Long Range Propagation on Continuous-Time Dynamic Graphs

Alessio Gravina [*1] Giulio Lovisotto [*2] Claudio Gallicchio [1] Davide Bacciu [1] Claas Grohnfeldt [2]

Abstract

Learning Continuous-Time Dynamic Graphs (C-TDGs) requires accurately modeling spatio-temporal information on streams of irregularly sampled events. While many methods have been proposed recently, we find that most message passing-, recurrent- or self-attention-based methods perform poorly on *long-range* tasks. These tasks require correlating information that occurred "far" away from the current event, either spatially (higher-order node information) or along the time dimension (events occurred in the past). To address long-range dependencies, we introduce continuous-time graph anti-symmetric network (CTAN). Grounded within the ordinary differential equations framework, our method is designed for efficient propagation of information. In this paper, we show how CTAN's (i) long-range modeling capabilities are substantiated by theoretical findings and how (ii) its empirical performance on synthetic long-range benchmarks and real-world benchmarks is superior to other methods. Our results motivate CTAN's ability to propagate long-range information in C-TDGs as well as the inclusion of long-range tasks as part of temporal graph models evaluation.

1. Introduction

Graphs are a highly expressive abstraction for modelling entities and their relations, e.g., molecular structures, recommender systems, or traffic networks (Monti et al., 2019; Derrow-Pinion et al., 2021; Gravina et al., 2022; Errica et al., 2023; Cini et al., 2023; Bacciu et al., 2024). Deep Graph Networks (DGNs) (Bacciu et al., 2020; Wu et al., 2021) have lately emerged as a family of deep learning models that can effectively process and learn such structured information. While most of the proposed DGNs have been designed for static graphs, many real-world scenarios are inherently *dynamic* in nature. Examples include the continual activities and interactions between members of social as well as communication networks, recurrent purchases by users on e-commerce platforms, or evolving interactions of processes with files in an operating system. A number of works investigated models that can process the temporal dimension of a dynamic graph (Kazemi et al., 2020; Gravina & Bacciu, 2024), with recent interest in graphs defined through irregularly sampled event streams, known as Continuous-Time Dynamic Graphs (C-TDGs). However, dynamic methods, which are based on static DGNs and recurrent neural networks (RNNs) as backbone architectures, often retain the limitations of their core components. Specifically, static DGNs suffer from the *over-squashing* phenomenon (Alon & Yahav, 2021), which prevents the final network to learn and propagate long range information (Gravina et al., 2023). Similarly, RNNs often face similar challenges in propagating long-term dependencies, as evidenced by Chang et al. (2019), mainly due to exploding or vanishing gradients. With growing evidence from the static and dynamic case (Dwivedi et al., 2022; Yu et al., 2023) that long-range dependencies are necessary for effective learning, the ability to learn properties beyond the event's temporal and spatial locality remains an open challenge in the C-TDG domain.

In this paper, we propose the continuous-time graph anti-symmetric network (CTAN), a framework for learning of C-TDGs with *scalable* long range propagation of information, thanks to properties inherited from stable and non-dissipative ordinary differential equations (ODEs). We establish theoretical conditions for achieving stability and non-dissipation in the CTAN ODE by employing anti-symmetric weight matrices, which is the key factor for modeling long-range spatio-temporal interactions. The CTAN layer is derived from the forward Euler discretization of the designed differential equation. The formulation of CTAN allows scaling the radius of propagation of information depending on the number of discretization steps, i.e., the number of layers in the final architecture. Remarkably, even with a limited number of layers, the non-dissipative behavior enables the transmission of information for a past event as new events occur, since node states are used to efficiently

[*]Equal contribution, work done while at Huawei Technologies [1]Department of Computer Science, University of Pisa, Pisa, Italy [2]Huawei Technologies, Munich, Germany. Correspondence to: Alessio Gravina <alessio.gravina@phd.unipi.it>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

retain and propagate historical information. This mechanism permits scaling the single event propagation to cover a larger portion of the C-TDG. The general formulation of the node update state function allows the implementation of the more appropriate dynamic to the problem at hand. Specifically, it allows the inclusion of static DGN dynamics, thus reinterpreting current state-of-the-art static DGNs as a discretized representation of non-dissipative ODEs tailored for C-TDGs, mirroring previous approaches in the static case (Poli et al., 2019; Gravina et al., 2023). To the best of our knowledge, CTAN is the first framework to effectively address the problem of long-range propagation in C-TDGs and the first to bridge the gap between ODEs and C-TDGs.

The key contributions of this work can be summarized as follows: (i) We introduce the problem of long-range propagation (i.e., non-dissipativeness) within C-TDGs; (ii) We introduce CTAN, a new deep graph network for learning C-TDGs based on ODEs, which enables stable and non-dissipative propagation to preserve long term dependencies in the information flow, and it does so in a theoretically founded way; (iii) We present novel benchmark datasets specifically designed to assess the ability of DGNs to propagate information over long spatio-temporal distances within C-TDGs; (iv) We conduct extensive experiments to demonstrate the benefits of our method, showing that CTAN not only outperforms state-of-the-art DGNs on synthetic long-range tasks but also outperforms them on several real-world benchmark datasets.

2. Preliminaries

We consider a *dynamic graph* as the tuple $\mathcal{G}(t) = (\mathcal{V}(t), \mathcal{E}(t), \mathbf{X}(t), \mathbf{E}(t))$, defined for any time $t \geq 0$, which models a dynamical system of interacting entities (also known as *nodes*) where interactions (or *edges*) evolve over time, i.e., they are dynamic in nature. Here, $\mathcal{V}(t)$ is the set of nodes that are present in the graph at time t, and $\mathcal{E}(t) \subseteq \{\{u, v, t^-\} \mid u, v \in \mathcal{V}(t), t^- < t\}$ defines the edges between them, with t^- the time in which the edge $\{u, v\}$ was created. Matrices $\mathbf{X}(t) \in \mathbb{R}^{|\mathcal{V}(t)| \times d_n}$ and $\mathbf{E}(t) \in \mathbb{R}^{|\mathcal{E}(t)| \times d_e}$ contain node and edge features, respectively. The u-th row of $\mathbf{X}(t)$ is denoted as $\mathbf{x}_u$ and represents the features of the single node u. Similarly, we indicate the feature vector of the edge between nodes u and v created at time t as $\mathbf{e}_{uvt}$. Each node u is also associated to state $\mathbf{h}_u(t) \in \mathbf{H}(t) \in \mathbb{R}^{|\mathcal{V}(t)| \times d}$, which encodes node evolution over time t.

In our setting, the dynamic graph is observed as a *stream of events*, also known as observations, that can appear *irregularly* over time. Therefore, the system of interacting entities is not fully observed over time, and it is known as *C-TDG* (Nguyen et al., 2018; Kazemi et al., 2020; Gravina & Bacciu, 2024). In this scenario, the dynamic graph can be rewritten as $\mathcal{G} = \{o_t \mid t \in [t_0, t_n]\}$, where each event $o_t = (t,\, EventType,\, u,\, v,\, \mathbf{x}_u,\, \mathbf{x}_v,\, \mathbf{e}_{uvt})$ is a tuple containing information regarding the timestamp, the event type, the involved nodes, and their states. The event types can be grouped into three main classes, which are node-wise events (i.e., a node is updated or created), interaction events (i.e., an edge is created), and deletion events (i.e., a node/edge is deleted). In the following, we will refer to $\mathcal{V}_\oplus$ as the event "node creation", and to $\mathcal{E}_\oplus$ as "edge addition". We present in Figure 1 a visual exemplification of a C-TDG.

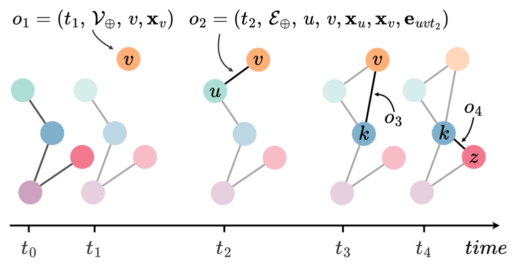

$o_1 = (t_1, \mathcal{V}_\oplus, v, \mathbf{x}_v)$
$o_2 = (t_2, \mathcal{E}_\oplus, u, v, \mathbf{x}_u, \mathbf{x}_v, \mathbf{e}_{uvt_2})$
v
u
v
v
k
o_3
k
o_4
z
t_0
t_1
t_2
t_3
t_4
time

Figure 1. The evolution of a Continuous-Time Dynamic Graph through the stream of events up to timestamp t_4. At each timestamp, the faded portion of the graph corresponds to historical information.

3. Continuous-Time Graph Anti-Symmetric Network (CTAN)

Learning the dynamics of a C-TDG can be cast as the problem of learning information propagation following newly observed events in the system. This entails learning a diffusion function that updates the state of node u as

$$\mathbf{h}_u(t) = F\left(t, \mathbf{x}_u, \mathbf{h}_u(t), \{\mathbf{h}_v(t)\}, \{\mathbf{e}_{uvt^-}\}\right), \qquad (1)$$

where $(v, t^-) \in \mathcal{N}_u^t$, and $\mathcal{N}_u^t = \{(v, t^-) \mid \{u, v, t^-\} \in \mathcal{E}(t)\}$ is the temporal neighborhood of a node u at time t, which consists of all the historical neighbors of u prior to current time t. In the following, we omit the time subscript from the edge feature vector to enhance readability, since it refers to a time in the past in which the edge appeared.

In recent literature, Eq. 1 is modeled through a dynamical system described by a learnable ordinary differential equation (ODE) (Poli et al., 2019; Chamberlain et al., 2021; Eliasof et al., 2021; Rusch et al., 2022; Gravina et al., 2023). Differently from discrete models, neural-ODE-based approaches learn more effective latent dynamics and have shown the ability to learn complex temporal patterns from irregularly sampled timestamps (Chen et al., 2018; Rubanova et al., 2019; Kidger et al., 2020), making them more suitable to address C-TDG problems.

In this paper, we leverage non-dissipative ODEs (Haber & Ruthotto, 2017; Chang et al., 2019; Gravina et al., 2023) for the processing of C-TDGs. Thus, we propose a framework as a solution to a *stable* and *non-dissipative* ODE over a streamed graph. The main goal of our work is therefore achieving preservation of long-range information between nodes over a stream of events. We do so by first showing how a generic ODE can learn the hidden dynamics of a C-TDG and then by deriving the condition under which the ODE is constrained to the desired behavior.

Modeling C-TDGs. First, we define a Cauchy problem in terms of the node-wise ODE defined in time $t \in [0, T]$

$$\frac{\partial \mathbf{h}_u(t)}{\partial t} = f_\theta\left(t, \mathbf{x}_u, \mathbf{h}_u(t), \{\mathbf{h}_v(t)\}_{v\in\mathcal{N}_u^t}, \{\mathbf{e}_{uv}\}_{v\in\mathcal{N}_u^t}\right) \tag{2}$$

and subject to an initial condition $\mathbf{h}_u(0) \in \mathbb{R}^d$. The term f_θ is a function parametrized by the weights θ that describes the dynamics of node state. We observe that this framework can naturally deal with events that arrive at an arbitrary time. Indeed, the original Cauchy problem in Eq. 2 can be divided into multiple sub-problems, one per each event in the C-TDG. The i-th sub-problem, defined in the interval $t \in [t_s, t_e]$, is responsible for propagating only the information encoded by the i-th event. Overall, when a new event o_i happens, the ODE in Eq. 2 computes new nodes representations $\mathbf{h}_u^i(t_e)$, starting from the initial configurations $\mathbf{h}_u^i(t_s)$. In other words, f_θ evolves the state of each node given its initial condition. The top-right of Figure 2 visually summarizes this concept, showing the nodes evolution given the propagation of an incoming event. We observe that the knowledge of past events is preserved and propagated in the system thanks to an initial condition that includes not only the current node input states but also the node representations computed in the previous sub-problem, i.e., $\mathbf{h}_u^i(t_s) = \psi(\mathbf{h}_u^{i-1}(t_e), \mathbf{x}_u(i))$. We notice that the terminal time t_e (treated as an hyper-parameter) is responsible for determining the extent of information propagation across the graph, since it limits the propagation to a constrained distance from the source. Consequently, smaller values of t_e allow only for localized event propagation, whereas larger values enable the dissemination of information to a broader set of nodes.

