Temporal Graph ODEs for Irregularly-Sampled Time Series

Alessio Gravina[1,*], **Daniele Zambon**[2], **Davide Bacciu**[1], **Cesare Alippi**[2,3]
[1]University of Pisa, Pisa, Italy
[2]The Swiss AI Lab IDSIA, Università della Svizzera italiana, Lugano, Switzerland
[3]Politecnico di Milano, Milan, Italy
alessio.gravina@phd.unipi.it, daniele.zambon@usi.ch, davide.bacciu@unipi.it, cesare.alippi@usi.ch

Abstract

Modern graph representation learning works mostly under the assumption of dealing with regularly sampled temporal graph snapshots, which is far from realistic, e.g., social networks and physical systems are characterized by continuous dynamics and sporadic observations. To address this limitation, we introduce the Temporal Graph Ordinary Differential Equation (TG-ODE) framework, which learns both the temporal and spatial dynamics from graph streams where the intervals between observations are not regularly spaced. We empirically validate the proposed approach on several graph benchmarks, showing that TG-ODE can achieve state-of-the-art performance in irregular graph stream tasks.

1 Introduction

Representation learning for graphs has been gaining increasing attention over recent years. Such popularity often builds on the fact that complex phenomena are frequently understood as systems of interacting entities described as a graph. For such a reason, learning on graph-structured data through Deep Graph Networks (DGNs) [Bacciu *et al.*, 2020; Wu *et al.*, 2021] has been adopted for solving problems in a variety of fields, such as biology, social science, and sensor networks [Gilmer *et al.*, 2017; Zitnik *et al.*, 2018; Monti *et al.*, 2019; Derrow-Pinion *et al.*, 2021; Gravina *et al.*, 2022; Bacciu *et al.*, 2024].

Graph-based processing methods turned out to be extremely effective in processing spatio-temporal data too [Li *et al.*, 2018; Wu *et al.*, 2019b; Bai *et al.*, 2021; Chen *et al.*, 2022; Zambon and Alippi, 2022; Marisca *et al.*, 2022; Jiang and Luo, 2022; Errica *et al.*, 2023; Cini *et al.*, 2023a]. Such a scenario is a setting where the temporal modeling accounts for functional dependencies – assimilated, in a broad sense, as spatial relationships – existing among the interacting entities. Real-world complex problems described as temporal graphs, e.g., those associated with social interactions, call for novel methods that can move beyond the common assumptions found in most of the methods proposed until now. Indeed, such problems require dealing with mutable relational information, irregularly and severely under-sampled data.

Some recent works propose to model input-output data relations as a continuous dynamic described by a learnable ordinary differential equation (ODE), instead of discrete sequences of layers commonly used in deep learning. Neural ODE-based approaches have been exploited to model non-temporal data, including message-passing functions for learning node-level embeddings [Poli *et al.*, 2019; Chamberlain *et al.*, 2021; Eliasof *et al.*, 2021; Rusch *et al.*, 2022; Gravina *et al.*, 2023]. Notably, relying on ODEs has shown promising for modeling complex temporal patterns from irregularly and sparsely sampled data [Chen *et al.*, 2018; Rubanova *et al.*, 2019; Kidger *et al.*, 2020].

In this paper, we formulate *Temporal Graph Ordinary Differential Equation* (TG-ODE), a general continuous-time modeling framework for temporal graphs that encompasses some methods from the literature as its specific instances, and demonstrate that TG-ODE can be an effective design choice to operate with irregularly and sparsely sampled observations. TG-ODE is designed through the lens of ODEs for effective learning of irregularly sampled temporal graphs. With TG-ODE, a differential equation is learned directly from data to solve a downstream task and predictions are trajectories obtained by numerical integration of the learned ODE.

The key contributions of this work can be summarized as follows:

(i) we introduce TG-ODE, a general modeling framework suited for handling irregularly sampled temporal graphs;

(ii) we introduce new benchmarks of synthetic and real-world scenarios for evaluating forecasting models on irregularly sampled temporal graphs; and

(iii) we conduct extensive experiments to demonstrate the benefits of our method and show that TG-ODE outperforms state-of-the-art DGNs on all benchmarks.

Finally, we stress that, other than the outstanding empirical performance achieved by even simple TG-ODE instances, the framework allows us to reinterpret many state-of-the-art DGNs as a discretized solution of an ODE, thus facilitating their extension to handle graph streams with irregular sampling.

*Corresponding author.

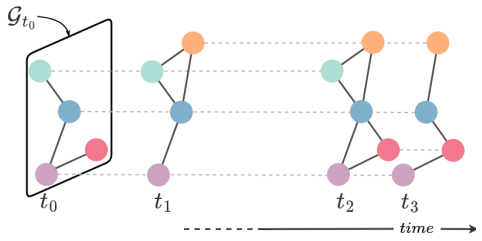


Figure 1: An example of a non-uniform sampling of a temporal graph with snapshots over a set of 5 nodes.

2 Problem Statement

We consider a dynamical system of interacting entities $u \in \mathcal{V}$, which we refer to as *nodes*, that is described by a Cauchy problem defined on an ODE of the form

$$\frac{d\mathbf{X}(t)}{dt} = F(\mathbf{X}(t), \mathbf{E}(t), \mathbf{z}(t)), \tag{1}$$

with initial condition $\mathbf{X}(0) = \mathbf{X}_0$. System state $\mathbf{X}(t) = \{\mathbf{x}_u(t) : u \in \mathcal{V}(t)\}$ is a function of time t and collects the node-level states $\mathbf{x}_u(t) \in \mathbb{R}^{d_x}$ associated with each node $u \in \mathcal{V}(t)$. The node set $\mathcal{V}(t)$ is allowed to vary over time. We denote the dynamic set of edges encoding the node relations as $\mathcal{E}(t) \subseteq \mathcal{V}(t) \times \mathcal{V}(t)$. $\mathbf{E}(t) = \{\mathbf{e}_{uv}(t) : (u, v) \in \mathcal{E}\}$ is a set of edge-level attributes $\mathbf{e}_{vu}(t) \in \mathbb{R}^{d_e}$ defining the type or strength of the interaction between nodes $u, v \in \mathcal{V}(t)$. The system can also be driven by vector $\mathbf{z}_u(t) \in \mathbb{R}^c$ accounting for exogenous variables relevant to the problem at hand, such as weather conditions, hour of the day, or day of the week. Accordingly, for all $u \in \mathcal{V}(t)$, we write

$$\frac{d\mathbf{x}_u(t)}{dt} = F\left(\mathbf{x}_u(t), \mathbf{z}_u(t), \{\mathbf{x}_v(t)\}_{v\in\mathcal{N}_u(t)}, \{\mathbf{e}_{vu}(t)\}_{v\in\mathcal{N}_u(t)}\right), \tag{2}$$

to emphasize the local dependencies of node state $\mathbf{x}_u(t)$ at a time t from its neighboring nodes $v \in \mathcal{N}_u(t) = \{v \in \mathcal{V}(t) : (v, u) \in \mathcal{E}(t)\}$ at the corresponding time.

