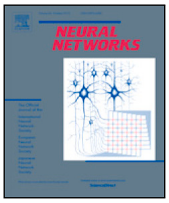
Neural Networks 172 (2024) 106151

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet



Full Length Article

ConTIG: Continuous representation learning on temporal interaction graphs

Zihui Wang [a,b,1], Peizhen Yang [a,b,1], Xiaoliang Fan [a,b,*], Xu Yan [a,b], Zonghan Wu [c], Shirui Pan [d], Longbiao Chen [a,b], Yu Zang [a,b], Cheng Wang [a,b], Rongshan Yu [a,b]

[a] *Fujian Key Laboratory of Sensing and Computing for Smart Cities, School of Informatics, Xiamen University, Xiamen, 361000, Fujian, China*
[b] *Key Laboratory of Multimedia Trusted Perception and Efficient Computing, Ministry of Education of China, Xiamen University, 361005, PR China*
[c] *University of Technology Sydney, Ultimo, 2007, NSW, Australia*
[d] *School of ICT, Griffith University, Gold Coast, 4222, Queensland, Australia*

ARTICLE INFO

Keywords:
Graph representation
Graph embedding
Temporal interaction graph
Mining and learning in graphs
Graph neural networks

ABSTRACT

Representation learning on temporal interaction graphs (TIG) aims to model complex networks with the dynamic evolution of interactions on a wide range of web and social graph applications. However, most existing works on TIG either (a) rely on discretely updated node embeddings merely when an interaction occurs that fail to capture the continuous evolution of embedding trajectories of nodes, or (b) overlook the rich temporal patterns hidden in the ever-changing graph data that presumably lead to sub-optimal models. In this paper, we propose a two-module framework named ConTIG, a novel representation learning method on TIG that captures the continuous dynamic evolution of node embedding trajectories. With two essential modules, our model exploits three-fold factors in dynamic networks including *latest interaction*, *neighbor features*, and *inherent characteristics*. In the first update module, we employ a continuous inference block to learn the nodes' state trajectories from time-adjacent interaction patterns using ordinary differential equations. In the second transform module, we introduce a self-attention mechanism to predict future node embeddings by aggregating historical temporal interaction information. Experiment results demonstrate the superiority of ConTIG on temporal link prediction, temporal node recommendation, and dynamic node classification tasks of four datasets compared with a range of state-of-the-art baselines, especially for long-interval interaction prediction.

1. Introduction

Graph representation learning has attracted a surge of research attention owing to the widespread existence of graph-structured data in the real world such as social networks, Recommendation (Wen et al., 2023; Zhang et al., 2023) and other user-item interaction systems (Hamilton et al., 2017b). Learning graph embedding is a powerful approach of graph representation learning, which maps the characteristics of nodes to a low-dimensional vector space so that the proximities of nodes in topological space can be well reserved (Cui et al., 2018). It has shown great success in graph representation learning from shallow graph embedding methods (Su et al., 2022) to deep graph neural networks (GNNs) (Wang et al., 2022; Wu et al., 2020). Recently, representation learning on the dynamic graph has attracted many research attention (Gao et al., 2022; Kazemi et al., 2020; Li et al., 2022), and they mainly model temporal graphs either as a sequence of snapshots (Gong et al., 2020; Pareja et al., 2020) or as real events with timestamps (An et al., 2022). Specially, the events only occur between users and items to get a temporal interaction graph (TIG) while events would not happen among users or items themselves (Zhang et al., 2021), which is also concluded as a bipartite graph, e.g., Fig. 1 shows the interactions that users edit posts on a social platform with timestamps which can be regarded as a sequence of interactions.

For existing works learning temporal interaction embedding from the sequence of interactions, a number of studies discretely update the embeddings of the interactive nodes once an interaction occurs (Kumar et al., 2019; Zhang et al., 2021). Other works learn the interactions by

* Corresponding author.
E-mail addresses: wangziwei@stu.xmu.edu.cn (Z. Wang), yangpz@stu.xmu.edu.cn (P. Yang), fanxiaoliang@xmu.edu.cn (X. Fan), yanxu97@stu.xmu.edu.cn (X. Yan), zonghan.wu-3@student.uts.edu.au (Z. Wu), s.pan@griffith.edu.au (S. Pan), longbiaochen@xmu.edu.cn (L. Chen), zangyu7@126.com (Y. Zang), cwang@xmu.edu.cn (C. Wang), rsyu@xmu.edu.cn (R. Yu).
[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.neunet.2024.106151
Received 21 January 2023; Received in revised form 15 December 2023; Accepted 26 January 2024
Available online 29 January 2024

Table 1
Table comparing the three-fold factors of existing temporal interaction graph methods learning temporal interaction embedding and our proposed ConTIG. ConTIG satisfies all the factors.

Name	Latest interaction	Neighbor features	Inherent characteristics
TDGNN (Qu et al., 2020)	✗	✓	✗
TGAT (Xu et al., 2020)	✗	✓	✗
JODIE (Kumar et al., 2019)	✓	✗	✗
TigeCMN (Zhang et al., 2021)	✓	✗	✗
ConTIG (ours)	✓	✓	✓

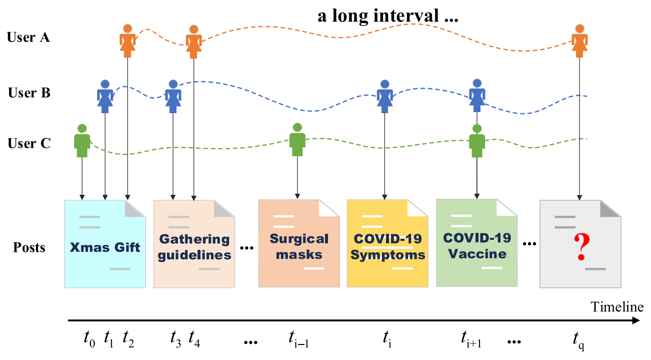


Fig. 1. An illustrated example of the inactive user (i.e., User A) with long-interval interaction in a temporal interaction graph. In the long-interval of User A, her future posts about COVID-19 might be affected by three factors: (1) her latest interaction with "Gathering guidelines"; (2) her neighbors, user B and C; and (3) her inherent characteristics as a healthcare personnel.

aggregating the temporal neighbor features to pass messages between nodes (Qu et al., 2020; Xu et al., 2020), which model discrete dynamic of node representation in multiple propagation layers (Oono & Suzuki, 2020). Although these discrete works have made substantial advances in learning temporal interaction embedding, they fail to capture the continuous dynamic evolution of node embedding trajectories thus losing temporal information for non-ignorable and inactive nodes with long-interval interactions (e.g., User A in Fig. 1). Therefore, we aim to learn nodes embedding trajectories and capture the continuous dynamic of node representations.

Learning dynamic node embedding trajectories is extremely challenging due to the complex non-linear dynamic in temporal interaction graphs. We conclude the challenges in three folds under the scenario of posting in Reddit[2] website. **First**, *latest interaction* information may have a significant impact on the current interaction, e.g., in Fig. 1, the posts made by User A at t_4 would be influential for t_q, since guidelines of holiday gathering under COVID-19 pandemics will be regularly implemented. **Second**, the node state is also affected by their *neighbors' features* over time. For instance, User A is expected to emulate their neighbors by posting "COVID-19 vaccine" at t_q in Fig. 1. **Third**, *inherent characteristics* of nodes is vitally important that would fatally determine the state regardless of aforementioned two factors (i.e., latest interaction, and neighbors' features). For example in Fig. 1, User A is a frontline healthcare personnel and she could inherently pay much attention to COVID-19 training on infection control regardless of the latest interaction and neighbor features. In summary, existing methods hold partial considerations for these three-fold factors. We compare existing temporal interaction graph (TIG) methods as is shown in Table 1. Existing TIG methods lack of comprehensive solution with the consideration of all three-fold factors.

To cover the shortcomings of previous methods, we propose a **Con**tinuous representation learning method on **T**emporal **I**nteraction **G**raphs (**ConTIG**). The proposed ConTIG contains two modules: the update module and transform module. When an interactive message comes, in the update module, inspired by Xhonneux et al. (2020), we define a neural ordinary differential equations (ODE) (Chen et al., 2018) with the three aforementioned factors affecting node states (i.e., latest interaction, neighbors' feature and inherent characteristics), and incorporate them with the continuous-time encoding function to trace dynamic knowledge and learn node state trajectories. Then the node embeddings at the ending time of neural ODE is used to update embeddings of interacting nodes. In the transform module, a self-attention mechanism is introduced to awaken historical interactive information of current interacting nodes and convert them to generate future node representations. The results on temporal link prediction, temporal node recommendation and dynamic node classification tasks show the effectiveness of our proposed model compared with a range of state-of-the-art baselines, especially for long-interval interactions prediction.

The contributions of this work are summarized as follows:

- We introduce ConTIG, a novel continuous representation learning framework that employs an encoder–decoder architecture to constantly capture the dynamic evolution of node embedding trajectories for temporal interaction prediction.
- We design the update module which assembles three-fold factors (i.e., latest interaction, neighbor features, and inherent characteristics) into a neural ODE. The module allows us to estimate the probable updating direction of embedding trajectories for all nodes, including inactive ones.
- We introduce the transform module combining the aforementioned update module with a self-attention mechanism. These two modules guarantee that our framework is effective on long-interval interactions prediction.
- We evaluate our ConTIG on three representation learning tasks with four real-world datasets. Extensive experiments demonstrate that ConTIG compares favorably against state-of-the-art methods

The rest of this paper is organized as follows. In Section 2, we discuss some related work. Section 3 and 4 describes the notations and our proposed model in detail. In Section 5, we conduct experiments on several datasets and compare them with state-of-the-art methods. In Section 6, the conclusion and our future work are presented.

2. Related work

In this section, we introduce the embedding methods for both the static graph and temporal graph, as well as the neural ordinary differential equation (ODE).

