Hawkes Point Process-enhanced Dynamic Graph Neural Network

Zhiqiang Wang
Shanxi Taihang Laboratory,
School of Computer and Information Technology,
Shanxi University
Taiyuan, Shanxi, China
wangzq@sxu.edu.cn

Baijing Hu
Shanxi Taihang Laboratory,
School of Computer and Information Technology,
Shanxi University
Taiyuan, Shanxi, China
202222409014@email.sxu.edu.cn

Kaixuan Yao*
Shanxi Taihang Laboratory,
School of Computer and Information Technology,
Shanxi University
Taiyuan, Shanxi, China
ykx@sxu.edu.cn

Jiye Liang
Shanxi Taihang Laboratory,
School of Computer and Information Technology,
Shanxi University
Taiyuan, Shanxi, China
ljy@sxu.edu.cn

Abstract

Dynamic graph representation learning aims to capture the evolution of graph structures and obtain accurate node embeddings, a crucial task in graph machine learning. The Hawkes point process, a mathematical framework effective for modeling the influence of historical events on future occurrences, has been validated as a powerful tool for capturing the dynamics of graph evolution in dynamic graph representation learning. However, existing dynamic graph representation learning methods based on the Hawkes point process primarily model excitation at the individual node level, failing to adequately account for structural influences during graph evolution. This limitation restricts their ability to comprehensively capture network evolution patterns. To address this limitation, we propose a Hawkes Point Process-enhanced Dynamic Graph Neural Network (HP-DGNN) model. This model leverages the Hawkes point process to model both individual node histories and structural histories, capturing their respective influences on future node interactions. By integrating individual and structural influences in computing Hawkes conditional intensity, the model comprehensively captures the impacts of both layers on future node interactions. We evaluate our proposed model on two downstream tasks of dynamic graph representation learning: dynamic link prediction and future node degree prediction. Compared to 12 state-of-the-art methods, our model consistently demonstrates superior performance, underscoring its effectiveness in capturing the complexities of graph evolution.

CCS Concepts

• **Computing methodologies** → **Neural networks**.

*Kaixuan Yao is the corresponding author.

WSDM '25, March 10–14, 2025, Hannover, Germany

ACM ISBN 979-8-4007-1329-3/25/03
https://doi.org/10.1145/3701551.3703520

Keywords

Graph Neural Network; Graph Representation; Dynamic Graph; Hawkes Point Process

ACM Reference Format:
Zhiqiang Wang, Baijing Hu, Kaixuan Yao, and Jiye Liang. 2025. Hawkes Point Process-enhanced Dynamic Graph Neural Network. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining (WSDM '25), March 10–14, 2025, Hannover, Germany*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3701551.3703520

1 Introduction

In the real world, networks like social, e-commerce, and transportation networks are complex and dynamic, with connections between nodes evolving over time, leading to changes in both network topology and node structures [25]. Effective dynamic graph representations are key to capturing temporal changes [32], essential for predicting links, node behavior, and graph evolution, while also enhancing recommendation precision in social networks [13], e-commerce, and optimizing traffic systems.

Currently, two important approaches to dynamic graph representation learning include discrete and continuous methods [12]. Discrete methods convert dynamic graphs into snapshots at fixed intervals, applying static graph representation learning techniques at each timestamp. For example, TDGE [37] and CNR [23] use matrix decomposition to obtain node representations, while GAE [14] and VGAE [14] utilize encoding-decoding frameworks. Techniques like DeepWalk [27] and Node2vec [7] generate node sequences through random walks for representations, with DRS-SpikeNet [42] and SpikeNet [19] focusing on efficient network evolution detection. However, these snapshot-based approaches often lead to information loss and fail to capture node states in real-time, prompting a shift toward continuous representation learning [38].

Continuous dynamic graph representation learning uses event sequences to capture structural and attribute changes in the network at any time, resulting in finer-grained node representations [34]. For instance, CTDNE [9] aggregates node walk sequences to obtain representation, while GraphSAGE+T [8] and TGAT [35] leverage Graph Neural Networks (GNNs) to aggregate neighborhood information for representation. Furthermore, approaches like

xGCN [29] and TGNs [28] combine the strengths of GNNs and Recurrent Neural Networks (RNNs) to effectively capture temporal dynamics. In contrast, methods like DGCN [6] employ an enhanced Graph Convolutional Networks (GCNs), and [43] employs the Dual Feature Interaction-based GCNs to capture node features and obtain node embeddings. Self-supervised graph contrastive learning methods, such as DySubc [3] and MNCS [17], are also used to obtain representations. There are some methods that DyGFormer [40] and S2GAE [31] use the encoder-decoder approach to obtain dynamic representations. However, robust node representations need to incorporate not only immediate states but also historical information, as nodes' historical interaction events reveal behavioral patterns that significantly impact future evolution.

To better capture the dynamics of interactions between nodes, some models introduce the Hawkes point process [30] to model the temporal evolution of node interactions. For example,the HTNE [44] and TREND [33] models acquire node representations through the excitation effects of historical events, simulating the "chain reaction" effect in the evolution of dynamic networks. However, existing Hawkes point process-based models often focus on node-level analysis, neglecting the critical role of local topology in predicting future behavior. In dynamic networks, changes in local structures like ego networks—formed by users and their connections, customers and their purchases, or vehicles and traffic signals—profoundly impact node interactions.

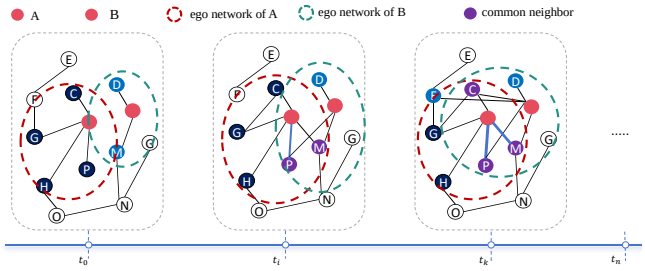


Figure 1: Dynamic evolution of user ego networks in a social network.

Figure 1 illustrates the evolution of collaboration in a co-author network, where edges represent relationships. Researchers A and B, initially unconnected, gradually form a link through the integration of their local networks. As A collaborates more with B's partners P and M, A's network evolves, leading to future interactions between A and B. The evolution underscores the significant influence of topological changes in predicting node behavior.