We express any solution of ODE (1) as the temporal graph

$$\mathcal{G}(t) = (\mathcal{V}(t), \mathcal{E}(t), \mathbf{X}(t), \mathbf{E}(t)) \tag{3}$$

defined for $t \geq 0$. However, we assume to observe system (1) only as a (discrete) sequence of snapshot graphs

$$\mathcal{G} = \{\mathcal{G}_{t_i} : i = 0, 1, 2, \ldots, T\} \tag{4}$$

that arrive at irregular timestamps, i.e., the sampling is not uniform and, in general, $t_i - t_{i-1} \neq t_{i+1} - t_i$. Each snapshot $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t, \overline{\mathbf{X}}_t, \mathbf{E}_t)$ corresponds to an observation of the system state at a specific timestamp $t \in \mathbb{R}$. Edge set $\mathcal{E}_t$ contains the functional relations between the nodes $u, v \in \mathcal{V}_t$ at time t (thus, $\mathcal{E}_t$ can vary over time), while $\overline{\mathbf{X}}_t = \{\overline{\mathbf{x}}_{t,v} : v \in \mathcal{V}_t\}$ gathers the observed node states, with $\overline{\mathbf{x}}_{t,v}$ the state of node v associated to the observed graph snapshot at time t. Edge attributes, if present, are denoted as $\mathrm{e}_{u,v} \in \mathbf{E}_t$. Figure 1 visually summarizes this concept.

In this paper, we address the problem of learning a model of the differential equation underlying the observed data, which is subsequently exploited to provide estimates of unobserved system's node states and make forecasts. To ease readability, in the following we drop the time variable t.

3 Temporal Graph Ordinary Differential Equation

To learn the function F in (2), we consider a family of models

$$f_\theta(\mathbf{x}_u, \mathbf{z}, \{\mathbf{x}_v\}_{v\in\mathcal{N}_u}, \{\mathbf{e}_{vu}\}_{v\in\mathcal{N}_u}) \tag{5}$$

parameterized by vector θ, and optimized so that the solution $\hat{\mathbf{x}}$ of the differential equation

$$\frac{d\mathbf{x}_u}{dt} = f_\theta\left(\mathbf{x}_u, \mathbf{z}_u, \{\mathbf{x}_v\}_{v\in\mathcal{N}_u}, \{\mathbf{e}_{vu}\}_{v\in\mathcal{N}_u}\right), \quad \forall u \in \mathcal{V} \tag{6}$$

minimizes the discrepancy with the observed sequence of graphs in (4). We follow the message-passing paradigm [Gilmer *et al.*, 2017] and instantiate (6) as

$$\frac{d\mathbf{x}_u}{dt} = \phi_U\left(\mathbf{x}_u, \mathbf{z}_u, \rho\left(\{\phi_M(\mathbf{x}_u, \mathbf{x}_v, \mathbf{e}_{vu})\}_{v\in\mathcal{N}_u}\right)\right), \tag{7}$$

with message function ϕ_M, aggregation operator ρ (e.g., the mean), and update function ϕ_U. Note that functions ϕ_U, ϕ_M, and ρ have learnable parameters θ, which are shared across all nodes and time steps. An example of a message passing operator is when $\phi_M(\mathbf{x}_u, \mathbf{x}_v, \mathbf{e}_{vu}) = \mathbf{x}_v$, ρ is the mean, and ϕ_U is a dense feed-forward network of its inputs. We refer to the above framework in (7) as *Temporal Graph Ordinary Differential Equation* (TG-ODE).

We observe that, since our framework relies on ODEs, it can naturally deal with snapshots that arrive at an arbitrary time. Indeed, the original Cauchy problem can be divided into multiple sub-problems, one per snapshot in the temporal graph. Here, the i-th sub-problem is defined for all $u \in \mathcal{V}_t$ as

$$\begin{cases} \frac{d\mathbf{x}_u}{dt} = \phi_U\left(\mathbf{x}_u, \mathbf{z}_u, \rho\left(\{\phi_M(\mathbf{x}_u, \mathbf{x}_v, \mathbf{e}_{vu})\}_{v\in\mathcal{N}_u}\right)\right), \\ \mathbf{x}_u(0) = \psi(\overline{\mathbf{x}}_{t_{i-1},u}, \hat{\mathbf{x}}_u(t_{i-1})) \end{cases} \tag{8}$$

in the time span between the two consecutive timestamps, i.e., $t \in [t_{i-1}, t_i]$, where ψ is a function that combines the i-th observed state of the node u related to the snapshot graph $\mathcal{G}_{t_{i-1}}$ in (4) (i.e., $\overline{\mathbf{x}}_{t_{i-1},u}$) and the prediction $\hat{\mathbf{x}}_u(t_{i-1})$ obtained by solving (8) at the previous step. When given, we consider the true – potentially variable – topology $\mathcal{E}(t)$ to define the neighborhoods for $t \in [t_{i-1}, t_i]$, otherwise, we set $\mathcal{E}(t) \equiv \mathcal{E}_{t_{i-1}}$ for every t, i.e., equal to the last observed topology associated with $\mathcal{G}_{t_{i-1}}$. Accordingly, we optimize θ in order to minimize the mean of some loss $\mathcal{L}$,

$$\frac{1}{T}\sum_{i=1}^{T}\frac{1}{|\mathcal{V}_{t_i}|}\sum_{u\in\mathcal{V}_{t_i}}\mathcal{L}(\overline{\mathbf{x}}_{t_i,u}, \hat{\mathbf{x}}_u(t_i)) \tag{9}$$

where prediction $\hat{\mathbf{x}}_u(t_i)$ at time t_i is obtained by solving (8).

We observe that for most ODEs it is not possible to compute analytical solutions. For such a reason, it is common practice to resort to numerical approximations that leverage discretization strategies (e.g., forward Euler's method). In this case, the solution is computed through iterative applications of the method over a discrete set of points in the time interval and, as observed in [Chen *et al.*, 2018; Haber and Ruthotto, 2017], the process can be assimilated to that of a Recurrent Neural Network (RNN). For simplicity, here we employ the *forward Euler's method* according to

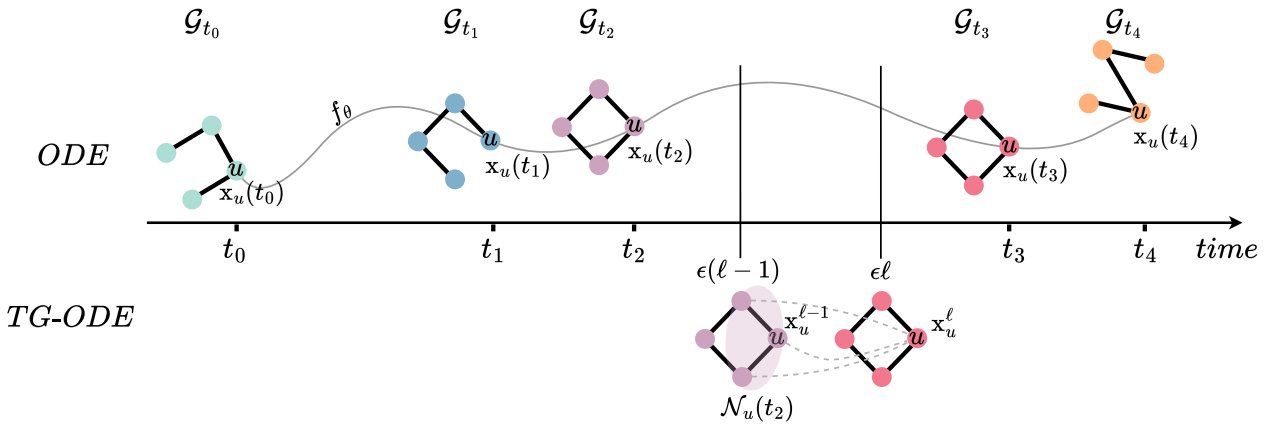


Figure 2: The continuous processing of node u's state in a discrete-time dynamic graph with irregularly-sampled snapshots over a set of 4 nodes and fixed edge set. At the top, the node-wise ODE function f_θ defines the evolution of the states $\mathbf{x}_u(t)$. At the bottom, the discretized solution of the node-wise ODE, which corresponds to our framework TG-ODE. The node embedding $\mathbf{x}_u^\ell$ is computed iteratively over a discrete set of points by leveraging the temporal neighborhood and self-representation at the previous step.