2.1. Static graph embedding

Early works for representation learning are mainly shallow models including graph factorization approaches (Ahmed et al., 2013) and skip-gram models (Grover & Leskovec, 2016; Perozzi et al., 2014; Tang et al., 2015), which learn node representations by random walk objectives. With the success of deep learning, GNNs (Velickovic et al., 2018; Wang et al., 2022; Wu et al., 2020) have gradually attracted great research interest. They are effective approaches to learn node representations by updating each node according to messages from neighboring nodes in graphs in each layer. GNNs essentially model

[2] https://www.reddit.com/

discrete dynamics of node representations with multiple propagation layers (Oono & Suzuki, 2020). However, all the above mentioned approaches are limited to learning node embeddings on static graph structure information, and the temporal behavior of interaction over time is generally ignored.

2.2. *Temporal graph embedding*

One common way to handle temporal graphs is to decompose it into multiple static graphs snapshots by a regular time interval. Some works embed the graph convolution into the recurrent neural network (RNN) based models or attention mechanism (Vaswani et al., 2017), which learns to exploit the dynamic information in the graph evolution within a period of time (Gong et al., 2020; Liu et al., 2020; Pareja et al., 2020; Sankar et al., 2020; Yang et al., 2021). Some other works are dynamic extensions of ideas applied in the static case inspired by methods such as PageRank (Guo et al., 2021), matrix factorization (Zhu et al., 2016), graph autoencoder (Goyal et al., 2020, 2018; Jiao et al., 2021) and the topic model (Spasov et al., 2020) to capture both the temporal community dynamics and evolution of graph structure. However, learning embedding from the sequence of graph snapshots sampled from the temporal graph by a regular time interval may lose information by only looking at some snapshots of the graph over time.

Therefore, recent works learn temporal graph embedding from the sequence of timed interactions. A number of studies learn the sequence of interactions by discretely updating the node embeddings once an interaction occurs by RNNs (Rossi et al., 2020; Xu et al., 2021), memory networks (Zhang et al., 2021), time point process (Lu et al., 2019; Trivedi et al., 2019), transformer network (Wang, Chang, Li, et al., 2021), generative models (Zhou et al., 2020), contrasting learning (Jiang et al., 2021; Tian, Wu, et al., 2021), meta-learning (Yang et al., 2022), etc. Other works learn the interactions by aggregating the temporal neighbor features to pass messages between nodes (Liu, Tu, et al., 2021; Rossi et al., 2020; Tian, Xiong, & Shi, 2021; Xu et al., 2020) or learning node representation from random walk objects (Nguyen et al., 2018) and temporal motifs (Fu et al., 2020; Liu, Ma, & Li, 2021; Paranjape et al., 2017; Wang, Chang, Liu, et al., 2021). TGAT (Xu et al., 2020) proposes the temporal graph attention layer to aggregate temporal-topological neighborhood features. TGN (Rossi et al., 2020) makes a combination of updating operators and aggregating operators. However, these methods learn the discrete dynamic of node representations, and it is not beneficial to learn the node state trajectories. Moreover, they ignore the influence of the inherent characteristics of the change on the node states, which is not beneficial to capture the complex non-linear dynamic of node representations. Different from the aforementioned works, we assembled the latest interaction, neighbor features and inherent characteristics of the nodes into a neural ODE to learn node state trajectories and update node embeddings. As a result, our method captures the continuous dynamic of node representation.

2.3. *Neural ODE*

Neural ordinary differential equations (ODE) (Chen et al., 2018) is an approach for modeling a continuous dynamics on hidden representation, where the dynamic is characterized through an ODE parameterized by a neural network (i.e. ODE Solver). Recently, there are some works devote to use neural ODE (Chen et al., 2018) combined with GNN to characterize the continuous dynamics of node representations (Xhonneux et al., 2020; Zang & Wang, 2020). However, these methods are not built for temporal graphs, which are continuous-time GNNs learning the dynamic of node representation in static graph scenario to build "deeper" networks. For temporal graphs, (Ding et al., 2021) learns the evolution of the temporal knowledge graph dynamics over time using ODEs as a tool. CG-ODE (Huang et al., 2021) presents a latent ODE generative model that evolves node representations through neighbor aggregation, natural recovery, and inherent physical characteristics. This model aims to capture the coupled dynamics of nodes and edges with a GNN-based ODE in a continuous manner. GNG-ODE (Guo et al., 2022) extends the concept of neural ODEs which capture the dynamics of latent user preference by the original node representation and the neighborhood representations. GDERec (Qin et al., 2023) introduces a novel ODE-based GNN that updates the node representations by incorporating both the initial node embeddings and the normalized adjacency, implicitly capturing the temporal evolution of the user-item interaction graph. Different from them, we stand at the perspective of node state modeling in temporal interaction graph, using a differential equation integrating the three-fold factors (i.e., latest interaction, neighbor features, and inherent characteristics). In addition, for practical stability graph learning algorithms, we have thoughtfully designed our neighbor matrix to get the positive eigenvalues. This allows us to accurately capture continuous dynamic of node embedding trajectories.

3. Problem definition

We summarize some notations and state the problem we want to solve in this section.

Definition 1 (***Temporal interaction graph***)**.** Temporal interaction graph is a graph with temporal and attributed interaction edges between nodes. Temporal interaction graph is defined as a pair $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ represents vertices sets, and $\mathcal{E}$ is a set of the sequences of interactions with time label between two vertices. An interaction e is a tuple of form (u, i, t, f^{ui}), where u and $i \in \mathcal{V}$ represent two vertices that have an interaction at timestamp t, and f^{ui} represents the feature of interaction e.

Definition 2 (***Temporal interaction graph embedding***)**.** Given the temporal interaction graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we aim to learn a mapping function: $\phi : \mathcal{V} \to \mathbb{R}^d$ to map node features into a low-dimensional latent space and generate node embeddings $H \in \mathbb{R}^{|\mathcal{V}| \times d}$ to represent nodes, where $d \ll |\mathcal{V}|$ representing the dimension of node embeddings.

Definition 3 (***Interaction intervals***)**.** In the temporal interaction graph, each interaction is an edge connected by a source node and a target node, so we can use source nodes as the label of interaction to calculate the time interval Δt_e between current interaction and its latest interaction, where the latest interaction is the edge where the source node last appeared. For example, the latest interaction of the current interaction, $e_p^j = (u^j, i, t_p, f^{ui})$, is the interaction $e_q^{j-1} = (u^{j-1}, r_u, \tau_u, f^{ur_u})$ in which the node u at the last time appeared. As a result, the time interval Δt_{e_p} between e_p^j and e_q^{j-1} is $t_p - \tau_u$, where r_u and τ_u represents the latest interactive node and timestamp of u, respectively. While, p and q represent the interaction id, and j represents that how many times u appears. Here, if the node appears for the first time (i.e., $j = 0$), the time interval of the interaction is 0.

4. The proposed model

In this section, we will describe our model in detail. The proposed model – ConTIG consists of two essential modules (i.e., update and transform module). We will briefly introduce them in Section 4.1 and describe each module in detail in the rest of sections.

4.1. *Overview of the ConTIG*

Fig. 2 provides the framework of our proposed ConTIG. The encoder–decoder framework (Fig. 2 left) consists of two modules: the update module and the transform module, which constantly updates node embeddings as the interaction messages emerges. In the update module (Fig. 2 middle), a continuous inference block is used to update

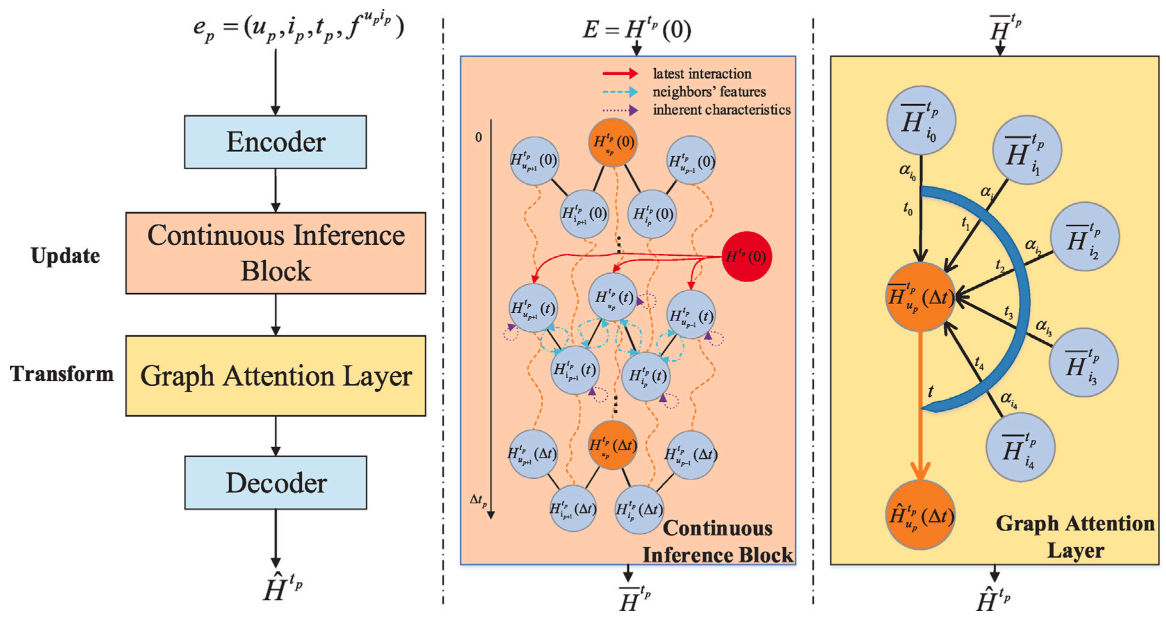


Fig. 2. ConTIG Framework. The encoder–decoder framework (left) consists of a continuous inference block (middle) and a graph attention layer (right). The former contains a neural ODE defined with three-fold factors (i.e., latest interaction, neighbors' feature and inherent characteristics) to update the node embeddings according to the output of encoder. The latter estimates the future embedding of nodes by aggregating observed historical temporal interaction information.

the embeddings of the interacting nodes in a continuous setting. In the transform module (Fig. 2 right), a graph attention layer is applied to estimate the future embedding of nodes by capturing observed historical temporal interaction information. When an interaction message $e_p = (u, i, t_p, f^{ui})$ comes, we first employ a neural encoder to project the latest interaction message $e_q = (u, r_u, \tau_u, f^{ur_u})$ of input interacting node into a latent space, *i.e.*, $E = f(X)$. Specifically, the latest interaction message e_q of each node u will be instantly maintained. Afterwards, treating E as the initial value $H^{t_p}(0)$ in continuous inference block, we utilize a ODE to learn the nodes' continuously changing state trajectories at a certain time interval $[0, \Delta t]$, and the embeddings of nodes $\bar{H}_u^{t_{p-1}}$ is updated as $\bar{H}_u^{t_p}$ by the node embeddings obtained in nodes' state trajectory at ending time $H^{t_p}(\Delta t)$. Then, selecting k observed historical neighbors of current nodes u, we introduce a self-attention mechanism on graphs to convert these interactions information to generate future node embedding $\hat{H}_u^t$. Finally, the decoder uses the future node embeddings $\hat{H}_u^t$ for downstream tasks (i.e., temporal link prediction, temporal node recommendation, and dynamic node classification).