To efficiently model the critical role of historical topology in network evolution and enhance dynamic node representations, we propose a Hawkes Point Process-enhanced dynamic graph neural network (HP-DGNN). This model captures the excitation effects of both individual and structural histories on future interactions by integrating their influences within the Hawkes point process, offering a comprehensive approach to predicting future node behavior. The main contributions include:

- We propose a Hawkes Point Process-enhanced dynamic graph neural network model that can simultaneously model the excitation effects of individual and structural historical behaviors on future behaviors.
- We design a method to compute Hawkes point process excitation, with adaptive fusion parameters, to effectively embed this mechanism within the graph neural network framework by integrating individual and structural influences.
- Extensive experiments demonstrate that the proposed model surpasses state-of-the-art methods in dynamic graph representation tasks, confirming its effectiveness in modeling graph evolution.

2 Related Work

Dynamic graph representation learning methods can be broadly categorized into two main approaches: discrete representation learning methods and continuous representation learning methods [39].

Discrete representation learning methods divide dynamic graphs into multiple fixed time interval snapshots, and then perform representation learning on each snapshot independently. These methods typically use techniques such as matrix factorization, random walk, and encoder-decoder architectures to combine the representation of snapshots into a comprehensive representation of the dynamic graph [1]. For example, the CNR [23] obtains node representations by decomposing the adjacency matrix of static graph snapshots and incorporating temporal information. Models like, GAE [14], and VGAE [14] utilize an encoder-decoder framework to encode the topology and attributes of the network, respectively, and reconstruct the original graph through a decoder. VGAE [14] introduces variational inference, representing node representations as probability distributions, from which node representation are sampled. Additionally, random walk-based methods like DeepWalk [27] and Node2Vec [7] generate node sequences by simulating random walks and use techniques like skip-gram to obtain node representations. Node2Vec [7] enhances the flexibility of random walks by introducing a biased sampling strategy. To improve computational efficiency, SpikeNet [19] replaces traditional recurrent neural networks with spiking neural networks to obtain efficient node representations. Despite the simplicity and effectiveness of discrete representation methods, they have notable drawbacks. First, determining the optimal time interval for the dynamic graph is crucial—intervals that are too fine increase computational overhead, while coarse intervals result in information loss. Second, static snapshot aggregation can only reflect dynamic graph changes at a coarse level, making it difficult to capture continuous temporal evolution. Furthermore, these methods can only obtain node representations at fixed time points, preventing dynamic adjustments at any arbitrary moment [18].

Continuous representation learning methods model directly on the original dynamic graph, preserving the temporal continuity of the network, thereby obtaining precise node representations[4]. CTDNE [9] uses temporal random walks to simulate the optimized framework of DeepWalk [27] within a time window to obtain node representations. Methods like GraphSAGE+T [8] and TGAT [35] leverage GNNs to aggregate neighborhood information, with TGAT [35] introducing temporal information to assign different weights to different neighbors [41]. EvolveGCN [25] combines the advantages of GCNs and RNNs, extracting node features and dynamically adjusting model parameters. TGNs [28] use a memory module to store long-term features of nodes, capturing dynamic behaviors

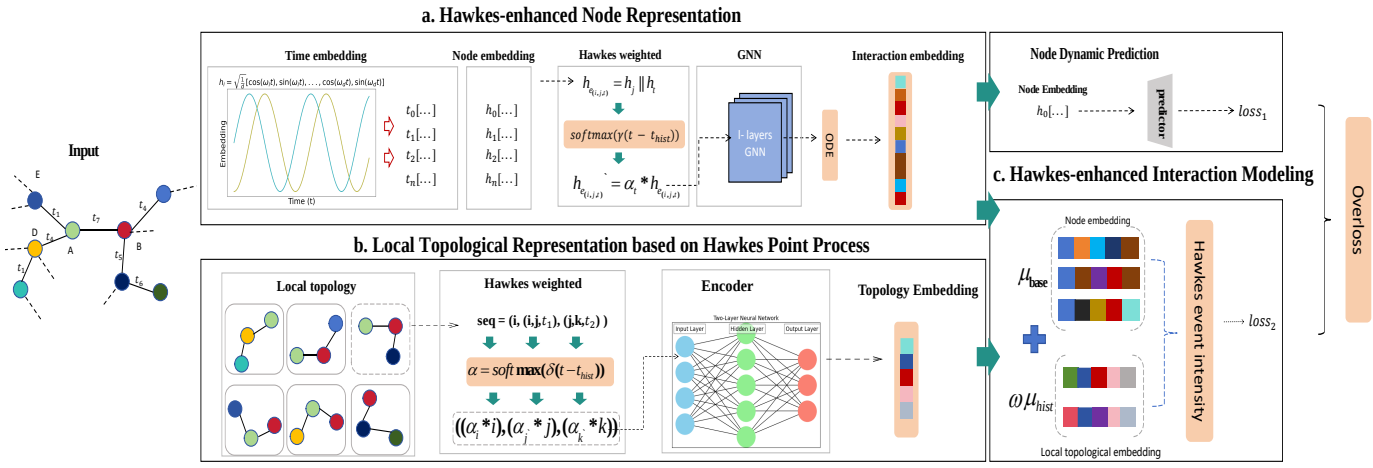


Figure 2: The architecture of the Hawkes Point Process-enhanced Dynamic Graph Neural Network.

in dynamic graphs. Additionally, there are some graph contrastive learning models, the SUGRL [22]: a new graph contrastive representation learning method. Although deep learning models perform well in modeling continuous evolution in dynamic graphs, they often overlook dependencies between interactions. Whether there is an incentive effect among interactions and its decay over time is a topic worthy of in-depth study. Temporal point processes provide a theoretical foundation for modeling such dependencies. In models such as HTNE [44], MMDNE [20], and TREND [33], the Hawkes point process is employed to model the excitation effects of events. HTNE [44] utilizes the Hawkes point process to update the weights of neighboring nodes, while MMDNE [20] applies it to model the occurrence of edge events, capturing the micro-dynamics in network evolution. TREND [33] combines the Hawkes point process to capture both individual and dynamic features of links. However, existing Hawkes point process-based models primarily focus on node-level analysis, often neglecting the substantial impact of local topology on predicting future behaviors.

3 HP-DGNN

3.1 Overview of HP-DGNN

The architecture of HP-DGNN, illustrated in Figure 2, leverages the Hawkes point process to model dynamic graph, which includes three primary modules: (1) Hawkes-enhanced Node Embedding, (2) Local topological Embedding based on Hawkes point process, and (3) Hawkes-enhanced Interaction Modeling.

