^{R} E_{rr} \otimes \mathbf{W}^{(2),r} \in \mathbb{R}^{R \times R \times |\mathcal{V}| \times |\mathcal{V}|}, \tag{24}$$

$$\tilde{\mathbf{W}}^{(L)} = \sum_{r=1}^{R} E_{rr} \otimes \mathbf{W}^{(L),r} \in \mathbb{R}^{R \times R \times d \times d}, \tag{25}$$

and $E_{rr}$ is the $R \times R$ matrix with a one at $(r, r)$. Define the row-major flattening maps

$$\mathrm{vec}_{|\mathcal{V}|} : \mathbb{R}^{|\mathcal{V}| \times d} \to \mathbb{R}^{|\mathcal{V}| d}, \qquad Z_{ij} \longmapsto z_a,\ a = i + |\mathcal{V}|(j-1),$$

$$\mathrm{vec}_{|\mathcal{V}|^2} : \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|} \to \mathbb{R}^{|\mathcal{V}|^2}, \qquad X_{kl} \longmapsto x_b,\ b = k + |\mathcal{V}|(l-1).$$

TABLE 4: The comparison of different interpolation schemes.

| Interpolation Schemes | Properties | | | |
|---|---|---|---|---|
| | Smoothness | Dependency on Future | Interpolation Complexity | Integral Difficulty |
| Linear | (piecewise) | One | Low | High |
| Rectilinear | (piecewise) | No | Lowest | High |
| Natural Cubic | ✓ | All | Highest | Lowest |
| Cubic Hermite | ✓ | One | High | Low |

Setting $z_t = \mathrm{vec}_{|\mathcal{V}|}(Z_t)$ and $x_t = \mathrm{vec}_{|\mathcal{V}|^2}(X_t)$,

$$\mathrm{d}z_{a,t} = \tilde{B}_{abc} z_{c,t} \mathrm{d}x_{b,t}, \qquad \tilde{B}_{abc} = \sum_{r=1}^{R} W_{ik}^{3,r} W_{lm}^{1,r} W_{oj}^{2,r}. \tag{26}$$

Take $R$ to be the number of non-zero entries in $A_{abc}$, i.e. number of triples $(a, b, c)$ which satisfy $a = 1+b+n^2(c-1)$, and for each triple $(a_r, b_r, c_r)$ let $(i_r, j_r, k_r, l_r, m_r, o_r)$ be the corresponding indices. Taking $d$ such that $nd > \kappa(n^2, N)$ and

$$W^{1,r} = E_{l_r m_r}, \quad W^{2,r} = E_{o_r j_r} \quad W^{3,r} = E_{i_r k_r}, \tag{27}$$

then,

$$\tilde{B}_{abc} = \begin{cases} A_{abc}, & 1 \le a, c \le \kappa(n^2, N), 1 \le b \le n^2, \\ 0 & \text{otherwise.} \end{cases} \tag{28}$$

Finally, taking $z_0 = (1, 0, \dots, 0)$, the first $\kappa(n^2, N)$ entries of $z_t$ will be the depth $N$ truncated signature and all other elements will be zero.

# Appendix C
# Different Interpolation Schemes

We consider four different interpolation schemes for our GN-CDE model: 1) Linear control; 2) Rectilinear control; 3) Natural cubic splines; 4) Cubic Hermite splines with backward differences. As we have analyzed the continuity of vector field $f_\theta(\mathbf{Z}_s, \mathbf{A}_s)$ and the existence and uniqueness of solutions for our framework in Section 3.2, in this part, we analyze the practical performance of them in the smoothness property, interpolation complexity and optimization difficulty when utilizing some ODE solvers ( *e.g.*, Euler method, Dormand-Prince (DOPRI) method [44]) for the integral. We summarize the results in Table 4.

Commonly, the ODE solvers calculate Eq. (5) use another form as

$$\frac{\mathrm{d}\mathbf{Z}_t}{\mathrm{d}t} = f_\theta(\mathbf{Z}_t, \mathbf{A}_t) \frac{\mathrm{d}\hat{\mathbf{A}}t}{\mathrm{d}t}. \tag{29}$$

We also utilize this formula to show different effects caused by the interpolation schemes for our GN-CDE model.

**Linear control.** If we have two observations $(t_0, \mathbf{A}_{t_0})$ and $(t_2, \mathbf{A}_{t_2})$ collected at time stamps $t_0$ and $t_2$, respectively, and we want the get the value for time $t_1$. Linear control is the interpolating along the straight line between these two observations. Formally, $\mathbf{A}_{t_1}$ can be evaluated by solving the equation

$$\frac{\mathbf{A}_{t_1} - \mathbf{A}_{t_0}}{t_1 - t_0} = \frac{\mathbf{A}_{t_2} - \mathbf{A}_{t_1}}{t_2 - t_1}.$$

For our GN-ODE model, the vector field $f_\theta(\mathbf{Z}_t, \mathbf{A}_t)$ in Eq. (29) is implemented as GCN layers with the formula $f_\theta(\mathbf{Z}_s, \mathbf{A}_s) = \sigma(\mathbf{A}_s \mathbf{Z}_s^{(l)} W^{(l)})$, it is Lipschitz continuous. While for the derivation $\frac{\mathrm{d}\hat{\mathbf{A}}t}{\mathrm{d}t}$, its value will be a constant when there exists a graph structural change otherwise zero. The multiplication of these two terms will exhibit some jumps due to the gradient discontinuities at the structural changing moment $(t_k, \mathbf{A}_{t_k})$. Thus, this scheme owns moderate complexity for interpolation and high integral difficulty for the solvers to resolve the jumps.