which a solution to (8) is obtained by the recursion

$$\mathbf{x}_u^{\ell+1} = \mathbf{x}_u^\ell + \epsilon\phi_U\Big(\mathbf{x}_u^\ell, \mathbf{z}_u(t_\ell), \rho\big(\{\phi_M(\mathbf{x}_u^\ell, \mathbf{x}_v^\ell, \mathbf{e}_{vu}^\ell)\}_{v\in\mathcal{N}_u(t_\ell)}\big)\Big), \quad (10)$$

starting from initial condition $\mathbf{x}_u^0 = \mathbf{x}_u(0) = \psi(\overline{\mathbf{x}}_{t_{i-1},u}, \hat{\mathbf{x}}_u(t_{i-1}))$ and is reiterated until $\epsilon\ell \geq t_i - t_{i-1}$. In (10), $\epsilon \ll t_i - t_{i-i}$ is the step size, while ℓ indicates the generic iteration step, and $t_\ell = t_{i-1} + \epsilon\ell$. Finally, a solution $\hat{\mathbf{x}}(t)$ to (8) in the interval $[t_{i-1}, t_i]$ is provided by the discretization $\hat{\mathbf{x}}_u(t_{i-1} + \epsilon\ell) = \mathbf{x}_u^\ell$, for all ℓ, and interpolated elsewhere. The process is visually summarized at the bottom of Figure 2.

We acknowledge that not all resulting ODEs allow unique solutions and yield numerical stable problems. Generally, numerical stability is associated with the ODE to solve rather than the input data. Thus, a proper design of the considered family of ODEs in (7) can prevent stability issues. Indeed, the solution of a Cauchy problem exists and is unique if the differential equation is uniformly Lipschitz continuous in its input and continuous in t, as states in the Picard–Lindelöf theorem [Coddington and Levinson, 1955]. Thus, different implementations of (7) should address the continuous behavior of the differential equation. We note that this theorem holds for our model if the underlying neural network has finite weights and uses Lipshitz non-linearities, e.g., the tanh.

By (10) and the generality of the message passing in (7), we observe that TG-ODE allows us to cast basically any standard DGN through the lens of an ODE for temporal graphs with irregular timestamps. Secondly, we stress that, even though TG-ODE is solved here by means of the forward Euler's method, other discretization methods can still be utilized. To conclude, our framework can be implemented using the aggregation function that is most suitable for the given task and the discretization method that best fits the computational resources and problem at hand, such as [Veličković *et al.*, 2018; Hu *et al.*, 2020; Choi *et al.*, 2023; Eliasof *et al.*, 2024a]. As a demonstration of this, in Section 5 we explore the neighborhood aggregation scheme proposed in [Du *et al.*, 2017]. Thus, (7) can be reformulated as

$$\frac{d\mathbf{x}_u}{dt} = \sigma\left(\sum_{k=0}^{K}\sum_{v\in\mathcal{N}_u^k\cup\{u\}} \alpha_{u,v}^{(k)}\,\mathbf{x}_v\theta_k\right), \quad (11)$$

where σ is an activation function, K is the number of hops in the neighborhood, θ_k is the k-th weight matrix, $\mathcal{N}_u^k$ is the k-hop neighborhood of u, and $\alpha_{u,v}^{(k)}$ is a normalization term. For instance, $\alpha_{u,v}^{(k)} = \left(\hat{d}_v^{(k)}\hat{d}_u^{(k)}\right)^{-1/2}$ weighs according to the degrees $\hat{d}_v^{(k)}$ and $\hat{d}_u^{(k)}$ of nodes v and u in the k-hop graph; other choices can include edge attributes as well. We note that θ_k is a parameter specific to the k-hop neighborhood of node u. Thus, it allows the model to learn different transformation patterns at different distances from the considered node u.

4 Related works

Deep Graph Networks for static graphs. In the static graph domain, most DGNs can be generalized by the concepts introduced by MPNN [Gilmer *et al.*, 2017], which is designed to capture and propagate information between nodes in a graph through a message-passing mechanism. Thus, each node updates its state by exchanging information with its neighboring nodes. MPNN can be formulated as

$$\mathbf{x}_u^\ell = \phi_U\left(\mathbf{x}_u^{\ell-1}, \sum_{j\in\mathcal{N}_u}\phi_M\left(\mathbf{x}_u^{\ell-1}, \mathbf{x}_v^{\ell-1}, \mathbf{e}_{uv}\right)\right) \quad (12)$$

where ϕ_U and ϕ_M are respectively the update and message functions, which are responsible for computing neighbor states and updating the node's state accordingly. Most of the state-of-the-art DGNs can be derived by (12). Indeed, the definition of the message and update function allows implementing DGNs with different properties [Kipf and Welling, 2017; Veličković *et al.*, 2018; Hamilton *et al.*, 2017; Xu *et al.*, 2019; Defferrard *et al.*, 2016; Hu *et al.*, 2020; Du *et al.*, 2017].

We can extend the abovementioned DGNs to the domain of temporal graphs by selecting appropriate operators in (7) and, in turn, it allows us to tailor the TG-ODE to exploit relational inductive biases and fulfill given application requirements. For instance, by considering a graph attentional operator [Veličković *et al.*, 2018] in (7), we can implement an anisotropic message passing within the temporal graph during the update of node states.

Deep Graph Networks for temporal graphs. Given the sequential structure of temporal graphs, a natural choice for many methods has been to extend Recurrent Neural Networks to graph data. Indeed, most of the models presented in the literature can be summarized as a combination of DGNs and RNNs.

Some approaches adopt a stacked architecture, where DGNs and RNNs are used sequentially [Seo *et al.*, 2018; Pareja *et al.*, 2020], enabling to separately model spatial and temporal dynamics. Other approaches integrate the DGN inside the RNN [Li *et al.*, 2019; Chen *et al.*, 2022; Seo *et al.*, 2018; Li *et al.*, 2018; Zhao *et al.*, 2020; Bai *et al.*, 2021], allowing to jointly capture the temporal evolution and the spatial dependencies in the graph. We refer to [Cini *et al.*, 2023b] and [Gravina and Bacciu, 2024] for a deeper discussion.

Differently from these approaches, which are intrinsically designed to deal with regular time series, TG-ODE can naturally handle arbitrary time gaps between observations. This makes our framework more suitable for realistic scenarios, in which data are irregular over time.

Continuous Dynamic Models. NeuralODE [Chen *et al.*, 2018] (NODE) has emerged as an effective class of neural network models suitable for learning systems' continuous dynamics, drawing a connection between RNNs and ODEs. Despite the similarity with RNNs, such architectures can deal with irregular time series since the continuously-defined dynamics can naturally incorporate data that arrive at arbitrary times [Chen *et al.*, 2018; Rubanova *et al.*, 2019].

Inspired by the NODE approach, GDE [Poli *et al.*, 2019] links DGNs for static graphs with ODEs. In the static graph domain, ODE-based architectures have been proposed with different aims, such as preserving long-range dependencies [Gravina *et al.*, 2023], reducing the computational complexity of message passing [Wang *et al.*, 2021; Wu *et al.*, 2019a], and mitigating the over-smoothing phenomena [Eliasof *et al.*, 2021; Rusch *et al.*, 2022; Kang *et al.*, 2024].