While this approach is applicable to all ODE-based DGNs for C-TDGs, we note that we are the first to introduce this truncated history propagation method in C-TDGs.

Non-Dissipativeness in C-TDGs. We now proceed to derive the condition under which the ODE is constrained to a stable and non-dissipative behavior, allowing for the propagation of long-range dependencies in the information flow. Non-Dissipativeness[1] in C-TDGs can be dissected into two components: non-dissipativeness over space and over time.

Definition 3.1 (*Non-dissipativeness over space*)**.** *Let $u, v \in \mathcal{V}(t)$ be two nodes of the C-TDG at some time t, connected by a path of length L. If an event o_i occurs at node u, then the information of o_i is propagated from u to v, $\forall L \geq 0$.*

We start by instantiating Eq. 2 as

$$\frac{\partial \mathbf{h}_u(t)}{\partial t} = \sigma\Big(\mathbf{W}_t\mathbf{h}_u(t) + \Phi\left(\{\mathbf{h}_v(t), \mathbf{e}_{uv}, t_v^-, t\}_{v\in\mathcal{N}_u^t}\right)\Big) \tag{3}$$

where σ is a monotonically non-decreasing activation function; Φ is the aggregation function that computes the representation of the neighborhood of the node u considering node states and edge features; t_v^- is the time point of the previous event for node v; and $\mathbf{W}_t \in \mathbb{R}^{d\times d}$. Here and in the following, the bias term is omitted for simplicity. We notice that including t_v^- in Φ encodes the time elapsed since the previous event involving node v. This inclusion allows for smooth updates of the node's current state during the time interval to prevent the staleness problem (Kazemi et al., 2020).

As discussed in (Haber & Ruthotto, 2017), a non-dissipative propagation is directly linked to the sensitivity of the solution of the ODE to its initial condition, thus to the stability of the system. Such sensitivity is controlled by the Jacobian's eigenvalues of Eq. 3. Given $\lambda_i(\mathbf{J}(t))$ the i-th eigenvalue of the Jacobian, when $Re(\lambda_i(\mathbf{J}(t))) = 0$ for $i = 1, ..., d$ the initial condition is effectively propagated into the final node representation, making the system both stable and non-dissipative[2].

Definition 3.2 (*Non-dissipativeness over time*)**.** *Let $u \in \mathcal{V}(t)$ be a node in the C-TDG at time t and o_t an event that occurs at node u at time t. A DGN for C-TDGs is non-dissipative over time if, regardless of how many more events subsequently occur at u, the information of event o_t will persist in u's embedding.*

In essence, Definition 3.2 captures the idea that the embedding computed by a DGN for a node in a C-TDG retains the information from a specific event indefinitely, ensuring that the historical context is preserved and not forgotten despite the occurrence of additional events at that node.

To show the property of non-dissipativity over time, we analyze the entire system defined in Eq. 3 from a temporal perspective. Thus, Eq. 3 can be reformulated as:

$$\frac{\partial \mathbf{h}_u(t)}{\partial t} = \sigma\Big(\mathbf{W}_t\psi(\mathbf{h}_u(t), \mathbf{x}_u(t)) + \Phi\left(\{\psi(\mathbf{h}_v(t), \mathbf{x}_v(t)), \mathbf{e}_{uv}, t_v^-, t\}_{v\in\mathcal{N}_u^t}\right)\Big) \tag{4}$$

where ψ is the function that computes the initial condition for the propagation of each event considering the node representations computed in the previous event propagation $\mathbf{h}_u(t)$ and the current node input state $\mathbf{x}_u(t)$ (as before).

[1]The reader is referred to (Glendinning, 1994; Ascher et al., 1995) for an in-depth analysis of dissipative dynamical systems.

[2]This result holds also when the eigenvalues of the Jacobian are still bounded in a small neighborhood around the imaginary axis (Gravina et al., 2023).

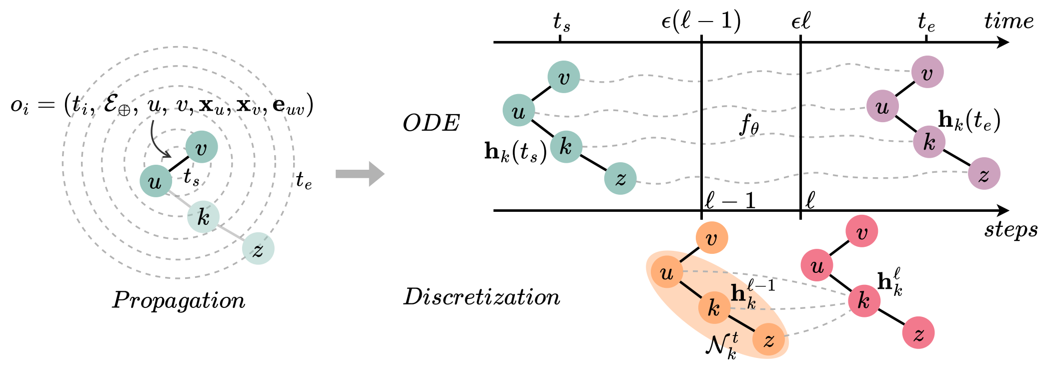


Figure 2. A high-level overview of the proposed framework illustrated for the i-th Cauchy sub-problem. On the left, we depict the propagation of the information of event o_i through the graph. The faded portion of the graph corresponds to historical information, while the rest is the incoming event. On the right, we illustrate the evolution of node states given the propagation of the incoming event. Specifically, the top right shows the evolution as an ODE, f_θ, that computes the node representation for a node k, $\mathbf{h}_k(t)$. Such computation is subject to an initial condition $\mathbf{h}_k(t_s) = \psi(\mathbf{h}_k^{i-1}(t_e), \mathbf{x}_k(i))$ that includes the node representations computed in the previous sub-problem $\mathbf{h}_k^{i-1}(t_e)$ and the current node input state. In the bottom right, the discretized solution of the ODE is computed as iterative steps of the method over a discrete set of points in the time interval $[t_s, t_e]$.

In this context, we can view the system as having $e\Delta t$ steps, where e denotes the number of events and Δt represents the propagation time of an event. Furthermore, the input state of the node $\mathbf{x}_u(t)$ is only present upon occurrence of a new event, meaning that during the propagation of events, $\mathbf{x}_u(t)$ is set to 0. Therefore, its impact on information propagation is confined to the event's specific occurrence and does not affect each step of the propagation process.

The next proposition ensures that when the eigenvalues of the Jacobian matrix of Eq. 4 are placed only on the imaginary axes, then the ODE in Eq. 4 is non-dissipative in both space and time. Thus, we guarantee the preservation of historical context over time and the propagation of event information through the C-TDGs.

Proposition 3.3. *Provided that the weight matrix $\mathbf{W_t}$ is anti-symmetric*[3] *and the aggregation function Φ does not depend on $\mathbf{h}_u(t)$, then the ODE in Eq. 4 is stable and non-dissipative over space and time if the resulting Jacobian matrix has purely imaginary eigenvalues, i.e.,*

$$Re(\lambda_i(\mathbf{J}(t))) = 0, \forall i = 1, ..., d.$$

For the proof, we refer the reader to (Gravina et al., 2023)(Appendix B) and substitute the Jacobian computed w.r.t. space with a Jacobian computed w.r.t. time.

By constraining weight matrix $\mathbf{W_t}$ to be anti-symmetric we obtain that the ODE in Eq. 3 is not-dissipative in both space and time, guaranteeing the preservation of historical node context over time while propagating event information "spatially" through the C-TDGs. We provide a more in-depth analysis of non-dissipativeness over time in Appendix A

[3]A matrix $\mathbf{M} \in \mathbb{R}^{d\times d}$ is anti-symmetric if $\mathbf{M}^\top = -\mathbf{M}$.

where we show that varying the formulation of ψ can yield to diverse behaviors.

Numerical discretization. Now that we have defined the conditions under which the ODE in Eq. 3 is stable and non-dissipative, i.e., it can propagate long-range dependencies between nodes in the C-TDG, we observe that computing the analytical solution of an ODE is usually infeasible. It is common practice to rely on a discretization method to compute an approximate solution by multiple applications of the method over a discrete set of points in the time interval $[t_s, t_e]$. This process is visually summarized at the bottom of Figure 2. We employ the *forward Euler's method* to discretize Eq. 3 for the i-th Cauchy sub-problem, yielding the following node state update equation for the node u at step ℓ:

$$\mathbf{h}_u^\ell = \mathbf{h}_u^{\ell-1} + \epsilon\sigma\Big((\mathbf{W}_\ell - \mathbf{W}_\ell^\top - \gamma\mathbf{I})\mathbf{h}_u^{\ell-1} + \Phi\left(\{\mathbf{h}_v(t), \mathbf{e}_{uv}, t_v^-, t\}_{v\in\mathcal{N}_u^t}\right)\Big), \quad (5)$$

with $\epsilon > 0$ being the discretization step size. We notice that the anti-symmetric weight matrix $(\mathbf{W}_\ell - \mathbf{W}_\ell^\top)$ is subtracted by the term $\gamma\mathbf{I}$ to preserve the stability of the forward Euler's method, see Appendix B for a more in-depth analysis. We refer to $\mathbf{I}$ as the identity matrix and γ to a hyper-parameter that regulates the stability of the discretized diffusion. We note that the resulting neural architecture contains as many layers as the discretization steps, i.e., $L = t_e/\epsilon$.

Truncated non-dissipative propagation. As previously discussed, the number of iterations in the discretization (i.e., the terminal time t_e) plays a crucial role in the propagation. Specifically, few iterations result in a localized event propagation. Consequently, the non-dissipative event propagation does not reach each node in the graph, causing a *truncated* non-dissipative propagation. This method allows scaling the radius of propagation of information depending on the number of discretization steps, thus allowing for a scalable long-range propagation in C-TDGs. Crucially, we notice that, even with few discretization steps, it is still possible to propagate information from a node u to z (if a path of length P connects u and z). As an example, consider the situation depicted in the left segment of Figure 2, where nodes u and v establish a connection at some time t, and our objective is to transmit this information to node z. In this scenario, we assume $L = 1$, thus the propagation is truncated before z. Upon the arrival of the event at time t, this is initially relayed (due to the constraint of $L = 1$) to node k, which then captures and retains this information. If a future event at time $t+\tau$ involving node k occurs, its state is propagated, ultimately reaching node z. Consequently, the information originating from node u successfully traverses the structure to reach node z. More formally, if it exists a sequence of (at least P/L) successive events, such that each future i-th event is propagated to an intermediate node at distance iP/L from u, then u is able to directly share its information with z. Therefore, even with a limited number of discretization steps, the non-dissipative behavior enables scaling the single event propagation to cover a larger portion of the C-TDG. We also notice that if the number of iterations is at least equal to the longest shortest path in the C-TDG, then each event is always propagated throughout the whole graph.

The CTAN framework. We name the framework defined above continuous-time graph anti-symmetric network (CTAN). Note that Φ in Eq. 3 and 5 can be any function that aggregates nodes and edges states. Then, CTAN can leverage the aggregation function that is more adequate for the specific task. As an exemplification of this, in Section 4 we leverage the aggregation scheme based on the one proposed by (Shi et al., 2021):

$$\Phi\left(\{\mathbf{h}_v(t), \mathbf{e}_{uv}, t_v^-, t\}_{v\in\mathcal{N}_u^t}\right) = \\ = \sum_{v\in\mathcal{N}_u^t\cup\{u\}} \alpha_{uv}\left(\mathbf{V}_n\mathbf{h}_v^{\ell-1} + \mathbf{V}_e\hat{\mathbf{e}}_{uv}\right) \quad (6)$$

where $\hat{\mathbf{e}}_{uv} = \mathbf{e}_{uv}\|\left(\mathbf{V}(t-t_v^-)\right)$ is the new edge representation computed as the concatenation between the original edge attributes and a learned embedding of the elapsed time from the previous neighbor interaction, and $\alpha_{uv} = \text{softmax}\left(\frac{\mathbf{q}^\top\mathbf{K}}{\sqrt{d}}\right)$ is the attention coefficient with d the hidden size of each head, $\mathbf{q} = \mathbf{V}_q\mathbf{h}_u^{\ell-1}$, and $\mathbf{K} = \mathbf{V}_k\mathbf{h}_v^{\ell-1} + \mathbf{V}_e\hat{\mathbf{e}}_{uv}$.

