When Speed meets Accuracy: an Efficient and Effective Graph Model for Temporal Link Prediction

Haoyang LI
PolyU
haoyang-comp.li@polyu.edu.hk

Yuming XU
PolyU & SCUT
fm.martinx@gmail.com

Yiming LI
HKUST
yliix@connect.ust.hk

Hanmo LIU
HKUST
hliubm@connect.ust.hk

Darian LI
PolyU & GDUT
jinlulhc@outlook.com

Chen Jason ZHANG
PolyU
jason-c.zhang@polyu.edu.hk

Lei CHEN
HKUST(GZ) & FYTRI
leichen@cse.ust.hk

Qing LI
PolyU
csqli@comp.polyu.edu.hk

ABSTRACT

Temporal link prediction in dynamic graphs is a critical task with applications in diverse domains such as social networks, recommendation systems, and e-commerce platforms. While existing Temporal Graph Neural Networks (T-GNNs) have achieved notable success by leveraging complex architectures to model temporal and structural dependencies, they often suffer from scalability and efficiency challenges due to high computational overhead. In this paper, we propose EAGLE, a lightweight framework that integrates short-term temporal recency and long-term global structural patterns. EAGLE consists of a time-aware module that aggregates information from a node's most recent neighbors to reflect its immediate preferences, and a structure-aware module that leverages temporal personalized PageRank to capture the influence of globally important nodes. To balance these attributes, EAGLE employs an adaptive weighting mechanism to dynamically adjust their contributions based on data characteristics. Also, EAGLE eliminates the need for complex multi-hop message passing or memory-intensive mechanisms, enabling significant improvements in efficiency. Extensive experiments on seven real-world temporal graphs demonstrate that EAGLE consistently achieves superior performance against state-of-the-art T-GNNs in both effectiveness and efficiency, delivering more than a 50× speedup over effective transformer-based T-GNNs.

PVLDB Reference Format:
Haoyang LI, Yuming XU, Yiming LI, Hanmo LIU, Darian LI, Chen Jason ZHANG, Lei CHEN, and Qing LI. When Speed meets Accuracy: an Efficient and Effective Graph Model for Temporal Link Prediction. PVLDB, 18(10): 3396 - 3405, 2025.
doi:10.14778/3748191.3748203

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/TreeAI-Lab/EAGLE.

Proceedings of the VLDB Endowment, Vol. 18, No. 10 ISSN 2150-8097.
doi:10.14778/3748191.3748203

1 INTRODUCTION

Temporal graphs [1, 4, 7, 30, 38, 58], also known as dynamic graphs, represent dynamically evolving interactions, relationships, and events among nodes over time. These graphs are widely applied across diverse domains, including social networks [6, 14], recommendation systems [8, 40], and e-commerce platforms [26, 27]. A critical task in temporal graph analysis is temporal link prediction [3, 7, 47], which aims to predict whether a link (or interaction) will occur between two nodes at a specific future time based on historical data. Temporal link prediction is pivotal for applications, such as recommending friends or content in social networks, predicting user-item interactions for personalized recommendations, and identifying emerging patterns in dynamic systems.

Recently, temporal graph neural networks (T-GNNs) [5, 16, 18, 35, 37, 39, 43, 48, 51, 55], such as DyGFormer [55], GraphMixer [5], TGN [39], etc, have achieved success in modeling temporal graphs, which capture both temporal and structural information. These T-GNNs have different model architectures, which leverage multiple-hop neighbor aggregation [50], multi-head self-attention mechanisms [44], recurrent neural networks [32], and memory modules [39]. While these T-GNNs subsequently improve performance on temporal link prediction tasks, they often require extensive computational resources, which limits their scalability, efficiency, and generalization in real-world applications [17, 24]. To address these computational challenges, researchers have explored data management techniques [9, 10, 20, 21, 23, 24, 48, 57, 59, 61] to accelerate the T-GNNs without sacrificing accuracy. For instance, methods, such as APAN [48], TGL [61], CacheGNN [20], Orca [23, 25], and GENTI [57], have introduced techniques such as asynchronous message propagation, parallel sampling, pruning computational graphs, and caching intermediate results.

However, despite extensive research on T-GNNs, a fundamental question remains unanswered:

What inherent attributes of temporal graphs are truly significant for temporal link prediction, and do we really need so complex T-GNN architectures to achieve high performance?

In this paper, we conduct motivational experiments to highlight two orthogonal yet complementary attributes that significantly contribute to temporal link prediction: most recent neighbors and

global influential nodes. Recent neighbors capture short-term temporal recency, reflecting a node's current preferences and behavioral trends. These interactions are often the strongest indicators of immediate future connections. For instance, in e-commerce networks like eBay or Taobao, a customer's next purchase is frequently influenced by their most recent transaction. Similarly, in social networks such as Twitter or Facebook, users who actively engage with specific accounts (e.g., liking or retweeting) are likely to continue interacting with them in the near future. In contrast, global influential nodes capture long-term dependencies and structural patterns, representing a node's broader context and position within the network. While these connections may not be recent, they remain impactful due to their structural significance. For example, recommendations based on highly-rated or well-connected items are often driven by global structural patterns rather than recent activity. However, existing T-GNNs with complex architectures fail to explicitly and efficiently integrate both factors, leading to suboptimal performance and significant computational overhead.