In Fig. 2(a), the node features are integrated with low-dimensional time embeddings to construct the embeddings of historical node interactions. Node embeddings are generated by applying the Hawkes matrix, followed by multi-layer temporal GNN aggregation [10]. Ordinary Differential Equations (ODE) are subsequently solved to produce node embedding vectors at arbitrary timestamps [26], which are then fed into the base intensity module of the conditional probability intensity in the Hawkes point process.

In Fig. 2(b), temporal random walks are employed to generate node sequences for each node. These sequences are initially weighted according to the interaction times via the Hawkes point process, after which the local structural embeddings of the nodes are derived using a positional encoder. These embeddings are input into the historical excitation intensity module of the conditional probability intensity in the Hawkes point process.

In Fig. 2(c) demonstrates the integration of local topological embeddings (derived in Fig. 2(b)) with node features (from Fig. 2(a)) to construct the conditional probability intensity in the Hawkes point process. This model encodes node sequences to establish the conditional probability intensity, thereby modeling the influence of local structural evolution around nodes on their future interactions.

The overall loss function of HP-DGNN is composed of two parts: node dynamic loss and interaction loss. The node dynamic loss is calculated by predicting node dynamics using the node embeddings obtained in Fig. 2(a) via a node dynamic predictor. The interaction loss is derived based on the interaction probability intensity constructed in Fig. 2(c), capturing the impact of historical and local structural changes on node interactions.

3.2 Hawkes-enhanced Node Embeddings

Historical Interaction Embedding. In a dynamic network, a node's state is shaped not only by its intrinsic attributes but also by its temporal interactions with other nodes. These historical interactions encode crucial information that significantly influences both the node's current and future states. To capture these temporal dependencies more effectively, we aggregate the historical interaction data of nodes rather than merely relying on standard GNN aggregation of neighboring node information.

To enrich the historical interaction embeddings, we incorporate temporal embeddings with the node embeddings. For the interaction generated by node i at time t, the edge $e_{(i,j,t)} \in E$ is considered. The timestamp t generates a d-dimensional temporal embedding h_t using a temporal encoding function, which is then concatenated with the embedding h_j of node j to construct the edge embedding $h_{e_{(i,j,t)}}$:

$$h_t = \sqrt{\frac{1}{d}} \left[\cos(\omega_1 t), \sin(\omega_1 t), \ldots, \cos(\omega_d t), \sin(\omega_d t)\right], \quad (1)$$
$$h_{e_{(i,j,t)}} = h_j \parallel h_t.$$

This design effectively encodes temporal dynamics, allowing the model to retain key historical interaction details that are crucial for accurately modeling the temporal evolution of a node's state.

Hawkes Weight Coefficient. Temporal interactions exert varying degrees of influence on a node's state, with the impact naturally attenuating over time. To model this decay, we introduce a softmax function to compute the Hawkes weight coefficient at the timestamp t, which quantifies the temporal influence of past interactions.

$$\alpha_t = \text{softmax}(\gamma(t - t_{\text{hist}})), \tag{2}$$

where γ is a learnable parameter, and t_{hist} denotes the timestamp of the historical interaction events. The **softmax** normalization ensures that the resulting time weight coefficient α_t accurately reflects the relative influence of historica interactions, thereby capturing the temporal decay in a principled manner.

Hawkes-weighted Historical Interaction Embedding. The final historical interaction embeddings for the node is computed as:

$$h^{Hawkes}_{e_{(i,j,t)}} = \alpha_t \cdot h_{e_{(i,j,t)}}, \tag{3}$$

where α_t serves as the Hawkes weight coefficient at the timestamp t, modulating the influence of the historical interaction $h_{e_{(i,j,t)}}$ based on the temporal context. This weighted embeddings effectively captures the decaying significance of historical interactions, ensuring a more accurate modeling of temporal dynamics in node embeddings.

DGNN based on the Hawkes Point Process. The two-layers Dynamic Graph Neural Network (DGNN) leverages Hawkes-weighted historical interaction information to compute the node embeddings $h^l_t(i)$:

$$h^l_t(i) = \delta\left(h_{e_{(i,j,t)}} W^l_{\text{self}} + \alpha_t \left(\sum_{(i,j,t)\in E} h_{e_{(i,j,t)}} W^{l-1}_{\text{hist}}\right)\right), \tag{4}$$

where $h_{e_{(i,j,t)}}$ is the d-dimensional historical interaction embeddings, W^l_{self} and W^{l-1}_{hist} are trainable parameters of the neural network, l indicates which layer of the neural network, and α_t is the temporal weighting factor of the t.

Given that node interactions may remain stable over certain time intervals—resulting in low discriminability when relying solely on GNN-derived node embeddings—particularly in regions where node interactions exhibit minimal change, we introduce an ODE solver [11] to refine node embeddings across different timestamps, where the ODE is differentiable:

$$\hat{h} = \text{ODE}(f(\cdot), h, [t_0, t_1]), \tag{5}$$

here, $f(\cdot)$ represents a known differential equation. For this study, we use:

$$\frac{dy}{dt} = -2y + \sin t, \tag{6}$$

where y denotes the initial node embeddings, t represents time, and y is the dependent variable. $\hat{h}$ is the node embeddings obtained at time t_1 after processing through the ODE solver. The initial node embeddings h is derived from GNN aggregation at time t_0, and $[t_0, t_1]$ indicates the time interval over which the computation is performed.

3.3 Local Topological Embedding based on Hawkes Point Process

This section presents the derivation of local topological embeddings for nodes, which is accomplished through a time-biased random walk over neighboring nodes, followed by encoding the sampled sequences and aggregating them via attention mechanisms to obtain the final node embeddings.

Time-biased Random Walk. For a node $i \in V$ involved in an interaction event $e = (i, j, t) \in E$, where $N_t(i)$ and $N_t(j)$ denote the neighborhoods of nodes i and j at time t, a time-biased random walk is conducted over these neighborhoods as formulated below [11] :

$$\text{Pr}_t(a) = \frac{\exp(\alpha(t_a - t))}{\sum_{a'\in N_t(i)} \exp(\alpha(t_{a'} - t))}, \tag{7}$$

$$\text{Pr}_s(a) = \frac{\exp(-\beta/d_a)}{\sum_{a'\in N_t(j)} \exp(-\beta/d_{a'})}, \tag{8}$$

here, α and β are hyperparameters, t_a represents the timestamp of node a's interaction with its neighbors, and d_a indicates the degree of node a at time t.