Despite CTAN being designed from the general perspective of layer-dependent weights, it can be used with weight sharing between layers (as in Section 4).

4. Experiments

To evaluate the performance of CTAN, we design two novel temporal tasks which require propagation of long-range information by design, Section 4.1.1 and Section 4.1.2. Afterwards, we assess the performance of the proposed CTAN approach on classical benchmarks for C-TDGs in Section 4.2. We complement these classical benchmarks with a larger evaluation on the TGB framework (Huang et al., 2023) in Appendix G, showcasing the model capabilities in diverse settings, covering evaluations with (i) improved negative sampling techniques and (ii) transductive and inductive settings. In Appendix H we conduct an investigation on the scalability property of CTAN. In Appendix D, we present comprehensive descriptions and statistics of the datasets. We release the long-range benchmarks and the code implementing our methodology and reproducing our analysis at `https://github.com/gravins/non-dissipative-propagation-CTDGs`.

Shared Experimental Settings. In the following experiments, we consider weight sharing of CTAN parameters across the neural layers. We compare CTAN against four popular dynamic graph network methods (i.e., DyRep (Trivedi et al., 2019), JODIE (Kumar et al., 2019), TGAT (Xu et al., 2020), and TGN (Rossi et al., 2020)) and include recent methods GraphMixer (Cong et al., 2023) and DyGFormer (Yu et al., 2023) for evaluation in long-range tasks. To ensure fair comparison and efficient implementation, we implement these methods in our framework. With the same purpose, we reuse the graph convolution operators in the original literature, considering for all methods the aggregation function defined in Eq. 6. We designed each model as a combination of two components: (i) the DGN (i.e., CTAN or a baseline) which is responsible to compute the node representations; (ii) the readout that maps node embeddings into the output space. The readout is a 2-layer MLP, used in all models with the same architecture. We perform hyper-parameter tuning via grid search, considering a fixed parameter budget based on the number of graph convolutional layers (GCLs). Specifically, for the maximum number of GCL in the grid, we select the embedding dimension so that the total number of parameters matches the budget; such embedding dimension is used across every other configuration. We report more detailed information on each task in their respective subsections. Detailed information about hyper-parameter grids and training of models are in Appendix E. While we do not directly investigate the optimal terminal time t_e within the hyper-parameter space, we implicitly address this aspect through the choice of the step size ϵ and the maximum number of layers L, as they jointly determine the terminal time, i.e., $t_e = \epsilon L$.

4.1. Long Range Tasks

Here, we introduce two temporal tasks which contain long-range interaction (LRI). The first is a *Sequence Classification* task on path graphs (Bondy & Murty, 1976) and the second an extension to the temporal domain of the classification task *PascalVOC-SP* introduced in the Long Range Graph Benchmark (Dwivedi et al., 2022).

4.1.1. Sequence Classification on Temporal Path Graph

Setup. Inspired by the tasks in (Chang et al., 2019), we consider a sequence classification task requiring long-range information on a temporal interpretation of a path graph (Bondy & Murty, 1976). Here, the nodes of the path graph appear sequentially over time from first to last, i.e., each event in the C-TDG connects each node to the previous one in the path graph (see Appendix D for a reference to our dataset). The task objective is to predict the feature observed at the source node in the first event after having traversed the entire temporal path graph, i.e., after reaching the last event in the stream. After the model processes the last event in the graph, the output prediction for the whole graph is computed by a readout that takes as input the updated embedding of the destination node of the last event in the C-TDG. The task requires models to propagate the information seen at the first node through the entire sequence. Models that exhibit smoothing or dissipative behavior will fail to transmit relevant information to the destination node for longer sequences, resulting in poor performance.

When creating the dataset, we set the feature of the first source node to be either 1 or -1, and we use uniformly random sampled features for intermediate nodes and edges to ensure the only task-relevant information is on the earliest node. We forward events one at a time to update neighboring nodes representations (i.e., batch size is 1). We considered graphs of different sizes, from length 3 to 20, to test how long information is propagated, i.e., longer graphs force models to propagate information for longer. During training, we optimize the binary cross-entropy loss over two classes corresponding to the two possible signals (1 or -1) placed on the initial node. Each experimental run is repeated 10 times for different weight initializations; the grid is computed considering a budget of $\sim$20k trainable parameters. The best performing configuration is chosen based on the validation loss. Appendix E reports more training details and the grid of hyper-parameters.

Results. The test accuracy on the sequence classification task is in Table 1 (comprehensive results are reported in Appendix F). CTAN exhibits exceptional performance in comparison to reference state-of-the-art methods. This result highlights the capability of our method to propagate information seen on the first node throughout long paths. Meanwhile, several baseline models struggle in solving such a task because the information is lost through the time-steps: in practice, informative gradients vanish over time. Note that, memory-less methods such as TGAT, GraphMixer and DyGformer can not effectively propagate information past the number of layers (i.e., hops) used in the neighbor aggregation. Note that while the latter two methods are designed for 1-hop aggregation, TGAT allows for variable number of GCLs aggregations, which we test up to 5. We notice TGAT can solve the task at distance 5, but fails for longer graphs. JODIE and TGN are memory-based methods, which grants them the ability to solve tasks for longer distances, but being RNN-based methods inherently struggle to maintain long-term dependencies (Bengio et al., 1994; Chang et al., 2019). TGN fails at distance 7, while JODIE at distance 15. CTAN on the other hand, better propagates information for longer distances, solving the task even at length 20.

Table 1. Results of the sequence classification on path graph long-range task, for increasing graph length n. The performance metric is the mean test set accuracy score, averaged over 10 different random weights initializations for each model configuration.

	n=3	n=9	n=15	n=20
DyGFormer	**100.0**$\pm$0.0	53.02$\pm$6.06	42.80$\pm$16.25	42.79$\pm$19.62
DyRep	**100.0**$\pm$0.0	47.93$\pm$2.73	48.60$\pm$2.48	50.47$\pm$2.88
GraphMixer	**100.0**$\pm$0.0	52.80$\pm$5.56	52.49$\pm$5.36	52.04$\pm$8.20
JODIE	**100.0**$\pm$0.0	**100.0**$\pm$0.0	60.0$\pm$14.91	50.87$\pm$2.46
TGAT	**100.0**$\pm$0.0	47.87$\pm$2.72	50.53$\pm$2.15	49.07$\pm$1.55
TGN	**100.0**$\pm$0.0	48.13$\pm$1.63	48.67$\pm$2.76	50.13$\pm$2.17
Our	**100.0**$\pm$0.0	99.93$\pm$0.21	**93.47**$\pm$8.78	**88.93**$\pm$12.06

4.1.2. Classification on Temporal Pascal-VOC

Setup. We consider edge classification on a temporal interpretation of the `PascalVOC-SP` dataset, which has been previously employed by (Dwivedi et al., 2022) as a benchmark to show the efficacy of capturing LRI in static graphs. Here, we adapt the task to the C-TDG domain: we forward edges one at a time and predict the class of the destination node. We generate temporal graphs starting from the dataset of `rag-boundary` graphs extracted from Pascal VOC 2011 provided in (Dwivedi et al., 2022) (more details are provided in Appendix D). We consider two degrees of SLIC superpixels compactness, i.e., 10 and 30. Larger compactness means more patches, with less information included in each patch and more to be propagated.

During training, we optimize for the F1-score as in (Dwivedi et al., 2022). To benchmark the ability of models to propagate information through the graph, we test model performance for an increasing number of GCLs. Fewer GCLs require models to store and transmit relevant information along node embeddings rather than relying on effectively aggregating information from increasingly larger neighborhoods. Each experimental run is repeated 5 times for multiple weight initializations. The hyper-parameter grid is computed considering a budget of trainable parameters per model equal to $\sim$40k. Appendix E provides further training and model selection details.

Results. Table 2 reports the average F1-score on the temporal PascalVOC-SP task. Note that DyRep, JODIE, GraphMixer and DyGFormer, in their original definition, do not support a variable number of GCLs, hence the results of such models are presented in the table under "1 GCL" for clarity. CTAN largely outperforms reference methods. We observe that for SLIC compactness equal to 30, CTAN achieves a 65% and 16% improvement against the second best performing model (i.e., TGAT), for one and three GCLs, respectively. Interestingly, TGAT almost matches the performance of CTAN when considering five GCLs. This is in line with the excellent results of computationally expensive Transformers-based models in the static case (Dwivedi et al., 2022), corroborating the advantages of self-attention blocks in modeling long-range dependencies between far away nodes. This result also suggests that the majority of the relevant information necessary to solve the temporal Pascal VOC task may lie within neighborhoods five hops away. We note that at SLIC compactness 10, DyGFormer benefits from the shorter long-range propagation (when sc=10 the graph contains fewer patches, hence fewer nodes and spatially closer relevant information compared to sc=30), and from its deeper architecture compared to CTAN's single-layer design, when considering the same number of spatial hops. In fact, in this setting DyGFormer contains two transformer blocks, while CTAN does not. However, we observe that by including multiple layers of CTAN (i.e., no.GCLs > 1), our method effectively propagates information and outperforms DyGFormer even in the sc=10 task. Nevertheless, the results indicate how CTAN is capable of propagating relevant information across the time-steps to achieve accurate predictions, even when the model is only allowed to extract information from limited, very local neighborhoods.

4.2. Future Link Prediction Task

Setup. For the C-TDG benchmarks we consider four well-known datasets proposed by (Kumar et al., 2019), Wikipedia, Reddit, LastFM, and MOOC, to assess the model performance in real-world setting, with the task of future link prediction (Kumar et al., 2019). We perform hyper-parameter tuning via grid search by optimizing the area under the roc curve (AUC). Results for the best configuration are provided as average on 5 random weight initializations. To give models a fair setting for comparison, the grid is computed considering a budget of $\sim$140k trainable parameters per model and the neighbor sampler size is set to 5. Appendix E provides additional experimental details.

Results. Table 3 reports the average test AUC on the C-TDG benchmarks. CTAN shows remarkable performance, ranking first across datasets. Our method achieves a score that on average is 4.7% better than other baselines. This finding shows the importance of a non-dissipative behavior of the method even on real-world tasks, since more information need to be retained and propagated from the past to improve the final performance. Our results demonstrate that CTAN is able to better capture and exploit such information. Nevertheless, note that not all real-world datasets inherently present long-range dependencies. To evaluate how CTAN fares against state-of-the-art methods on several datasets, we complement this analysis with an evaluation on the TGB Benchmark (Huang et al., 2023), see Appendix G. In this setting, CTAN characterizes by the best performing behaviour when considering the combination of TGB datasets.

5. Related Work

Deep Graph Network for C-TDGs. Nowadays, most of the DGNs tailored for learning C-TDGs can be generalized within the Temporal Graph Network (TGN) framework (Rossi et al., 2020). This architecture comprises three main modules: a message module, which is responsible for computing a message that encodes the incoming event; a memory module, which stores the node histories; and a graph propagation module, which aggregates information from the local neighborhood to produce the final node representation. Usually, the memory module is implemented as a Recurrent Neural Network (RNN) and the graph propagation module as a DGN for the processing of static graphs. Many state-of-the-art architectures (Kumar et al., 2019; Trivedi et al., 2019; Xu et al., 2020; Ma et al., 2020; Souza et al., 2022) fit this framework, with later methods outperforming earlier ones thanks to advances in the local message passing part or even in the encoding of positional features. Two recent methods (Cong et al., 2023; Yu et al., 2023) focus on modeling long-range (time) dependencies by including longer node histories in the context while not relying on memory modules. While recent methods often provide improved results, none of them explicitly models long-range *temporal and spatial* dependencies between nodes or events in the C-TDG. As increasingly evidenced both in sequence-model architectures (Chang et al., 2019), and in the static graph case (Dwivedi et al., 2022), propagating information across various time steps is extremely beneficial for learning.