In the temporal domain, TDE-GNN [Eliasof *et al.*, 2024b] employs higher-order ODEs to capture the temporal graph dynamic, while NDCN [Zang and Wang, 2020] extends GDE to learn continuous-time dynamics on both static and temporal graphs by diffusing input features until the termination time. MTGODE [Jin *et al.*, 2022] adopts an ODE-based approach to deduce missing graph topologies from the time-evolving node features in regularly sampled temporal graphs. Differently, [Huang *et al.*, 2020] and [Huang *et al.*, 2021] propose an ODE-based model in the form of a variational auto-encoder for learning latent dynamics from sampled initial states. To infer missing observations, the methods consider both past and future neighbors' information. This prevents them from being used in an online setting, where data becomes available in a sequential order. Lastly, STG-NCDE [Choi *et al.*, 2022] employs a stacked architecture of two neural controlled differential equations to model temporal and spatial information, respectively. In the STG-NCDE's paper, irregular data are considered, yet they are handled by making them regular via interpolation.

In contrast to these approaches, in this paper we explicitly address irregularly-sampled temporal graphs and we propose a simple model to showcase the effectiveness and efficiency of the TG-ODE framework in working with such data, eliminating the need for additional strategies, such as interpolation. It should be noted that while many of the ODE-based approaches mentioned earlier can be viewed as instances of the introduced TG-ODE framework, our model is specifically designed to demonstrate the benefits of this approach.

5 Experiments

We provide an empirical assessment of our method against related temporal DGN models from the literature[1]. First, we test the efficacy in handling dynamic graphs with irregularly sampled time series by evaluating the models on several heat diffusion scenarios (see Section 5.1). Afterward, we assess and discuss the performance on real graph benchmarks on traffic forecasting problems (see Section 5.2). We report in Table 1 the grid of hyper-parameters employed in our experiments by each method. We carried out the experiments on 7 nodes of a cluster with 96 CPUs per node.

Hyper-parameters	**Values**	
	Heat	***Bench***
Learning rate	$10^{-2}, 10^{-3}, 10^{-4}$	
Weight decay	$10^{-2}, 10^{-3}$	
ψ	concat, sum, $\psi(\overline{\mathbf{x}}, \hat{\mathbf{x}}) = \overline{\mathbf{x}}$	
Activation fun.	tanh, relu, identity	
Embedding dim.	None, 8	64, 32
ϵ	10^{-3}	$1, 0.5, 10^{-1}, 10^{-2}, 10^{-3}$
# hops	5	1, 2, 5

Table 1: The grid of hyper-parameters employed during model selection for the heat diffusion tasks (*Heat*) and graph benchmark tasks (*Bench*). The ϵ hyper-parameter is only used by our method (i.e., TG-ODE), and *embedding dim* equal to *None* means that no encoder and readout are employed.

5.1 Heat Diffusion

In this section, we focus on simulating the heat diffusion over time on a graph. The data is composed of irregularly sampled graph snapshots providing the temperature of the graph's nodes at the given timestamp. We address the task of predicting the nodes' temperature at future (irregular) timestamps.

[1]We release the code implementing our methodology and reproducing our empirical analysis at https://github.com/gravins/TG-ODE.

Datasets

In our experiment, we consider a grid graph consisting of 70 nodes, each of which is characterized by an initial temperature $\mathbf{x}_u(0)$ randomly sampled in the range between 0 and 0.2. We randomly alter the initial temperature profile by generating hot and cold spikes located at some nodes. A hot spike is characterized by a temperature between 10 and 15, while a cold spike is between -15 and -10. Each altered node has a 40% chance of being associated with a cold spike and 60% with a hot spike. We considered two different experimental scenarios depending on the number of altered nodes. In the first scenario, we alter the temperature of a single node. In the second one, we alter the temperature of one third of the graph's nodes. We will refer to these settings as *single-spike* and *multi-spikes*, respectively.

We collected the ground truth by simulating the heat diffusion equation through the forward Euler's method with step size $\epsilon = 10^{-3}$. Figure 3 illustrates two snapshot graphs from the simulated heat diffusion. The training set consists of 100 randomly selected timestamps over the 1000 steps used to simulate the diffusion process. The validation and test sets are generated from two different simulations similar to the one used for building the training set. However, validation and test sets are obtained through 500-step simulations, and only 50 of them are kept as validation/test sets. We simulated seven different diffusion functions, i.e., $-\mathbf{L}\mathbf{X}(t)$, $-\mathbf{L}^2\mathbf{X}(t)$, $-\mathbf{L}^5\mathbf{X}(t)$, $-\tanh(\mathbf{L})\mathbf{X}(t)$, $-5\mathbf{L}\mathbf{X}(t)$, $-0.05\mathbf{L}\mathbf{X}(t)$, and $-(\mathbf{L}+\mathcal{N}_{0,1})\mathbf{X}(t)$. Here, $\mathcal{N}_{0,1}$ stands for a noise sampled from a standard normal distribution, and $\mathbf{L}$ is the normalized graph Laplacian.

TG-ODE and Baseline Models

We explored the performance of TG-ODE leveraging the aggregation scheme in [Du *et al.*, 2017] and the forward Euler's method as discretization procedure, for simplicity. Thus, the nodes' states for the entire snapshot are updated as

$$\mathbf{X}^{\ell} = \mathbf{X}^{\ell-1} + \epsilon\sigma\left(\sum_{k=0}^{K}\mathbf{L}^k\mathbf{X}^{\ell-1}\theta_k\right), \qquad (13)$$

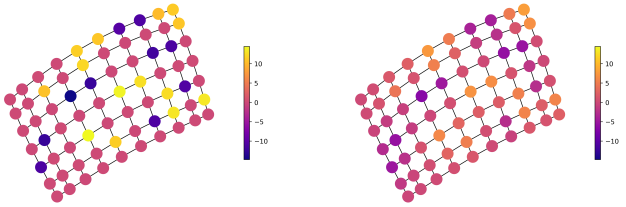
where K corresponds to the number of neighborhood hops and θ_k is the k-th weight matrix. We recall that other choices of aggregation and discretization schemes are possible. We compared our method with six common DGNs for dynamic graphs: A3TGCN [Bai *et al.*, 2021], DCRNN [Li *et al.*, 2018], TGCN [Zhao *et al.*, 2020], GCRN-GRU [Seo *et al.*, 2018], GCRN-LSTM [Seo *et al.*, 2018], and NDCN [Zang and Wang, 2020]. We note that whenever we used the NDCN model with embedding dimension set to none (see Table 1), the resulting model corresponds to DNND [Liu *et al.*, 2023].

Moreover, we considered two additional baselines: NODE [Chen *et al.*, 2018] and LB-baseline. NODE represents an instance of our approach that does not take into account node interactions. Instead, LB-baseline returns the same node states received as input (i.e., , the prediction of $\overline{\mathbf{X}}_{t_{i+1}}$ is $\hat{\mathbf{X}}(t_{i+1}) = \overline{\mathbf{X}}_{t_i}$) and provides a lower bound on the performance we should expect from the learned models.

We designed each model as a combination of three main components. The first is the encoder which maps the node input features into a latent hidden space; the second is the temporal graph convolution (i.e., TG-ODE or the DGN baselines) or the NODE baseline; and the third is a readout that maps the output of the convolution into the output space. The encoder and the readout are Multi-Layer Perceptrons that share the same architecture among all models in the experiments.

To allow all considered baseline models to handle irregular timestamps, we used a similar strategy employed for TG-ODE. Specifically, we selected the unit of time, τ, and then we iteratively applied the temporal graph convolution for a number of steps equal to the ratio between the time difference between two consecutive timestamps and the time unit, i.e., $\#steps = (t_{i+1} - t_i)/\tau$.

We performed hyper-parameter tuning via grid search, optimizing the Mean Absolute Error (MAE). We trained the models using the Adam optimizer for a maximum of 3000 epochs and early stopping with patience of 100 epochs on the validation error.



