Temporal Graph ODEs for Irregularly-Sampled Time Series

Alessio Gravina1,∗, Daniele Zambon2, Davide Bacciu1, Cesare Alippi2,3

1University of Pisa, Pisa, Italy
2The Swiss AI Lab IDSIA, Università della Svizzera italiana, Lugano, Switzerland
3Politecnico 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; Marisa 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.
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

\[
\frac{dX(t)}{dt} = F(X(t), E(t), z(t)),
\]

with initial condition \( X(0) = X_0 \). System state \( X(t) = \{x_u(t) : u \in \mathcal{V}(t)\} \) is a function of time \( t \) and collects the node-level states \( x_u(t) \in \mathbb{R}^{d_u} \) 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 \( E(t) \subseteq \mathcal{V}(t) \times \mathcal{V}(t) \). \( E(t) = \{e_{uv}(t) : (u, v) \in E\} \) is a set of edge-level attributes \( e_{uv}(t) \in \mathbb{R}^{d_e} \) defining the type or strength of the interaction between nodes \( u, v \in \mathcal{V}(t) \). The system can also be driven by vector \( z_u(t) \in \mathbb{R}^c \) 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

\[
\begin{aligned}
\frac{dx_u(t)}{dt} &= F(x_u(t), z_u(t), \{x_v(t)\}_{v \in \mathcal{V}(t)}, \{e_{uv}(t)\}_{v \in \mathcal{V}(t)}), \\
\end{aligned}
\]

(2)

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

We express any solution of ODE (1) as the temporal graph \( G(t) = (\mathcal{V}(t), E(t), X(t), E(t)) \)

(3)
defined for \( t \geq 0 \). However, we assume to observe system (1) only as a (discrete) sequence of snapshot graphs

\[
G = \{G_t : t = 0, 1, 2, \ldots, T\}
\]

(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 \( G_t = (\mathcal{V}_t, E_t, X_t) \) corresponds to an observation of the system state at a specific timestamp \( t \in \mathbb{R} \). Edge set \( E_t \) contains the functional relations between the nodes \( u, v \in \mathcal{V}_t \) at time \( t \) (thus, \( E_t \) can vary over time), while \( X_t = \{x_{i,v} : v \in \mathcal{V}_t\} \) gathers the observed node states, with \( x_{i,v} \) the state of node \( v \) associated to the observed graph snapshot at time \( t \). Edge attributes, if present, are denoted as \( e_{uv} \in E_t \). 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

\[
f_\theta(x_u, z_u, \{x_v\}_{v \in \mathcal{V}_t}, \{e_{uv}\}_{v \in \mathcal{V}_t})
\]

(5)

parameterized by vector \( \theta \), and optimized so that the solution \( x \) of the differential equation

\[
\frac{dx_u}{dt} = f_\theta(x_u, z_u, \{x_v\}_{v \in \mathcal{V}_t}, \{e_{uv}\}_{v \in \mathcal{V}_t}), \quad \forall u \in \mathcal{V}
\]

(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

\[
\frac{dx_u}{dt} = \phi_U(x_u, z_u, \rho(\{\phi_M(x_u, x_v, e_{uv})\}_{v \in \mathcal{V}_t})),
\]

(7)

with message function \( \phi_M \), aggregation operator \( \rho \) (e.g., the mean), and update function \( \phi_U \). Note that functions \( \phi_U, \phi_M \), and \( \rho \) have learnable parameters \( \theta \), which are shared across all nodes and time steps. An example of a message passing operator is when \( \phi_M(x_u, x_v, e_{uv}) = x_u \), \( \rho \) is the mean, and \( \phi_U \) 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 \( u \in \mathcal{V}_i \) as

\[
\begin{aligned}
\frac{dx_u}{dt} &= \phi_U(x_u, z_u, \rho(\{\phi_M(x_u, x_v, e_{uv})\}_{v \in \mathcal{V}_t})), \\forall u \in \mathcal{V}_i, \\
x_u(0) &= \psi(\mathcal{X}_{t_i-1}, u, x_u(t_{i-1}))
\end{aligned}
\]