Node Sampling Sequence [27]. Let $\text{Seq}_t(i) = \{\text{seq}_1, \text{seq}_2, \ldots, \text{seq}_m\}$ and $\text{Seq}_t(j) = \{\text{seq}_1, \text{seq}_2, \ldots, \text{seq}_m\}$ represent the sampled sequences for nodes i and j, respectively. Each sequence element $\text{seq}_n = \{(i, j, k)\}$ is such that $j, k \in V$ and $n < m$, ensuring that the interaction between (i, j) occurs after that between (j, k).

Hawkes Weighting Coefficient. The influence of interactions varies with time, and this temporal impact is quantified using a Hawkes weighting coefficient computed as follows:

$$\alpha_t = \text{softmax}(\delta(t - t_{\text{hist}})), \tag{9}$$

where δ is a learnable parameter, and t_{hist} is the timestamp of historical interactions.

Sequence Encoding and Weighting. For each sampled sequence $\text{seq}_n = \{(i, j, k), \ldots\}$, the sequence embedding is computed by:

$$h^t_{\text{seq}_n} = \text{linear}(\alpha^t_i * h^l_t(i), \alpha^t_j * h^l_t(j), \alpha^t_k * h^l_t(k)), \tag{10}$$

where α^t_i, α^t_j, and α^t_k are the Hawkes weights corresponding to nodes i, j, and k at time t, and $h^l_t(i)$, $h^l_t(j)$, and $h^l_t(k)$ are their respective embeddings at time t.

Node Local Topological Representation. The sequence embeddings are then aggregated using a neural network to produce the final local topological embeddings:

$$S_t(i) = f(h^{(t_1)}_{\text{seq}_1}, \ldots, h^{(t_n)}_{\text{seq}_n}, \ldots, h^{(t_m)}_{\text{seq}_m}), \quad t_1 < t_n < t_m < t \tag{11}$$

where $S_t(i)$ and $S_t(j)$ represent the temporal random walk encodings for nodes i and j, respectively, serving as their local topological embeddings.

3.4 Hawkes-Enhanced Interaction Modeling

In this section, we detail the construction of the conditional intensity function for interaction occurrences, which is subsequently used as the probability of interaction. We also introduce the full loss function designed to optimize the model.

Conditional Probability Intensity. The conditional probability intensity in the Hawkes process quantifies the probability of an interaction occurring:

$$\lambda(t) = \mu + \sum_i \omega(t - t_i), \tag{12}$$

where μ is the base intensity of interaction occurrence, and $\sum_i$ captures the impact of historical interactions occurring at times t_i before t. The function $\omega(t - t_i)$ models the influence of each historical interaction, expressed as:

$$\omega(t - t_i) = \alpha \exp(-\beta(t - t_i)), \tag{13}$$

here, α denotes the weight of each interaction's influence, while β represents the decay rate, illustrating that the influence of past interactions diminishes over time.

Hawkes point Process for Dynamic Graphs. In dynamic graphs, the interaction probability p between nodes i and j at time t is influenced by both the base intensity $\mu_{base}^{(i,j)}$ and the historical interaction influence $\mu_{hist}^{(i,j)}$ prior to t [36]:

$$\lambda_{(i,j)}(t) = \mu_{base}^{(i,j)} + \omega_h^t \mu_{hist}^{(i,j)}, \tag{14}$$

where ω_h^t adjusts the weight between historical influence and base intensity.

Base Intensity.

$$\mu_{base}^{(i,j)} = \text{sigmoid}\left(FCL_e\left(\left(h_t^l(i), h_t^l(j)\right)^2, \theta_e\right)\right), \tag{15}$$

here, $h_t^l(i)$ and $h_t^l(j)$ are the feature vectors of nodes i and j at time t. The fully connected layer FCL_e projects the difference of these feature vectors into the interaction intensity, represented by the base intensity. The trainable parameters of the layer are denoted by θ_e.

Historical Incentive Effect. The influence of historical interactions is modulated by their temporal incentive effect. We calculate the historical incentive by weighting different historical times based on the dissimilarity between local embeddings:

$$\tau_t = \text{softmax}(-\omega(t - t')), \tag{16}$$

where $t - t'$ represents the time difference between the current and historical interactions, and ω is a hyperparameter controlling the temporal decay. The local embeddings of the node is then weighted as (11).

The historical incentive effect is then computed as:

$$\mu_{hist}^{(i,j)} = \text{sigmoid}(FCL_e((S_t(i), S_t(j))^2, \theta_s)), \tag{17}$$

where $S_t(i)$ and $S_t(j)$ are the local structural embeddings of nodes i and j at time t, respectively. The fully connected layer FCL_e and parameter vector θ_s are used to compute the historical incentive.

3.5 Loss Function

In this section, we will introduce the composition of the model's loss function.

Node Dynamic Loss. By inputting the node embeddings vectors into the node dynamic estimator, the loss $loss_d$ is obtained [33].

$$\text{loss}_d = f(h_t^l(i)), \tag{18}$$

where $h_t^l(i)$ represents the node embeddings.

Interaction Loss. For a time series sample G, the Hawkes loss function is defined as:

$$\text{loss}_\theta = -\log(\lambda_{(i,j)}(t)) - Q\mathbb{E}_{k \in P_n} \log(1 - \lambda_{(i,o)}(t)), \tag{19}$$

where $\theta = (\alpha, \beta, \theta_t, \theta_s, \omega, \omega_h)$ represents the model parameters, Q is the number of negative samples, and p_n is the node degree distribution, defined as $p_n \propto \deg(v)^{\frac{3}{4}}$, to ensure robust performance across varying node degrees.

Overall Loss. The global loss consists of the dynamic loss of nodes and the interaction loss. The μ represents the proportion of dynamic node loss, which is artificially set.

$$\text{loss} = \text{loss}_\theta + \mu \text{loss}_d. \tag{20}$$

In summary, by calculating the conditional intensity of interactions using both node representations and expanded embeddings, the model minimizes loss and optimizes its parameters. The training process for this model is detailed in Algorithm 1.