**Rectilinear control.** For the observations $\{(t_0, \mathbf{A}_{t_0}), ..., (t_N, \mathbf{A}_{t_N})\}$, rectilinear control updates the time and feature channels separately in lead-lag fashion as $\hat{\mathbf{A}}_t : [0, 2n] \to \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|\times 2}$ such that $\hat{\mathbf{A}}_{2k} = (t_k, \mathbf{A}_{t_k})$ and $\hat{\mathbf{A}}_{2k+1} = (t_{k+1}, \mathbf{A}_{t_k})$.

When using rectilinear control as the interpolation scheme to build the path, the derivation $\frac{\mathrm{d}\hat{\mathbf{A}}_t}{\mathrm{d}t}$ for each possible interaction could appear with a large value when there exists a graph structural change. This leads to the derivation in Eq. (29) much more non-smooth than linear control. Besides, the length of the created path is twice as long as other schemes, thus it takes a longer time for our model to evaluate and train. The merit of this scheme is its low interpolation complexity since it only needs to pad the value for a time with the previous one.

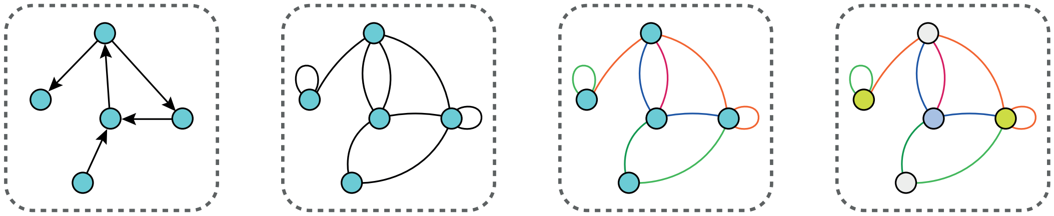
**Natural cubic splines.** The natural cubic spline is a spline that uses the third-degree polynomial which satisfies the given control points. This is also the recommended interpolation scheme by [20].

With natural cubic splines, the created paths are twice differentiable. This means the ODE solvers do not need to resolve the jumps of $\frac{\mathrm{d}\mathbf{Z}_t}{\mathrm{d}t}$ since the derivation is smooth enough. Thus, the integral difficulty for this scheme is the lowest among the four discussed schemes. However, this scheme can lead to unacceptable time costs for path creation, especially when the scale of the graph is large.

**Cubic Hermite splines with backward differences.** For each interval $[t_k, t_{k+1})$ in among the whole observations $\{(t_0, \mathbf{A}_{t_0}), ..., (t_N, \mathbf{A}_{t_N})\}$, cubic Hermite spline keeps that $\hat{\mathbf{A}}_{t_k} = (t_k, \mathbf{A}_{t_k})$ and $\hat{\mathbf{A}}_{t_{k+1}} = (t_{k+1}, \mathbf{A}_{t_{k+1}})$, and that the gradient at each node matches the backward finite difference as

$$\frac{\mathrm{d}\hat{\mathbf{A}}_{t_k}}{\mathrm{d}t} = \hat{\mathbf{A}}_{t_k} - \hat{\mathbf{A}}_{t_{k-1}} \quad \text{and} \quad \frac{\mathrm{d}\hat{\mathbf{A}}_{t_{k+1}}}{\mathrm{d}t} = \hat{\mathbf{A}}_{t_{k+1}} - \hat{\mathbf{A}}_{t_k}.$$

This scheme can smooth gradient discontinuities of linear control, thus its integration difficulty is lower than linear control. Besides, since it only needs to solve a single equation on each control point, its interpolation complexity



(a) Directed graph (b) Multigraph (c) Knowledge graph (d) Heterogeneous information graph

Fig. 5: Illustration of other graph types.

is lower than natural cubic splines. However, the spurious delays in the spline will degrade the accuracy of our model.

See [37] for more descriptions of these schemes.

## Appendix D
## Other Types of Graph

Our proposed GN-CDE model is not only a generic framework for tackling different graph-related tasks, such as the prediction of node attributes, dynamic nodes classification and temporal link prediction but can also be easily extended to more complex graph structures. We discuss how our model can be applied to solve tasks on directed graphs, multigraph, knowledge graphs, and heterogeneous information networks. An illustration of these graph structures is presented in Fig. 5.

**Directed graph.** Vertices in a directed graph are connected by directed edges (See Fig. 5(a)). In order to apply our GN-CDE model to tasks established upon directed dynamic graphs, it is necessary to modify our implementation of the vector field from graph convolutional layers defined for undirected graphs to directed graphs. In addition to utilizing an asymmetric adjacency matrix directly, we can also represent the graph structure with spectral-based methods that leverage edge direction proximity [45], transforms [46, 47] or local graph motifs [48].