CTAN, instead, provably enables effective long-range propagation by design. Note that our approach does not require the co-existence of memory *and* graph propagation module, as in the TGN framework. CTAN stores all necessary information within the node embeddings themselves as computed by the graph convolution, while achieving non-dissipative propagation by design. This makes CTAN more lightweight.

Table 2. Results of the classification on the Temporal PascalVOC task, for increasing number of GCLs. The performance metric is the mean test set F1-score, averaged over 5 different random weights initializations for each model configuration.

	Temporal Pascal VOC (sc=10)			**Temporal Pascal VOC (sc=30)**		
no. GCLs	1	3	5	1	3	5
DyGFormer	**8.45**$_{\pm 0.13}$	–	–	8.07$_{\pm 0.27}$	–	–
DyRep	5.29$_{\pm 0.47}$	–	–	5.23$_{\pm 0.11}$	–	–
GraphMixer	6.60$_{\pm 0.11}$	–	–	5.88$_{\pm 0.08}$	–	–
JODIE	6.33$_{\pm 0.41}$	–	–	5.76$_{\pm 0.35}$	–	–
TGAT	5.39$_{\pm 0.19}$	6.53$_{\pm 0.58}$	8.23$_{\pm 0.73}$	6.04$_{\pm 0.26}$	8.79$_{\pm 0.29}$	10.38$_{\pm 0.7}$
TGN	6.04$_{\pm 0.27}$	6.55$_{\pm 0.46}$	7.51$_{\pm 0.80}$	5.59$_{\pm 0.24}$	7.26$_{\pm 0.82}$	7.90$_{\pm 1.31}$
Our	7.89$_{\pm 0.33}$	**8.53**$_{\pm 1.06}$	**8.88**$_{\pm 0.98}$	**9.98**$_{\pm 0.33}$	**10.16**$_{\pm 0.52}$	**10.41**$_{\pm 0.52}$

Table 3. Results of the future link prediction task. We report the mean test set AUC and std in percent averaged over 5 random weight initializations.

	Wikipedia	**Reddit**	**LastFM**	**MOOC**
DyRep	88.64$_{\pm 0.15}$	97.51$_{\pm 0.10}$	77.89$_{\pm 1.39}$	81.87$_{\pm 2.47}$
JODIE	94.68$_{\pm 1.05}$	96.34$_{\pm 0.83}$	69.76$_{\pm 2.74}$	81.90$_{\pm 9.03}$
TGAT	94.91$_{\pm 0.25}$	98.18$_{\pm 0.05}$	81.53$_{\pm 0.34}$	87.61$_{\pm 0.15}$
TGN	95.60$_{\pm 0.18}$	98.23$_{\pm 0.10}$	79.18$_{\pm 0.79}$	90.74$_{\pm 0.99}$
Our	**97.55**$_{\pm 0.09}$	**98.61**$_{\pm 0.04}$	**83.81**$_{\pm 0.92}$	**92.47**$_{\pm 0.78}$

Lastly, as TGN allows for different graph propagation modules, the general formulation of the aggregation function Φ in Eq. 5 allows extending state-of-the-art DGNs for static graphs to the domain of C-TDGs through the lens of non-dissipative and stable ODEs.

Continuous Dynamic Models. Neural Differential Equations have emerged as a class of neural networks suitable for learning continuous dynamics of systems. (Chen et al., 2018) and (Chang et al., 2019) parameterize the continuous dynamic of RNNs through an ordinary differential equation. Similarly, (Gallicchio, 2022) draws a connection with Reservoir Computing. Despite the similarity with RNNs, such architectures have shown the ability to naturally incorporate data that arrive at arbitrary times (Chen et al., 2018; Rubanova et al., 2019). Inspired by the NeuralODE approach, GDE (Poli et al., 2019) links DGNs for static graphs with ODEs. In this scenario, the inter-layer dynamic of DGN's node representation is designed as a continuous information processing system defined by an ODE, which, starting from the input configuration of the nodes' states, computes the final node representations. In the static graph domain, ODE-based architectures have been proposed with different aims, such as reducing the computational complexity of message passing (Wu et al., 2019; Wang et al., 2021), or mitigating the over-smoothing phenomena (Eliasof et al., 2021; Rusch et al., 2022).

Gravina et al. (2023) proposes A-DGN, an ODE-based model achieving non-dissipative propagation through static graphs, i.e., in the time-unaware spatial domain. We note that time-aware nodes and edges combined with possibly irregularly sampled repetitive edges between the same pair of nodes natively render A-DGN (as well as other methods designed for static graphs) inapplicable to C-TDGs. Less trivially, non-dissipative propagation in C-TDGs cannot be achieved through mere non-dissipative propagation through space. On the contrary, non-dissipative propagation of information through time is a property unique to DGNs designed for C-TDG, necessary for their overall non-dissipativeness.

To the best of our knowledge, we are the first to propose an ODE-based architecture suitable for C-TDGs that can effectively propagate long-range information between nodes.

6. Conclusion

We presented continuous-time graph anti-symmetric network (CTAN), a new framework based on stable and non-dissipative ODEs for learning long-range interactions in Continuous-Time Dynamic Graphs (C-TDGs). Differently from previous approaches, CTAN's formulation allows scaling the radius of effective propagation of information in C-TDGs (i.e., allowing for a scalable long-range propagation in C-TDGs) and reimagines state-of-the-art static DGNs as a discretization of non-dissipative ODEs for C-TDGs. To the best of our knowledge, CTAN is the first framework to address the long-range propagation problem in C-TDGs, while bridging the gap between ODEs and C-TDGs.

Our experimental investigation reveals, at first, that when it comes to capturing long-range dependencies in a task, our framework significantly surpasses state-of-the-art DGNs for C-TDGs. Our experiments indicate that CTAN is capable of propagating relevant information incrementally across time to achieve accurate predictions, even when the model is only allowed to extract information from very local neighborhoods, i.e., by using only a single or few layers. Thus, CTAN enables scaling the extent of information propagation in C-TDG data structures without increasing the number of layers nor incurring in dissipative behaviors. Moreover, our results indicate that CTAN is effective across various graph benchmarks in both real and synthetic scenarios. In essence, CTAN showcased its ability to explore long-range dependencies (even with limited resources), suggesting its potential in mitigating over-squashing in C-TDGs.

We believe that CTAN lays down the basis for further investigations of the problem of over-squashing and long-range interaction learning in the C-TDG domain. Looking ahead to future developments, we plan to extend this study to explore alternative architectures resulting from different discretization methods, such as adaptive multi-step schemes (Ascher & Petzold, 1998). Additionally, we aim to assess the framework's impact in the realm of efficient neural networks, such as in Reservoir Computing (Nakajima & Fischer, 2021).

Impact Statement

This paper aims to contribute to the field of Machine Learning, specifically focusing on advancing Deep Graph Networks (DGNs) in the Continuous-Time Dynamic Graph (C-TDG) setting. The research presented herein has the potential to positively impact the ongoing exploration and applications of DGNs designed for C-TDGs. As far as we are aware, our work does not raise any ethical issues.

Acknowledgments

The authors would like to thank Michele Russo, Huawei Technologies, Munich, Germany, and Jakub Reha, University of Amsterdam, Amsterdam, Netherlands, for the insightful discussions throughout the development of this work. The work has been partially supported by EU-EIC EMERGE (Grant No. 101070918).

References

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., and Süsstrunk, S. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282, 2012.

Alon, U. and Yahav, E. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=i80OPhOCVH2.

Ascher, U., Mattheij, R., and Russell, R. *Numerical solution of boundary value problems for ordinary differential equations*. Classics in applied mathematics. Society for Industrial and Applied Mathematics (SIAM), United States, unabridged, corr. republication. edition, 1995. ISBN 0-89871-354-4.

Ascher, U. M. and Petzold, L. R. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. Society for Industrial and Applied Mathematics, USA, 1st edition, 1998. ISBN 0898714125.

Bacciu, D., Errica, F., Micheli, A., and Podda, M. A gentle introduction to deep learning for graphs. *Neural Networks*, 129:203–221, 2020. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2020.06.006. URL https://www.sciencedirect.com/science/article/pii/S0893608020302197.

Bacciu, D., Errica, F., Gravina, A., Madeddu, L., Podda, M., and Stilo, G. Deep Graph Networks for Drug Repurposing With Multi-Protein Targets. *IEEE Transactions on Emerging Topics in Computing*, 12(1):177–189, 2024. doi: 10.1109/TETC.2023.3238963.

Bengio, Y., Simard, P., and Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994. doi: 10.1109/72.279181.

Bondy, J. A. and Murty, U. S. R. *Graph Theory with Applications*. Elsevier, New York, 1976.

Chamberlain, B., Rowbottom, J., Gorinova, M. I., Bronstein, M., Webb, S., and Rossi, E. Grand: Graph neural diffusion. In *International Conference on Machine Learning*, pp. 1407–1418. PMLR, 2021.

Chang, B., Chen, M., Haber, E., and Chi, E. H. AntisymmetricRNN: A dynamical system view on recurrent neural networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryxepo0cFX.

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/69386f6bb1dfed68692a24c8686939b9-Paper.pdf.

Cini, A., Marisca, I., Bianchi, F. M., and Alippi, C. Scalable Spatiotemporal Graph Neural Networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(6):7218–7226, Jun. 2023. doi: 10.1609/aaai.v37i6.25880. URL https://ojs.aaai.org/index.php/AAAI/article/view/25880.

Cong, W., Zhang, S., Kang, J., Yuan, B., Wu, H., Zhou, X., Tong, H., and Mahdavi, M. Do we really need complicated model architectures for temporal networks? *arXiv preprint arXiv:2302.11636*, 2023.

Derrow-Pinion, A., She, J., Wong, D., Lange, O., Hester, T., Perez, L., Nunkesser, M., Lee, S., Guo, X., Wiltshire, B., Battaglia, P. W., Gupta, V., Li, A., Xu, Z., Sanchez-Gonzalez, A., Li, Y., and Velickovic, P. Eta prediction with graph neural networks in google maps. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, CIKM '21, pp. 3767–3776, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450384469. doi: 10.1145/3459637.3481916. URL https://doi.org/10.1145/3459637.3481916.

Dwivedi, V. P., Rampášek, L., Galkin, M., Parviz, A., Wolf, G., Luu, A. T., and Beaini, D. Long range graph benchmark. *Advances in Neural Information Processing Systems*, 35:22326–22340, 2022.

Eliasof, M., Haber, E., and Treister, E. PDE-GCN: Novel architectures for graph neural networks motivated by partial differential equations. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=wWtk6GxJB2x.

Errica, F., Gravina, A., Bacciu, D., and Micheli, A. 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.

Everingham, M., Eslami, S. A., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111:98–136, 2015.

Gallicchio, C. Euler state networks: Non-dissipative reservoir computing. *arXiv preprint arXiv:2203.09382*, 2022.

Glendinning, P. *Stability, Instability and Chaos: An Introduction to the Theory of Nonlinear Differential Equations*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 1994. doi: 10.1017/CBO9780511626296.

Gravina, A. and Bacciu, D. Deep Learning for Dynamic Graphs: Models and Benchmarks. *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–14, 2024. doi: 10.1109/TNNLS.2024.3379735.

Gravina, A., Wilson, J. L., Bacciu, D., Grimes, K. J., and Priami, C. Controlling astrocyte-mediated synaptic pruning signals for schizophrenia drug repurposing with deep graph networks. *PLOS Computational Biology*, 18(5):1–19, 05 2022. doi: 10.1371/journal.pcbi.1009531. URL https://doi.org/10.1371/journal.pcbi.1009531.