4.2. Encoder-decoder

The proposed ConTIG adapts an encoder–decoder architecture (Fig. 2 left). For each interaction, we first project the features into a latent space to generate the initial hidden representation of nodes. After learning node embeddings at current timestamp, we finally design a decoder for the specific downstream tasks.

Before the encoder, we initialize the latest interactive node r_v and timestamp τ_v as 0 for all node $v \in \mathrm{V}$.

In the encoder, to learn the temporal pattern of each interaction, we use a time encoding function in Xu et al. (2019) to obtain a continuous functional mapping $F_T : T \rightarrow \mathbb{R}^{d_T}$ from time domain to the d^T-dimensional vector space, which projects the timestamps of interactions into the continuous-time space. For any given time t, we will generate its time embeddings as follows:

$$F_T(t) = \frac{1}{\sqrt{d_T}}[cos\left(\omega_1 t\right), sin\left(\omega_1 t\right), \\ \ldots, cos\left(\omega_{d_T} t\right), sin\left(\omega_{d_T} t\right)], \tag{1}$$

where $\omega_1, \omega_2, \ldots, \omega_{d_T}$ are trainable parameters. To learn informative node representations H^{t_p} at timestamp t_p, we concatenate the latest node embedding of the interactive node $\bar{H}_u^{t_{p-1}}$, the latest interactive node $\bar{H}_{r_u}^{t_{p-1}}$, the last interaction message feature f^{ur_u}, and the time embeddings of the time interval $F_T(\Delta t)$ to represent new node features of u at timestamp t (line 3 in Algorithm 1). Here, as we focus on the interval between the current interaction and its latest interaction, we use the time embedding $F_T(\Delta t)$. Afterwards, we adopt a fully connected layer as an encoder, and the hidden representations can be defined as follows:

$$E = f(\bar{H}_u^{t_{p-1}} \,||\, \bar{H}_{r_u}^{t_{p-1}} \,||\, f^{ur_u} \,||\, F_T(\Delta t)), \tag{2}$$

where $||$ is the concatenation operation, and $f(\cdot)$ is a linear projection as follows:

$$f(x) = xW + b, \tag{3}$$

where W and b are learnable parameters.

For the decoder, two fully connected layers are designed for temporal link prediction and temporal node recommendation tasks, and three fully connected layers are designed for dynamic node classification tasks.

4.3. Update: Continuous inference block

In the update module (Fig. 2 middle), to capture the continuous dynamic of node representation, inspired by Xhonneux et al. (2020), we define the ODE with the latest interaction information, neighbor features, and inherent characteristics of nodes, and use a neural solver to generate the node state trajectories at a certain time interval $[0, \Delta t]$. In this way, our method can estimate the possible direction of embedding trajectory for an inactive node.

We select the interactions between each node u in temporal interaction graph and its last interactive node r_u, and divide them into a set $\mathcal{E}_p$ to generate an adjacency matrix S_p describing their relationships. $S_p \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ is defined by $\mathcal{E}_p$ as follows:

$$S_p^{ui} = \begin{cases} 1 & \text{if } (u, r_u) \in \mathcal{E}_p \\ 0 & \text{otherwise,} \end{cases} \tag{4}$$

As the degree of nodes can be very different, we typically normalize the adjacency matrix as $D_p^{-\frac{1}{2}} S_p D_p^{-\frac{1}{2}}$, where $D_p = \operatorname{diag}\left(\sum_j S_p^{ij}\right) \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ is the degree matrix of S_p. As such a normalized adjacency matrix always has an eigenvalue decomposition, and the eigenvalues are in the interval $[-1, 1]$. To get the positive eigenvalues and make graph learning algorithms stable in practice, we follow (Kipf & Welling, 2017) and leverage the following regularized matrix for characterizing graph structures:

$$A_p = \frac{\beta}{2}\left(I_N + D_p^{-\frac{1}{2}} S_p D_p^{-\frac{1}{2}}\right), \tag{5}$$

where $\beta \in (0, 1)$ is a hyperparameter, $N = |\mathrm{V}|$. $I_N \in \mathbb{R}^{N\times N}$ is merely the identity matrix that are designed to guarantee the positive eigenvalue of the adjacency matrix (Kipf & Welling, 2017). As a result, I_N is completely unrelated to any node, thus it has no information about inherent characteristics of nodes. A_p only denotes the normalization of the adjacency matrix S_p without information on inherent characteristics of nodes, where the eigenvalues of A_p are in $[0, \beta]$.

Afterwards, to capture the complex non-linear dynamic of node embeddings, we assume that there are three possible factors affecting the node states: (1) latest interaction information exhibiting the latest nodes states; (2) neighbors' features affecting the change of node states; and (3) the inherent characteristics of nodes determining the influence of the aforementioned two factors. As a continuous network with dynamics, ODE can naturally adapt to observation data at any time interval. ODE can automatically adjust their evaluation strategy based on the input provided, leading to faster and more effective training processes. When the network layer of the model is large and the learning rate is small, it is possible to utilize the ODE specified by the neural network to parameterize the continuous dynamic change of the hidden unit. This method enables us to effectively learn the continuous change dynamic of node representations. Based on these considerations, we define an ODE in $\mathcal{E}_p$ to solve the node embedding trajectories, which adaptively fuse them with a gated mechanism. Here, we treat the encoder E_{t_p} as the initial value of ODE $H^{t_p}(0)$, i.e., $H^{t_p}(0) = E_{t_p}$. Then, inherent characteristics will inhibit the functionality of both latest interaction and neighbor features. Inspired by Xhonneux et al. (2020) we use a differential equation defined as follows to learn the continuous node representations in the interval of interactions:

$$\frac{\mathrm{d}H^{t_p}(t)}{\mathrm{d}t} = \underbrace{z_l \odot H^{t_p}(0)}_{\text{latest interaction}} + \underbrace{z_n \odot A_p H^{t_p}(t)}_{\text{neighbor features}} - \underbrace{z_i \odot H^{t_p}(t)}_{\text{inherent characteristics}}, \tag{6}$$

where $H^{t_p}(0)$ represents the latest interaction information of nodes, $H^{t_p}(t)$ represents the inherent characteristics of nodes, while $A_p H^{t_p}(t)$ only denotes the influence from neighbors on a specific node where the node state is always affected by their neighbors' features over time. As a result, neighbor features do not contain any information about inherent characteristics in Eq. (6). $\odot$ denotes the element-wise product operation. It is noted that "+/-" in Eq. (6) indicates the direction of three factors ("+" means the latest interaction and neighbor features are in the same direction to mutually promote each other; "-" means the inherent characteristics will inhibit the functionality of aforementioned two factors) rather than indicating a degree of importance. z_l, z_n, and z_i represents three gates, respectively. They are computed as:

$$z_l = \sigma(W_l H^{t_p}(0) + b_l), \tag{7}$$

$$z_n = \sigma(W_n H^{t_p}(0) + b_n), \tag{8}$$

$$z_i = \sigma(W_i H^{t_p}(0) + b_i), \tag{9}$$

where W_l, W_n, W_i and b_l, b_n, b_i are trainable parameters, and $\sigma(\cdot)$ is the sigmoid activation function to normalize the output into $[0, 1]$. The gated fusion mechanism can adaptively fuse three factors according to their importance calculated by the initial value of ODE. In addition, the updating process starts at the timestamp of the latest interaction of the node. To this end, following (Rossi et al., 2020), when an interactive message comes, we use the latest interaction information to update node embeddings and save the current interaction message for the next time the node appears. Meanwhile, due to the integration of time encoding in $H^{t_p}(0)$, the temporal behaviors of nodes could be captured. As a result, we can leverage the aforementioned ODE to learn the nodes' state trajectories at the interval $[0, \Delta t]$, and the three main factors mentioned above jointly capture the continuous dynamic of node representations. Then the node representations are updated by ODE solver as follows:

$$H^{t_p}(\Delta t) = \mathrm{ODESolve}(g(t), H^{t_p}(0), \Delta t) \tag{10}$$

where $g(t) = \frac{\mathrm{d}H^{t_p}(t)}{\mathrm{d}t}$.

Finally, we use the hidden state at the end time $H(\Delta t_p)$ to update the previous embedding memory $\bar{H}^{t_{p-1}}$ as $\bar{H}^{t_p}$ (i.e., $\bar{H}^{t_p} = H(\Delta t_p)$) (line 5 in Algorithm 1).

4.4. Transform: Graph attention layer

After capturing the continuous dynamic of node representations from the latest interaction sets and updating the embeddings of nodes, a graph attention layer in transform module (Fig. 2 right) is applied to convert the historical observed interaction features of nodes to generate future representations (Line 6 and 16–23 in Algorithm 1).

