Continuous Temporal Graph Networks for Event-Based Graph Data

Anonymous ACL submission

Abstract

There has been an increasing interest in modeling continuous-time dynamics of temporal graph data. Previous methods encode time-evolving relational information into a low-dimensional representation by specifying discrete layers of neural networks, while real-world dynamic graphs often vary continuously over time. Hence, we propose Continuous Temporal Graph Networks (CTGNs) to capture continuous dynamics of temporal graph data. We use both the link starting timestamps and link duration as evolving information to model continuous dynamics of nodes. The key idea is to use neural ordinary differential equations (ODE) to characterize the continuous dynamics of node representations over dynamic graphs. We parameterize ordinary differential equations using a novel graph neural network. The existing dynamic graph networks can be considered as a specific discretization of CTGNs. Experiment results on both transductive and inductive tasks demonstrate the effectiveness of our proposed approach over competitive baselines.

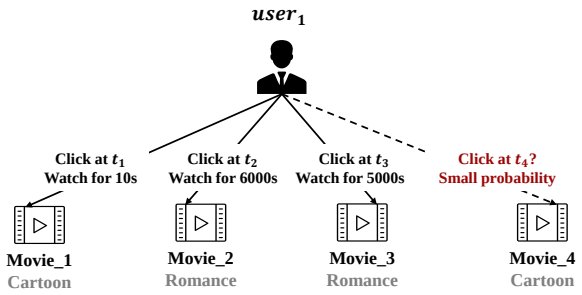


Figure 1: The importance of link duration. Consider the behavior of a user watching movies. There are two types of nodes in the graph: $user$ nodes and $item$ nodes. Given the user's historical behavior, the predicted target is ($user_1$, *don't_click*, *Movie_4*). If we ignore the link duration information, $user_1$ seems interested in cartoon movies because he clicked on it at timestamp t_1. But $user_1$ only watched the *Movie_ 1* for 10s. The link duration indicated that although the user clicked, he was not interested.

1 Introduction

Graph neural networks (GNNs) have attracted growing interest in the past few years due to their universal applicability in various fields, *e.g.*, social networks (Fan et al., 2019) and natural language processing (Liu et al., 2021a). Graph neural networks (GNNs) learn a lower-dimensional representation for a node in a vector space by aggregating the information from its neighbors using discrete hidden layers. Then the embedding can be used for downstream tasks such as node classification (Atwood and Towsley, 2015), link prediction (Zhang and Chen, 2018; Li et al., 2020), and knowledge completion (Liu et al., 2021b).

Most graph neural networks only accept static graphs as input, although real-life graphs of interactions, such as user-item interactions, often change over time. Learning the node representation on dynamic graphs is a very challenging task. Dynamic graph methods can be divided into discrete-time dynamic graph (DTDG) models and continuous-time dynamic graph (CTDG) models. More recently, an increasing interest in CTDG-based graph representation learning algorithms can be observed (Xu et al., 2020; Trivedi et al., 2018; Kumar et al., 2019; Rossi et al., 2020; Wang et al., 2020b; Ding et al., 2021).

Although the above continuous-time dynamic methods have achieved impressive results, they still have limitations. The majority of research (Rossi et al., 2020; Wang et al., 2020b; Xu et al., 2020; Trivedi et al., 2018; Kumar et al., 2019) pays attention to the contact sequence dynamic graphs, in which the links are permanent, and no link duration is provided (*e.g.*, email networks and citation networks). However, most real-life networks are event-based dynamic graphs in which the interactions between source nodes and destination nodes are not permanent (*e.g.*, employment networks and proximity networks). The event-based dynamic

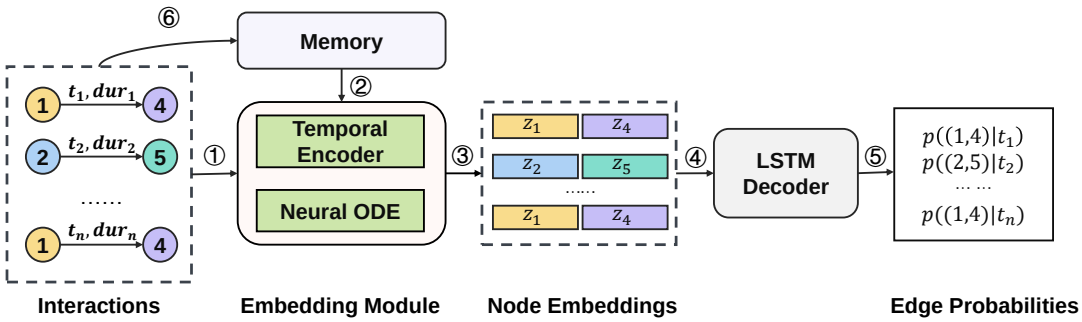


Figure 2: Overview of our Continuous Temporal Graph network.