Gravina, A., Bacciu, D., and Gallicchio, C. Anti-Symmetric DGN: a stable architecture for Deep Graph Networks. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=J3Y7cgZOOS.

Haber, E. and Ruthotto, L. Stable architectures for deep neural networks. *CoRR*, abs/1705.03341, 2017. URL http://arxiv.org/abs/1705.03341.

Huang, S., Poursafaei, F., Danovitch, J., Fey, M., Hu, W., Rossi, E., Leskovec, J., Bronstein, M., Rabusseau, G., and Rabbany, R. Temporal graph benchmark for machine learning on temporal graphs. *arXiv preprint arXiv:2307.01026*, 2023.

Kazemi, S. M., Goel, R., Jain, K., Kobyzev, I., Sethi, A., Forsyth, P., and Poupart, P. Representation Learning for Dynamic Graphs: A Survey. *J. Mach. Learn. Res.*, 21(1), jan 2020. ISSN 1532-4435.

Kidger, P., Morrill, J., Foster, J., and Lyons, T. Neural Controlled Differential Equations for Irregular Time Series. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Kumar, S., Zhang, X., and Leskovec, J. Predicting Dynamic Embedding Trajectory in Temporal Interaction Networks. In *Proceedings of the 25th ACM SIGKDD KDD*. ACM, 2019.

Ma, Y., Guo, Z., Ren, Z., Tang, J., and Yin, D. Streaming Graph Neural Networks. In *Proceedings of the 43rd International ACM SIGIR*, pp. 719–728. Association for Computing Machinery, 2020. ISBN 9781450380164. doi: 10.1145/3397271.3401092.

Monti, F., Frasca, F., Eynard, D., Mannion, D., and Bronstein, M. M. Fake news detection on social media using geometric deep learning. *CoRR*, abs/1902.06673, 2019. URL http://arxiv.org/abs/1902.06673.

Nakajima, K. and Fischer, I. *Reservoir Computing: Theory, Physical Implementations, and Applications*. Natural Computing Series. Springer Nature Singapore, 2021. ISBN 9789811316876. URL https://books.google.it/books?id=AQc8EAAAQBAJ.

Nguyen, G. H., Lee, J. B., Rossi, R. A., Ahmed, N. K., Koh, E., and Kim, S. Continuous-Time Dynamic Network Embeddings. In *Companion Proceedings of the*

The Web Conference 2018, WWW '18, pp. 969–976, Republic and Canton of Geneva, CHE, 2018. International World Wide Web Conferences Steering Committee. ISBN 9781450356404. doi: 10.1145/3184558.3191526.

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

Poursafaei, F., Huang, S., Pelrine, K., , and Rabbany, R. Towards Better Evaluation for Dynamic Link Prediction. In *Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks*, 2022.

Rossi, E., Chamberlain, B., Frasca, F., Eynard, D., Monti, F., and Bronstein, M. Temporal Graph Networks for Deep Learning on Dynamic Graphs. In *ICML 2020 Workshop on Graph Representation Learning*, 2020.

Rubanova, Y., Chen, R. T. Q., and Duvenaud, D. K. Latent Ordinary Differential Equations for Irregularly-Sampled Time Series. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/42a6845a557bef704ad8ac9cb4461d43-Paper.pdf.

Rusch, T. K., Chamberlain, B. P., Rowbottom, J., Mishra, S., and Bronstein, M. M. Graph-coupled oscillator networks. *arXiv preprint arXiv:2202.02296*, 2022.

Shi, Y., Huang, Z., Feng, S., Zhong, H., Wang, W., and Sun, Y. Masked Label Prediction: Unified Message Passing Model for Semi-Supervised Classification. In Zhou, Z.-H. (ed.), *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pp. 1548–1554. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/214. URL https://doi.org/10.24963/ijcai.2021/214. Main Track.

Souza, A. H., Mesquita, D., Kaski, S., and Garg, V. K. Provably expressive temporal graph networks. In *NeurIPS*, 2022.

Trivedi, R., Farajtabar, M., Biswal, P., and Zha, H. DyRep: Learning Representations over Dynamic Graphs. In *ICLR*, 2019.

Wang, Y., Wang, Y., Yang, J., and Lin, Z. Dissecting the diffusion process in linear graph convolutional networks. *CoRR*, abs/2102.10739, 2021. URL https://arxiv.org/abs/2102.10739.

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. Simplifying graph convolutional networks. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6861–6871. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/wu19e.html.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2021. doi: 10.1109/TNNLS.2020.2978386.

Xu, D., Ruan, C., Korpeoglu, E., Kumar, S., and Achan, K. Inductive representation learning on temporal graphs. In *ICLR*, 2020.

Yu, L. An empirical evaluation of temporal graph benchmark. *arXiv preprint arXiv:2307.12510*, 2023.

Yu, L., Sun, L., Du, B., and Lv, W. Towards better dynamic graph learning: New architecture and unified library. *arXiv preprint arXiv:2303.13047*, 2023.

A. Non-dissipativeness Over Time

We note that the non-dissipative behavior of the system in Eq. 4 is contingent on the specific definition of the function ψ. Varying the formulation of ψ can yield to diverse behaviors, significantly impacting the system's ability to either preserve or dissipate information over time.

Proposition A.1. *Provided that the aggregation function Φ does not depend on $\mathbf{h}_u(t)$, the Jacobian matrix resulting from the ODE in Eq. 4 has purely imaginary eigenvalues, i.e., $Re(\lambda_i(\mathbf{J}(t))) = 0, \forall i = 1, ..., d$ if the function ψ is implemented as one of the following functions:*

- *addition, i.e., $\psi = \mathbf{h}_u(t) + \mathbf{x}_u(t)$;*
- *concatenation, i.e., $\psi = \mathbf{h}_u(t) \| \mathbf{x}_u(t)$;*
- *composition of tanh and concatenation, i.e., $\psi = tanh(\mathbf{h}_u(t) \| \mathbf{x}_u(t))$.*

Proof. Let's consider $\psi = \mathbf{h}_u(t) + \mathbf{x}_u(t)$, i.e., **addition**. In this case Eq. 4 can be reformulated as

$$\frac{\partial \mathbf{h}_u(t)}{\partial t} = \sigma\Big(\mathbf{W}_t\mathbf{h}_u(t) + \mathbf{W}_t\mathbf{x}_u(t) + \Phi\left(\{(\mathbf{h}_u(t) + \mathbf{x}_u(t)), \mathbf{e}_{uv}, t_v^-, t\}_{v\in\mathcal{N}_u^t}\right)\Big). \tag{7}$$

The Jacobian matrix of Eq. 7 is defined as

$$\mathbf{J}(t) = \text{diag}\left[\sigma'\Big(\mathbf{W}_t\mathbf{h}_u(t) + \mathbf{W}_t\mathbf{x}_u(t) + \Phi\left(\{(\mathbf{h}_u(t) + \mathbf{x}_u(t)), \mathbf{e}_{uv}, t_v^-, t\}_{v\in\mathcal{N}_u^t}\right)\Big)\right]\mathbf{W}_t. \tag{8}$$

Thus, it is the result of a matrix multiplication between invertible diagonal matrix and a weight matrix. Imposing $\mathbf{A} = \text{diag}\left[\sigma'\left(\mathbf{W}_t\mathbf{h}_u(t) + \Phi\left(\{\mathbf{h}_v(t), \mathbf{e}_{uv}, t_v^-, t\}_{v\in\mathcal{N}_u^t}\right) + \mathbf{b}_t\right)\right]$, then the Jacobian can be rewritten as $\mathbf{J}(t) = \mathbf{A}\mathbf{W}_t$.

Let us now consider an eigenpair of $\mathbf{A}\mathbf{W}_t$, where the eigenvector is denoted by $\mathbf{v}$ and the eigenvalue by λ. Then:

$$\begin{aligned}\mathbf{A}\mathbf{W}_t\mathbf{v} &= \lambda\mathbf{v},\\ \mathbf{W}_t\mathbf{v} &= \lambda\mathbf{A}^{-1}\mathbf{v},\\ \mathbf{v}^*\mathbf{W}_t\mathbf{v} &= \lambda(\mathbf{v}^*\mathbf{A}^{-1}\mathbf{v})\end{aligned} \tag{9}$$

where $*$ represents the conjugate transpose. On the right-hand side of Eq. 9, we can notice that the $(\mathbf{v}^*\mathbf{A}^{-1}\mathbf{v})$ term is a real number. If the weight matrix $\mathbf{W}_t$ is anti-symmetric (i.e., skew-symmetric), then it is true that $\mathbf{W}_t^* = \mathbf{W}_t^\top = -\mathbf{W}_t$. Therefore, $(\mathbf{v}^*\mathbf{W}_t\mathbf{v})^* = \mathbf{v}^*\mathbf{W}_t^*\mathbf{v} = -\mathbf{v}^*\mathbf{W}_t\mathbf{v}$. Hence, the $\mathbf{v}^*\mathbf{W}_t\mathbf{v}$ term on the left-hand side of Eq. 9 is an imaginary number. Thereby, λ needs to be purely imaginary, and, as a result, all eigenvalues of $\mathbf{J}(t)$ are purely imaginary.

Let's now consider $\psi = \mathbf{h}_u(t) \| \mathbf{x}_u(t)$, i.e., **concatenation**. In this case, the product $\mathbf{W}_t(\mathbf{h}_u(t) \| \mathbf{x}_u(t))$ can be decomposed as $\mathbf{K}_t\mathbf{h}_u(t) + \mathbf{V}_t\mathbf{x}_u(t)$, with $\mathbf{K}_t$ and $\mathbf{V}_t$ weight matrices. Similarly to the addition case, the Jacobian has purely imaginary eigenvalues.

Lastly, we consider the case of $\psi = tanh(\mathbf{h}_u(t) \| \mathbf{x}_u(t))$, i.e., the **composition of *tanh* and concatenation**. Here, Eq. 4 is

$$\frac{\partial \mathbf{h}_u(t)}{\partial t} = \sigma\left(\mathbf{W}_t tanh(\mathbf{h}_u(t)) + \mathbf{V}_t tanh(\mathbf{x}_u(t)) + \Phi\left(\{tanh(\mathbf{h}_u(t)\|\mathbf{x}_u(t)), \mathbf{e}_{uv}, t_v^-, t\}_{v\in\mathcal{N}_u^t}\right)\right). \tag{10}$$

The Jacobian matrix is the results of the multiplication of three matrices, i.e., $\mathbf{J}(t) = \mathbf{A}\mathbf{B}\mathbf{W}_t$, with $\mathbf{A} = \text{diag}\left[\sigma'\left(\mathbf{W}_t tanh(\mathbf{h}_u(t)) + \mathbf{V}_t tanh(\mathbf{x}_u(t)) + \Phi(...) + \mathbf{b}\right)\right]$ and $\mathbf{B} = \text{diag}[1 - tanh^2(\mathbf{h}_u(t))]$. Thanks to the associative property of multiplication $\mathbf{J}(t) = \mathbf{A}\mathbf{B}\mathbf{W}_t = (\mathbf{A}\mathbf{B})\mathbf{W}_t = \mathbf{D}\mathbf{W}_t$, where $\mathbf{D}$ is the result of the multiplication of two diagonal matrices, thus $\mathbf{D}$ is diagonal. As detailed for the addition case, we can conclude that the Jacobian matrix has purely imaginary eigenvalues. □

As a counterexample, if $\psi = \mathbf{x}_u(t)$, Eq. 4 can result in a dissipative behavior, leading to the loss of information over time and compromising the model's ability to preserve historical context, since past node information is always discarded between new events. As a result, the function ψ can function as a parameter to control the balance between the dissipative and non-dissipative behavior of CTAN.