In this module, for the current interaction e connected by u and v at time t, their temporal neighbors and the interaction information between them is token as input. We introduce a self-attention mechanism to distinguish different neighbors, and take account of the structural information with temporal information (Xu et al., 2020). For node u at time t, we consider its neighbors $N(u,t) = \{i_0, \dots, i_{k-1}\}$, where the interactions between u and $i_j \in N(u,t)$ occurred at time t_j prior to time t, and the sampling process of temporal neighbors of node i is the same as u. Then we take the node information with time t encoding $z_u^t = \bar{h}_u^t \,||\, F_T(t)$ and the neighborhood information with the time interval $t - t_j$ encoding $Z_N^t = \bar{H}_{N(u,t)}^t \,||\, f^{uN(u,t)} \,||\, F_T(t-t_j)$ as the input of attention, where the time interval $t - t_j$ is between current interaction e and the interaction of u and its neighbors $i_j \in N(u,t)$. In attention, three different linear projections are used to obtain the query Q_t, key K_t, and value V_t:

$$q_t = Z_u^t W_Q, K_t = Z_N^t W_K, V_t = Z_N^t W_V, \tag{11}$$

where W_Q, W_K, W_V are the weight matrices. The attention weights α_j is given by:

$$\alpha_j = \frac{\exp(q^{\mathrm{T}} K_j / \sqrt{d})}{\sum_c \exp\left(q^{\mathrm{T}} K_c / \sqrt{d}\right)}, \tag{12}$$

where the attention weight α_j reveals how node u attends to the features of node i_j within the topological structure defined as $N(u,t)$ after accounting for their interaction time with i_j. Specifically, we have updated the target node by considering the three-fold factors (i.e., latest interaction, neighbor features, and inherent characteristics) in the update module. In the transfer module, the inputs already incorporate both the inherent characteristics and the latest interaction. Therefore, the layer produces the time-aware representation of node u at time t, $\hat{h}_u^t$, which represents the future hidden representations generated by historical observed interactions as follows:

$$\hat{h}_u^t = \sum_{j=0}^{k-1} \alpha_j V_t, \tag{13}$$

To stabilize the learning process and improves performances, we extend the self-attention mechanism to be multi-head ones (Vaswani et al., 2017). Specifically, we concatenate M parallel attention mechanisms with different learnable projections:

$$\hat{h}_u^t = ||_{m=1}^{M} \left\{ \sum_{j=0}^{k-1} \alpha_j^{(m)} V_t^{(m)} \right\}, \tag{14}$$

where $V_t^{(m)}$ represent the value with different projections in the mth head attention, and $\alpha_j^{(m)}$ represent the attention weight calculated by the query and key with different projections in the mth head attention.

Finally, the future node embeddings $\hat{H}_u^{t_p}$ is generated by calculating $\hat{h}_u^{t_p}$ for each node in interactions (line 6 in Algorithm 1). The latest interactive node r_u and r_i are updated as i and u, respectively. In addition, the latest interactive timestamps τ_u and τ_i are both updated as t_p (Lines 7–8 Algorithm 1).

4.5. Binary cross-entropy loss function

In this work, we adopt time-sensitive link prediction binary cross-entropy loss function to learn ConTIG's parameters. The binary cross-entropy loss function is defined as follows and our goal is to maximize this likelihood function:

$$\mathcal{L} = \sum_{(u_p,i_p,t_p)\in\mathcal{E}} -\log\sigma(-\hat{h}_{u_p}^{t_p\top}\hat{h}_{i_p}^{t_p}) - Q\mathbb{E}_{i_q\sim P(i)}\log\sigma(\hat{h}_{u_p}^{t_p\top}\hat{h}_{i_q}^{t_p}), \tag{15}$$

where the summation is over the observed edges on u_p and i_p that interact at time t_p, Q is the number of negative samples and $P(i)$ is the negative sampling distribution over the node space.

4.6. Complexity analysis

The time complexity of our proposed method mainly consists of two portions.

First, for the update module (i.e., continuous inference block), we consider that both adjacency matrices are stored as sparse matrices. And the runtime of the ODE solver depends on the length on the time interval (i.e., the end time of ODE solver) Δt and the complexity of the dynamics. Then, the time complexity of the continuous inference block is $O(\Delta t\,|\mathcal{E}|)$.

Second, for the transform module (i.e., graph attention layer), since the masked self-attention operation is parallelizable, as suggested by Vaswani et al. (2017). The per-batch time complexity of the graph attention layer with m heads can be expressed as $O(mk)$, where k is the average neighborhood size. Since the batch is divided by edges, the time complexity of the graph attention layer is $O(mk\,|\mathcal{E}|)$, where $m \ll |\mathcal{E}|$ and $k \ll |\mathcal{E}|$.

Therefore, the time complexity of ConTIG is $O((\Delta t + mk)\,|\mathcal{E}|) \approx O(C\,|\mathcal{E}|)$, where C is a constant and is relative to the runtime of the ODE Solver.

5. Experiments

In this section, we will utilize four networks to compare the performance of our model with four static and five temporal methods. We conduct three main tasks, link prediction, node recommendation, and node classification, to evaluate the influence of introducing temporal information and learn continuous dynamic of node embedding trajectories.

5.1. Experimental setup

Datasets We evaluate the performance of ConTIG on temporal link prediction, node recommendation and dynamic node classification tasks with four public datasets, where three datasets are user-item networks selected by Kumar et al. (2019) and one dataset is e-mail network. The statistics of the four datasets are summarized in Table 2, where the number of interactions is $|\mathcal{E}|$.

Algorithm 1 Continuous Representation Learning on Temporal Interaction Graphs

```
Input: Interaction Stream $e = (u, i, t, f^{ui}) \in \mathcal{E}$ in temporal interaction graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
Output: Node Embeddings $\hat{h}_v^t, \forall v \in \mathcal{V}$
 1: $\bar{h}_v^0 \leftarrow x_v, r_v \leftarrow 0, \tau_v \leftarrow 0, \forall v \in \mathcal{V}$
 2: for $e_p = (u, i, t_p, f^{ui})$ in $\mathcal{E}$
 3:     Learning $F_T(t_p - \tau_v)$ by Eqn.(1)
 4:     $E_v^{t_p} = f(\bar{h}_v^{t_{p-1}} \,||\, \bar{h}_{r_v}^{t_{p-1}} \,||\, f^{vr_v} \,||\, F_T(t_p - \tau_v)), \forall v \in \mathcal{V}$
 5:     $\bar{h}_u^{t_p}, \bar{h}_i^{t_p} \leftarrow$ CONTINUOUS INFERENCE BLOCK $(E^{t_p}, u, i)$
 6:     $\hat{h}_u^{t_p}, \hat{h}_i^{t_p} \leftarrow$ GRAPH ATTENTION LAYER $(\bar{h}_u^{t_p}, \bar{h}_i^{t_p}, u, i, t_p)$
 7:     $r_u \leftarrow i, \tau_u \leftarrow t_p$
 8:     $r_i \leftarrow u, \tau_i \leftarrow t_p$
 9: end for
10: return $\hat{h}_v^t, \forall v \in \mathcal{V}$
11: CONTINUOUS INFERENCE BLOCK$(E^{t_p}, u, i)$
12:     $h_v(0) \leftarrow E_v^{t_p}, v \in \mathcal{V}$
13:     $(v, r_v), \forall v \in \mathcal{V}$ generates $A_p$ by Eqn.(4)-(5)
14:     $h_v(\Delta t) \leftarrow$ ODESolve$(g(t), h_v(0), \Delta t), \forall v \in \mathcal{V}$
15:     return $h_u(\Delta t), h_i(\Delta t)$
16: GRAPH ATTENTION LAYER$(\bar{h}_u^t, \bar{h}_i^t, u, i, t)$
17:     for $v$ in $\{u, i\}$
18:         Sample $k$ temporal neighbors $N(v, t)$
19:         $z_v^t \leftarrow \bar{h}_v^t \,||\, F_T(t)$
20:         $Z_{N(v,t)}^t \leftarrow \bar{H}_{N(v,t)}^t \,||\, f^{vN(v,t)} \,||\, F_T(t - t_j)$
21:         $\hat{h}_v^t \leftarrow$ Attn$(z_v^t, Z_{N(v,t)}^t, Z_{N(v,t)}^t)$
22:     end for
23:     return $\hat{h}_u^t, \hat{h}_i^t$
```

- **Wikipedia.**[3] The dataset describes the interactions between active users and pages they edit the contents with unique timestamps and the dynamic labels indicating whether a user is banned from editing.
- **Reddit.**[4] The dataset describes the interactions between active users and the posts they submit on subreddits and the dynamic labels indicating if users are banned from posting.
- **Mooc.**[5] The dataset describes the interactions between students and MOOC online courses, e.g., viewing a video, submitting an answer, etc.
- **CollegeMsg.**[6] The dataset is an online social network at the University of California and describes the interaction between users by sending private messages at different timestamps. The dataset is without node labels and edge features.

Baselines To evaluate the performance of ConTIG, we compare our method with state-of-the-art graph embedding methods on both static and temporal graphs.

Static graph embedding methods: GAE (Kipf & Welling, 2017) utilizes GCN to encode the graph and then decodes it with the transpose of the node hidden feature product to reconstruct the adjacency matrix of the graph. VGAE (Kipf & Welling, 2016) employs VAE to capture the key features of nodes within the graph. Different from GAE (Kipf & Welling, 2017), VGAE encodes the content as parameters of a Gaussian distribution rather than fixed node parameters in the encoding phase. GraphSAGE (Hamilton et al., 2017a) samples a fixed number of neighbors for each node level in the graph, and updates the node representation by combining the aggregated neighbor node features

[3] http://snap.stanford.edu/jodie/wikipedia.csv
[4] http://snap.stanford.edu/jodie/reddit.csv
[5] http://snap.stanford.edu/jodie/mooc.csv
[6] https://snap.stanford.edu/data/CollegeMsg.html

Table 2
Statistics of the datasets.