graph includes the time at which the link appeared and the duration of the link. Link duration reflects the degree of association between the two nodes, *e.g.*, user i browses item j for 2 seconds and k for 20 seconds. It means that the user's interest in the two items j, k is different. Ignoring the link duration information can reduce the link prediction ability and even result in questionable inference. Thus, it is crucial to consider the influence of link duration on node relationship prediction (Zhang and Chen, 2018; Li et al., 2020) and knowledge completion (Liu et al., 2021b).

The existing GNN-based methods (Weinan, 2017; Oono and Suzuki, 2019) that learn the node representation over dynamic graphs can be considered discrete dynamical systems. Chen *et al.* (2018) demonstrate that the continuous dynamical systems are more efficient for modeling continuous-time dynamic data. The discrete networks can roughly be regarded as continuous networks by stacking enough layers. However, Onno and Suzuki (2019) point out that graph neural networks (GNNs) exponentially lose expressive power for downstream tasks, which will lead to over-smoothing problems as we add more hidden layers. Therefore, designing effective continuous Graph Neural Networks to model continuous-time dynamics of node representation on dynamic graphs is critical. To this end, many continuous graph neural networks (Chen et al., 2018; Xhonneux et al., 2019) have been proposed recently. Although those mentioned above continuous dynamic neural networks are more efficient to model the graph data, few approaches have been proposed for dealing with dynamic graphs using continuous-time dynamic neural networks.

This paper proposes a general framework of continuous temporal graph networks (CTGNs) to model continuous-time representations for dynamic graph-structured data. We combine Ordinary Differential Equation Systems (ODEs) and graphs methods. Instead of specifying discrete hidden layers, we integrate neural layers over continuous time. Figure 2 illustrates the workflow of the proposed CTGN method. There is an interaction between two nodes. First, a novel temporal graph network (TGN) is applied as the encoder to learn the latent states using the updated memory. Then, the neural ODE module is used to model the node's continuous-time representation. Considering that the link duration reflects the degree of association between the two nodes, we use the link duration as the integration variable to control the weights of different interactions. After that, we use the LSTM (Shi et al., 2015) as the decoder to compute the probability of interaction between the two given nodes. Finally, the memory is updated as the input of the encoder. Memory is a compressed representation of the historical behavior of all nodes defined in Section 3.1. Experimental results on five real-world datasets of link prediction demonstrate the effectiveness of the proposed method over the state-of-art baselines. The main contributions of this paper are:

- We present a novel Continuous Temporal Graph Network (CTGN) inspired by the neural ODE method.

- CTGNs pay attention to the event-based dynamic graph. CTGNs update the node's representation with both the valid discrete timestamps when the link appears and the link duration between two linked nodes as evolving information.

- We show that our model can outperform existing state-of-the-art methods on both transductive and inductive tasks.

2 Background

2.1 Dynamic Graph Methods

The existing dynamic graph representation learning methods can be divided into two categories, discrete-time dynamic graphs and continuous-time dynamic graphs.

Discrete-time dynamic graphs (DTDGs) are a sequence of snapshots at different time intervals.

$$DG = \{G^1, G^2, ..., G^T\}, \tag{1}$$

where T is the number of snapshots. Current dynamic graph methods (Wang et al., 2020a; Trivedi et al., 2017; Xiong et al., 2019) have been mostly designed for discrete-time dynamic graphs (DTDGs).

Continuous-time dynamic graphs (CTDGs) can be viewed as a set of observations/events (Kazemi et al., 2019), and the network evolution information is retained. There are only a few works on CTDG. But recently, more attention has been paid to continuous-time graphs. All three representations of CTDG are described in more detail below.