To address these limitations, we propose EAGLE, a lightweight and effective framework for temporal link prediction, which explicitly captures short-term temporal trends and long-term structural dependencies. Specifically, EAGLE consists of a time-aware module and a structure-aware module. The time-aware module explicitly captures short-term trends by aggregating information from a node's top-k_r most recent neighbors, which reflect its immediate interactions and preferences. Meanwhile, the structure-aware module explicitly learns long-term dependencies by leveraging Temporal Personalized PageRank (T-PPR) to select and aggregate information from the top-k_s most influential nodes in the graph. To balance the contributions of these modules, EAGLE employs an adaptive weighting mechanism that dynamically adjusts the influence of temporal and structural factors based on varying data characteristics, such as graphs with high temporal activity but minimal structure or vice versa. From the architecture perspective, EAGLE employs simple 2-layer MLPs and aggregates information from only $k_r + k_s$ nodes, enabling it to achieve significantly higher efficiency than existing T-GNNs that rely on complex architectures such as transformer and multi-hop message passing.

In summary, this work makes the following key contributions:

- We identify and validate two critical attributes for each node in temporal link prediction: (i) most recent neighbors, which capture a node's short-term temporal recency and preferences, and (ii) global influential nodes, which provide long-term global structural patterns specific to each node.
- We propose a lightweight and effective model EAGLE for temporal link prediction, which explicitly integrates short-term temporal recency and long-term global structural patterns.
- Extensive experiments on seven real-world temporal graph datasets show that EAGLE outperforms state-of-the-art T-GNNs in both effectiveness and efficiency.

2 PRELIMINARY AND RELATED WORK

2.1 Temporal Graphs and Link Prediction

Temporal graphs are very popular in real-world applications, such as Twitter and Facebook. In real-world web platforms, users may subscribe/unsubscribe to other users, join and leave the platforms, or change their attributes. These activities can be regarded as temporal graphs. Formally, we denote a temporal graph at time t as $G(t) = (V(t), \mathbf{A}(t), \mathbf{X}(t), \mathbf{E}(t))$, where $\mathbf{A}(t) \in \{0,1\}^{|V(t)|\times|V(t)|}$ is the adjacency matrix, $\mathbf{X}(t) \in \mathbb{R}^{|V(t)|\times d_x}$ is the node feature matrix, and $\mathbf{E}(t) \in \mathbb{R}^{n_e \times d_e}$ is the edge feature matrix. In general, $G(t)$ can be constructed by a sequence of edge interaction events $\mathcal{E}(t) = \{\gamma(1), \gamma(2), \ldots, \gamma(t)\}$ arranged in ascending chronological order. Each interaction $\gamma(\tau_i)$ is characterized by $\gamma(\tau_i) = (v, u, \mathbf{e}_{v,u}(\tau_i), \tau_i)$, where v and u denote nodes, $\mathbf{e}_{v,u}(\tau_i)$ is edge features, $\tau_i \in \mathbb{N}^+$ denotes the timestamp of $\gamma(\tau_i)$. Particularly, the neighbors of each $v \in V(t)$ at time t can be denoted as $N_v(t) = \{(u, \mathbf{e}_{v,u}(\tau_i), \tau_i) \mid (v, u, \mathbf{e}_{v,u}(\tau_i), \tau_i) \in \mathcal{E}(t), \tau_i \le t\}$.

Given a continuous-time dynamic graph $G(t)$ at time t and two nodes v and u, the temporal link prediction task aims to predict the occurrence of an interaction between v and u after time t. Temporal link prediction plays a crucial role in practical applications across diverse domains, including social networks, recommendation systems, and e-commerce. Early temporal link prediction methods focus on acquiring temporal embeddings with random walks (e.g., CTDNE [33]) or temporal point processes (e.g., HTNE [62], M2DNE [31]). Inspired by random walks on static graphs, CTDNE [33] extends the principles of random walk-based embedding methods, such as DeepWalk [36], node2vec [13], and LINE [42], to learn time-dependent representations in dynamic graphs. However, such random walk-based approaches fail to effectively capture the joint structural and temporal dependencies in dynamic graphs, leading to the development of Temporal Graph Neural Networks [29, 60] to overcome these limitations.

Table 1: Summary of primary notations.