B. Stability of the Forward Euler's Method

Following (Ascher & Petzold, 1998), the Euler's forward method applied to Eq. 3 is considered stable when $(1 + \epsilon\lambda(\mathbf{J}(t)))$ lies within the unit circle in the complex plane for all eigenvalues of the system. However, since the eigenvalues of the Jacobian matrix are exclusively imaginary, it follows that $|1 + \epsilon\lambda(\mathbf{J}(t))| > 1$, thus Eq. 3 is unstable when solved with forward Euler's method.

To enhance the stability of the numerical discretization method, we subtract a small positive constant $\gamma > 0$ from the diagonal elements of the weight matrix $\mathbf{W}$. This adjustment allows the eigenvalues of the Jacobian to possess a slightly negative real part, which positions $(1 + \epsilon\lambda(\mathbf{J}(t)))$ within the unit circle and enhancing the stability of the numerical discretization method. However, as detailed in Section 3, since $Re(\lambda_i(\mathbf{J}(t))) < 0$, the ODE becomes slightly dissipative. In conclusion, the term γ can serve as a parameter for balancing the dissipative and non-dissipative behavior.

C. Summary of CTAN's Propagation Capacity

In this section, we gather the information regarding the theoretically infinite propagation capacity of our method CTAN. Section 3 provides the theoretical conditions (see Proposition 3.3) under which CTAN is non-dissipative over space and time (Definition 3.1 and 3.2), allowing for the preservation of historical node context over time while propagating event information spatially through the C-TDG. When Proposition 3.3 is satisfied, the information propagation rate is constant independently of time, since the magnitude of $\partial\mathbf{h}(t)/\partial\mathbf{h}(0)$ is constant over time. As a consequence, theoretically, there is always information flowing within the CTDG modeled by a non-dissipative (space and time) model.

For propagation over time, Appendix A discusses which choices of ψ guarantee non-dissipativeness over time, in this case the upper bound of the range length of propagation is theoretically infinite. See Appendix A for examples of dissipative and non-dissipative ψ functions.

For propagation over space, Section 3 - paragraph "Truncated non-dissipative propagation" discusses how the terminal diffusion time t_e influences the range length of propagation, which is lower bounded by the number of GCLs.

D. Datasets Description and Statistics

Table 4 contains the statistics of the employed datasets. In the following, we describe the datasets and their generation.

Table 4. Statistics of the datasets used in our experiments. We report the total number of nodes and edges in the dataset for the temporal path graph (i.e., T-PathGraph) and temporal Pascal VOC (i.e., T-PascalVOC).

	# Nodes	# Edges	# Edge ft.	Split	Surprise Index
T-PathGraph	3,000-20,000	2,000-19,000	1	70/15/15	1.0
T-PascalVOC$_{10}$	2,671,704	2,660,352	14	70/15/15	1.0
T-PascalVOC$_{30}$	2,990,466	2,906,113	14	70/15/15	1.0
Wikipedia	9,227	157,474	172	70/15/15, Chronological	0.42
Reddit	11,000	672,447	172	70/15/15, Chronological	0.18
LastFM	2,000	1,293,103	2	70/15/15, Chronological	0.35
MOOC	7,144	411,749	4	70/15/15, Chronological	0.79
tgbl-wiki-v2	9,227	157,474	172	70/15/15, Chronological	0.108
tgbl-review-v2	352,637	4,873,540	-	70/15/15, Chronological	0.987
tgbl-coin-v2	638,486	22,809,486	-	70/15/15, Chronological	0.120
tgbl-comment	994,790	44,314,507	-	70/15/15, Chronological	0.823

Sequence classification on temporal path graphs. To craft a temporal long-range problem, we first introduced a sequence classification problem on path graphs (Bondy & Murty, 1976), which is a simple linear graph consisting of a sequence of nodes where each node is connected to the previous one. In the temporal domain, the nodes of the path graph appear sequentially over time from first to last (e.g., bottom-to-top in Figure 3).

We define the task objective as the prediction of the feature seen in the first node (colored in orange in Figure 3) by making the prediction leveraging only the last node representation (colored in red in Figure 3) computed at the end of the sequence, i.e., when the last event appears. Note that this task is akin to the sequence classification task designed in (Chang et al.,

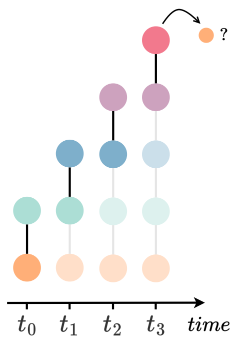


Figure 3. The illustration of the sequence classification task on a temporal path graph consisting of 5 nodes. The first node (colored in orange) has an initial feature that can be either 1 or -1. All the other nodes and edges have a feature set to random value sampled uniformly in $[-1, 1]$. At the end of the sequence, the representation computed for the last node (colored in red) is used to predict the original value of the first node. At each timestamp, the faded portion of the graph corresponds to historical information.

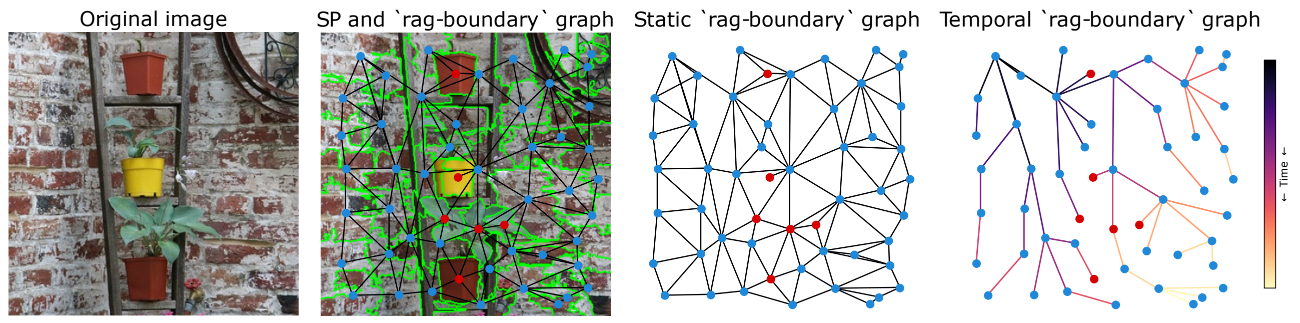

Original image
SP and `rag-boundary` graph
Static `rag-boundary` graph
Temporal `rag-boundary` graph
← Time ←

Figure 4. Construction of the Temporal `PascalVOC-SP` dataset. The SLIC algorithm extracts patches from an image. We create the `rag-boundary` graph connecting neighboring patches based on spatial closeness. We construct a temporal graph by traversing from the topleftmost node with BFS. The goal of the task is to predict the class of the destination node at each visited edge - in the figure, either 'potted plant' (red) or 'background' (blue). For clarity in this visualization, the compactness of the SLIC algorithm is low.

2019), with the addition of a graph convolution. We set the feature of the first node to be either 1 or -1, while we set every other node and edge feature to be sampled uniformly in the range $[-1, 1]$. In other words, the feature $\mathbf{x}_{u_0}$ of the first node u_0 contains a signal to be remembered as noise is added through the propagations steps along the graph. Formally, we create a C-TDG: $\mathcal{G} = \{o_t \,|\, t \in [t_0, t_n]\}$, such that

$$o_t = (t,\, \mathcal{E}_{\oplus},\, u_t,\, u_{t+1}, \mathbf{x}_{u_t}, \mathbf{x}_{u_{t+1}}, \mathbf{e}_{u_t,u_{t+1}}),$$

where $\mathbf{x}_{u_0} \sim \text{Bernoulli}(0.5)$[4], and $\mathbf{x}_{u_j} \sim \mathcal{U}_{[-1,1]}, \forall j > t_0$ and $\mathbf{e}_{u_t,u_{t+1}} \sim \mathcal{U}_{[-1,1]}, \forall t$.

For this task we considered 8 temporal graph path datasets with different sizes, ranging from $n = 3$ to $n = 20$, with n the number of nodes. For every graph size we generate 1,000 different graphs, and we split the dataset into train/val/test with the ratios 70%-15%-15%.

Temporal Pascal-VOC. We use the `PascalVOC-SP` (Dwivedi et al., 2022) dataset to design a new temporal long-range task for edge classification. `PascalVOC-SP` is a node classification dataset composed of graphs created from the images in the Pascal VOC 2011 dataset (Everingham et al., 2015). A graph is derived from each image by extracting superpixel nodes using the SLIC algorithm (Achanta et al., 2012) and constructing a `rag-boundary` graph to interconnect these nodes. Each node in a graph corresponds to one region of the image belonging to a particular class, see Figure 4 for an example. `PascalVOC-SP` contains long-range interactions between spatially distant image patches, evidenced by its average shortest path length of 10.74 and average diameter of 27.62 (Dwivedi et al., 2022).

To craft a temporal task, we consider that nodes in a `rag-boundary` graph appear from the top-left to the bottom-right of

[4]Note that we sample 1 or -1 rather than 0 or 1 to make the problem balanced around zero.

the image, sequentially. We do so by selecting the top-leftmost node, i.e., the one closest (by means of L_1 norm) to the origin in image coordinates. From this node, we traverse the graph with a Breadth-First-Search, visiting each node exactly once. The order of edge traversal corresponds to the timestamp of edge appearance in the temporal task. We set the task's objective to be the prediction of the class of the node that is being visited by the current edge. Note that the traversal removes a large number of edges from the initial graph, making the propagation of class information more difficult, see Figure 4.

Neighborhoods are constructed based on coordinates, connecting a node with its 8 spatially closest neighbors. Nodes have 12 features extracted by channel-wise statistics on the image (mean, std, max, min) and 2 features defining the spatial location of the superpixel; we normalize these spatial features in the [0, 1] range. We consider two SLIC superpixels compactness of 10 and 30 (smaller compactness means fewer patches). To allow for batching, we fix the number of nodes in each graph, allowing batching of edges that occur at the same timestep across different graphs together. To do so, we discard `rag-boundary` graphs with fewer nodes than the limit, and discard excess nodes on graphs with more nodes than the limit, according to time (i.e., the most recent nodes are dropped). This removes a small number of nodes corresponding to image patches on the bottom-right of the image. In practice, for the two compactness levels 10 and 30, we set the number of minimum nodes per graph to be 434 and 474, which gives us 6,156 and 6,309 temporal graphs (out of the total 11,355 images in the dataset). The resulting temporal datasets have 2,660,352 and 2,906,113 edges respectively.

C-TDG benchmarks. For the C-TDG benchmarks on future link prediction we consider four well-known datasets proposed by (Kumar et al., 2019):

- **Wikipedia**: one month of interactions (i.e., 157,474 interactions) between user and Wikipedia pages. Specifically, it corresponds to the edits made by 8,227 users on the 1,000 most edited Wikipedia pages;
- **Reddit**: one month of posts (i.e., interactions) made by 10,000 most active users on 1,000 most active subreddits, resulting in a total of 672,447 interactions;
- **LastFM**: one month of who-listens-to-which song information. The dataset consists of 1000 users and the 1000 most listened songs, resulting in 1,293,103 interactions.
- **MOOC**: it consists of actions done by students on a MOOC online course. The dataset contains 7,047 students (i.e., users) and 98 items (e.g., videos and answers), resulting in 411,749 interactions.

Since the datasets do not contain negative instances, we perform negative sampling by randomly sampling non-occurring links in the graph, as follows: (i) during training we sample negative destinations only from nodes that appear in the training set, (ii) during validation we sample them from nodes that appear in training set or validation set and (iii) during testing we sample them from the entire node set.

For all the datasets, we considered the same chronological split into train/val/test with the ratios 70%-15%-15% as proposed by (Xu et al., 2020).