Datasets	$\|\mathcal{V}\|$	$\|\mathcal{E}\|$	Feature	Label
Wikipedia	9,227	157,474	172	2
Reddit	10,984	672,447	172	2
Mooc	7,144	411,749	0	2
CollegeMsg	1,899	59,835	0	0

with the features of the target node itself. GAT (Velickovic et al., 2018) uses the attention mechanism to calculate the attention scores of neighbor nodes to the target node. These scores are subsequently as weights to aggregate the features of the neighboring nodes.

Temporal graph embedding methods: CTDNE (Nguyen et al., 2018) defines the temporal random walk requiring the walks to obey the temporal order. JODIE (Kumar et al., 2019) learns to generate embedding trajectories of all users and items from temporal interactions by update and project operations. DyRep (Trivedi et al., 2019) uses deep recurrent architecture and attention mechanism to effectively model fine-grained temporal dynamic. TGAT (Xu et al., 2020) introduces a self-attention mechanism and a functional time encoding technique to learn the time-feature interactions. TGN (Rossi et al., 2020) is a generic inductive framework operating on continuous-time dynamic graphs (Kazemi et al., 2020) represented as a sequence of events.

Parameter Settings For parameter settings, the dimension of both node representations and time representations are determined by grid search in the range of 16, 64, 128, 172, 256, 512 following (Rossi et al., 2020; Xu et al., 2020). The optimizer is Adam algorithm, learning rate is 0.0001, and dropout probability is 0.1. In continuous inference block, parameter β in adjacency matrix regularization is 0.95, and the end time of ODE is set as 1.0 instead of the real interval Δt. In graph attention layer, the number of selected neighbors k is set as 15 and heads M in attention is set as 2, and the negative sampling distribution $P(i)$ is a uniform distribution.

Settings for Baselines For static baselines, we adopt the same baseline training procedures as in Xu et al. (2020). We refer to the PyTorch geometric library for implementing the GAE and VGAE baselines, and develop off-the-shelf implementation for GraphSAGE and GAT by referencing their original implementations to accommodate the temporal setting and incorporate edges features. For dynamic baselines, we use the open-source implementations for CTDNE and use the source code released by the authors to implement TGAT and TGN, where we implement JODIE and DyRep in the version of TGN in PyTorch following TGN.

Implementation All our experiments are implemented on a 32g-MEM Ubuntu 20.04 server with Intel(R) Core(TM) i7-9700K CPU @ 3.60 GHz and NVidia(R) 2080Ti GPUs. All code is implemented in PyTorch version 1.5.1.

5.2. Temporal link prediction

The goal of the temporal link prediction task is to predict the probability that there exists a link between the two nodes given two nodes and a certain timestamp in the future. For this task, we evaluate our method on both the transductive and inductive settings. In the transductive task, we predict the links between nodes observed during training. In the inductive task, we predict the links between new nodes which have not been observed in the past. Our model is tuned on the validation set and we report the average precision (AP) on the test set. We divide the training, validation, and testing sets into a 70%-15%–15% split. After the division, the nodes that do not appear in the training set are considered as new nodes. The model is trained by temporal link prediction.

The results comparison between our method and baseline methods in temporal link prediction tasks are shown in Table 3. We observe that: (1) static graph embedding methods including GAE, VGAE,

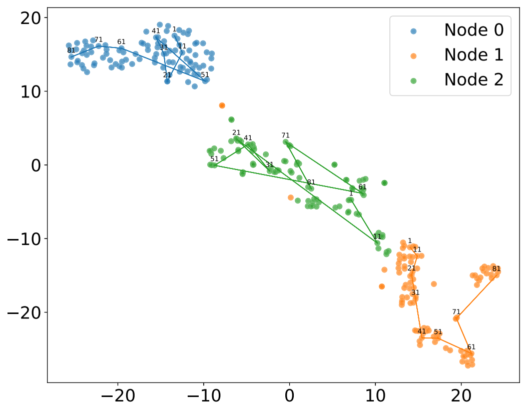


Fig. 3. A t-SNE plot of three node representations under dynamic changes over time with ConTIG on the Wikipedia dataset. The larger the number on the data, the longer the training time. The length of the line represents the distance that the node moves. The trajectory of the three node representations during the training phase is relatively steady, indicating the robustness of ConTIG in handling dynamic changes over time.

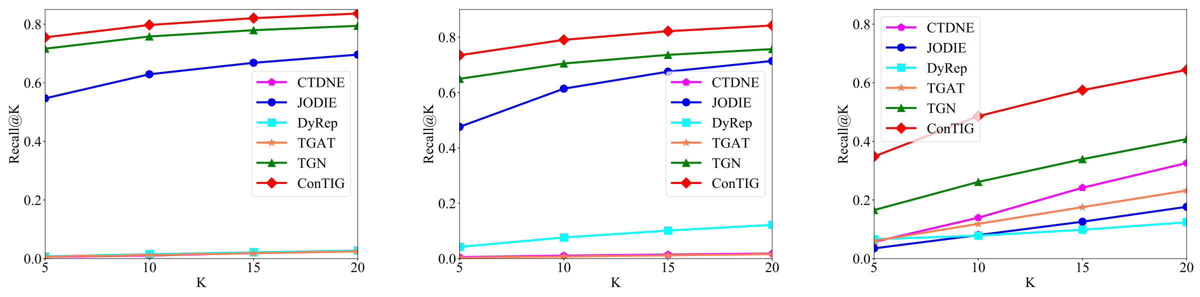
GraphSAGE, and GAT, perform worse than those baselines modeling temporal interaction information. Because most of the interactions in the real-world networks are time-stamped; (2) among the dynamic graph embedding works, both temporal neighbor information (i.e., CTDNE, TGAT) and latest interaction information (i.e., JODIE, DyRep) methods perform worse than fusing methods (i.e., TGN, ConTIG); and (3) ConTIG achieves the best prediction performance on the datasets Wikipedia, Reddit, and CollegeMsg for both transductive and inductive settings, and the second best performance on Mooc for the inductive task. This observation demonstrates the advantage of our proposed method compared to existing methods. In fact, by considering the inherent node properties and modeling the three important factors (i.e., latest interaction, neighbor features, and inherent characteristics) in temporal interaction graphs, our method can capture the complex non-linear dynamics of node representations effectively, thereby achieving superb performance.

Fig. 3 represents the visualization results of three node representations under dynamic changes over time with ConTIG on the Wikipedia dataset. One of the key observations from Fig. 3 is the stability of these trajectories of the three node representations (i.e., Node 0, 1, and (2) during the training phase. For example in Fig. 3, the representation of Node 0, 1, and 2 predominantly evolves within a confined area over time respectively (e.g., the representation of Node 0 is restricted in the upper-left corner of Fig. 3). This consistent pattern of evolution within a limited space is indicative of the stability in our ConTIG model.

5.3. Temporal node recommendation

The goal of temporal node recommendation task is to predict the top-K possible neighbors of node u at t given the node u and future timestamp t. This task is also used to evaluate the performance of temporal network embedding methods. For this task, we evaluate all methods on transductive setting, each method outputs the user u's preference scores over all the items at time t in test set. We sort scores in a descending order and record the rank of the paired node u and i. We evaluate the task on three user-item networks (i.e., Wikipedia, Reddit, and Mooc), where CollegeMsg dataset is not included because it is not a user-item network. set, and divide the training, validation, and testing sets into a 70%-15%–15% split. Our evaluation metric in this task is Recall@K, where $K \in \{5, 10, 15, 20\}$.

The results comparison between our method and baseline methods in temporal node recommendation task is shown in Fig. 4, showing that our model ConTIG performs better than all the baselines. Compared with the best competitors (i.e., TGN), the recommendation performance



(a) Recall@K on Wikipedia (b) Recall@K on Reddit (c) Recall@K on Mooc

Fig. 4. Recall@K for the transductive temporal node recommendation on Wikipedia, Reddit and Mooc.

Table 3
ROC AUC(%) and Average Precision(%) for the transductive temporal link prediction on Wikipedia, Reddit, Mooc, and CollegeMsg. The means and standard deviations are computed for ten runs.