Notation	Description
G	A continuous-time dynamic graph
$\gamma(t)$	The interaction arriving at timestamp t
t^-, t^+	The time just before and after t
τ, τ_i	Time index
$\Delta(t)$	The difference between t and t^-
v, u	Node index
$\mathbf{x}_v(t), \mathbf{e}_{v,u}(t)$	Node and edge feature
$N_v(t)$	Neighbors of v
$N_v(t, k_r)$	k_r most recent neighbors from $N_v(t)$
k_r	Number of neighbors with highest recency
k_s	Number of nodes with highest PPR scores
$E_v(t)$	Edges connecting with v before t
$\mathbf{h}_v(t)$	Node representation
$\pi_v(t)$	T-PPR score from v to all nodes
$\mathbf{P}(t)$	Probability transition matrix
$s^{ta}_{v,u}(t)$	Time-aware score
$s^{sa}_{v,u}(t)$	Structure-aware score
$s^{hy}_{v,u}(t)$	Hybrid score
d_h, d_x, d_t, d_e	Dimensions of $\mathbf{h}_v(t)$, $\mathbf{x}_v(t)$, $\mathbf{t}$, $\mathbf{e}_{v,u}(t)$

2.2 Temporal Graph Neural Networks

We firstly introduce the architecture design of temporal GNNs, followed by a review of existing data management approaches.

2.2.1 Temporal GNN Architecture Design. To encode a node v at time t, T-GNNs compute the representation $\mathbf{h}_v(t)$ by sampling from its neighbor set $N_v(t)$ and aggregating the features of v and its neighbors, along with time encoding $\mathbf{t}$ and edge features, across L layers of the network. This process can be formalized as follows:

$$N_v(t,k) = \texttt{SAMPLE}(N_v(t),k), \tag{1}$$

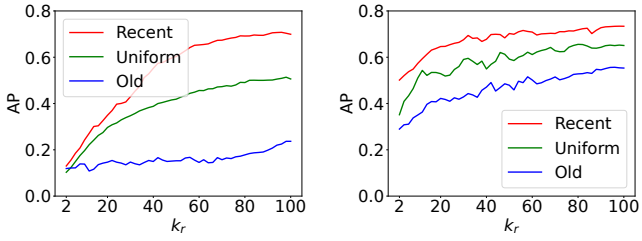
$$\mathbf{h}_v^l(t) = \texttt{ENC}\Big(\mathbf{h}_v^{l-1}(t^-), \mathbf{t}, \big\{\mathbf{h}_u^{l-1}(\tau), \mathbf{e}_{v,u}(\tau) | (u,\tau) \in N_v(t)\big\}\Big), \tag{2}$$

where $1 \le l \le L$ denotes the layer index, and the initial node feature is given by $\mathbf{h}_v^0(t) = \mathbf{x}_v(t)$. The representation at the L-th layer is denoted as $\mathbf{h}_v(t)$. The function $\texttt{SAMPLE}(\cdot)$ in Equation (1) selects k neighbors from $N_v(t)$, either randomly or based on recency. The function $\texttt{ENC}(\cdot)$ in Equation (2) defines the specific aggregation mechanism and varies across T-GNN models. Common designs for $\texttt{ENC}(\cdot)$ include RNNs, attention mechanisms, and transformers [44]. Depending on the method, the inputs and configuration of $\texttt{SAMPLE}(\cdot)$ and $\texttt{ENC}(\cdot)$ may differ or be omitted [18, 37].

As a representative work, JODIE [18] uses an RNN [28] to update node representations based on direct edge information $\gamma(t)$. DyRep [43] and TGAT [51] use a multi-layer self-attention mechanism to encode temporal information from neighbors. Recognizing the importance of capturing the dynamics within each node, memory modules are introduced in APAN [48] and TGN [39] to store and reflect node-specific historical information. Beyond message-passing mechanisms, CAWN [49] employs anonymized temporal random walks to capture temporal motifs, combined with RNNs for encoding. Taking a step further, Zebra [24] adopts a time-aware personalized PageRank (T-PPR) algorithm for neighbor aggregation to learn representations. Moreover, TCL [46] and DyGFormer [55] utilize transformers [45] to encode nodes, resulting in a quadratic time complexity with respect to the number of neighbors. To address the growing complexity of models, GraphMixer [5] uses MLPs and mean pooling to encode node and neighbor information for efficient temporal link prediction.

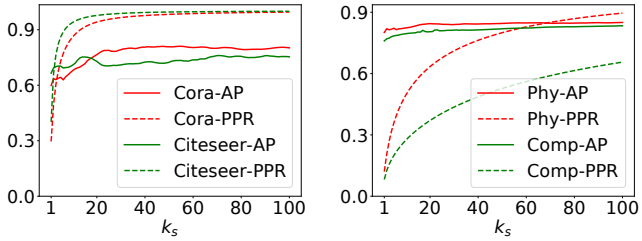
Unlike existing T-GNNs that use complex architectures like transformers or memory modules, EAGLE adopts a fundamentally different approach. EAGLE explicitly and efficiently integrates short-term temporal recency via recent neighbors and long-term global structural patterns via influential nodes identified by T-PPR, using lightweight MLPs and a modular design. Therefore, EAGLE achieves high performance while greatly reducing computational cost and memory usage.