Algorithm 1: The training process for HP-DGNN

Input: A temporal graph $G : G = \{V, E, T, X\}$; feature dimensions d; the learning rate η; sequence coding method f and encoding step size ω

Output: Optimized parameters $\theta = (\alpha, \beta, \theta_t, \theta_s, \omega, \omega_h)$

Initialize: Set Spatial-bias = 0.5 and Time-bias = 10^{-6}; the weight of node degree loss: nocedf (different datasets have different settings);

for *each* $c_i = (i, j, t) \in E$ **do**
  for *each* $p_i \in N_i$ *and* $p_j \in N_j$ **do**
    Select Node v_m ; // Eq. (7) and (8)
    Append v_m to $Seq(i)$; // Append to sequence
    Append v_m to $Seq(j)$
    $S_t(i) = f(Seq(i), Seq(j))$; // Eq. (11)
    Get the embeddings of nodes i and j;
  $\lambda_{(i,j)}(t) = f_\theta(h_t^l(i), h_t^l(j), S_t(i), S_t(j))$; // The event intensity of $c_i = (i, j, t) \in E$
  $\text{loss}_\theta = -\log(\lambda_{(i,j)}(t)) - Q\mathbb{E}(\log(1 - \lambda_{(i,o)}(t)))$; // Eq. (19)
  loss_d ; // Eq. (18)
  The overall loss: $loss$; // Eq. (20)
  Update $\theta = (\alpha, \beta, \theta_t, \theta_s, \omega, \omega_h)$

4 Experiments and Results

4.1 Experimental Setup

Datasets. We evaluated the generalization capability of our proposed model on three real-world datasets, as summarized in Table 1. The "Proportion of New Nodes" indicates the percentage of testing events involving nodes that were not observed during the training phase.

Table 1: Datasets Overview

Dataset	CollegeMsg	cit-Hepth	Wikipedia
Edges	59,835	21,315	157,474
Nodes	1,899	7,577	8,227
Node Features	–	128	172
Multiple Edges	Yes	No	Yes
Proportion of New Nodes	22.79%	100%	7.26%

- **CollegeMsg [24]:** This dataset records private message exchanges between users within an online social network. Each event represents a message sent between two users. Users could search for others on the platform and send messages based on their profile information. The dataset includes edge timestamps, with nodes indexed starting from zero.
- **cit-Hepth [16]:** This dataset contains citation relationships in the domain of high-energy physics, specifically from the arXiv HEP-TH collection. Each edge represents a citation between two papers. The paper content was embedded into a feature space using word2vec [21].
- **Wikipedia [15]:** This dataset tracks user edit histories on Wikipedia pages. Each event denotes a user's edit on a Wikipedia page within a specific month. The text features are represented as 172-dimensional LIWC [5] features, while edit frequency serves as a measure of node activity.

Comparison Methods. We benchmark our proposed method against the following state-of-the-art models:

- **DeepWalk [27]:** An embedding method utilizing random walks combined with the Skip-gram model to derive node representations from sequences of nodes.
- **Node2vec [7]:** An extension of DeepWalk that introduces flexible parameters to guide the random walk strategy, effectively capturing a diverse range of node features.
- **VGAE [14]:** A variational graph autoencoder that represents graph structures probabilistically, learning latent node representations through a variational approach.
- **GAE [14]:** A GCN method that encodes node features into low-dimensional embeddings using an encoder-decoder architecture.
- **GraphSAGE [8]:** A framework that samples and aggregates neighborhood information to produce embeddings for target nodes, designed to generalize across different graphs.
- **CTDNE [9]:** Employs a time-aware random walk approach to capture the dynamic characteristics of networks by incorporating temporal contexts.
- **EvolveGCN [25]:** A dynamic GCN model that utilizes RNN to update GCN parameters, adapting to temporal changes within the graph.
- **GraphSAGE+T [8]:** Enhances the GraphSAGE framework by integrating temporal information, enabling the model to better capture time-evolving dynamics.
- **TGAT [35]:** A temporal graph attention network that uses time encoding and attention mechanisms to model time-dependent relationships in dynamic graphs.
- **HTNE [44]:** A Hawkes process-based model that quantifies the influence of temporal neighborhood sequences on nodes, effectively modeling historical interactions through event excitation.
- **MMDNE [20]:** An embedding approach that combines macro and micro perspectives to dynamically update node representations, capturing temporal structural changes.
- **TREND [33]:** Integrates the Hawkes process with graph neural networks to model dynamic interactions and excitation features between nodes.

Downstream Tasks. We evaluate our proposed method using two downstream tasks in dynamic graph representation learning: Dynamic Link Prediction and Node Degree Prediction.

- **Dynamic Link Prediction** [33]: This task is to predict whether a link will form in the future. Given a dataset, divide it according to a specific timestamp. Interactions before time t are the training set t^{train}, and interactions after time t^{train} are the test set. Given an edge (i, j, t) where $t > t^{\text{train}}$, use HP-DGNN to generate node representations, and input these node representations into a downstream logistic classifier to determine whether it is positive or negative, in order to judge whether two nodes will form a link. The task's performance is assessed using Accuracy and F1-score as evaluation metrics.
- **Node Degree Prediction** [33]: The mean absolute error(MAE) between the number of new connections generated at future time points and the predicted values. Compare the number of predicted new connections obtained by inputting the node representations generated by the model into the node dynamic predictor with the actual number of connections, and calculate the MAE.

4.2 Parameter Settings

For the CollegeMsg and cit-Hepth datasets, the model's embedding dimension is configured to 32, while for the Wiki dataset, it is set to 16. The neural network architecture comprises 3 layers. The ODE solver employs the RK-4 method [2], with both the GNN depth and the sampling depth set to 10. To ensure robustness and generalizability of the experimental results, the experiments are repeated multiple times across varying time windows, ranging from 1 to 10.

4.3 Experimental Comparison

Table 2 presents the performance comparison between the proposed HP-DGNN model and several baseline models on the link prediction task, evaluated across three datasets. The metrics used for evaluation are accuracy and F1-score, with all values expressed as percentages.

As illustrated in Table 2, the HP-DGNN model consistently demonstrates superior performance in link prediction tasks across all datasets. This performance advantage, particularly over other temporal models such as HTNE, MMDNE, and TREND, can be attributed to HP-DGNN's comprehensive modeling approach, which integrates node representations with local historical topological information. This method effectively captures the temporal evolution of a node's local topology, leading to more accurate predictions of future links.

Among dynamic graph models, GraphSAGE shows improved predictive accuracy over static models like DeepWalk and Node2Vec by incorporating temporal aspects. The HTNE model, which accounts for the excitation effects of historical interactions, also shows strong predictive capabilities for future links. MMDNE and TREND further

Table 2: Performance of Different Models on Temporal Link Prediction Task.
The table data is presented as percentages, with the best result in each column bolded and the second-best underlined.