Task	Methods	Wikipedia		Reddit		Mooc		CollegeMsg	
		AUC	AP	AUC	AP	AUC	AP	AUC	AP
Transductive	GAE	91.47 ± 0.3	91.12 ± 0.1	95.87 ± 1.2	96.57 ± 1.0	87.89 ± 0.6	90.70 ± 0.3	73.15 ± 1.5	70.00 ± 1.17
	VGAE	82.43 ± 1.6	82.50 ± 4.0	92.70 ± 0.4	91.53 ± 0.7	88.21 ± 0.6	**91.00** ± 0.3	74.07 ± 0.9	70.66 ± 1.0
	GraphSAGE	92.00 ± 0.3	92.34 ± 0.3	97.75 ± 0.1	97.85 ± 0.1	56.17 ± 0.3	60.63 ± 0.2	62.38 ± 1.3	62.48 ± 0.9
	GAT	92.76 ± 0.5	93.17 ± 0.5	97.90 ± 0.1	97.07 ± 0.1	67.24 ± 0.1	66.66 ± 0.8	78.09 ± 0.5	75.97 ± 0.7
	CTDNE	82.36 ± 0.7	80.86 ± 0.7	85.32 ± 2.0	87.31 ± 1.4	88.97 ± 2.6	89.27 ± 2.0	81.88 ± 0.7	80.25 ± 0.8
	JODIE	94.94 ± 0.3	94.65 ± 0.6	97.62 ± 0.2	97.07 ± 0.4	79.75 ± 2.8	74.85 ± 3.1	59.85 ± 6.0	54.50 ± 4.4
	DyRep	94.22 ± 0.2	94.63 ± 0.2	98.01 ± 0.1	98.05 ± 0.1	80.57 ± 2.1	77.30 ± 2.2	54.75 ± 6.8	51.89 ± 4.8
	TGAT	94.99 ± 0.3	95.29 ± 0.2	98.07 ± 0.1	98.17 ± 0.1	66.02 ± 1.0	63.82 ± 0.9	81.05 ± 0.6	79.16 ± 0.6
	TGN	98.42 ± 0.1	98.50 ± 0.1	98.69 ± 0.1	98.73 ± 0.1	89.07 ± 1.6	86.96 ± 2.1	85.06 ± 5.9	85.38 ± 6.4
	ConTIG (our work)	**98.50** ± 0.2	**98.62** ± 0.2	**98.71** ± 0.3	**98.75** ± 0.3	**90.34** ± 1.6	88.87 ± 1.9	**90.11** ± 1.2	**90.54** ± 1.2
Inductive	GraphSAGE	88.60 ± 0.3	88.94 ± 0.5	94.28 ± 0.4	94.51 ± 0.1	53.68 ± 0.4	55.35 ± 0.4	49.64 ± 1.5	51.83 ± 0.8
	GAT	89.11 ± 0.5	89.82 ± 0.4	94.30 ± 0.4	94.58 ± 0.3	53.43 ± 2.1	54.80 ± 0.9	68.98 ± 1.2	66.22 ± 1.2
	JODIE	92.75 ± 0.3	93.11 ± 0.4	95.42 ± 0.2	94.50 ± 0.6	81.43 ± 0.8	76.82 ± 1.4	51.59 ± 3.2	50.02 ± 2.2
	DyRep	91.03 ± 0.3	91.96 ± 0.2	95.79 ± 0.5	95.75 ± 0.5	82.06 ± 1.7	79.17 ± 1.6	49.05 ± 4.1	49.30 ± 2.6
	TGAT	93.37 ± 0.3	93.86 ± 0.3	96.46 ± 0.1	96.61 ± 0.2	69.09 ± 0.8	67.65 ± 0.7	72.27 ± 0.5	72.53 ± 0.6
	TGN	97.72 ± 0.1	97.83 ± 0.1	97.54 ± 0.1	97.63 ± 0.1	**89.03** ± 1.6	**86.70** ± 2.0	78.54 ± 3.9	80.77 ± 3.7
	ConTIG (our work)	**98.44** ± 0.2	**98.41** ± 0.2	**98.26** ± 0.3	**98.31** ± 0.2	86.77 ± 2.0	85.44 ± 2.0	**83.63** ± 0.9	**85.37** ± 0.9

Table 4
ROC AUC(%) for the transductive dynamic node classification on Wikipedia, Reddit, and Mooc. The means and standard deviations are computed for ten runs.

	Wikipedia	Reddit	Mooc
CTDNE	84.86 ± 1.5	54.38 ± 7.5	71.84 ± 1.0
JODIE	84.40 ± 0.9	61.51 ± 1.2	70.03 ± 0.5
DyRep	83.25 ± 0.5	60.86 ± 1.7	64.64 ± 1.4
TGAT	84.41 ± 1.5	65.98 ± 1.6	65.79 ± 0.5
TGN	**87.56** ± 0.7	69.78 ± 0.8	63.93 ± 0.3
ConTIG (our work)	87.13 ± 0.6	**69.99** ± 0.5	**73.56** ± 0.4

of ConTIG improves by 110.51%, 13.13%, and 3.25% in terms of Recall@5 on Mooc, Reddit, and Wikipedia. These significant improvements verify that the differential equation fused with three factors (i.e., latest interaction, neighbor features, and inherent characteristics) proposed in ConTIG is capable of learning the trend of node state trajectories in the network. Additionally, the significant improvement of ConTIG benefits from the combination of continuous node state modeling in update module and dynamic sub-graph structure capturing in transform module on node embeddings, which is good for the down-stream node recommendation task.

5.4. Dynamic node classification

The goal of dynamic node classification task is to predict the state label of user given the user, item, and future timestamp. For this task, we evaluate our method on transductive setting, predicting the state labels of users who have been observed during training. We evaluate the task on three datasets with dynamic node labels (i.e., Wikipedia, Reddit, and Mooc), where CollegeMsg dataset is not included because there is no node label, and divide the training, validation, and testing sets into a 70%-15%–15% split. Specifically, we train a decoder after the model trained by the temporal link prediction task. Our evaluation metric in this task is the area under the ROC curve (AUC).

The results comparison between our method and baseline methods in dynamic node classification tasks are shown in Table 4. Again, our algorithm achieves the best or comparable performance compared with existing dynamic graph embedding approaches, demonstrating its effectiveness for the down-stream node classification task. s

5.5. Performance on long-interval interactions

The goal of this task is to observe the link prediction performance of our method for interaction sets with different time intervals. To categorize the interactions, first, we calculate the time interval Δt between each interaction and its latest interaction as mentioned in Section 3, and describe the distribution of the time intervals of interactions. Then, we equally divide the interactions into five sets according to the four quantiles: 20%, 40%, 60%, 80%, where the quantiles are cut points dividing the interactions into continuous intervals with equal probabilities in terms of the intervals Δt of interactions. For example, the interactions in 0%–20% set are of shorter intervals, while the interactions in 80%–100% set are of longer intervals. Finally, we calculate the AP score for each set.

Our results of comparison between our method and dynamic graph baselines in temporal link prediction task on five interaction sets of Wikipedia and CollegeMsg are shown in Fig. 5. Comparing the AP results in each set, we find that our method outperforms TGN, especially in long-interval interactions (e.g., the frequency interval 60%–80% and 80%–100%), demonstrating the superiority of ConTIG in capturing

Table 5
Average Precision(%) for both the transductive and inductive temporal link prediction on Wikipedia, Reddit, Mooc, and CollegeMsg.

	Wikipedia		Reddit		Mooc		CollegeMsg	
	Transductive	Inductive	Transductive	Inductive	Transductive	Inductive	Transductive	Inductive
ConTIG w/o transform	95.99	96.76	98.25	97.88	82.33	75.22	87.66	67.65
ConTIG w/o update	95.19	95.03	95.85	93.36	75.51	73.10	82.78	80.90
ConTIG w/o adaptive	98.60	98.36	98.75	98.15	68.56	62.42	87.72	84.23
ConTIG w/o latest	98.50	98.45	98.45	97.79	90.25	86.76	90.66	85.42
ConTIG w/o neighbor	98.38	98.25	98.02	96.95	84.03	79.57	89.08	83.13
ConTIG w/o inherent	**98.75**	98.56	**98.79**	98.35	86.74	84.27	90.66	82.72
ConTIG	98.61	**98.58**	**98.79**	**98.43**	**90.56**	**87.31**	**91.40**	**86.42**

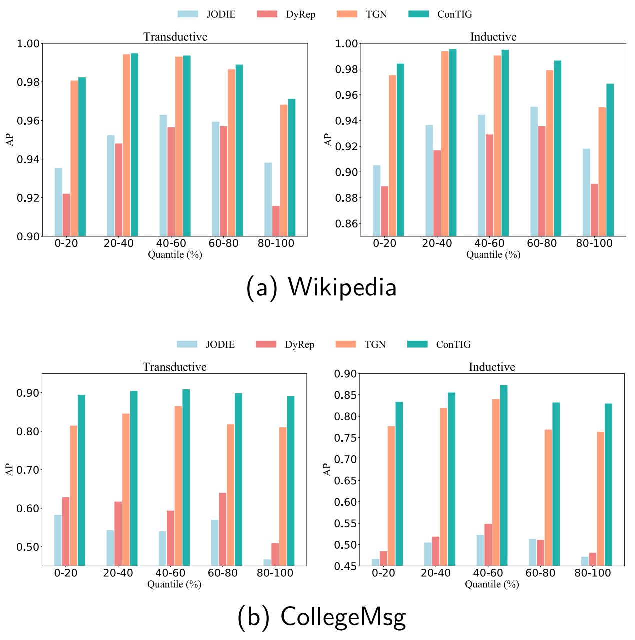


Fig. 5. Average Precision(%) for both the transductive and inductive temporal link prediction on five interaction sets of Wikipedia and CollegeMsg, divided by quantiles of time interval of the interaction.

continuous dynamics of node representations. And we argue that the long-interval link prediction is more beneficial to practical applications because quite a few users are always inactive on social networks, citation networks, and other user-item interaction systems.

5.6. Ablation study

To further investigate the effect of each component in our model, we compare ConTIG with its variants as follows:

- ConTIG w/o transform: ConTIG without the transform module (i.e. graph attention layer).
- ConTIG w/o update: ConTIG without the update module (i.e. continuous inference block).
- ConTIG w/o adaptive: ConTIG without adaptively fusing three factors in update module. We replace it by directly adding the three factors in the update module.
- ConTIG w/o latest: ConTIG without the latest interaction factor in update module.
- ConTIG w/o neighbor: ConTIG without the neighbor features factor in update module.
- ConTIG w/o inherent: ConTIG without the inherent characteristics factor in update module.

Table 5 shows the AP results of each model. ConTIG performs better than ConTIG w/o transform and ConTIG w/o update by a large margin, which demonstrates the effectiveness of the main modules of our model. Especially, the introduction of the continuous update module significantly improves the results, indicating that the latest knowledge between two consecutive interactions is an essential feature to learn the node representations. By further modeling and aggregating the historical interaction information of nodes, ConTIG consistently improves the performance, showing the importance of the historical interaction information.

Specifically, we investigate the effect of each component in update module. By removing the adaptive fusion approach, the performance of ConTIG w/o adaptive degrades obviously, pointing out that the adaptive fusion is a key factor to the success of the ODE solver in update module. It helps the network to focus on the most correlated factor to update the node state, and adaptively fuses three factors in a data-dependent way. Besides, ConTIG outperforms ConTIG w/o latest, ConTIG w/o neighbor and ConTIG w/o inherent by a small margin, which indicates that the three factors are of great importance for learning the change of node state in the update module. In particular, by comparing ConTIG with ConTIG w/o latest, ConTIG w/o neighbor, and ConTIG w/o inherent, respectively, we observe that the neighbor features contribute most to the performance in all datasets, which indicates the importance of neighbors in the latest changing of node states. For ConTIG w/o inherent, we observe that it has obvious effects on the datasets without edge features (i.e., Mooc and CollegeMsg), although it has tiny effects on the datasets with edge features (i.e., Wikipedia and Reddit).