1. **The contact sequence dynamic graph** is the simplest representation form of CTDG.

$$CS = (u_i, v_i, t_i), \tag{2}$$

where u is the source node, v is the destination node, and t is the timestamp when the link appears. In the contact sequence dynamic graph, the link is permanent (*e.g.*, citation networks) or instantaneous (*e.g.*, email networks). Therefore, this graph has no link duration.

There has been a lot of research on contact sequence dynamic graphs. Trivedi *et al.* (2018) learn the representation of node i by aggregating the node destination's neighborhood information and updating the embedding for the node using a recurrent architecture after an interaction involving node i. Kumar *et al.* (2019) employ two recurrent neural networks to update the embedding of a user and an item at every interaction. TGAT (Xu et al., 2020) proposes a novel functional time encoding method and uses self-attention to inductive representation learning on temporal graphs. Wang *et al.* (2020b) propose the asynchronous propagation attention network (APAN) for real-time temporal graph embedding.

2. **The event-based dynamic graph** consists of the node pairs (u, v), the edge appears timestamp t and the link duration Δt . Link duration indicates how long the edge lasts until it disappears.

$$EB = (u_i, v_i, t_i, \Delta t_i). \tag{3}$$

Rossi et al. (2020) proposes a generic inductive framework operating on contact sequence dynamic graphs by adding a memory module on TGAT (Xu et al., 2020). TGN can also operate on the event-based dynamic graph by simply replacing the timestamp t with link duration Δt in the memory module.

3. **The streams graph** can be viewed as a particular case of the event-based dynamic graph. The streams graph includes the edge label δ, which indicates edge removal or edge addition.

$$GS = (u_i, v_i, t_i, \delta_i), \delta_i \in [-1, 1]. \tag{4}$$

TGN (Rossi et al., 2020) converts the streams graph into an event-based graph for processing. According to the edge label, the event can be reorganized as (u_i, v_i, t', t), which was created at time t' and deleted at time t, then two messages can be computed for the source and target nodes.

The existing CTDG methods model discrete dynamics representations of continuous-time graph data with multiple discrete propagation layers. Our proposed method focuses on the event-based temporal graph and updates the node's representation with both the timestamps and the link duration between the two nodes. CTGN also supports contact sequence dynamic graph. The model details will be slightly different from event-based dynamic graph. We will clarify this point in Chapter 3.

2.2 Continuous-time Dynamical Systems

Continuous-time dynamical systems mean that the system's behavior changes with time development in the continuous-time domain. There have been related works that view data as a continuous object in artificial intelligence, *e.g.*, pictures (Chen et al., 2018) and static graphs (Xhonneux et al., 2019; Poli et al., 2019). The continuous-time dynamic graph (CTDG) we introduced in Section 2.1 is also a continuous-time dynamical system in

which nodes' state changes over time. Therefore, it is necessary to model the continuous dynamical system of CTDG data. To the best of our knowledge, our CTGN is the first approach that learn continuous-time dynamics on CTDG.

2.3 Neural Ordinary Differential Equations and Continuous Graph Neural Networks

Considering a residual network:

$$\mathbf{h}_{t+1} = \mathbf{h}_t + f(\mathbf{h}_t, \theta_t) \tag{5}$$

A theoretical method to improve the performance of discrete networks is to stack more neural layers and take smaller steps (Chen et al., 2018). However, this scheme is not feasible because of the limited computer resources and over-fitting problems. Oono and Suzuki (2019) point out that Graph Neural Networks (GNNs) exponentially lose expressive power for downstream tasks when adding more hidden layers because of over-smoothness problems.

Inspired by residual network and ordinary difference, neural ordinary difference is proposed to solve this problem. Neural ODE models continuous-time dynamical systems by parameterizing the hidden state's derivative using a neural network.

$$\frac{d_{\mathbf{z}}}{d_t} = f(\mathbf{z}, t), \mathbf{z}(0) = \mathbf{x}, \tag{6}$$

NeuralODE can be regarded as a discrete network with an infinitesimal learning rate and infinite layers. Weinanl (2017) proposes the idea of using continuous dynamical systems to model hidden layers. Chen *et al.* (2018) introduce neural ODE, a continuous-depth model by parameterization the derivative of the hidden state using a neural network. Neural ODE only focuses on unstructured data. Xhonneux *et al.* (2019) apply continuous dynamical methods to static graph-structured data. They propose Continuous Graph Neural Networks (CGNNs), which solve the over-smoothing caused by stacking more layers and improve the performance of GNNs. Zang and Wang (2019) learn continuous-time dynamics on complex networks. However, continuous graph neural networks (CGNN) can only deal with static data.