Model	CollegeMsg		cit-HepTh		Wikipedia	
	Accuracy	F1	Accuracy	F1	Accuracy	F1
DeepWalk	66.54% ± 5.36	67.86% ± 5.86	51.5% ± 50.90	50.39% ± 0.98	65.12% ± 0.94	64.25% ± 1.32
Node2Vec	65.82% ± 4.12	69.10% ± 3.50	62.68% ± 1.90	66.13% ± 2.15	75.52% ± 0.58	75.61% ± 0.52
VGAE	65.82% ± 5.68	68.73% ± 4.49	66.79% ± 2.58	67.27% ± 2.84	66.35% ± 1.48	68.04% ± 1.18
GAE	62.54% ± 5.11	66.97% ± 3.22	69.52% ± 1.10	70.28% ± 1.33	68.70% ± 1.34	69.74% ± 1.43
GraphSAGE	58.91% ± 3.67	60.45% ± 4.22	70.72% ± 1.96	71.27% ± 2.41	72.32% ± 1.25	73.39% ± 1.25
CTDNE	62.55% ± 3.67	65.56% ± 2.34	49.42% ± 1.86	44.23% ± 3.92	60.99% ± 1.26	62.71% ± 1.49
EvolveGCN	63.27% ± 4.42	65.44% ± 4.72	61.57% ± 1.53	62.42% ± 1.54	71.20% ± 0.88	73.43% ± 0.51
GraphSAGE+T	69.09% ± 4.91	69.41% ± 5.45	67.80% ± 1.27	69.12% ± 1.12	57.93% ± 0.93	63.41% ± 0.91
TGAT	58.18% ± 4.78	57.23% ± 7.57	78.02% ± 1.93	78.52% ± 1.61	76.45% ± 0.91	76.99% ± 1.16
HTNE	73.82% ± 5.36	74.24% ± 5.36	66.70% ± 1.80	67.47% ± 1.16	77.88% ± 1.56	78.09% ± 1.40
MMDNE	73.82% ± 5.36	74.10% ± 3.70	66.28% ± 3.87	66.70% ± 3.39	79.76% ± 0.89	79.87% ± 0.95
TREND	74.55% ± 1.95	75.64% ± 2.09	80.37% ± 2.08	81.13% ± 1.92	83.75% ± 1.19	83.86% ± 1.24
HP-DGNN	**91.22%** ± 0.33	**91.1%** ± 0.33	**87.16%** ± 0.17	**87.47%** ± 1.05	**84.61%** ± 1.57	**84.64%** ± 1.48

refine dynamic network modeling by combining both macro- and micro-structural information. HP-DGNN surpasses these models by holistically integrating both local historical topology and historical interactions, utilizing the Hawkes point process to effectively track changes in node relationships. This capability allows HP-DGNN to more precisely predict shifts in interaction patterns, offering a more reliable and comprehensive tool for dynamic network analysis.

Table 3: Performance comparison on the node degree prediction task across different datasets. The results are reported as mean absolute error (MAE). The best results are highlighted in bold.

Model	CollegeMsg	cit-Hepth	Wikipedia
CTDNE	10.0360	3.0173	7.3265
EvolveGCN	3.1964	2.5610	6.8651
GraphSAGE+T	21.9444	2.2421	5.9231
TGAT	2.6903	2.8094	7.7737
HTNE	12.3587	3.2781	6.8860
MMDNE	8.0555	2.7456	6.9552
TREND	2.3549	**2.2066**	5.9140
HP-DGNN	**0.6506**	2.7972	**5.6319**

In addition, we conducted comparative experiments on the node degree prediction task, as shown in Table 3. The results indicate that HP-DGNN outperforms other baseline models on the CollegeMsg and Wikipedia datasets, though it underperforms on cit-Hepth. This discrepancy may be due to the distinct characteristics of the cit-Hepth dataset, where connections are unevenly distributed, with 21,315 edges concentrated within just 78 timestamps. In contrast, the CollegeMsg and Wikipedia datasets have more uniformly distributed edges, with 59,835 and 157,474 edges respectively, spanning a longer timeframe. This disparity in edge distribution could have influenced the model's predictive performance on cit-Hepth.

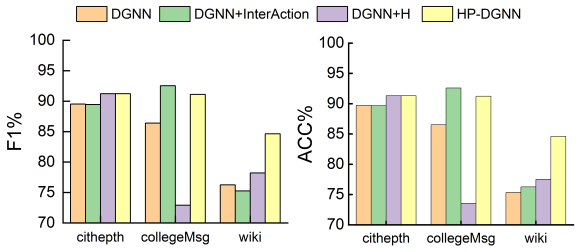
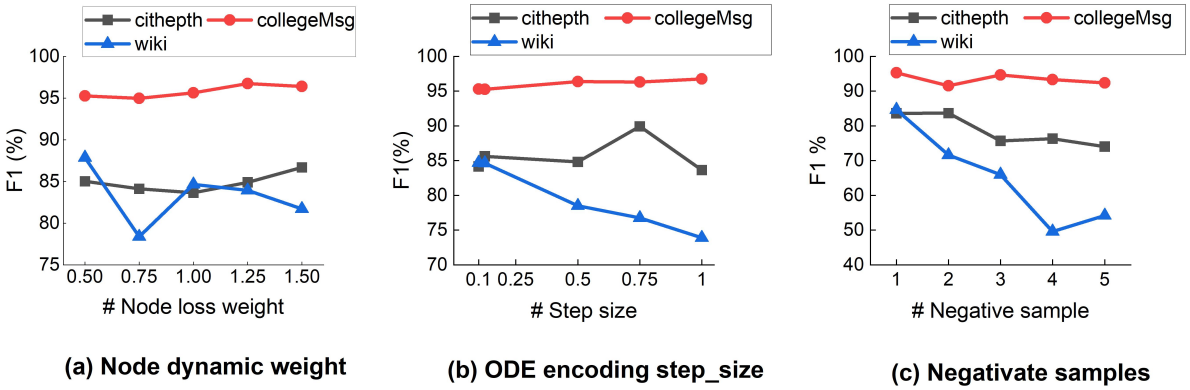


Figure 3: Ablation study results: Performance comparison across different model configurations on the cit-Hepth, CollegeMsg, and Wikipedia datasets. The full HP-DGNN model, which integrates both neighbor and historical interaction information.

4.4 Ablation Study

To evaluate the contribution of different components within our model to the temporal link prediction task, we conducted an ablation study. The study examined the performance under the following configurations: (1) Using traditional GNNs for node embedding without additional modules, (2) Integrating only the component: Hawkes-enhanced node embeddings, (3) Integrating only the component: local topological embedding based on Hawkes point process, and (4) Incorporating historical interaction information to construct the Hawkes conditional intensity function.