2.2.2 Data Management for T-GNNs. Data management techniques are orthogonal to the design of T-GNN architectures, as they accelerate T-GNNs without interfering with their core architecture [2, 9, 10, 20, 21, 48, 56, 57, 59, 61]. For instance, APAN [48] propagates messages asynchronously between nodes in a non-blocking manner, efficiently updating node embeddings by reducing the overhead associated with sequential processing. Similarly, TGL [61] and GENTI [57] employ parallel sampling techniques that enable faster neighbor sampling by leveraging distributed or parallel computation frameworks. To address the challenges of GPU-CPU communication overhead, SIMPLE [10] and ETC [9] utilize data management and batch construction strategies to minimize excessive GPU-CPU data transfer time. Additionally, caching techniques have proven effective for improving efficiency, as demonstrated by CacheGNN [20] and Orca [23, 25], which reuse frequently accessed embeddings and intermediate results in memory, thus avoiding redundant computations during neighbor aggregation and message passing. Unlike previous data management techniques that accelerate T-GNNs using specialized sampling, caching, or parallel processing, which often bring extra complexity, EAGLE achieves efficiency through a simple and lightweight modular design.



(a) Wikipedia. **(b) AskUbuntu.**

Figure 1: Time influence of neighbors.



(a) Cora and Citeseer graphs. **(b) Computer and Physics graphs.**

Figure 2: Global influential nodes.

3 METHODS

In this section, we begin by presenting a series of motivational experiments to highlight the critical role of time and structural information. We introduce EAGLE, a highly efficient and effective framework for temporal graph link prediction.

3.1 Motivational Observations

3.1.1 The Importance of Time Information. In this section, we evaluate the impact of time information on link prediction. Specifically, we perform link prediction experiments on two widely used temporal graphs, i.e., Wikipedia [34] and AskUbuntu [34]. The split ratios of each graph for training, validation, and test sets are 8:1:1 in chronological order, and the evaluation metric is Average Precision (AP). At each time step t, when learning the representation for a node v, we use three strategies to select its k_r neighbors: (1) **Recent**: select its top-k_r most recent neighbors, (2) **Old**: select its top-k_r oldest neighbors, and (3) **Uniform**: uniformly sample k_r neighbors from all neighbors.

As shown in Figure 1 (a) and (b), the Recent strategy consistently outperforms the Uniform and Old strategies on both datasets, highlighting the relevance of recent neighbors in improving model performance. Uniform performs moderately, while Old exhibits the lowest performance, suggesting that older interactions contribute

less useful information. Additionally, as the number of neighbors k_r increases, the performance improves across all strategies. However, the performance begins to stabilize when k_r becomes sufficiently large (e.g., beyond $k_r = 40$), indicating diminishing returns from adding more neighbors. These findings highlight the importance of time-aware neighbor selection and show that recent neighbors provide more valuable information for improving model effectiveness.

3.1.2 The Importance of Structure Information. We investigate the significance of global graph structure for each node. Specifically, the importance of each node u to the node representation of each node v is proportional to the personalized PageRank score from node v to node u. We utilize the following theorem.

THEOREM 3.1. *[52] Given a graph $G(V, \mathbf{A}, \mathbf{X})$, a L-layer GCN-mean model learns node representations $\mathbf{H} \in \mathbb{R}^{|V| \times d_h}$ as follows:*

$$\mathbf{H}^{l+1} = \mathsf{ReLU}\left(\mathbf{A}_n \mathbf{H}^l \mathbf{W}^l\right), \quad l = 0, 1, \ldots, L-1 \tag{3}$$

$$\mathbf{H} = \mathbf{H}^L, \ \ \mathbf{H}^0 = \mathbf{X} \tag{4}$$

where $\mathbf{A}_n = D^{-1}\mathbf{A}$ and D is the degree of nodes V. Then, the importance score of node u on node v after L propagation steps is the expected Jacobian matrix: $I(v, u, L) = \left\| \mathbb{E}\left[\frac{\partial \mathbf{H}[v]}{\partial \mathbf{X}[u]}\right] \right\|_1$. The normalized importance score $\tilde{I}(v, u, L)$ of node u on v is defined as:

$$\tilde{I}(v, u, L) = \frac{I(v, u, L)}{\sum_{w \in \mathcal{V}} I(v, w, L)} = c \cdot \sum_{Path_L^{v \to u}} \prod_{i=L}^{1} \frac{1}{D[v_{i-1}]}. \tag{5}$$

where $\tilde{I}(v, u, L)$ is the sum probabilities of all possible influential paths from v to node u, $Path_L^{v \to u}$ is a path $[v, v_2, \ldots, u]$ of length L from node v to node u, $D[v_{i-1}]$ is the degree of v_{i-1}, and c is an constant.