(a) (b)

Figure 3: (a) A grid graph consisting of 70 nodes in which each node is characterized by an initial temperature. Darker colors correspond to colder temperatures, while brighter colors mean warmer temperatures. (b) The heat diffusion simulation is computed through 1000 steps forward Euler's method leveraging $-\mathbf{L}\mathbf{X}(t)$ as diffusion.

Results

We present the results on the heat diffusion tasks in Table 2 and Table 3, using the $log_{10}(\text{MAE})$ as performance metric in both the single-spike and multi-spikes scenarios. The first observation is that TG-ODE has outstanding performance compared to literature models and the baseline. Despite its simpler architecture, our method produces an error that is significantly lower than the runner-up in each task. In the single-spike setting, TG-ODE achieves a $log_{10}(\text{MAE})$ that is on average 308% to 628% better than the competing models in each task.

Interestingly, not all DGN-based models are capable of improving the results of the LB-baseline. This situation suggests that such approaches attempt to merely learn the mapping function between inputs and outputs rather than learning the actual latent dynamics of the system. Such behavior becomes more evident in the more complex multi-spike scenario. Here, our method achieves up to almost 2080% better $log_{10}(\text{MAE})$ score and more literature models fail in improving the performance with respect to the LB-baseline. These results indicate that capturing the latent dynamics is fundamental, in particular, when the time intervals between observations are not regular over time. We conclude that such methods from the literature might not be suitable for more

	$-\mathbf{L}\mathbf{X}(t)$	$-\mathbf{L}^2\mathbf{X}(t)$	$-\mathbf{L}^5\mathbf{X}(t)$	$-\tanh(\mathbf{L})\mathbf{X}$	$-5\mathbf{L}\mathbf{X}(t)$	$-0.05\mathbf{L}\mathbf{X}(t)$	$-(\mathbf{L}+\mathcal{N}_{0,1})\mathbf{X}(t)$
LB-baseline	-0.557	-0.572	-0.562	-0.538	-0.337	-0.565	-0.837
NODE	-2.828$_{\pm 0.063}$	-2.657$_{\pm 0.053}$	-2.139$_{\pm 0.005}$	-2.711$_{\pm 0.136}$	-2.313$_{\pm 0.016}$	-3.983$_{\pm 0.003}$	-2.059$_{\pm 0.005}$
A3TGCN	-0.834$_{\pm 0.145}$	-0.902$_{\pm 0.093}$	-0.819$_{\pm 0.036}$	-0.890$_{\pm 0.035}$	-1.084$_{\pm 0.004}$	-0.653$_{\pm 0.001}$	-0.781$_{\pm 0.094}$
DCRNN	-1.320$_{\pm 0.163}$	-0.913$_{\pm 0.242}$	-0.867$_{\pm 0.305}$	-1.273$_{\pm 0.075}$	-1.098$_{\pm 0.154}$	-0.964$_{\pm 0.366}$	-1.150$_{\pm 0.375}$
GCRN-GRU	-0.474$_{\pm 0.232}$	-0.633$_{\pm 0.004}$	-0.464$_{\pm 0.064}$	-0.621$_{\pm 0.047}$	-0.695$_{\pm 0.002}$	-0.640$_{\pm 0.019}$	-0.490$_{\pm 0.094}$
GCRN-LSTM	-0.430$_{\pm 0.140}$	-0.323$_{\pm 0.019}$	-0.405$_{\pm 0.053}$	-0.351$_{\pm 0.097}$	-0.511$_{\pm 0.157}$	-0.428$_{\pm 0.140}$	-0.367$_{\pm 0.790}$
NDCN	-1.497$_{\pm 0.034}$	-1.337$_{\pm 0.070}$	-0.350$_{\pm 0.328}$	-1.485$_{\pm 0.075}$	-1.097$_{\pm 0.046}$	-2.408$_{\pm 0.183}$	-0.414$_{\pm 0.155}$
TGCN	-0.825$_{\pm 0.108}$	-0.900$_{\pm 0.143}$	-0.804$_{\pm 0.074}$	-0.834$_{\pm 0.149}$	-1.051$_{\pm 0.020}$	-0.653$_{\pm 0.001}$	-0.781$_{\pm 0.094}$
Ours	**-4.087**$_{\pm 0.171}$	**-3.106**$_{\pm 0.181}$	**-2.265**$_{\pm 0.053}$	**-4.166**$_{\pm 0.140}$	**-2.351**$_{\pm 0.036}$	**-4.811**$_{\pm 0.198}$	**-2.069**$_{\pm 0.001}$

Table 2: Test $log_{10}(\mathrm{MAE})$ score and std in the single-spike heat diffusion experiments, averaged over 5 separate runs.

	$-\mathbf{L}\mathbf{X}(t)$	$-\mathbf{L}^2\mathbf{X}(t)$	$-\mathbf{L}^5\mathbf{X}(t)$	$-\tanh(\mathbf{L})\mathbf{X}(t)$	$-5\mathbf{L}\mathbf{X}(t)$	$-0.05\mathbf{L}\mathbf{X}(t)$	$-(\mathbf{L}+\mathcal{N}_{0,1})\mathbf{X}(t)$
LB-baseline	0.490	0.517	0.552	0.523	0.256	0.561	0.666
NODE	-1.708$_{\pm 0.016}$	-1.426$_{\pm 0.021}$	-1.093$_{\pm 0.004}$	-1.671$_{\pm 0.006}$	-1.198$_{\pm 0.016}$	-2.749$_{\pm 0.016}$	-0.979$_{\pm 0.047}$
A3TGCN	0.443$_{\pm 0.087}$	0.244$_{\pm 0.124}$	0.174$_{\pm 0.071}$	0.509$_{\pm 0.058}$	0.187$_{\pm 0.010}$	0.628$_{\pm 0.023}$	0.328$_{\pm 0.060}$
DCRNN	-0.140$_{\pm 0.092}$	-0.143$_{\pm 0.111}$	-0.123$_{\pm 0.132}$	-0.122$_{\pm 0.120}$	-0.421$_{\pm 0.227}$	-0.002$_{\pm 0.125}$	-0.212$_{\pm 0.333}$
GCRN-GRU	0.586$_{\pm 0.003}$	0.614$_{\pm 0.004}$	0.639$_{\pm 0.002}$	0.610$_{\pm 0.002}$	0.440$_{\pm 0.003}$	0.629$_{\pm 0.001}$	0.719$_{\pm 0.003}$
GCRN-LSTM	0.584$_{\pm 0.001}$	0.610$_{\pm 0.002}$	0.637$_{\pm 0.002}$	0.612$_{\pm 0.003}$	0.440$_{\pm 0.005}$	0.631$_{\pm 0.002}$	0.705$_{\pm 0.002}$
NDCN	0.120$_{\pm 0.325}$	-0.070$_{\pm 0.056}$	0.315$_{\pm 0.245}$	-0.128$_{\pm 0.020}$	0.146$_{\pm 0.107}$	-1.357$_{\pm 0.053}$	0.384$_{\pm 0.013}$
TGCN	0.404$_{\pm 0.236}$	0.313$_{\pm 0.072}$	0.113$_{\pm 0.071}$	0.493$_{\pm 0.056}$	0.113$_{\pm 0.086}$	0.615$_{\pm 0.023}$	0.364$_{\pm 0.134}$
Ours	**-4.259**$_{\pm 0.037}$	**-3.705**$_{\pm 0.143}$	**-1.314**$_{\pm 0.249}$	**-3.572**$_{\pm 0.010}$	**-2.350**$_{\pm 0.083}$	**-4.567**$_{\pm 0.109}$	**-1.021**$_{\pm 0.002}$

Table 3: Test $log_{10}(\mathrm{MAE})$ score and std in the multi-spikes heat diffusion experiments, averaged over 5 separate runs.

realistic settings characterized by continuous dynamics and sporadic observations.