**Multigraph.** A graph in which there are multiple edges between two nodes (See Fig. 5(b)). The most common operations for performing graph convolution on a multigraph are graph fusion and separate subgraphs [49]. These techniques can cooperate with our GN-CDE model for representation learning on evolving multigraph.

**Knowledge graph.** This graph is a collection of real-world entities and the relational facts between pairs of entities. The underlining graph structure of the knowledge graph is commonly a multi-digraph with labeled edges, where the labels indicate the types of relationships (See Fig. 5(c)). In order to learn the graph embeddings, we would first utilize the graph fusion or separate subgraphs technique proposed for processing multigraph to acquire a single graph, then we employ the GN-CDE model presented in Eq. (11) but without node attributes matrix for the continuous inference, that is

$$\mathbf{Z}_t = \mathbf{Z}_{t_0} + \int_{t_0}^{t} f_\theta\Big(\mathbf{Z}_s, \mathbf{A}_s, \mathbf{E}\Big) \mathrm{d}\hat{\mathbf{A}}_s \quad \text{for } t \in (t_0, t_N], \tag{30}$$

where $\mathbf{Z}_{t_0} = \zeta_\theta(t_0, \mathbf{A}_{t_0}, \mathbf{E})$.

**Heterogeneous information networks.** This is a complex graph type that consists of multiple types of nodes or edges (See Fig. 5(d)). In order to deal with the dynamic node types and edge features of heterogeneous information networks, we can directly apply the GN-CDE model presented in Eq. (11).

We recommend a comprehensive overview of graph neural networks [49] for readers who are interested in exploring the extension of our GN-CDE model to various graph types.

## Appendix E
## Details of Experimental Setup

### E.1 Heat Diffusion Dynamics

The dynamics are governed by Newton's law of cooling as follows,

$$\frac{\mathrm{d}\boldsymbol{x}_t(v_i)}{\mathrm{d}t} = -k^{(i,j)} \sum_{j=1}^{n} \mathbf{A}^{(i,j)} \left(\boldsymbol{x}(v_i) - \boldsymbol{x}(v_j)\right), \tag{31}$$

where $\mathbf{x}_t(v_i)$ represents the state of node $v_i$ at time $t$ and $\mathbf{A}^{(i,j)}$ is the heat capacity matrix represents the neighbors of each node $v_i$.

**Data generation.** To generate the data for our experiments, we first initialize a graph network with $400$ nodes using a network generator for one of Grid network, Random network, Power-law network, Small-world network and Community network. After that, we randomly select $10$ time stamps from $(t_0, T]$ as the occurrence time of structural changes. To mimic the dynamic structural changes, we randomly add or remove some edges with a uniform probability $p = 0.02$ at these time stamps. Provided the initial energy for all nodes (the value of $\mathbf{F}_{t_0}$), we let the graph structure during its persistent period as a static graph, and numerically solve the heat diffusion system of Eq. (31) segment by segment, using the Dormand–Prince method. This allows us to simulate the continuously dynamic evolution of the node attributes over time, taking into account the structural changes in the graph.

### E.2 Gene regulatory dynamics

The dynamics for gene regulatory networks are governed by Michaelis-Menten equation as follows,

$$\frac{\mathrm{d}\boldsymbol{x}_t(v_i)}{\mathrm{d}t} = -b_i \boldsymbol{x}(v_i)^f + \sum_{j=1}^{n} \mathbf{A}^{(i,j)} \frac{\boldsymbol{x}^h(v_j)}{\boldsymbol{x}^h(v_j)+1}, \tag{32}$$

where the first term models the degradation when $f = 1$ or dimerization when $f = 2$, and the second term represents genetic activation, with the Hill coefficient $h$ determining the level of cooperation in the regulation of the gene.

**Data generation.** The data generation process for gene regulation is similar to that of heat diffusion, except that the differential equation used to model the dynamic system is replaced by Eq. (32).

### E.3 tgbn-trade

This dataset from TGB [41] represents the international agricultural trading network among United Nations (UN) member nations from 1986 to 2016, comprising 255 nations (nodes) and 468,245 trade relationships (edges). Each edge denotes the total trade value of all agricultural products exported from one nation to another within a given year. As the data is reported annually, the time granularity of the dataset is yearly. The primary task associated with this dataset is to predict the proportion of agricultural trade values from one nation to other nations in the subsequent year.

### E.4 tgbn-genre

This dataset from TGB [41] is a bipartite, weighted interaction network between users and music genres, comprising 1,505 nodes and 17,858,395 edges across 133,758 time steps. Each node represents either a user or a music genre, and each edge indicates that a user listened to a genre at a specific time, with the edge weight reflecting the percentage of a song's association with that genre. The dataset integrates listening records and genre metadata, retaining only genres that account for at least 10% genre weight per song and that appear at least 1,000 times. Genre names have been standardized to remove typos. The primary task is to predict the frequency with which each user will interact with music genres over the subsequent week, facilitating personalized music recommendations as user preferences evolve over time.