We conduct experiments on the widely used Cora [53], Citeseer [53], Amazon-Computers (COMP) [41], and Coauthor-Physics (Phy) [41] datasets for the link prediction with the APPNP model [11]. These graphs are split into training, validation, and test sets with ratios of 8:1:1. The goal is to evaluate the impact of selecting a limited number of influential neighbors for each node on the performance of the link prediction task. To begin, we compute the Personalized PageRank (PPR) score for each pair of nodes V in the graph. For each node $v \in V$, we retain only the top-k_s nodes from V with the highest PPR scores as its neighbors, thereby reducing the number of neighbors while focusing on the most influential ones.

As shown in Figure 2 (a) and (b), as k_s increases, the average precision (AP) initially improves and then stabilizes, even when k_s is significantly smaller than the total number of nodes $|V|$. Interestingly, the average accumulated PPR scores of the retained top-k_s neighbors do not necessarily account for the largest proportion of all PPR scores. This observation highlights that retaining only a small number of influential nodes is sufficient.

3.2 The EAGLE Framework

In this subsection, we introduce our framework, EAGLE, for temporal link prediction, inspired by the two motivational experiments discussed earlier. EAGLE is both highly efficient and effective at predicting links between nodes. It consists of two simple components, i.e., a time-aware module and a structure-aware module.

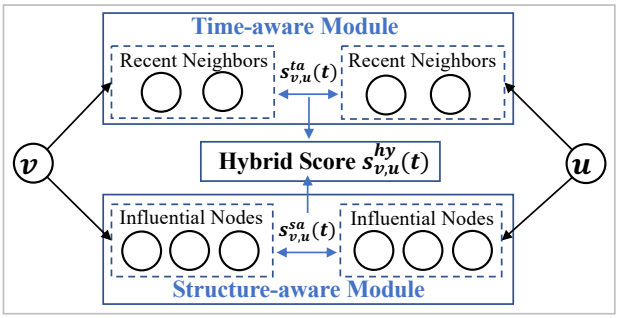


Figure 3: EAGLE framework overview. EAGLE predicts whether there is a link between nodes v and u at time t.

3.2.1 Time-aware Module. As discussed in Section 3.1.1, from a temporal perspective, the most recent neighbors are more significant than older ones, as they reflect the current preferences of nodes. Leveraging these top-k_r most recent neighbors enables the model to better capture such recent preferences. Formally, given a graph $G(t) = (V(t), \mathbf{A}(t), \mathbf{X}(t), \mathbf{E}(t))$ and a node v, we denote the top-k_r most recent neighbors of node v as $N_v(t, k_r) = \{(v_i, \mathbf{e}_{v,v_i}(\tau_i), \tau_i) \mid (v, v_i, \mathbf{e}_{v,v_i}(\tau_i), \tau_i), \tau_i \leq t\}$, where $|N_v(t, k_r)| = k_r$. Then, we can learn the time-aware node representation for each node $v \in V(t)$ by averaging the concatenation of the node feature $\mathbf{x}_{v_i}(t)$ and the edge feature $\mathbf{e}_{v,v_i}(\tau_i)$ of the selected nodes in $N_v(t, k_r)$ as follows:

$$\mathbf{h}_v(t) = \frac{1}{k_r} \sum_{(v_i, \mathbf{e}_{v,v_i}(\tau_i), \tau_i) \in N_v(t, k_r)} [\mathbf{x}_{v_i}(t), \mathbf{e}_{v,v_i}(\tau_i)], \tag{6}$$

After obtaining representations $\mathbf{h}_v(t)$ and $\mathbf{h}_u(t)$ for each pair of nodes v and u in Equation (6), the time-aware module predicts the probability $s^{ta}_{v,u}(t) \in [0, 1]$ of a link between v and u as follows:

$$s^{ta}_{v,u}(t) = \sigma\big(\mathbf{W}_2 \, \mathsf{ReLU}(\mathbf{W}_1[\mathbf{h}_v(t), \mathbf{h}_u(t)] + \mathbf{b}_1) + \mathbf{b}_2\big), \tag{7}$$

where $\sigma(\cdot)$ is `Sigmoid` function, and $\mathbf{W}_1$, $\mathbf{b}_1$, $\mathbf{W}_2$, and $\mathbf{b}_2$ are simple trainable parameters.