Transductive vs Inductive Settings. In Section 4.2 we employed transductive setting and random negative sampling as in (Kumar et al., 2019; Xu et al., 2020; Rossi et al., 2020; Yu et al., 2023; Cong et al., 2023). We chose not to employ an inductive setting as it is not easily applicable to C-TDGs. Specifically, there is no clear consensus in the literature regarding the definition of inductive settings, making it difficult to identify the nodes considered for assessing this experimental setup (e.g. (Xu et al., 2020) differs from (Rossi et al., 2020)). Some definitions of inductive settings lead to the number of sampled inductive nodes to be not statistically relevant for evaluation. Other interpretations of inductive settings disrupt the true dynamics of the graph, i.e., in (Rossi et al., 2020), certain nodes and their associated edges are removed from the training set with the purpose of isolating an inductive set of nodes. Thanks to the analysis performed in (Yu et al., 2023), we can also observe that among all the considered datasets in our paper there is mix of inductive and transductive edges, which can be measured with the *surprise index* from (Yu et al., 2023), measuring the proportion of unseen edges at test time; reported in Table 4. Hence, achieving strong performance on tasks with a high surprise index offers valuable insights into the model's capability to address the inductive setting. Comparing CTAN performance to the surprise index, it is clear that CTAN can cope reasonably well even in fully inductive tasks, such as those in Section 4.1 where it generally ranks first among other baselines.

E. Explored Hyper-Parameter Space

In Table 5 we report the grids of hyper-parameters employed in our experiments by each method. We recall that the hyper-parameters ϵ, γ, and ψ refer only to our method. We used dropout only for GraphMixer and DyGFormer, where the values are loosely based on best-performing values in Yu (2023).

Table 5. The grid of hyper-parameters employed during model selection for the following three tasks: Sequence classification on temporal path graphs, Temporal Pascal-VOC, and Link Prediction – here, abbreviated as and color-coded in (*Seq*, orange), (*Pasc*, green), and (*Link*, blue), respectively. For *Seq* and *Pasc*, we conducted 10 runs and 5 runs with different random seeds for different weight initializations *for each configuration*, whereas for *Link*, we conducted 5 runs only for the configuration that resulted in the best performance in the initial run. For the three tasks, the models were configured to have a maximum number of learnable parameters of ~20k, ~40k, and ~140k, respectively. Training was conducted for 20 epochs, 200 epochs, and 1000 epochs, respectively. For *Seq* and *Pasc*, we employed a scheduler *halving the learning rate* with a patience of 5 epochs, 20 epochs, respectively, whereas for *Link* we used early stopping with a patience of 50 epochs. For all tasks, the neighbor sampler size was set to 5. The batch size was set to 128, 256, and 256, respectively. We used the loss, F1-score, and AUC on the validation set to optimize for the hyper-parameters. We used dropout only for GraphMixer and DyGFormer, where the values are loosely based on best-performing values in (Yu, 2023).

Hyper-parameter	**Method**	**Values**		
		Seq	Pasc	Link
optimizer		Adam		
learning rate		$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$	10^{-4}, 10^{-5}
weight decay		10^{-7}	10^{-5}	10^{-6}
n. GCLs		1, 3, 5		
σ		tanh		
ϵ		1, 0.5, 10^{-1}, 10^{-2}	1, 0.5, 10^{-1}, 10^{-2}	0.5, 10^{-1}, 10^{-2}, 10^{-3}
γ		1, 0.5, 10^{-1}, 10^{-2}	1, 0.5, 10^{-1}, 10^{-2}	0.5, 10^{-1}, 10^{-2}, 10^{-3}
ψ		concat, $\psi = \tanh(\mathbf{h}^{i-1}(t_e)\|\|\mathbf{x}(i))$		
dropout		0.1, 0.2	0.1, 0.2	–
time dim		1	1	16
memory dim (= DGN dim)	DyGFormer	10, 5	14, 7	–
	DyRep	53, 26	74, 37	118, 87
	GraphMixer	30, 15	24, 12	–
	JODIE	69, 34	97, 48	164, 122
	TGAT	24, 12	24, 12	33, 23
	TGN	19, 9	21, 10	33, 20
	CTAN	53, 26	74, 37	128, 96

F. Complete Results

In Table 6 we report the results on the sequence classification task on temporal path graphs, and in Table 7 we show the complete results on the link prediction task, including the performance of EdgeBank (Poursafaei et al., 2022) with different time window sizes. EdgeBank is a memorization-based method without learning that simply stores previously observed edges from a fixed-size time-window from the immediate past, and predicts stored edges as positive. We evaluated EdgeBank with different time windows spanning from a size of 1% of the training set to infinite size, i.e., all observed edges are stored in memory.

In this scenario, we observe that EdgeBank is particularly good at capturing long-range information along the time dimension in the LastFM task, surpassing all the baselines and CTAN as the time window increases. We highlight that the experiments in Section 4.2 are meant to outline how CTAN outperforms baselines under an *even field* of number of trainable parameters (i.e., 140k) and restricted range of hyper-parameter values, e.g., sampler size equal to 5. On the other hand, EdgeBank is a non-parametric method that at the time of inference accesses the entire temporal adjacency matrix. In LastFM, the median node degree after training is 903 (mean 1152 ± 1722), which is high compared to other datasets. At validation time, for the average node in LastFM, EdgeBank pools information from 903 node neighbors, while the setting in Section 4.2 allows baselines to pool information from 5 randomly sampled neighbors. As nodes have larger degrees, sampling larger

Table 6. Results of the sequence classification on path graph long-range task, for increasing graph length n. The performance metric is the mean test set accuracy score, averaged over 10 different random weights initializations for each model configuration. Models have a maximum budget of learnable parameters equal to ∼20k.

	n=3	n=5	n=7	n=9	n=11	n=13	n=15	n=20
DyGFormer	**100.0**$_{\pm 0.0}$	42.55$_{\pm 16.95}$	52.94$_{\pm 7.3}$	53.02$_{\pm 6.06}$	51.80$_{\pm 9.52}$	51.70$_{\pm 8.52}$	42.80$_{\pm 16.25}$	42.79$_{\pm 19.62}$
DyRep	**100.0**$_{\pm 0.0}$	49.20$_{\pm 2.10}$	51.00$_{\pm 1.76}$	47.93$_{\pm 2.73}$	44.87$_{\pm 0.89}$	46.73$_{\pm 1.55}$	48.60$_{\pm 2.48}$	50.47$_{\pm 2.88}$
GraphMixer	**100.0**$_{\pm 0.0}$	42.58$_{\pm 21.2}$	55.40$_{\pm 6.44}$	52.80$_{\pm 5.56}$	44.65$_{\pm 19.42}$	43.77$_{\pm 16.51}$	52.49$_{\pm 5.36}$	52.04$_{\pm 8.20}$
JODIE	**100.0**$_{\pm 0.0}$	**100.0**$_{\pm 0.0}$	**100.0**$_{\pm 0.0}$	**100.0**$_{\pm 0.0}$	98.53$_{\pm 4.64}$	97.4$_{\pm 7.99}$	60.0$_{\pm 14.91}$	50.87$_{\pm 2.46}$
TGAT	**100.0**$_{\pm 0.0}$	**100.0**$_{\pm 0.0}$	50.67$_{\pm 4.12}$	47.87$_{\pm 2.72}$	42.67$_{\pm 2.15}$	43.53$_{\pm 0.83}$	50.53$_{\pm 2.15}$	49.07$_{\pm 1.55}$
TGN	**100.0**$_{\pm 0.0}$	**100.0**$_{\pm 0.0}$	60.20$_{\pm 13.2}$	48.13$_{\pm 1.63}$	45.07$_{\pm 1.64}$	44.40$_{\pm 0.64}$	48.67$_{\pm 2.76}$	50.13$_{\pm 2.17}$
Our	**100.0**$_{\pm 0.0}$	**100.0**$_{\pm 0.0}$	**100.0**$_{\pm 0.0}$	99.93$_{\pm 0.21}$	**99.6**$_{\pm 0.56}$	**98.67**$_{\pm 1.89}$	**93.47**$_{\pm 8.78}$	**88.93**$_{\pm 12.06}$

neighborhoods is fundamental to access and therefore retain information. To show that CTAN performance is limited by the considered range of hyper-parameter values, we present in Table 8 the performance of CTAN by solely adjusting the neighbor sampler size, while maintaining a budget of ∼350k learnable parameters. The evaluation involves substituting various sampler size values into the optimal combination of hyper-parameters obtained for CTAN on the LastFM dataset (as in Section 4.2), with the embedding dimension configured to achieve the target of ∼350k learnable parameters (i.e., 192). The results indicate that CTAN performs better by adjusting the sampler size alone.

Table 7. Mean test set AUC and std in percent averaged over 5 random weight initializations. Each model have a maximum budget of learnable weights equal to ∼140k. The higher, the better.

	Wikipedia	**Reddit**	**LastFM**	**MOOC**
EdgeBank$_{1\%\ tr\ set}$	71.03	71.92	77.59	61.29
EdgeBank$_{5\%\ tr\ set}$	81.65	85.07	86.75	63.93
EdgeBank$_{10\%\ tr\ set}$	85.26	89.07	89.87	65.18
EdgeBank$_{25\%\ tr\ set}$	88.31	92.92	92.74	67.49
EdgeBank$_{50\%\ tr\ set}$	90.29	94.82	94.06	69.63
EdgeBank$_{75\%\ tr\ set}$	91.11	95.63	94.55	70.46
EdgeBank$_{100\%\ tr\ set}$	91.52	96.08	94.69	70.80
EdgeBank$_{\infty}$	91.82	96.42	**94.72**	70.85
DyRep	88.64$_{\pm 0.15}$	97.51$_{\pm 0.10}$	77.89$_{\pm 1.39}$	81.87$_{\pm 2.47}$
JODIE	94.68$_{\pm 1.05}$	96.34$_{\pm 0.83}$	69.76$_{\pm 2.74}$	81.90$_{\pm 9.03}$
TGAT	94.91$_{\pm 0.25}$	98.18$_{\pm 0.05}$	81.53$_{\pm 0.34}$	87.61$_{\pm 0.15}$
TGN	95.60$_{\pm 0.18}$	98.23$_{\pm 0.10}$	79.18$_{\pm 0.79}$	90.74$_{\pm 0.99}$
Our	**97.55**$_{\pm 0.09}$	**98.61**$_{\pm 0.04}$	**83.81**$_{\pm 0.92}$	**92.47**$_{\pm 0.78}$

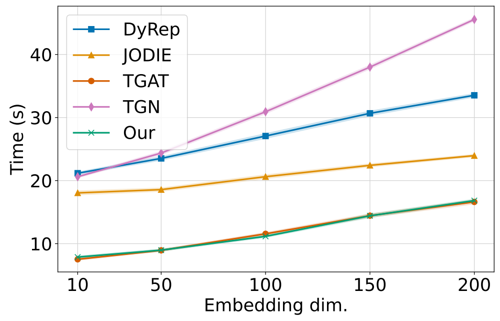
Table 8. Mean test set AUC and std on LastFM (in percent) for increasing size of sampled neighbors, averaged over three different weights initializations. The model has a budget of learnable weights equal to ∼350k. When nodes have large degrees as in LastFM, accessing larger neighborhoods with the neighbor sampler is fundamental to access and retain important information.

Sampler size	2	8	16	32	64	128
CTAN	82.64$_{\pm 0.93}$	86.21$_{\pm 0.58}$	86.16$_{\pm 0.55}$	86.27$_{\pm 0.55}$	86.32$_{\pm 0.81}$	**87.82**$_{\pm 0.42}$

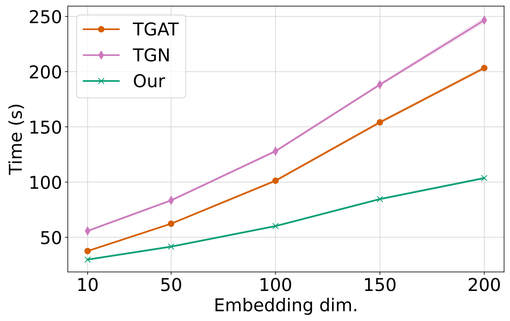
We report the average time per epoch (measured in seconds) for each model on the four considered link prediction datasets in Table 9. In this evaluation, each model has the same embedding dimension and number of GCLs. Similarly, Figure 5 shows the average time per epoch of each model on the Wikipedia dataset. Here, the time is reported with respect to a varying embedding size and similar number of GCLs. We observe that our method has a speedup on average of $1.3\times$ to $2.2\times$ on the four benchmarks when one layer of graph convolutions is considered, and $1.5\times$ to $1.9\times$ when five layers are used.