3 The Proposed Method: CTGN

In this section, we introduce our proposed approach. The key idea of the CTGN is to build continuous-time hidden layers which can learn continuous informative node representations over event-based dynamic graphs. To characterize the continuous dynamics of node representation, we use ordinary differential equations (ODEs) parameterized by a neural network, which is a continuous function of time. We study both transductive and inductive settings. In the transductive task, we predict future links of the nodes observed during the training phase. In the inductive tasks, we predict future links of the nodes never seen before. We first employ a temporal graph attention layer (Xu et al., 2020) to project each node into a latent space based on its features and neighbors. And then, an ODE module is designed to define the continuous dynamics on the node's latent representation $\mathbf{h}_i(t)$.

3.1 Temporal Graph Network

Memory Passing. Memory $\mathbf{s}_i(t)$ is used to record the historical information of each node i the model has seen so far. It is a compressed representation of the historical behavior of all nodes. Memory $\mathbf{s}_i(t)$ is updated when there is an interaction involving node i. At the end of each batch, we firstly compute memory $\mathbf{s}_i(t)$ using the last time message $\mathbf{m}_i(t^-)$ and memory $\mathbf{s}_i(t^-)$:

$$\mathbf{s}_i(t) = mem(\mathbf{m}_i(t), \mathbf{s}_i(t^-)) \,. \tag{7}$$

Here, $mem(\cdot)$ is a learnable memory update function. In all experiments, we choose the memory function as GRU. $\mathbf{s}_i(0)$ is initialized as a zero vector. At the end of each batch, the message $\mathbf{m}_i(t)$ for the node can be updated to compute i's memory:

$$\begin{aligned} \mathbf{m}_i(t) &= msg_s(\mathbf{s}_i(t^-)||\mathbf{s}_j(t^-)||\Delta t||\mathbf{e}_{ij}(t)) \,, \\ \mathbf{m}_j(t) &= msg_s(\mathbf{s}_j(t^-)||\mathbf{s}_i(t^-)||\Delta t||\mathbf{e}_{ij}(t)) \,. \end{aligned} \tag{8}$$

Here $||$ is the concatenation operator, Δt is the link duration between node i and j, . In the contact sequence dynamic graph, the link duration property is not available. We use $(t - t^-)$ as Δt. There may be multiple events $\mathbf{e}_{i1}(t_1), \ldots, \mathbf{e}_{iN}(t_N)$ involving the same node i in the same batch. In the experiment, we only use the latest interaction $\mathbf{e}_{iN}(t_N)$ to compute i's message. $msg(\cdot)$ is a learnable function, and we use an RNN network in our experiment:

Multi-head Attention. Given an observed event $p = (i, j, t, \Delta t)$, we can compute the node latent representation respectively for i and j using:

$$\mathbf{H}^{(l)}(t) = Attn^{(l)}(\mathbf{Q}^{(l)}(t), \mathbf{K}^{(l)}(t), \mathbf{V}^{(l)}(t)) \,, \tag{9}$$

	Event-based dynamic graph			Contact sequence dynamic graph	
	NetFlix	Mooc	Lastfm	Wikipedia	Reddit
Nodes	18672	13374	7353	9227	10984
Edges	163417	131660	73358	157474	672447
Chronological Split	70%-15%-15%	70%-15%-15%	70%-15%-15%	70%-15%-15%	70%-15%-15%
Unseen nodes	10%	10%	10%	10%	10%
Timespan	2 years	2 years	2 years	30 days	30 days

Table 1: Statistics of the datasets used in our experiments.

$$Attn(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = softmax(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}})\mathbf{V}\,, \quad (10)$$

where $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$ denote the 'querys', 'keys', 'values', respectively. $\mathbf{H}^{(l)} = [\mathbf{h}_1^{(l)}, ..., \mathbf{h}_i^{(l)}]$ are the embedding of the graph nodes of l-th layers. The multi-head attention layer compute the node i's representation by aggregating it's N-hop neighbors.