3.2.2 Structure-aware Module. As discussed in Section 3.1.2, nodes with higher PPR scores are more important to a given node v. Therefore, we employ an efficient Temporal PPR (T-PPR) algorithm [24] to estimate the influence score of nodes in temporal graphs. T-PPR builds upon the traditional PPR approach but extends it to dynamic graphs, where the importance of temporal information is emphasized. Formally, the T-PPR [24] score of each node v with α-termination probability for each node v is defined as follows:

$$\pi_v^{l+1}(t) = \alpha \cdot \mathbf{P}(t)\pi_v^l(t) + (1 - \alpha) \cdot \mathbf{r}_v(t) \tag{8}$$

where $\pi_v^{l+1}(t) \in [0, 1]^{|V(t)|}$ is the transition probability from node v to all nodes $V(t)$ after l+1 iterations , $\pi_v^0(t) = \{1\}^{|V(t)|}$, $\alpha \in (0, 1)$ is termination rate, and $\mathbf{r}_v(t) \in \{0, 1\}^{|V(t)|}$ is the one-hot vector of v, i.e., $\mathbf{r}_v(t)[u] = 1$ if $u = v$ otherwise 0. Also, $\mathbf{P}(t) \in [0, 1]^{|V(t)| \times |V(t)|}$ is the transition probability matrix at time t. We denote the final T-PPR score of node v (resp., nodes $V(t)$) as $\pi_v(t)$ (resp., $\pi_V(t)$) after L iterations, and $\pi_v(t)[u]$ denotes T-PPR score from v to u at time t. Then, we introduce how to compute the transition probability $\mathbf{P}(t)[v_m][v_n]$. Given a node v_m and its neighbors $N_{v_m}(t) = \{(v_n, \mathbf{e}_{v_m,v_n}(\tau), \tau) \mid (v_m, v_n, \mathbf{e}_{v_m,v_n}(\tau), \tau) \in \mathcal{E}(t), \tau \leq t\}$ at time t, the transition probability $\mathbf{P}(t)[v_m][v_n]$ from v_m to each

neighbor $(v_n, \mathbf{e}_{v_m,v_n}(\tau), \tau) \in N_{v_m}(t)$ is as:

$$\mathbf{P}(t)[v_m][v_n] = \frac{\beta^{|\{(v',\tau')|(v',\mathbf{e}_{v_m,v'}(\tau'),\tau') \in N_{v_m}(t), \tau' \geq \tau\}|}}{\sum_{z=1}^{|N_{v_m}(t)|} \beta^z}, \quad (9)$$

where $\beta \in (0, 1)$ is an exponential decay factor. Equation (9) ensures that more recent temporal neighbors have higher probabilities. Specifically, it assigns the transition probability of the k-th most recent temporal neighbor to be proportional to β^k.

At time t, given a node v with its T-PPR vector $\pi_v(t)$, nodes with the top-k_s highest T-PPR scores in $\pi_v(t)$ are $N_v(\pi_v(t), k_s) = \{(v_i, \pi_v(t)[v_i])\}_{i=1}^{k_s}$. Formally, given two nodes $v, u \in V(t)$, the set of common nodes in their top-k_s T-PPR nodes is $N_v(\pi_v(t), k_s) \cap N_u(\pi_u(t), k_s)$ Then, the structure-aware module predicts the probability $s^{sa}_{v,u}(t)$ of a link between node v and node u by summing the product of T-PPR scores of shared influential neighbors as follows:

$$s^{sa}_{v,u}(t) = \sum_{v_i \in N_v(\pi_v(t),k_s) \cap N_u(\pi_u(t),k_s)} \pi_v(t)[v_i] \cdot \pi_u(t)[v_i] \quad (10)$$

A higher $s^{sa}_{v,u}(t)$ indicates a stronger structural connection between v and u, increasing the likelihood of a link existing between them.

3.2.3 Hybrid Model. We discuss how to integrate the time-aware and structure-aware modules to calculate the final probability for nodes v and u. Each module has its own limitations. The time-aware module focuses solely on temporal recency, such as the most recent interactions of a node, but overlooks global structural patterns and long-term relevance. On the other hand, the structure-aware module effectively captures global structural patterns but fails to account for time-sensitive interactions, which can result in suboptimal predictions. For example, in e-commerce networks, such as eBay or Taobao, a customer's next purchase is often influenced by their most recent transaction.

Given each pair of nodes v and u, the final score for determining whether an edge exists between them incorporates both the temporal score and structural score with an adaptive weighting. Specifically, given the top-k_r most recent neighbors $N_v(t, k_r)$ of node v, the average interaction time interval between node v and its neighbors $N_v(t, k_r)$ can be computed as $\bar{t}_v = \frac{1}{k_r} \sum_{(v_i,\mathbf{e}_{v,v_i}(\tau_i),\tau_i) \in N_v(t,k_r)} (t - \tau_i)$. Smaller $\bar{t}_v$ indicates more recent interactions, while larger $\bar{t}_v$ indicates older interactions. Therefore, given nodes v and u, together with their time-aware score $s^{ta}_{v,u}(t)$ and structure-aware score $s^{sa}_{v,u}(t)$, the hybrid score $s^{hy}_{v,u}(t)$ is defined as follows:

$$s^{hy}_{v,u}(t) = \lambda \cdot (\exp(-\bar{t}_v) + \exp(-\bar{t}_u)) \cdot s^{ta}_{v,u}(t) + s^{sa}_{v,u}(t), \quad (11)$$

where λ is a trade-off parameter,which can be automatically decided based on validation datasets. The terms $\exp(-\bar{t}_v)$ and $\exp(-\bar{t}_u)$ enable the score $s^{hy}_{v,u}(t)$ to adaptively prioritize the temporal recency score $s^{ta}_{v,u}(t)$ for recent interactions (small $\bar{t}_v$ and $\bar{t}_u$) while relying more on the structural score $s^{sa}_{v,u}(t)$ for nodes with older interactions or sparse temporal activity. Thus, the model effectively balances short-term temporal relevance and long-term structural patterns.