Table 9. Mean time (in seconds) and std averaged over 10 epochs. Each model is run with an embedding dimension equal to 100 on an Intel(R) Xeon(R) Gold 6278C CPU @ 2.60GHz.

		Wikipedia	**Reddit**	**LastFM**	**MOOC**
1 layer	DyRep	$27.07_{\pm 0.32}$	$161.43_{\pm 0.96}$	$216.88_{\pm 2.83}$	$53.32_{\pm 0.56}$
	JODIE	$20.62_{\pm 0.24}$	$131.71_{\pm 0.85}$	$176.61_{\pm 3.02}$	$43.92_{\pm 0.68}$
	TGAT	$11.56_{\pm 0.14}$	$67.83_{\pm 0.64}$	$139.79_{\pm 20.78}$	$\mathbf{33.92}_{\pm 0.50}$
	TGN	$30.92_{\pm 0.25}$	$196.87_{\pm 1.35}$	$289.22_{\pm 30.38}$	$53.46_{\pm 0.62}$
	Our	$\mathbf{11.16}_{\pm 0.11}$	$\mathbf{64.48}_{\pm 0.56}$	$\mathbf{123.19}_{\pm 11.33}$	$34.42_{\pm 0.50}$
5 layer	TGAT	$101.26_{\pm 0.46}$	$895.35_{\pm 5.46}$	$862.47_{\pm 217.38}$	$73.77_{\pm 1.29}$
	TGN	$127.99_{\pm 0.60}$	$1099.19_{\pm 3.91}$	$1034.24_{\pm 221.04}$	$95.45_{\pm 1.07}$
	Our	$\mathbf{60.16}_{\pm 0.20}$	$\mathbf{532.36}_{\pm 9.87}$	$\mathbf{495.18}_{\pm 111.13}$	$\mathbf{56.19}_{\pm 0.63}$



(a)



(b)

Figure 5. Average time per epoch (measured in seconds) and std with respect to the embedding size computed on the Wikipedia dataset, averaged over 10 epochs. The experiments were carried out on an Intel(R) Xeon(R) Gold 6278C CPU @ 2.60GHz. On the left (a), each model has 1 DGN layer (when possible), while on the right (b) the models have 5 GCLs.

G. Results on TGB Benchmarks

We evaluate CTAN on the Temporal Graph Benchmark (TGB) (Huang et al., 2023). TGB contains a set of real-world small-to-large scale benchmark datasets with varying graph properties. While TGB contains two different graph tasks, namely dynamic link property prediction and dynamic node property prediction, for consistency with the rest of the presentation in this paper we focus on the former. To overcome the existing limitations on negative edge sampling, i.e., where only one random negative edge is sampled per each positive edge, TGB provides pre-sampled negative edge sets with both *random* and *historical* negatives (Poursafaei et al., 2022). Here, for each positive edge, several negatives are sampled for the evaluation (Huang et al., 2023). Please refer to (Huang et al., 2023) for more information on datasets and tasks.

Experimental Setup. We adapt and re-use the TGB training loop and evaluation loop to fit our framework, and we select four datasets for measuring CTAN performance: *tgbl-wiki-v2, tgbl-review-v2, tgbl-coin-v2, tgbl-comment*. We use the configuration of CTAN, reported in Table 10. Note that for computational efficiency, since validation passes are extremely costly given the large number of negative edges, we only do a validation pass every three training epochs.

Results. In Table 11, we report the test Mean Reciprocal Rank (MRR) for the experiments. We note that CTAN performs quite well in general: its average rank across the four datasets is 3.25 which is the highest, together with DyGFormer (Yu et al., 2023). CTAN performs quite well on tgbl-review-v2, even significantly outperforming state-of-the-art methods DyGFormer (Yu et al., 2023) and GraphMixer (Cong et al., 2023). In such dataset the *surprise index* (Poursafaei et al., 2022) is 0.987, meaning that nodes do not have large histories. In this case, it seems that CTAN better propagates information from neighbors compared to methods focusing on first-hop information passing such as (Cong et al., 2023) and (Yu et al., 2023). On the other hand, it seems that the model in (Yu et al., 2023) is well suited in propagating long-range *time* information by modeling a large number of previous node interactions within the transformer input sequence, given enough computational budget, particularly in tgbl-wiki-v2, where nodes have long histories. Nevertheless, we notice that even with limited number of parameters, CTAN is extremely competitive within the leaderboard.

Table 10. The grid of hyper-parameters employed during model selection for CTAN on the Dynamic Link Property Prediction task on the three TGB benchmark datasets considered: tgbl-wiki-v2, tgbl-review-v2, tgbl-coin-v2, tgbl-comment. For tgbl-wiki-v2 we conducted five runs with different random seeds for different weight initializations for each configuration, whereas for the other datasets we conducted three different runs. The rest of the training configuration is taken from the TGB codebase: batch size is 200, weight decay penalty was 0, the optimized metric is Mean Reciprocal Rank and is evaluated with the TGB evaluator.

Hyper-parameter	**tgbl-wiki-v2**	**tgbl-review-v2**	**tgbl-coin-v2**	**tgbl-comment**
optimizer	Adam			
σ	tanh			
γ	0.1	0.1, 0.01	0.1, 0.01	0.1
ψ	concat, $\psi = \tanh(\mathbf{h}^{i-1}(t_e)\vert\vert\mathbf{x}(i))$			
n. GCLs	1, 2, 3	1,2	1	1
ϵ	1.0	0.5, 1.0	0.5, 1.0	1.0
embedding dim	256			
sampler size	32			
learning rate	10^{-3}, 10^{-4}, $3 \cdot 10^{-4}$, $3 \cdot 10^{-5}$	$3 \cdot 10^{-6}$	10^{-4}	10^{-4}, $3 \cdot 10^{-4}$, 10^{-5}, $3 \cdot 10^{-5}$
epochs	200	50	50	50
LR scheduler patience	20	3	3	3

H. Scalability of CTAN

Here we conduct an investigation on the scalability property of CTAN. Note that while in some related works the term scalable refers to the computational complexity of methods, here we use scalable to refer to how the range of information propagation can be controlled by increasing the number of graph convolutions in CTAN. To show this property, we show the results of the task in Section 4.1.1 at different values of GCLs (when possible). We report the results in Figure 6, which shows how for increasing GCLs, CTAN is capable of conveying information further away in the graph compared to other graph convolutional based models. In addition, we observe that both DyGFormer and GraphMixer may have increased capability to capture long-range dependencies, however, this is only applicable to *time*-only dependencies, and not spatial ones. Indeed, DyGFormer and GraphMixer model long-range time dependencies on node representations by fetching previous interactions for a node, both only relying on first-hop neighbors information and not considering spatial propagation of higher-order node information, which is in fact mentioned as a limitation of DyGFormer. Comparably, CTAN remains a graph convolution-based model, hence capable of propagating information in a non-dissipative way over time as well as over the spatial dimension of the graph, scaling the range of propagation with the number of discretization steps (equivalently, the termination time t_e). This property enables propagating information to neighbors beyond first-hop ones, which in turns allows solving tasks such as those in Section 4.1.1 and 4.1.2 in the paper.

Table 11. Results of the Dynamic Link Property Prediction task on the TGB benchmark datasets (Huang et al., 2023). The table reports the average MRR on the test split of the datasets over the considered weight initializations. For CTAN, the average is taken over a maximum of five runs with different random seeds for different weight initializations. All baselines' results are taken from (Yu, 2023). The number of parameters is computed from the TGB Baselines repository (Huang et al., 2023) by loading the best performing model across the model selection search.

	N. params	**tgbl-wiki-v2**	**tgbl-review-v2**	**tgbl-coin-v2**	**tgbl-comment**	**Avg. rank**
EdgeBank$_\infty$	–	52.50	2.29	35.90	10.87	11
EdgeBank$_{\text{tw-ts}}$	–	63.25	2.94	57.36	12.44	8.25
EdgeBank$_{\text{re}}$	–	65.88	2.84	59.15	–	8.25
EdgeBank$_{\text{th}}$	–	52.81	1.97	43.36	–	11.33
CAWN	4M	$73.04_{\pm 0.60}$	$19.30_{\pm 0.10}$	–	–	5.50
DyRep	700k	$51.91_{\pm 1.95}$	$40.06_{\pm 0.59}$	$45.20_{\pm 4.60}$	$28.90_{\pm 3.30}$	8.00
GraphMixer	600k	$59.75_{\pm 0.39}$	$36.89_{\pm 1.50}$	**$75.57_{\pm 0.27}$**	**$76.17_{\pm 0.17}$**	4.25
DyGFormer	1.1M	**$79.83_{\pm 0.42}$**	$22.39_{\pm 1.52}$	$75.17_{\pm 0.38}$	$67.03_{\pm 0.14}$	**3.25**
JODIE	200k	$63.05_{\pm 1.69}$	**$41.43_{\pm 0.15}$**	–	–	4.50
TCL	900k	$78.11_{\pm 0.20}$	$16.51_{\pm 1.85}$	$68.66_{\pm 0.30}$	$70.11_{\pm 0.83}$	4.25
TGAT	1.1M	$59.94_{\pm 1.63}$	$19.64_{\pm 0.23}$	$60.92_{\pm 0.57}$	$56.20_{\pm 2.11}$	6.50
TGN	1M	$68.93_{\pm 0.53}$	$37.48_{\pm 0.23}$	$58.60_{\pm 3.70}$	$37.90_{\pm 2.00}$	5.25
Our	600k	$66.76_{\pm 0.74}$	$40.52_{\pm 0.41}$	$74.82_{\pm 0.42}$	$67.10_{\pm 6.72}$	**3.25**

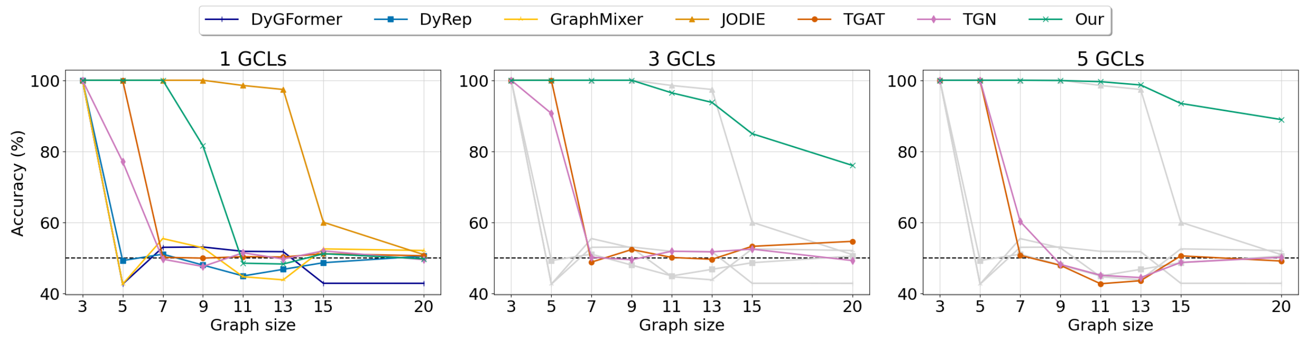


Figure 6. Mean accuracy on the T-PathGraph task on the experiment of Section 4.1.1, with distinction between the performance at different number of GCLs (whenever possible). With 3 and 5 GCLs we report in grey the results of DyGFormer, DyRep, GraphMixer, and JODIE, which are designed for 1-hop aggregation only. The plots show that not only CTAN can better retain information at low number of GCLs, but also that increasing the number of GCL enables solving the T-PathGraph task on longer graphs, where the task is harder because information needs to be propagated further away. The number of GCL allows CTAN to *scale* up the range of information propagation.