Finally, we observe that GCRN-GRU and GCRN-LSTM generate the highest error levels, while DCRNN, NDCN, and NODE are the best among the baselines. Since literature models use RNN architectures to learn temporal patterns, it is reasonable to assume that the poor performance might be due to the limited capacity of RNNs to handle non-uniform time gaps between observations. In contrast, ODE-based models (NODE, NCDN and ours) demonstrate enhanced learning capabilities in this scenario. The performance gap between our model and the considered baselines is an indication that the diverse spatial patterns learned by different DGN architectures can heavily impact the performance of the performed tasks.

5.2 Graph Benchmarks

This section introduces a set of graph benchmarks whose objective is to assess traffic forecasting performance from irregular time series; similar to the heat diffusion tasks, we predict the future node values given only the past history.

Datasets

We considered six real-world graph benchmarks for traffic forecasting: MetrLA [Li *et al.*, 2018], Montevideo [Rozemberczki *et al.*, 2021], PeMS03 [Guo *et al.*, 2022], PeMS04 [Guo *et al.*, 2022], PeMS07 [Guo *et al.*, 2022], and PeMS08 [Guo *et al.*, 2022]; we report additional details about the datasets in Table 4. We used a modified version of the original datasets where we employed irregularly sampled observations. We will refer to the datasets by using the subscript "i" – e.g., $\mathbf{MetrLA}_i$ – to make apparent the difference from the original versions.

We generated irregular time series by randomly selecting a third of the original graph snapshots for most of the experiments; ratios from 3% to 94% are studied in Figure 5. We considered a temporal data splitting in which 80% of the previously selected snapshots are used as training set, 10% as validation set, and the remaining as test set.

	# Steps	**# Nodes**	**# Edges**	**Timespan**
MetrLA	34,272	207	1,515	01/03 - 30/06 2012
Montevideo	739	675	690	01/10 - 31/10 2020
PeMS03	26,208	358	442	01/09 - 30/11 2018
PeMS04	16,992	307	209	01/01 - 28/02 2018
PeMS07	28,225	883	790	01/05 - 31/08 2017
PeMS08	17,856	170	137	01/07 - 31/08 2016

Table 4: Statistics of the original version of the datasets.

TG-ODE and Baseline Models

For these experiments, we considered the same models, baseline and architectural choices of the heat diffusion experiments. Since NODE does not take into account interactions between nodes for its predictions, we choose not to include it as a baseline in this scenario. Hyper-parameter tuning has been performed by grid search, optimizing the MAE. Optimizer settings are the same as for the previous experiments.

	MetrLA$_i$	**Montevideo**$_i$	**PeMS03**$_i$	**PeMS04**$_i$	**PeMS07**$_i$	**PeMS08**$_i$
LB-baseline	58.191	0.442	165.015	211.230	314.710	227.380
A3TGCN	$5.731_{\pm 0.011}$	$0.378_{\pm 4\cdot 10^{-4}}$	$28.897_{\pm 0.733}$	$32.221_{\pm 1.355}$	$38.303_{\pm 0.795}$	$30.652_{\pm 0.995}$
DCRNN	†	$0.332_{\pm 0.001}$	$18.652_{\pm 0.136}$	†	†	†
GCRN-GRU	$8.438_{\pm 0.004}$	$0.332_{\pm 0.001}$	$49.360_{\pm 18.619}$	$53.389_{\pm 4.728}$	$68.785_{\pm 5.787}$	$51.787_{\pm 10.872}$
GCRN-LSTM	$8.440_{\pm 0.009}$	$0.333_{\pm 0.002}$	$62.210_{\pm 0.923}$	$52.427_{\pm 4.162}$	$151.824_{\pm 17.654}$	$80.567_{\pm 24.891}$
NDCN	$8.471_{\pm 0.022}$	$0.435_{\pm 0.021}$	†	$127.202_{\pm 0.334}$	†	$129.667_{\pm 44.385}$
TGCN	$5.832_{\pm 0.125}$	$0.380_{\pm 4\cdot 10^{-4}}$	$28.506_{\pm 0.332}$	$33.059_{\pm 1.063}$	$38.750_{\pm 1.429}$	$33.114_{\pm 1.963}$
Ours	$\mathbf{2.828}_{\pm 0.001}$	$\mathbf{0.327}_{\pm 6\cdot 10^{-5}}$	$\mathbf{17.423}_{\pm 0.012}$	$\mathbf{24.739}_{\pm 0.014}$	$\mathbf{26.081}_{\pm 0.004}$	$\mathbf{18.818}_{\pm 0.021}$

Table 5: Test MAE score and std in the traffic forecasting setting, averaged over 5 separate runs. † means gradient explosion.

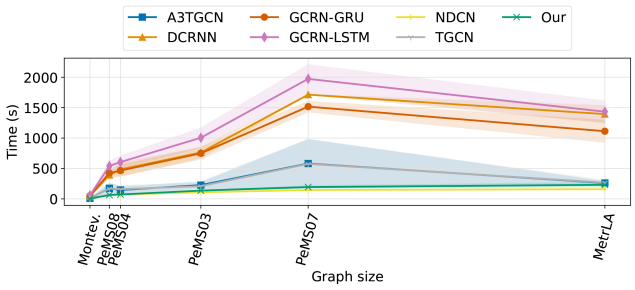


Figure 4: Average time per epoch (measured in seconds) and std computed using an Intel Xeon Gold 6240R CPU @ 2.40GHz. Each time is obtained using 5 neighbor hops (when possible) and embedding dimension equal to 64. The graph size is computed as $size = \#steps * \#edges$.

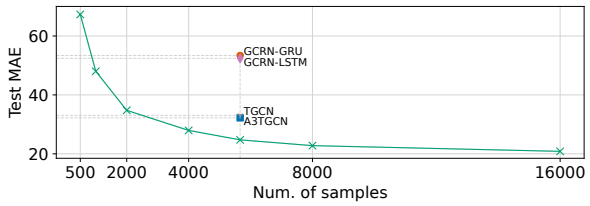


Figure 5: Test MAE scores and std of TG-ODE on PeMS04, averaged over 5 runs, for different sparsity levels.

Results

Table 5 reports the traffic forecasting results in terms of MAE. Similarly to the heat diffusion scenario, TG-ODE shows a remarkable performance improvement compared to literature models, achieving an MAE that is up to 202% better than the runner-up model. Moreover, as reported in Figure 4, we observe that TG-ODE is $2\times$ to $13\times$ faster than the other approaches under test. NDCN is the sole method matching the speed of our approach. However, it's noteworthy that NDCN utilizes only one neighbor hop, thereby simplifying the final computation.

We observe that the baseline performs poorly in these benchmarks, suggesting that such tasks are more complex than the ones based on heat diffusion. Despite all DGN models outperforming the LB-baseline, they still produce an error that is on average double than that of TG-ODE, highlighting the added value of our approach when dealing with temporal graphs characterized by irregular sampling.

Finally, we comment that the DCRNN and NDCN suffered from gradient issues in most of the tasks. We believe this is due to their inability to learn the latent dynamics of the system when the models' outputs are not computed over a regular time series.

Impact of the Sample Sparsity

To demonstrate the effectiveness of our approach, we study the prediction performance under different sparsity levels. We consider here the PeMS04 dataset. In this analysis, we systematically decreased the number of considered graph snapshots in the time series. This reduction makes the resulting task more challenging than the original one, as the snapshots become more sparse over time – the expected difference $t_{i+1} - t_i$ gets larger. Additionally, the model has fewer data to learn the task, thereby amplifying the task complexity. We generated the irregular time series by randomly selecting 500, 1000, 2000, 4000, 8000, or 16000 graph snapshots from the original dataset (from 3% to 94% of the original data), resulting in varying degrees of sparsity. For each dataset size, we used an 80/10/10 temporal data split and performed hyperparameter tuning, as previously done in this section.