$$\mathbf{Q}^{(l)}(t) = (\mathbf{H}^{(l-1)}(t) \,||\, \phi(0))\mathbf{W}_Q\,, \quad (11)$$

$$\mathbf{K}^{(l)}(t) = \mathbf{C}^{(l)}(t)\mathbf{W}_K\,, \quad (12)$$

$$\mathbf{V}^{(l)}(t) = \mathbf{C}^{(l)}(t)\mathbf{W}_V\,, \quad (13)$$

$$\begin{aligned}\mathbf{C}^{(l)}(t) = [\mathbf{H}_1^{(l-1)}(t) \,||\, \mathbf{E}_1(t_1) \,||\, \phi(t-t_1),\\ \ldots, \mathbf{H}_N^{(l-1)}(t) \,||\, \mathbf{E}_N(t_N) \,||\, \phi(t-t_N)]\,.\end{aligned} \quad (14)$$

Here $\phi(\cdot)$ represents a generic time encoder (Xu et al., 2020). $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{d_k \times d_k}$ are the projection matrices used to generate attention embedding. We define keys and values as the neighbor information. $\mathbf{h}_i^{(0)}(t) = \mathbf{s}_i(t) + \mathbf{v}_i$, $\mathbf{s}_i(t)$ is node i's memory which saves the history information for the node. $\mathbf{E}_n(t) = [e_{1n}(t), ..., e_{in}(t)]$, $e_{in}(t)$ is edge features between node i and it's n-hop neighbor at time t. Temporal graph network is a discrete method that can be thought of as a discretization of the continuous dynamical systems.

3.2 Model Continuous Dynamics of Node Representation

In order to characterize the continuous dynamics of node representations, instead of only specifying a discrete sequence of hidden layers, we parameterize the hidden layers using ordinary differential equations (ODEs), a continuous function of time.

$$\frac{d_{\mathbf{z}}}{d_t} = f(\mathbf{z}, t), \mathbf{z}(0) = \mathbf{x}. \quad (15)$$

Here, $\mathbf{x}$ is an initial vector, f is a learnable function, t is a time interval and $\mathbf{z}$ is a vector.

$$\mathbf{z}(t) = \mathbf{z}(0) + \int_0^t (f(t, z))d\tau. \quad (16)$$

We can compute the node's continuous-time dynamics representation by Equation 16 at arbitrary time t > 0.

Previous work (Zang and Wang, 2019; Poli et al., 2019) model continuous-time dynamics for data by setting integration variable $[0, t]$ as a hyper-parameter. Considering the influence of link duration on the interaction between two nodes, we choose the link duration as the integration variable, in our experiment $t = dur$.

Link duration shows how long it was (in seconds) until that user terminated browsing. Link duration can reflect the user's interest in different items. Take link duration as an integer variable that can control the weights of different interactions.

We parameterize the derivative of the hidden state using a neural network that takes the latent state, computed by the temporal graph network mentioned in Section 3.1 as input.

$$\mathbf{z}_i(t) = ODESolver(f(t, z), \mathbf{h}_i(t), \Delta t_i). \quad (17)$$

Here, $\mathbf{h}_i(t)$ is a discrete latent state computed by temporal graph networks, Δt_i is the link duration between source node i and destination j. $f(t, z)$ is ODE function, we choose $f(t, z)$ as MLP. A black-box ODE solver computes the final node continuous dynamics embedding $\mathbf{z}_i(t)$. We utilize the *torchdiffeq.odeint_adjoint* PyTorch package to solve reverse-time ODE and backpropagate.

3.3 Time Smoothness

The time-encoding method (Xu et al., 2020) used in this paper is an effective method to map timestamp t from the time domain to d-dim vector space.