As illustrated in Fig. 3, the experimental results suggest that the complete HP-DGNN model generally outperforms other configurations on the CITE and Wikipedia datasets, demonstrating strong performance. On the CollegeMsg dataset, the performance of the complete HP-DGNN model is slightly lower than that of the model with Hawkes-enhanced node representations. However, after incorporating the Hawkes-enhanced node representation module, the model achieves the best performance. Further analysis indicates that, in dynamic graph datasets, the presence of a large number of timestamps tends to enhance the model's performance (GNN + InterAction), while excessive edge sparsity may lead to a decline in performance (GNN + Hawkes). Overall, the complete HP-DGNN



(a) Node dynamic weight (b) ODE encoding step_size (c) Negativate samples

Figure 4: Parameter analysis results: (a) Node dynamic weight, (b) ODE encoding step size, and (c) Number of negative samples.

model performs well across the CITE, CollegeMsg, and Wikipedia datasets. The interaction between these modules enables the model to more effectively capture patterns in node evolution, leading to improved predictive accuracy, stable performance, and competitive potential.

4.5 Parameter Analysis

We examine three critical hyperparameters in the HP-DGNN model: node dynamic weight, ODE encoding step size, and the number of negative samples. This analysis provides insights into their influence on model performance in tasks such as link prediction and node degree prediction.

Node Dynamic Weight: The node dynamic weight balances the influence of historical interactions and immediate neighbors. As shown in Fig. 4 (a), increasing this weight generally enhances performance. For cit-Hepth, a weight of 1.5 is optimal, suggesting a greater emphasis on dynamic history is beneficial. In contrast, Wikipedia achieves the best results with a weight of 0.5, indicating that less focus on history suits its temporal characteristics. CollegeMsg performs best with a balanced weight of 1.25. These results highlight the need to adjust the node dynamic weight according to the dataset's specific temporal and structural properties.

ODE Encoding Step Size: The step size controls how the model captures the evolution of node states. As shown in Fig. 4 (b), a step size of 0.75 yields the best results for cit-Hepth, while 0.5 is optimal for CollegeMsg. An appropriate step size is crucial to balance capturing temporal dynamics without losing resolution or incurring overfitting.

Number of Negative Samples: According to the experimental results. Fig. 4 (c), increasing the number of negative samples does not improve the model's performance. The model achieved consistent results across different datasets, with optimal performance observed when there was only one negative sample.

In summary, optimal settings for these hyperparameters depend on the dataset's characteristics. Proper tuning allows the HP-DGNN model to effectively capture dynamic graph structures, leading to improved performance in graph representation tasks.

5 Conclusion and Future Work

This paper introduces a novel Hawkes Point Process-Enhanced Dynamic Graph Neural Network, designed to address the limitations of existing dynamic graph representation learning methods. By leveraging the Hawkes point process, HP-DGNN effectively models the excitation effects of both individual node histories and structural histories, offering a comprehensive framework that captures the complex temporal patterns governing network evolution. Unlike previous approaches that primarily focus on node-level dynamics, our method integrates both individual and structural influences into the computation of Hawkes conditional intensity, allowing for a more holistic understanding of graph evolution. Through extensive experiments on dynamic link prediction and future node degree prediction tasks, HP-DGNN consistently outperforms 12 state-of-the-art methods, demonstrating superior accuracy, generalization, and robustness. The integration of local topological information and historical interactions enables our model to capture the intricate dependencies within dynamic graphs, highlighting its potential as a powerful tool in graph machine learning. Future work will explore the model's applicability to multi-type nodes and edges, long-range dependencies, and large-scale dynamic graphs, with the aim of further enhancing the model's applicability and efficiency.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Nos.U21A20473, 62272285, 62072293, 62406180), the Key Technologies Program of Taihang Laboratory in Shanxi Province, China (THYF-JSZX-24010600), and the UK_China Joint Laboratory of Security and Control on Smart Energy (202104041101020).

References

[1] Moran Beladev, Lior Rokach, Gilad Katz, Ido Guy, and Kira Radinsky. 2020. tdgraphembed: Temporal dynamic graph-level embedding. In *the 29th ACM International Conference on Information & Knowledge Management*. 55–64.

[2] Eric Z Chen, Terrence Chen, and Shanhui Sun. 2020. MRI image reconstruction via learning optimization using neural ODEs. In *the Medical Image Computing and Computer Assisted Intervention–MICCAI 2020: 23rd International Conference*. 83–93.

[3] Kejia Chen, Linsong Liu, Linpu Jiang, and Jingqiang Chen. 2023. Self-Supervised dynamic graph representation learning via temporal subgraph contrast. *ACM Transactions on Knowledge Discovery from Data* 18, 1 (2023), 1–20.

[4] Pingtao Duan, Xiangsheng Ren, and Yuting Liu. 2023. Multi-relational dynamic graph representation learning. *NeuroComputing* 558 (2023), 126688.

[5] ME Francis and RJ Booth. 2001. Linguistic inquiry and word count: LIWC. *Mahway Lawrence Erlbaum Associates* (2001).

[6] Chao Gao, Junyou Zhu, Fan Zhang, Zhen Wang, and Xuelong Li. 2022. A novel representation learning for dynamic graphs based on graph convolutional networks. *IEEE Transactions on Cybernetics* 53, 6 (2022), 3599–3612.

[7] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *the 22nd ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 855–864.

[8] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *the Neural Information Processing Systems*. 1025–1035.

[9] Xin Huang, Yun Li, JinYu Xiong, Jie Liang, Jie Wu, and Jie Jing. 2020. Continuous-time dynamic network node classification algorithm Based on autoencoder. In *the IEEE 9th Joint International Information Technology and Artificial Intelligence Conference*. 1714–1722.

[10] Zijie Huang, Yizhou Sun, and Wei Wang. 2021. Coupled graph ode for learning interacting system dynamics. In *the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 705–715.

[11] Ming Jin, Yuan-Fang Li, and Shirui Pan. 2022. Neural temporal walks: Motif-aware representation learning on continuous-time dynamic graphs. In *the Neural Information Processing Systems*. 19874–19886.

[12] Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. 2020. Representation learning for dynamic graphs: A survey. *Journal of Machine Learning Research* 21, 70 (2020), 1–73.

[13] Mohamed Hassen Kerkache, Lamia Sadeg-Belkacem, and Fatima Benbouzid-Si Tayeb. 2023. A hybrid approach for enhanced link prediction in social networks based on community detection. *International Journal of General Systems* 53, 2 (2023), 154–183.

[14] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* (2016).