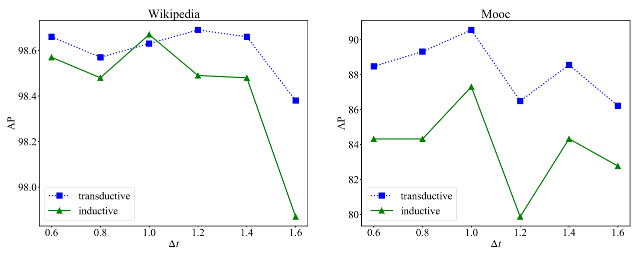
5.7. Parameter sensitivity

There are several hyper-parameters in our proposed method. It is necessary to analyze how these parameters influence the performance of our model in temporal link prediction task on the above datasets. In detail, these parameters include the end time Δt in continuous inference block, the number of neighbors k, heads M in graph attention layer, node embedding dimension d, and time embedding dimension d^T. We choose different values for them and use the AP score to evaluate their performance.

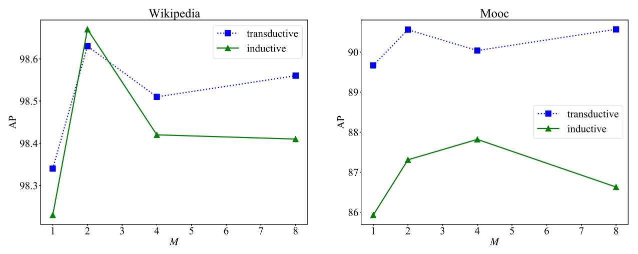
End time in continuous inference block (See Section 4.3). An important parameter is how long used to update the node states in the continuous inference block. In our experiment, the Δt ranges from 0.6 to 1.6 and the node and time embedding dimension is fixed to 172. As shown in Fig. 6(a), as Δt is larger, the AP score first increases and then decreases, demonstrating that more time for learning node changes between two consecutive interactions could yield better performance. However, when the updating time is very long, it would make the current node over rely on the latest information, and forget the historical information, which hinders the performance.

Heads in graph attention layer (See Section 4.4). Fig. 6(b) show the AP results under different number of heads M in the multi-head attention. We observed that the AP score first increases and then decreases or stabilizes, and there is a relatively stable and good performance at two-head attention. It means that less-head attention will make the results unstable and offset, but attention with many heads may pay attention to useless or unrelated information.

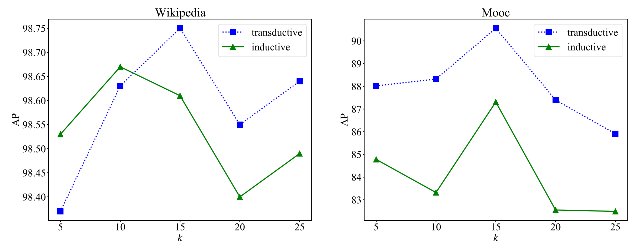
Number of neighbors (See Section 4.4). The influence of the number of neighbors will then be evaluated, which controls the amount



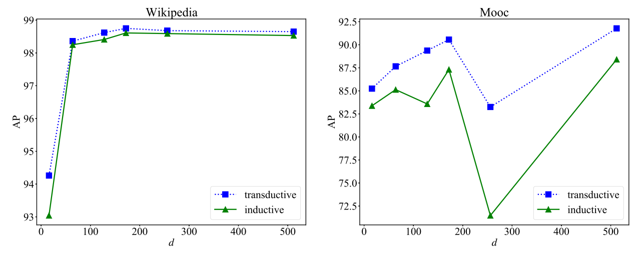
(a) end time in ODE Δt



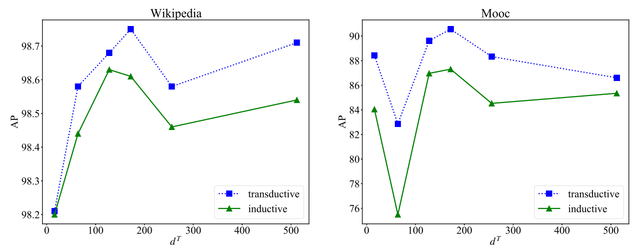
(b) number of heads M



(c) number of neighbors k



(d) node embedding dimension d



(e) time embedding dimension d^T

Fig. 6. Average Precision(%) for both transductive and inductive temporal link prediction on Wikipedia and Mooc with different hyper-parameters.

of historical interactive information is considered in the transform module. As shown in Fig. 6(c), in general, as the number of neighborhood k becomes larger, the model could achieve better performance because more historical information is considered. However, when the number of neighborhood is very large, it would introduce useless or overdue information into the learning process and thus degrade the performance.

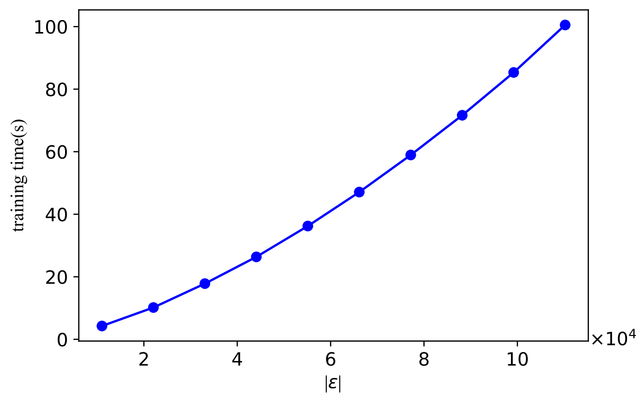


Fig. 7. Mean time over one epoch during the entire ConTIG training on Wikipedia with different $|\mathcal{E}|$ for training.

Node embedding dimension. The influence of different node embedding dimensions (e.g., 16, 64, 128, 172, 256, and 512) on our model is shown in Fig. 6(d). We discover the same trend on two datasets that as the dimension rises from a small value, the performance of ConTIG will improve rapidly and becomes relatively stable while the size is large. The reason is that a higher dimension enables the model to learn more information in the latent space. However, it would make computational consumption become very large, so we choose a reasonable dimension with the best performance (i.e., 172).

Time embedding dimension. The influence of different time embedding dimensions 16; 64; 128; 172; 256; 512 on our model is shown in Fig. 6(e). We observe the similar results with node embedding dimension, as d^T is larger, the AP score first increases and then tends to be stable. Different from node embedding dimension, there is a fluctuation in the high-dimension time embedding on Wikipedia and low-dimension time embedding on Mooc, which means high dimension and low dimensions may both hinder the performance, and shows the importance of choosing a reasonable time embedding dimension (i.e., 172).

5.8. *Complexity evaluation*

Here, we examine how the training time of ConTIG depends on the number of edges $|\mathcal{E}|$ which are used for training. We record the runtimes of ConTIG for training one epoch on the Wikipedia dataset using the best parameters in Section 5.7. Fig. 7 shows that the entire runtime for one-epoch training is close to linear with $|\mathcal{E}|$. This evaluation demonstrates our method's advantage of being scalable to process long edge streams.

6. Conclusion

In this paper, we present ConTIG, a novel representation learning method to capture the continuous dynamic of node embedding trajectories by identifying three-fold factors (i.e., latest interaction, neighbor features, and inherent characteristics). ConTIG contains two modules: a update module to learn the node embedding trajectories and a transform module to generate the future representations according to the historical interaction information. Experiments results demonstrate that ConTIG achieves state-of-the-art performances, especially for long-interval interactions on temporal link prediction tasks. In the future, besides node state trajectory, we plan to pay more attention to community evolution, exploring the impact of community on individuals during the graph evolution.

CRediT authorship contribution statement

Zihui Wang: Data curation, Investigation, Methodology, Writing – review & editing, Resources, Software. **Peizhen Yang:** Investigation, Methodology, Software, Writing – review & editing. **Xiaoliang Fan:** Conceptualization, Funding acquisition, Supervision, Writing – original draft, Writing – review & editing, Project administration, Resources. **Xu Yan:** Investigation, Methodology, Writing – original draft. **Zonghan Wu:** Writing – review & editing. **Shirui Pan:** Writing – review & editing. **Longbiao Chen:** Writing – review & editing. **Yu Zang:** Writing – review & editing. **Cheng Wang:** Writing – review & editing. **Rongshan Yu:** Conceptualization, Writing – review & editing.

Declaration of competing interest

We declare that we have no financial and personal relationships with other people or organizations that can inappropriately influence our work, there is no professional or other personal interest of any nature or kind in any product, service and/or company that could be construed as influencing the position presented in, or the review of, the manuscript entitled.

Data availability

Data will be made available on request.

Acknowledgments

The research was supported by Natural Science Foundation of China (62272403, 61872306), and Fundamental Research Funds for the Central Universities, China (20720200031).

References

Ahmed, A., Shervashidze, N., Narayanamurthy, S., Josifovski, V., & Smola, A. J. (2013). Distributed large-scale natural graph factorization. In *Proceedings of the 22nd international conference on world wide web* (pp. 37–48).

An, J., Liu, W., Liu, Q., Guo, L., Ren, P., & Li, T. (2022). DGInet: Dynamic graph and interaction-aware convolutional network for vehicle trajectory prediction. *Neural Networks*, *151*, 336–348.

Chen, R. T., Rubanova, Y., Bettencourt, J., & Duvenaud, D. K. (2018). Neural ordinary differential equations. In *Advances in neural information processing systems* (pp. 6571–6583).

Cui, P., Wang, X., Pei, J., & Zhu, W. (2018). A survey on network embedding. *IEEE Transactions on Knowledge and Data Engineering*, *31*(5), 833–852.

Ding, Z., Han, Z., Ma, Y., & Tresp, V. (2021). Temporal knowledge graph forecasting with neural ODE. arXiv preprint arXiv:2101.05151.