Figure 5 illustrates the performance of our method, TG-ODE, at various degrees of data sparsity. As expected, we observe that as the number of samples increases, the test MAE decreases. Notably, TG-ODE maintains robust performance even with higher degrees of sparsity, with a decrease in performance by only $\sim$7 points when reducing the size from 16000 to 4000 samples. While the prediction error is indeed relatively large when considering only 3-7% of the original data, we comment that it is still substantially better than that of the LB-baseline and comparable to that of the other DGN's which used 33% of the data. Overall, the model exhibits excellent performance even in situations of high to extreme sparsity (i.e., less than 8000 samples). The observed outcome supports the effectiveness of TG-ODE, emphasizing its potential for real-world applications with irregularly sampled temporal graphs.

6 Conclusions

We have presented *Temporal Graph Ordinary Differential Equation* (TG-ODE), a new general framework for effectively learning from irregularly sampled temporal graphs. Thanks to the connection between ODEs and neural architectures, TG-ODE can naturally handle arbitrary time gaps between obser-

vations, allowing to address a common limitation of DGNs for temporal graphs, i.e., the restriction to work solely on regularly sampled data.

To demonstrate the benefits of our approach, we conducted extensive experiments on ad-hoc benchmarks that include several synthetic and real-world scenarios. The results of our experimental analysis show that our method outperforms state-of-the-art models for temporal graphs by a large margin. Furthermore, our method benefits from a faster training, thus suggesting scalability to large networks.

Despite being appealing for many realistic application setups, we acknowledge that not all resulting ODEs allow unique solutions and yield numerical stability problems, thus requiring some additional care from the user.

Looking ahead to future developments, we intend to broaden the investigation of more sophisticated numerical methods to solve the learned temporal graph ODE, e.g., using adaptive multistep schemes [Ascher and Petzold, 1998]. Extending the proposed framework to the problem of reconstructing missing data is another interesting research direction to consider.

Acknowledgments

This work has been supported by EU-EIC EMERGE (Grant No. 101070918), by the EU NextGenerationEU programme under the funding schemes PNRR-PE-AI (PE00000013) FAIR - Future Artificial Intelligence Research, and by the Swiss National Science Foundation project HORD-GNN (FNS 204061).

References

[Ascher and Petzold, 1998] Uri M. Ascher and Linda R. Petzold. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. Society for Industrial and Applied Mathematics, USA, 1st edition, 1998.

[Bacciu *et al.*, 2020] Davide Bacciu, Federico Errica, Alessio Micheli, and Marco Podda. A gentle introduction to deep learning for graphs. *Neural Networks*, 129:203–221, 2020.

[Bacciu *et al.*, 2024] Davide Bacciu, Federico Errica, Alessio Gravina, Lorenzo Madeddu, Marco Podda, and Giovanni Stilo. Deep Graph Networks for Drug Repurposing With Multi-Protein Targets. *IEEE Transactions on Emerging Topics in Computing*, 12(1):177–189, 2024.

[Bai *et al.*, 2021] Jiandong Bai, Jiawei Zhu, Yujiao Song, Ling Zhao, Zhixiang Hou, Ronghua Du, and Haifeng Li. A3T-GCN: Attention Temporal Graph Convolutional Network for Traffic Forecasting. *ISPRS International Journal of Geo-Information*, 10(7), 2021.

[Chamberlain *et al.*, 2021] Ben Chamberlain, James Rowbottom, Maria I Gorinova, Michael Bronstein, Stefan Webb, and Emanuele Rossi. GRAND: Graph neural diffusion. In *International Conference on Machine Learning*, pages 1407–1418. PMLR, 2021.

[Chen *et al.*, 2018] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in neural information processing systems*, 2018.

[Chen *et al.*, 2022] Jinyin Chen, Xueke Wang, and Xuanheng Xu. GC-LSTM: graph convolution embedded LSTM for dynamic network link prediction. *Applied Intelligence*, 52(7):7513–7528, May 2022.

[Choi *et al.*, 2022] Jeongwhan Choi, Hwangyong Choi, Jeehyun Hwang, and Noseong Park. Graph Neural Controlled Differential Equations for Traffic Forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(6):6367–6374, Jun. 2022.

[Choi *et al.*, 2023] Jeongwhan Choi, Seoyoung Hong, Noseong Park, and Sung-Bae Cho. GREAD: Graph Neural Reaction-Diffusion Networks. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 5722–5747. PMLR, 23–29 Jul 2023.

[Cini *et al.*, 2023a] Andrea Cini, Ivan Marisca, Filippo Maria Bianchi, and Cesare Alippi. Scalable Spatiotemporal Graph Neural Networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(6):7218–7226, Jun. 2023.

[Cini *et al.*, 2023b] Andrea Cini, Ivan Marisca, Daniele Zambon, and Cesare Alippi. Graph Deep Learning for Time Series Forecasting. *arXiv preprint arXiv:2310.15978*, 2023.

[Coddington and Levinson, 1955] Earl A. Coddington and Norman Levinson. *Theory of Ordinary Differential Equations*. TATA McGraw-Hill, USA, 1955.

[Defferrard *et al.*, 2016] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[Derrow-Pinion *et al.*, 2021] Austin Derrow-Pinion, Jennifer She, David Wong, Oliver Lange, Todd Hester, Luis Perez, Marc Nunkesser, Seongjae Lee, Xueying Guo, Brett Wiltshire, Peter W. Battaglia, Vishal Gupta, Ang Li, Zhongwen Xu, Alvaro Sanchez-Gonzalez, Yujia Li, and Petar Velickovic. ETA Prediction with Graph Neural Networks in Google Maps. In *Proceedings of the 30th ACM CIKM*, page 3767–3776, New York, NY, USA, 2021. Association for Computing Machinery.

[Du *et al.*, 2017] Jian Du, Shanghang Zhang, Guanhang Wu, José M. F. Moura, and Soummya Kar. Topology adaptive graph convolutional networks. *arXiv preprint arXiv:1710.10370*, 2017.

[Eliasof *et al.*, 2021] Moshe Eliasof, Eldad Haber, and Eran Treister. PDE-GCN: Novel Architectures for Graph Neural Networks Motivated by Partial Differential Equations. In *Advances in neural information processing systems*, 2021.

[Eliasof *et al.*, 2024a] Moshe Eliasof, Eldad Haber, and Eran Treister. Feature Transportation Improves Graph Neural Networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(11):11874–11882, Mar. 2024.

[Eliasof *et al.*, 2024b] Moshe Eliasof, Eldad Haber, Eran Treister, and Carola-Bibiane B Schönlieb. On the temporal domain of differential equation inspired graph neural networks. In Sanjoy Dasgupta, Stephan Mandt, and Yingzhen Li, editors, *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pages 1792–1800. PMLR, 02–04 May 2024.

[Errica *et al.*, 2023] Federico Errica, Alessio Gravina, Davide Bacciu, and Alessio Micheli. Hidden Markov Models for Temporal Graph Representation Learning. In *Proceedings of the 31st European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, 2023.

[Gilmer *et al.*, 2017] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural Message Passing for Quantum Chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, page 1263–1272. JMLR.org, 2017.

[Gravina and Bacciu, 2024] Alessio Gravina and Davide Bacciu. Deep Learning for Dynamic Graphs: Models and Benchmarks. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–14, 2024.