(8)

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

\[
\frac{1}{T} \sum_{i=1}^{T} \sum_{u \in \mathcal{V}_i} \mathcal{L}(\mathcal{X}_{t_i}, u, \hat{x}_u(t_i))
\]

(9)

where prediction \( \hat{x}_u(t_i) \) 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

4026
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 $f_u$ defines the evolution of the states $x_u(t)$. At the bottom, the discretized solution of the node-wise ODE, which corresponds to our framework TG-ODE. The node embedding $x_u^\ell$ 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

$$x_u^{\ell+1} = x_u^{\ell} + \epsilon \sigma_U \left( M_u \left[ (x_u^\ell, x_v^\ell), \epsilon_{uv} \right] \right), \quad (10)$$

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

$$\frac{dx_u}{dt} = \sigma \left( \sum_{k=0}^{K} \sum_{v \in \mathcal{N}_u^{(k)} \cup \{u\}} \alpha_{u,v}^{(k)}(x_u \theta_v) \right), \quad (11)$$

where $\sigma$ is an activation function, $K$ is the number of hops in the neighborhood, $\theta_v$ 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, $\alpha_{u,v}^{(k)} = \left( \frac{\hat{d}_v}{d_u} \right)^{-1/2}$ weighs according to the degrees $\hat{d}_v$ and $d_u$ of nodes $v$ and $u$ in the $k$-hop graph; other choices can include edge attributes as well. We note that $\theta_v$ 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

$$x_u^\ell = \phi_U \left( x_u^{\ell-1}, \sum_{j \in \mathcal{N}_u} \phi_M \left( x_u^{\ell-1}, x_v^{\ell-1}, e_{uv} \right) \right), \quad (12)$$

where $\phi_U$ and $\phi_M$ 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; Velickovic 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 with temporal dependencies in the graph. 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-NODE [Choi et al., 2022] employs a stacked architecture of two neural controlled differential equations to model temporal and spatial information, respectively. In the STG-NODE’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. 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.

![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.](https://github.com/gravins/TG-ODE)

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.

---

1We 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 \( x_u(0) \) 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 stepsize \( \epsilon \). 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., 

\[
-LX(t), -L^2X(t), -L^3X(t), -\tanh(L)X(t), -5LX(t), -0.05LX(t), \text{ and } -(L + N_{0,1})X(t),
\]

where \( N_{0,1} \) stands for a noise sampled from a standard normal distribution, and \( L \) 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

\[
X^\ell = X^{\ell-1} + \epsilon \sigma \left( \sum_{k=0}^{K} L^k X^{\ell-1} \theta_k \right),
\]

where \( K \) corresponds to the number of neighborhood hops and \( \theta_k \) 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 [Chen 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 \( X_{t+1} \) is \( X(t+1) = X(t) \) 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, \( \tau \), 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 \( = \frac{(t_{i+1} - t_i)}{\tau} \).

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.

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
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., 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.

<table>
<thead>
<tr>
<th># Steps</th>
<th># Nodes</th>
<th># Edges</th>
<th>Timespan</th>
</tr>
</thead>
<tbody>
<tr>
<td>MetrLA</td>
<td>34,272</td>
<td>207</td>
<td>1,515</td>
</tr>
<tr>
<td>Montevideo</td>
<td>739</td>
<td>675</td>
<td>690</td>
</tr>
<tr>
<td>PeMS03</td>
<td>26,208</td>
<td>358</td>
<td>442</td>
</tr>
<tr>
<td>PeMS04</td>
<td>16,992</td>
<td>307</td>
<td>209</td>
</tr>
<tr>
<td>PeMS07</td>
<td>28,225</td>
<td>883</td>
<td>790</td>
</tr>
<tr>
<td>PeMS08</td>
<td>17,856</td>
<td>170</td>
<td>137</td>
</tr>
</tbody>
</table>

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.
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


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