Fu, D., Zhou, D., & He, J. (2020). Local motif clustering on time-evolving graphs. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 390–400).

Gao, C., Zhu, J., Zhang, F., Wang, Z., & Li, X. (2022). A novel representation learning for dynamic graphs based on graph convolutional networks. *IEEE Transactions on Cybernetics*, 1–14.

Gong, M., Ji, S., Xie, Y., Gao, Y., & Qin, A. (2020). Exploring temporal information for dynamic network embedding. *IEEE Transactions on Knowledge and Data Engineering*.

Goyal, P., Chhetri, S. R., & Canedo, A. (2020). dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. *Knowledge-Based Systems*, *187*, Article 104816.

Goyal, P., Kamra, N., He, X., & Liu, Y. (2018). Dyngem: Deep embedding method for dynamic graphs. arXiv preprint arXiv:1805.11273.

Grover, A., & Leskovec, J. (2016). Node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 855–864).

Guo, J., Zhang, P., Li, C., Xie, X., Zhang, Y., & Kim, S. (2022). Evolutionary preference learning via graph nested gru ode for session-based recommendation. In *Proceedings of the 31st ACM international conference on information & knowledge management* (pp. 624–634).

Guo, X., Zhou, B., & Skiena, S. (2021). Subset node representation learning over large dynamic graphs. In *Proc. of 2021 ACM SIGKDD int. conf. on knowledge discovery and data mining* (pp. 516–523).

Hamilton, W., Ying, Z., & Leskovec, J. (2017a). Inductive representation learning on large graphs. (pp. 1024–1234).

Hamilton, W. L., Ying, R., & Leskovec, J. (2017b). Representation learning on graphs: Methods and applications. arXiv preprint arXiv:1709.05584.

Huang, Z., Sun, Y., & Wang, W. (2021). Coupled graph ode for learning interacting system dynamics. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining* (pp. 705–715).

Jiang, L., Chen, K.-J., & Chen, J. (2021). Self-supervised dynamic graph representation learning via temporal subgraph contrast. arXiv preprint arXiv:2112.08733.

Jiao, P., Guo, X., Jing, X., He, D., Wu, H., Pan, S., Gong, M., & Wang, W. (2021). Temporal network embedding for link prediction via vae joint attention mechanism. *IEEE Transactions on Neural Networks and Learning Systems*.

Kazemi, S. M., Goel, R., Jain, K., Kobyzev, I., Sethi, A., Forsyth, P., & Poupart, P. (2020). Representation learning for dynamic graphs: A survey. *Journal of Machine Learning Research*, *21*(70), 1–73.

Kipf, T. N., & Welling, M. (2016). Variational graph auto-encoders. arXiv preprint arXiv:1611.07308.

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *International conference on learning representations*.

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

Li, M. M., Huang, K., & Zitnik, M. (2022). Graph representation learning in biomedicine and healthcare. *Nature Biomedical Engineering*, 1–17.

Liu, Y., Ma, J., & Li, P. (2021). Neural higher-order pattern (motif) prediction in temporal networks. arXiv preprint arXiv:2106.06039.

Liu, M., Tu, Z., Xu, X., & Wang, Z. (2021). Learning representation over dynamic graph using aggregation-diffusion mechanism. arXiv preprint arXiv:2106.01678.

Liu, J., Xu, C., Yin, C., Wu, W., & Song, Y. (2020). K-core based temporal graph convolutional network for dynamic graphs. *IEEE Transactions on Knowledge and Data Engineering*, *34*(8), 3841–3853.

Lu, Y., Wang, X., Shi, C., Yu, P. S., & Ye, Y. (2019). Temporal network embedding with micro-and macro-dynamics. In *Proceedings of the 28th ACM international conference on information and knowledge management* (pp. 469–478).

Nguyen, G. H., Lee, J. B., Rossi, R. A., Ahmed, N. K., Koh, E., & Kim, S. (2018). Continuous-time dynamic network embeddings. In *Companion proceedings of the the web conference 2018* (pp. 969–976).

Oono, K., & Suzuki, T. (2020). Graph neural networks exponentially lose expressive power for node classification. In *International conference on learning representations*.

Paranjape, A., Benson, A. R., & Leskovec, J. (2017). Motifs in temporal networks. In *Proceedings of the tenth ACM international conference on web search and data mining* (pp. 601–610).

Pareja, A., Domeniconi, G., Chen, J., Ma, T., Suzumura, T., Kanezashi, H., Kaler, T., Schardl, T. B., & Leiserson, C. E. (2020). EvolveGCN: Evolving graph convolutional networks for dynamic graphs. In *AAAI* (pp. 5363–5370).

Perozzi, B., Al-Rfou, R., & Skiena, S. (2014). Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 701–710).

Qin, Y., Ju, W., Wu, H., Luo, X., & Zhang, M. (2023). Learning graph ODE for continuous-time sequential recommendation. arXiv preprint arXiv:2304.07042.

Qu, L., Zhu, H., Duan, Q., & Shi, Y. (2020). Continuous-time link prediction via temporal dependent graph neural network. In *Proceedings of the web conference 2020* (pp. 3026–3032).

Rossi, E., Chamberlain, B., Frasca, F., Eynard, D., Monti, F., & Bronstein, M. (2020). Temporal graph networks for deep learning on dynamic graphs. arXiv preprint arXiv:2006.10637.

Sankar, A., Wu, Y., Gou, L., Zhang, W., & Yang, H. (2020). DySAT: Deep neural representation learning on dynamic graphs via self-attention networks. In *Proceedings of the 13th international conference on web search and data mining* (pp. 519–527).

Spasov, S., Di Stefano, A., Liò, P., & Tang, J. (2020). GRADE: Graph dynamic embedding. arXiv preprint arXiv:2007.08060.

Su, X., You, Z.-H., Huang, D.-s., Wang, L., Wong, L., Ji, B., & Zhao, B. (2022). Biomedical knowledge graph embedding with capsule network for multi-label drug-drug interaction prediction. *IEEE Transactions on Knowledge and Data Engineering*, 56–66.

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., & Mei, Q. (2015). Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web* (pp. 1067–1077).

Tian, S., Wu, R., Shi, L., Zhu, L., & Xiong, T. (2021). Self-supervised representation learning on dynamic graphs. In *Proceedings of the 30th ACM international conference on information & knowledge management* (pp. 1814–1823).

Tian, S., Xiong, T., & Shi, L. (2021). Streaming dynamic graph neural networks for continuous-time temporal graph modeling. In *2021 IEEE international conference on data mining* (pp. 1361–1366). IEEE.

Trivedi, R., Farajtabar, M., Biswal, P., & Zha, H. (2019). Dyrep: Learning representations over dynamic graphs. In *International conference on learning representations*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998–6008).

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph attention networks. In *International conference on learning representations*.

Wang, L., Chang, X., Li, S., Chu, Y., Li, H., Zhang, W., He, X., Song, L., Zhou, J., & Yang, H. (2021). TCL: Transformer-based dynamic graph modelling via contrastive learning. arXiv preprint arXiv:2105.07944.

Wang, Y., Chang, Y.-Y., Liu, Y., Leskovec, J., & Li, P. (2021). Inductive representation learning in temporal networks via causal anonymous walks. arXiv preprint arXiv:2101.05974.

Wang, Q., Huang, K., Chandak, P., Zitnik, M., & Gehlenborg, N. (2022). Extending the nested model for user-centric XAI: a design study on GNN-based drug repurposing. *IEEE Transactions on Visualization and Computer Graphics*, *29*(1), 1266–1276.

Wen, W., Wang, W., Hao, Z., & Cai, R. (2023). Factorizing time-heterogeneous Markov transition for temporal recommendation. *Neural Networks*, *159*, 84–96.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Philip, S. Y. (2020). A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, *32*(1), 4–24.

Xhonneux, L.-P., Qu, M., & Tang, J. (2020). Continuous graph neural networks. In *International conference on machine learning* (pp. 10432–10441). PMLR.

Xu, D., Liang, J., Cheng, W., Wei, H., Chen, H., & Zhang, X. (2021). Transformer-style relational reasoning with dynamic memory updating for temporal network modeling. In *Proceedings of the AAAI conference on artificial intelligence, vol. 35, no. 5* (pp. 4546–4554).

Xu, D., Ruan, C., Korpeoglu, E., Kumar, S., & Achan, K. (2019). Self-attention with functional time representation learning. *Advances in Neural Information Processing Systems*, *32*.

Xu, D., Ruan, C., Körpeoglu, E., Kumar, S., & Achan, K. (2020). Inductive representation learning on temporal graphs. In *International conference on learning representations*.

Yang, C., Wang, C., Lu, Y., Gong, X., Shi, C., Wang, W., & Zhang, X. (2022). Few-shot link prediction in dynamic networks. In *Proceedings of the fifteenth ACM international conference on web search and data mining* (pp. 1245–1255).

Yang, M., Zhou, M., Kalander, M., Huang, Z., & King, I. (2021). Discrete-time temporal network embedding via implicit hierarchical learning in hyperbolic space. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining* (pp. 1975–1985).

Zang, C., & Wang, F. (2020). Neural dynamics on complex networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 892–902).

Zhang, Z., Bu, J., Li, Z., Yao, C., Can, W., & Wu, J. (2021). TigeCMN: On exploration of temporal interaction graph embedding via coupled memory neural networks. *Neural Networks*, *140*, 13–26.

Zhang, C., Xue, S., Li, J., Wu, J., Du, B., Liu, D., & Chang, J. (2023). Multi-aspect enhanced graph neural networks for recommendation. *Neural Networks*, *157*, 90–102.

Zhou, D., Zheng, L., Han, J., & He, J. (2020). A data-driven graph generative model for temporal interaction networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 401–411).

Zhu, L., Guo, D., Yin, J., Ver Steeg, G., & Galstyan, A. (2016). Scalable temporal latent space inference for link prediction in dynamic social networks. *IEEE Transactions on Knowledge and Data Engineering*, *28*(10), 2765–2777.