[15] Srijan Kumar, Xikun Zhang, and Jure Leskovec. 2019. Predicting dynamic embedding trajectory in temporal interaction networks. In *the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1269–1278.

[16] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2005. Graphs over time: densification laws, shrinking diameters and possible explanations. In *the 11th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 177–187.

[17] Dong Li, Wenjun Wang, Minglai Shao, and Chen Zhao. 2023. Contrastive representation Learning based on multiple Node-centered Subgraphs. In *the 32nd ACM International Conference on Information and Knowledge Management*. 1338–1347.

[18] Hao Li, Hao Jiang, Dongsheng Ye, Qiang Wang, Liang Du, Yuanyuan Zeng, Yingxue Wang, Cheng Chen, et al. 2024. DHGAT: Hyperbolic representation learning on dynamic graphs via attention networks. *NeuroComputing* 568 (2024), 127038.

[19] Jintang Li, Zhouxin Yu, Zulun Zhu, Liang Chen, Qi Yu, Zibin Zheng, Sheng Tian, Ruofan Wu, and Changhua Meng. 2023. Scaling up dynamic graph representation learning via spiking neural networks. In *the AAAI Conference on Artificial Intelligence*. 8588–8596.

[20] Yuanfu Lu, Xiao Wang, Chuan Shi, Philip S Yu, and Yanfang Ye. 2019. Temporal network embedding with micro-and macro-dynamics. In *the 28th ACM International Conference on Information and Knowledge Management*. 469–478.

[21] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *the Neural Information Processing Systems*. 3111–3119.

[22] Yujie Mo, Liang Peng, Jie Xu, Xiaoshuang Shi, and Xiaofeng Zhu. 2022. Simple unsupervised graph representation learning. In *the AAAI Conference on Artificial Intelligence*. 7797–7805.

[23] Ikenna Victor Oluigbo, Hamida Seba, and Mohammed Haddad. 2023. Improving node embedding by a compact neighborhood representation. *Neural Computing and Applications* 35, 9 (2023), 7035–7048.

[24] Pietro Panzarasa, Tore Opsahl, and Kathleen M Carley. 2009. Patterns and dynamics of users' behavior and interaction: network analysis of an online community. *Journal of the American Society for Information Science and Technology* 60, 5 (2009), 911–932.

[25] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao Schardl, and Charles Leiserson. 2020. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *the AAAI Conference on Artificial Intelligence*. 5363–5370.

[26] Jiao Pengfei, Chen Shuxin, Guo Xuan, He Dongxiao, and Liu Dong. 2024. Survey on graph neural ordinary differential equations. *Journal of Computer Research and Development* 61, 8 (2024), 2045–2066.

[27] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *the 20th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 701–710.

[28] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. 2020. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637* (2020).

[29] Xiran Song, Jianxun Lian, Hong Huang, Zihan Luo, Wei Zhou, Xue Lin, Mingqi Wu, Chaozhuo Li, Xing Xie, and Hai Jin. 2023. xGCN: An extreme graph convolutional network for Large-scale social link prediction. In *the ACM Web Conference*. 349–359.

[30] Zhi-yan Song, Jian-wei Liu, Jie Yang, and Lu-ning Zhang. 2023. Linear normalization attention neural Hawkes process. *Neural Computing and Applications* 35, 1 (2023), 1025–1039.

[31] Qiaoyu Tan, Ninghao Liu, Xiao Huang, Soo-Hyun Choi, Li Li, Rui Chen, and Xia Hu. 2023. S2GAE: Self-Supervised graph autoencoders are generalizable learners with graph masking. In *the 16th ACM International Conference on Web Search and Data Mining*. 787–795.

[32] Samuel Unicomb, Gerardo Iñiguez, James P Gleeson, and Márton Karsai. 2021. Dynamics of cascades on burstiness-controlled temporal networks. *Nature Communications* 12, 1 (2021), 133.

[33] Zhihao Wen and Yuan Fang. 2022. Trend: Temporal event and node dynamics for graph representation learning. In *the ACM Web Conference*. 1159–1169.

[34] Yuxia Wu, Yuan Fang, and Lizi Liao. 2024. On the feasibility of simple transformer for dynamic graph modeling. In *the ACM Web Conference*. 870–880.

[35] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. 2020. Inductive representation learning on temporal graphs. *arXiv preprint arXiv:2002.07962* (2020).

[36] Sikun Yang and Hongyuan Zha. 2024. A variational autoencoder for neural temporal point processes with dynamic latent graphs. In *the AAAI Conference on Artificial Intelligence*. 16343–16351.

[37] Yu Yang, Jiannong Cao, Milos Stojmenovic, Senzhang Wang, Yiran Cheng, Chun Lum, and Zhetao Li. 2021. Time-capturing dynamic graph embedding for temporal linkage evolution. *IEEE Transactions on Knowledge and Data Engineering* 35, 1 (2021), 958–971.

[38] Nan Yin, Mengzhu Wang, Zhenghan Chen, Giulia De Masi, Huan Xiong, and Bin Gu. 2024. Dynamic spiking graph neural networks. In *the AAAI Conference on Artificial Intelligence*. 16495–16503.

[39] Jiaxuan You, Tianyu Du, and Jure Leskovec. 2022. ROLAND: Graph learning framework for dynamic graphs. In *the 28th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2358–2366.

[40] Le Yu, Leilei Sun, Bowen Du, and Weifeng Lv. 2023. Towards better dynamic graph learning: New architecture and unified library. In *the Neural Information Processing Systems*. 67686–67700.

[41] Guolin Zhang, Zehui Hu, Guoqiu Wen, Junbo Ma, and Xiaofeng Zhu. 2023. Dynamic graph convolutional networks by semi-supervised contrastive learning. *Pattern Recognition* 139 (2023), 109486.

[42] Han Zhao, Xu Yang, Cheng Deng, and Junchi Yan. 2024. Dynamic reactive spiking graph neural network. In *the AAAI Conference on Artificial Intelligence*. 16970–16978.

[43] Zhongying Zhao, Zhan Yang, Chao Li, Qingtian Zeng, Weili Guan, and MengChu Zhou. 2023. Dual Feature Interaction-Based Graph Convolutional Network. *IEEE Transactions on Knowledge and Data Engineering* 35, 9 (2023), 9019–9030.

[44] Yuan Zuo, Guannan Liu, Hao Lin, Jia Guo, Xiaoqian Hu, and Junjie Wu. 2018. Embedding temporal network via neighborhood formation. In *the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2857–2866.