[Gravina *et al.*, 2022] Alessio Gravina, Jennifer L. Wilson, Davide Bacciu, Kevin J. Grimes, and Corrado Priami. Controlling astrocyte-mediated synaptic pruning signals for schizophrenia drug repurposing with deep graph networks. *PLOS Computational Biology*, 18(5):1–19, 05 2022.

[Gravina *et al.*, 2023] Alessio Gravina, Davide Bacciu, and Claudio Gallicchio. Anti-Symmetric DGN: a stable architecture for Deep Graph Networks. In *International conference on learning representations (ICLR)*, 2023.

[Guo *et al.*, 2022] Shengnan Guo, Youfang Lin, Huaiyu Wan, Xiucheng Li, and Gao Cong. Learning Dynamics and Heterogeneity of Spatial-Temporal Graph Data for Traffic Forecasting. *IEEE Transactions on Knowledge and Data Engineering*, 34(11):5415–5428, 2022.

[Haber and Ruthotto, 2017] Eldad Haber and Lars Ruthotto. Stable Architectures for Deep Neural Networks. *Inverse problems*, 34(1):014004, 2017.

[Hamilton *et al.*, 2017] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. In *Advances in neural information processing systems*, 2017.

[Hu *et al.*, 2020] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for Pre-training Graph Neural Networks. In *International conference on learning representations (ICLR)*, 2020.

[Huang *et al.*, 2020] Zijie Huang, Yizhou Sun, and Wei Wang. Learning Continuous System Dynamics from Irregularly-Sampled Partial Observations. In *Advances in neural information processing systems*, volume 33, pages 16177–16187. Curran Associates, Inc., 2020.

[Huang *et al.*, 2021] Zijie Huang, Yizhou Sun, and Wei Wang. Coupled graph ode for learning interacting system dynamics. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, page 705–715, New York, NY, USA, 2021. Association for Computing Machinery.

[Jiang and Luo, 2022] Weiwei Jiang and Jiayun Luo. Graph neural network for traffic forecasting: A survey. *Expert Systems with Applications*, 207:117921, 2022.

[Jin *et al.*, 2022] Ming Jin, Yu Zheng, Yuan-Fang Li, Siheng Chen, Bin Yang, and Shirui Pan. Multivariate time series forecasting with dynamic graph neural odes. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[Kang *et al.*, 2024] Qiyu Kang, Kai Zhao, Qinxu Ding, Feng Ji, Xuhao Li, Wenfei Liang, Yang Song, and Wee Peng Tay. Unleashing the Potential of Fractional Calculus in Graph Neural Networks with FROND. In *The Twelfth International Conference on Learning Representations*, 2024.

[Kidger *et al.*, 2020] Patrick Kidger, James Morrill, James Foster, and Terry Lyons. Neural Controlled Differential Equations for Irregular Time Series. In *Advances in neural information processing systems*, 2020.

[Kipf and Welling, 2017] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *International conference on learning representations (ICLR)*, 2017.

[Li *et al.*, 2018] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In *International conference on learning representations (ICLR)*, 2018.

[Li *et al.*, 2019] Jia Li, Zhichao Han, Hong Cheng, Jiao Su, Pengyun Wang, Jianfeng Zhang, and Lujia Pan. Predicting Path Failure In Time-Evolving Graphs. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, page 1279–1289, New York, NY, USA, 2019. Association for Computing Machinery.

[Liu *et al.*, 2023] Bing Liu, Wei Luo, Gang Li, Jing Huang, and Bo Yang. Do we need an encoder-decoder to model dynamical systems on networks? In Edith Elkind, editor, *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, pages 2178–2186. International Joint Conferences on Artificial Intelligence Organization, 8 2023. Main Track.

[Marisca *et al.*, 2022] Ivan Marisca, Andrea Cini, and Cesare Alippi. Learning to Reconstruct Missing Data from Spatiotemporal Graphs with Sparse Observations. In *Advances in neural information processing systems*, pages 1–17, 2022.

[Monti *et al.*, 2019] Federico Monti, Fabrizio Frasca, Davide Eynard, Damon Mannion, and Michael M. Bronstein. Fake News Detection on Social Media using Geometric Deep Learning. *arXiv preprint arXiv:1902.06673*, 2019.

[Pareja *et al.*, 2020] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao B. Schardl, and Charles E. Leiserson. EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, 2020.

[Poli *et al.*, 2019] Michael Poli, Stefano Massaroli, Junyoung Park, Atsushi Yamashita, Hajime Asama, and Jinkyoo Park. Graph neural ordinary differential equations. *arXiv preprint arXiv:1911.07532*, 2019.

[Rozemberczki *et al.*, 2021] Benedek Rozemberczki, Paul Scherer, Yixuan He, George Panagopoulos, Alexander Riedel, Maria Astefanoaei, Oliver Kiss, Ferenc Beres, Guzman Lopez, Nicolas Collignon, and Rik Sarkar. PyTorch Geometric Temporal: Spatiotemporal Signal Processing with Neural Machine Learning Models. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*, page 4564–4573, 2021.

[Rubanova *et al.*, 2019] Yulia Rubanova, Ricky T. Q. Chen, and David K Duvenaud. Latent Ordinary Differential Equations for Irregularly-Sampled Time Series. In *Advances in neural information processing systems*, volume 32. Curran Associates, Inc., 2019.

[Rusch *et al.*, 2022] T Konstantin Rusch, Ben Chamberlain, James Rowbottom, Siddhartha Mishra, and Michael Bronstein. Graph-coupled oscillator networks. In *International Conference on Machine Learning*, pages 18888–18909. PMLR, 2022.

[Seo *et al.*, 2018] Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. Structured Sequence Modeling with Graph Convolutional Recurrent Networks. In *Neural Information Processing*, pages 362–373, Cham, 2018. Springer International Publishing.

[Veličković *et al.*, 2018] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *International conference on learning representations (ICLR)*, 2018.

[Wang *et al.*, 2021] Yifei Wang, Yisen Wang, Jiansheng Yang, and Zhouchen Lin. Dissecting the Diffusion Process in Linear Graph Convolutional Networks. In *Advances in Neural Information Processing Systems*, volume 34, pages 5758–5769. Curran Associates, Inc., 2021.

[Wu *et al.*, 2019a] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying Graph Convolutional Networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 6861–6871. PMLR, 09–15 Jun 2019.

[Wu *et al.*, 2019b] Z Wu, S Pan, G Long, J Jiang, and C Zhang. Graph WaveNet for Deep Spatial-Temporal Graph Modeling. In *The 28th IJCAI*. International Joint Conferences on Artificial Intelligence Organization, 2019.

[Wu *et al.*, 2021] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2021.

[Xu *et al.*, 2019] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks? In *International conference on learning representations (ICLR)*, 2019.

[Zambon and Alippi, 2022] Daniele Zambon and Cesare Alippi. AZ-whiteness test: A test for signal uncorrelation on spatio-temporal graphs. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 11975–11986. Curran Associates, Inc., 2022.

[Zang and Wang, 2020] Chengxi Zang and Fei Wang. Neural Dynamics on Complex Networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, page 892–902, New York, NY, USA, 2020. Association for Computing Machinery.

[Zhao *et al.*, 2020] Ling Zhao, Yujiao Song, Chao Zhang, Yu Liu, Pu Wang, Tao Lin, Min Deng, and Haifeng Li. T-GCN: A Temporal Graph Convolutional Network for Traffic Prediction. *IEEE Transactions on Intelligent Transportation Systems*, 21(9):3848–3858, 2020.

[Zitnik *et al.*, 2018] Marinka Zitnik, Monica Agrawal, and Jure Leskovec. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics*, 34(13):i457–i466, 06 2018.