3.2.4 Algorithm Summary. Algorithm 1 outlines the process of predicting future links for node pairs $\{(v_j, u_j)\}_{j=1}^m$. Specifically, the first step of the EAGLE is to update the graph $G(\tau_1^-)$ by incorporating the new events $\{\gamma(\tau_i)\}_{i=1}^n$, obtaining the updated graph $G(\tau_n)$. Secondly, EAGLE efficiently updates the T-PPR matrix $\pi_V(\tau_1^-)$ to

Algorithm 1: The EAGLE Framework

Input: Latest graph $G(\tau_1^-)$, T-PPR matrix $\pi_V(\tau_1^-)$, new events $\{\gamma(\tau_i)\}_{i=1}^n$, node pairs $\{(v_j, u_j)\}_{j=1}^m$, recent neighbor number k_r, and the number of top nodes ranked by T-PPR scores k_s.

Output: The updated graph $G(\tau_n)$, T-PPR matrix $\pi_V(\tau_n)$, and hybrid score $\{s^{hy}_{v_j,u_j}(\tau_n)\}_{j=1}^m$.

```
G(τ_n) = G(τ_1^-) + {γ(τ_i)}_{i=1}^n
π_V(τ_n) = T-PPR_UPDATING(π_V(τ_1^-), {γ(τ_i)}_{i=1}^n)
for j = 1 to m do
    s^ta_{v_j,u_j}(τ_n) ← Equation (7)
    s^sa_{v_j,u_j}(τ_n) ← Equation (10)
    s^hy_{v_j,u_j}(τ_n) ← Equation (11)
Return G(τ_n), π_V(τ_n), {s^hy_{v_j,u_j}(τ_n)}_{j=1}^m
```

$\pi_V(\tau_n)$ based on [24]. Then, for link prediction, EAGLE iterates over each node pair (v_j, u_j) in $\{(v_j, u_j)\}_{j=1}^m$. For each pair of nodes (v_j, u_j), EAGLE computes three key scores: the time-aware score $s^{ta}_{v_j,u_j}(\tau_n)$ in Equation (7), the structure-aware score $s^{sa}_{v_j,u_j}(\tau_n)$ in Equation (10), and the hybrid score $s^{hy}_{v_j,u_j}(\tau_n)$ in Equation (11). Finally, EAGLE returns the updated graph $G(\tau_n)$, the updated T-PPR matrix $\pi_V(\tau_n)$, and the hybrid scores $\{s^{hy}_{v_j,u_j}(\tau_n)\}_{j=1}^m$. EAGLE predicts links efficiently and effectively by leveraging a lightweight design that incorporates both temporal and structural features. The analysis of time and space complexity are as follows.

- **Time Complexity.** Given n new events $\{\gamma(\tau_i)\}_{i=1}^n$, EAGLE takes $O(nd_x)$ to update the graph $G(\tau_1^-)$ to $G(\tau_n)$ and takes $O(nk_s \log k_s)$ time to update the top-k_s T-PPR matrix $\pi_V(\tau_1^-)$ to $\pi_V(\tau_n)$ [24]. Then, for each pair of nodes (v_j, u_j), EAGLE takes $O(k_r d_x + d_x d_w)$ to compute $s^{ta}_{v_j,u_j}(\tau_n)$, takes $O(k_s)$ to compute $s^{sa}_{v_j,u_j}(\tau_n)$, and takes $O(k_r)$ to compute $s^{hy}_{v_j,u_j}(\tau_n)$. Thus, EAGLE takes $O(n(d_x + k_s \log k_s) + m(k_r d_x + d_x d_w + k_s)$ time in total.
- **Space Complexity.** EAGLE only needs to store the graph $G(t)$, a single top-k_s T-PPR matrix $\pi_V(t)$, and two MLPs in Equation (7), resulting in minimal storage requirements. Thus, the space complexity of EAGLE is $O(G) + O(k_s V(t)) + O(M)$, where $O(G)$ represents the storage needed for the graph $G(t)$, and $O(M)$ denotes the space required to store the model parameters.

4 EXPERIMENTS

We compare our model EAGLE with state-of-the-art baselines on the seven graphs for link prediction. In node classification, EAGLE outperforms all baselines in terms of effectiveness. Detailed results are provided in our technical report [22] due to space limitations.