	NetFlix		Mooc		Lastfm	
	Transductive	Inductive	Transductive	Inductive	Transductive	Inductive
GAT*	96.45 ± 0.2	92.09 ± 0.6	83.33 ± 10	77.39 ± 10	76.77 ± 0.5	62.81 ± 0.6
GraphSAGE*	95.14 ± 0.6	89.84 ± 1.7	82.01 ± 2.4	78.36 ± 2.2	77.41 ± 0.6	62.57 ± 0.3
CGNN*	91.82 ± 0.2	†	96.88 ± 0.2	†	74.93 ± 10	†
NDCN*	90.70 ± 0.9	†	96.07 ± 0.1	†	82.09 ± 1.4	†
DyRep	99.07 ± 0.1	97.36 ± 0.1	83.52 ± 6.5	68.96 ± 4.0	82.96 ± 0.3	68.06 ± 0.3
Jodie	99.20 ± 0.1	97.43 ± 0.1	93.12 ± 0.6	80.85 ± 1.2	84.41 ± 0.3	68.14 ± 0.5
TGAT	96.56 ± 0.2	93.04 ± 0.2	73.69 ± 1.3	68.76 ± 1.2	78.80 ± 0.8	64.19 ± 0.7
TGN	99.05 ± 0.2	97.38 ± 0.4	97.76 ± 0.4	93.86 ± 0.9	87.05 ± 0.1	72.89 ± 0.1
APAN	98.23 ±1.7	†	93.64 ± 1.3	†	82.65 ± 0.1	†
CTGN	**99.27 ± 0.1**	**97.84 ± 0.2**	**97.97 ± 0.4**	**94.89 ± 0.4**	**87.20 ± 0.1**	**74.05 ± 0.1**

Table 2: Experiments on event-based datasets. Average Precision (%) for future edge prediction task in transductive and inductive settings. **First** best performing method. *Static graph method. †Does not support inductive.

	node classification		link prediction-tranductive		link prediction-inductive	
	Wikipedia	Reddit	Wikipedia	Reddit	Wikipedia	Reddit
GAE*	74.85 ± 0.6	58.39 ± 0.5	91.44 ± 0.1	93.23 ± 0.3	†	†
VGAE*	73.67 ± 0.8	57.98 ± 0.6	91.34 ± 0.3	92.92 ± 0.2	†	†
GAT*	82.34 ± 0.8	64.52 ± 0.5	94.73 ± 0.2	97.33 ± 0.2	91.27 ± 0.4	95.37 ± 0.3
GraphSAGE*	82.42 ± 0.7	61.24 ± 0:6	93.56 ± 0.3	97.65 ± 0.2	91.09 ± 0.3	96.27 ± 0.2
DyRep	84.59 ± 2.2	62.91 ± 2.4	94.59 ± 0.2	97.98 ± 0.1	92.05 ± 0.3	95.68 ± 0.2
Jodie	84.84 ± 1.2	61.83 ± 2.7	94.62 ± 0.5	97.11 ± 0.3	93.11 ± 0.4	94.36 ± 1.1
TGAT	83.69 ± 0.7	65.56 ± 0.7	95.34 ± 0.1	98.12 ± 0.2	93.99 ± 0.3	96.62 ± 0.3
TGN	87.81 ± 0.3	67.06 ± 0.9	98.46 ± 0.1	98.70 ± 0.1	97.81 ± 0.1	97.55 ± 0.1
APAN	**89.86 ± 0.3**	65.34 ± 0.4	98.12 ± 0.2	**99.22 ± 0.2**	†	†
CTGN	88.01 ± 1.5	**68.38 ± 3.4**	**98.64 ± 0.1**	98.28 ± 0.2	**98.01 ± 0.1**	**98.05 ± 0.2**

Table 3: Experiments on contact sequence datasets. ROC AUC (%) for the dynamic node classification task, Average Precision (%) for link prediction task. *Static method, †Does not support inductive.

However, the learning process of each timestamp is independent of other timestamps. Independent learning of hyperplanes of adjacent time intervals may cause adjacent times to be farther apart in embedded space. Actually, adjacent states in the graph should be more similar. To avoid the problem mentioned above, we constrained the variation between hyperplanes at adjacent timestamps by minimizing the euclidean distance:

$$L_{smooth}(W) = \sum_{t=1}^{T-1} ||w_{t+1} - w_t||_2 \,. \qquad (18)$$

3.4 Model Learning

We use the link prediction loss function for training CTGN:

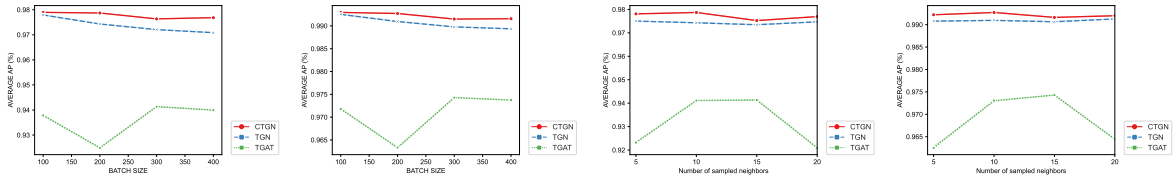
$$loss = \alpha L_{smooth}(W) + L_{task} \,, \qquad (19)$$

where α is a tradeoff parameter, l_{task} is a loss function defined as the cross-entropy of the prediction and the ground truth. Our experiment found a parameter α of 0.002 for contact sequence dynamic graphs and 0.7 for event-based dynamic graphs.

4 Experiment and Analysis

In this section, we first introduce datasets, baselines and parameter settings. Then we compare our proposed method with other strong baselines and competing approaches for both the inductive and transductive tasks for two benchmarks contact sequence dynamic graph datasets and three event-based dynamic graph datasets.

We study both transductive and inductive tasks. For event-based dynamic graphs, we learn link prediction tasks. For contact-sequence dynamic graphs, we learn dynamic node classification and link prediction tasks.



(a) (b) (c) (d)

Figure 3: Ablation studies on the Netflix dataset for both the transductive and inductive setting of the link prediction task. 3(a) Sensitivity study result of batch size in inductive setting. 3(b) Sensitivity study result of batch size in transductive setting. 3(c) The relationship between number of sampled neighbors and the model performance in inductive setting. 3(d) The relationship between number of sampled neighbors and the model performance in transductive setting.

4.1 Datasets

We use five real-world datasets in our experiments, three event-based dynamic graphs: Netflix [1], Mooc (Feng et al., 2019) and Lastfm (Cantador et al., 2011), two contact sequence dynamic graphs: Wikipedia (Kumar et al., 2019), Reddit (Kumar et al., 2019).

The statistics of the datasets used in our experiments are described in detail in Table 1.

4.2 Baseline

We compare our model with four CTDG methods: Jodie (Kumar et al., 2019), Dyrep (Trivedi et al., 2018), TGAT (Xu et al., 2020), TGN (Rossi et al., 2020), APAN (Wang et al., 2020b). And we also include four DTDG methods: GAE (Kipf and Welling, 2016), VGAE (Kipf and Welling, 2016), GAT (Veličković et al., 2018), GraphSAGE (Hamilton et al., 2017) as well as two state-of-the-art static graph neural ODE methods: CGNN (Xhonneux et al., 2019), NDCN (Zang and Wang, 2019).

4.3 Parameter Setup

We set the batch size to 200 for training and patience to 5 for early stopping in all experiments. The node embedding dimension is 172. During training, we used 0.0001 as the learning rate for contact sequence dynamic graph datasets (Wikipedia and Reddit) and 0.00009 for event-based dynamic graph datasets (Netflix, Mooc, Lastfm). The weight of time smoothness loss α is set to 0.002 on Wikipedia , Reddit and 0.7 on Netflix, Mooc, Lastfm. We choose the LSTM layer as the decoder for link prediction task and MLP for node classification task. We report mean and standard deviation across 10 runs.

[1] https://vodclickstream.com/

4.4 Result

To demonstrate the effectiveness of our proposed method, we compare CTGN with competitive baselines on five real-world event-based graph datasets. Table 2 shows the results on link prediction tasks in both transductive and inductive settings for three event-based datasets. It is evident that our approach has achieved better results than the discrete dynamics graph neural networks on almost all datasets, especially in the inductive setting.

Table 3 shows the dynamic node classification and link prediction results on two contact sequence-datasets. CTGN has a solid ability to embed dynamic graphs. The conclusion can be obtained from the Table 2 and Table 3.

Figure 3 shows ablation studies on the Netflix dataset for both the transductive and inductive setting of the link prediction task. As we can see from Figure 3(a) and 3(b), our model is not sensitive to batch size. When the training batch size is 100, CTGN has the same average precision as TGN. With the continuous increase of batch size, the performance of CTGN is more stable.

5 Conclusion

This paper introduces CTGN, a continuous temporal graph neural network for learning representation for event-based dynamic graphs. We build the connection between temporal graph networks and continuous dynamical systems inspired by neural ODE. Our framework allows the user to trade off speed for precision by selecting different learning rates and the weight of time smoothness loss parameters during training. We demonstrate on the link prediction task against competitive baselines that our model can outperform many existing state-of-the-art methods.

References

James Atwood and Don Towsley. 2015. Search-convolutional neural networks. *CoRR*, abs/1511.02136.

Iván Cantador, Peter Brusilovsky, and Tsvi Kuflik. 2011. 2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011). In *Proceedings of the 5th ACM conference on Recommender systems*, RecSys 2011, New York, NY, USA. ACM.

Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. 2018. Neural ordinary differential equations. *CoRR*, abs/1806.07366.

Zifeng Ding, Zhen Han, Yunpu Ma, and Volker Tresp. 2021. Temporal knowledge graph forecasting with neural ODE. *CoRR*, abs/2101.05151.

Wenqi Fan, Yao Ma, Qing Li, Yuan He, Yihong Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. *CoRR*, abs/1902.07243.

W. Feng, J. Tang, and T. X. Liu. 2019. Understanding dropouts in moocs. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:517–524.

William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *CoRR*, abs/1706.02216.

S. M. Kazemi, R. Goel, K. Jain, I. Kobyzev, A. Sethi, P. Forsyth, and P. Poupart. 2019. Representation learning for dynamic graphs: A survey.

Thomas N. Kipf and Max Welling. 2016. Variational graph auto-encoders.

Srijan Kumar, Xikun Zhang, and Jure Leskovec. 2019. Predicting dynamic embedding trajectory in temporal interaction networks. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*.

X Li, Y. Shang, Y. Cao, Y. Li, and Y. Liu. 2020. Type-aware anchor link prediction across heterogeneous networks based on graph attention network. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(1):147–155.

Shangqing Liu, Yu Chen, Xiaofei Xie, Jing Kai Siow, and Yang Liu. 2021a. Retrieval-augmented generation for code summarization via hybrid GNN. In *International Conference on Learning Representations*.

Xiyang Liu, Huobin Tan, Qinghong Chen, and Guangyan Lin. 2021b. Ragat: Relation aware graph attention network for knowledge graph completion. *IEEE Access*, 9:20840–20849.

Kenta Oono and Taiji Suzuki. 2019. Graph neural networks exponentially lose expressive power for node classification.

Michael Poli, Stefano Massaroli, Junyoung Park, Atsushi Yamashita, Hajime Asama, and Jinkyoo Park. 2019. Graph neural ordinary differential equations. *CoRR*, abs/1911.07532.

E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, F. Monti, and M. Bronstein. 2020. Temporal graph networks for deep learning on dynamic graphs.

Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. 2015. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *CoRR*, abs/1506.04214.

R. Trivedi, M. Farajtabar, Y. Wang, H. Dai, and S. Le. 2017. Know-evolve: Deep reasoning in temporal knowledge graphs.

Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. 2018. Representation learning over dynamic graphs. *CoRR*, abs/1803.04051.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks.

J. Wang, Y. Jin, G. Song, and X. Ma. 2020a. Epne: Evolutionary pattern preserving network embedding.

X. Wang, D. Lyu, M. Li, Y. Xia, Q. Yang, X. Wang, X. Wang, P. Cui, Y. Yang, and B. Sun. 2020b. Apan: Asynchronous propagation attention network for real-time temporal graph embedding.

Weinan. 2017. A proposal on machine learning via dynamical systems. *Communications in Mathematics amp; Statistics*, 5(1):1–11.

Lpac Xhonneux, M. Qu, and J Tang. 2019. Continuous graph neural networks.

Y. Xiong, Y. Zhang, H. Fu, W. Wang, and P. S. Yu. 2019. *DynGraphGAN: Dynamic Graph Embedding via Generative Adversarial Networks*. Grundlagen des MA-Geschäftes.

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

Chengxi Zang and Fei Wang. 2019. Neural dynamics on complex networks. *CoRR*, abs/1908.06491.

Muhan Zhang and Yixin Chen. 2018. Link prediction based on graph neural networks. *CoRR*, abs/1802.09691.