4.1 Experimental Settings

4.1.1 Datasets. Table 2 lists the seven widely-employed dynamic-graph benchmarks used in our experiments. Specifically, *Contacts* [37] is a contact network among university students where each link weight quantifies physical proximity. *LastFM* [37] is a bipartite user-song graph whose links denote a user playing a song. *Wikipedia* [18] and *Reddit* [18] track user edits and user posts, respectively, where

each link represents an action. *AskUbuntu* [34] and *SuperUser* [34] are Stack Exchange traces in which a link represents a user-user exchange. *Wiki-Talk* [19, 34] connects a contributor to another whenever the former edits the latter's Talk page. Following [17, 24, 39, 51], interaction links $\mathcal{E}(t) = \gamma(1), \ldots, \gamma(t)$ is chronologically split 70%/15%/15% into training, validation, and test sets.

4.1.2 Evaluation Metrics. For each test edge $(v, u, t) \in \mathcal{E}_{test}$, we randomly sample $Neg_t(v)$ negative nodes to replace node u, treating them as negative candidates for node v.

Effectiveness Metric. Following [5, 15, 17, 24, 55], we use the widely used **Average Precision (AP)**, **Mean Reciprocal Rank (MRR)**, and **Hit Ratio@N (HR@N)** metrics.

- **Average Precision (AP)**: AP measures the area under precision-recall curve, reflecting how well the model ranks true positives.
- **Mean Reciprocal Rank (MRR).** MRR is defined as $\textbf{MRR} = \frac{1}{|\mathcal{E}_{\text{test}}|} \sum_{(v,u,t)\in\mathcal{E}_{\text{test}}} \frac{1}{\text{rank}(v,u,t)}$, where $\text{rank}(v, u, t)$ is the rank of u among the $u \cup Neg_t(v)$.
- **Hit Ratio@N (HR@N).** Given each test edge (v, u, t), this metric evaluates whether the true positive node u is ranked within the top-N predictions among $u \cup Neg_t(v)$, which is defined as $\textbf{HR@N} = \frac{1}{|\mathcal{E}_{\text{test}}|} \sum_{(v,u,t)\in\mathcal{E}_{\text{test}}} \mathbb{I}[\text{rank}(v, u, t) \leq N]$, where $\mathbb{I}[\text{rank}(v, u, t) \leq N] = 1$ if $\text{rank}(v, u, t) \leq N$.

Efficiency Metric. We evaluate the efficiency of models based on the total **training time (s)**, **inference time (s)**, **the peak GPU memory usage (MB)** in training and inference phrases.

4.1.3 Baselines. We compare EAGLE with six state-of-the-art T-GNNs, which are widely used [12, 17, 24, 54, 55]. They cover main architectures for temporal GNNs, including RNNs, transformers, and multi-hop aggregation, ensuring comprehensive comparisons.

- **JODIE** [18] utilizes RNNs to propagate temporal interaction information, enabling dynamic updating of node representations.
- **TGAT** [51] simulates the message-passing of static GNNs while encoding temporal information using random Fourier features.
- **TGN** [39] dynamically keeps a state vector for each node and generalizes previous methods[18, 51] as special cases.
- **GraphMixier (GMixer)** [5] encodes link and node representations via simple MLPs and mean pooling.
- **Zebra** [24] learns node representations by aggregating neighbors with weights decided by the T-PPR process at the target node.
- **DyGFormer** [55] designs a transformer-based encoder to integrate time, node and edge features at once.
- **EAGLE (Ours)**: A lightweight framework with three variants. **EAGLE-Time** focuses on short-term temporal recency; **EAGLE-Struc.** leverages long-term global structural patterns; **EAGLE-Hybrid** adaptively combines both for temporal link prediction.

4.1.4 Hyperparameter and Hardware Setting. Based on validation datasets, we tune parameters α in Equation (8) and β in Equation (9) in T-PPR by $\alpha, \beta \in \{0.1, 0.2, \cdots, 1\}$, and tune $k_r, k_s \in \{10, 20, 30, 40, 50\}$. For the baselines, based on validation data, we use grid search to tune the hyperparameters for each baseline on each dataset [17, 24], ensuring fair comparisons. For all models, we set the number of negative samples in the training and validation phases to 1, following [17, 24, 39, 51]. For evaluation, we use 99 negative samples per node during testing, ranking the positive sample among 100 nodes. Early stopping is applied based on validation loss, with a patience of 5 epochs. All codes are implemented by Python and executed on a CentOS 7 machine with a 20-core Intel Xeon Silver 4210 CPU @ 2.20GHz, 8 NVIDIA GeForce RTX 2080Ti GPUs with 22GB memory each, and 256GB of RAM.

Table 2: Statistics of seven dynamic graphs. $|V|$ and $|E|$ are the numbers of nodes and interactions, while d_v and d_e are the dimensions of node and edge features.

Dataset	$\lvert V\rvert$	$\lvert E\rvert$	d_v	d_e	Timespan
Contacts [37]	692	2,426,279	172	172	30 days
LastFM [37]	1980	1,293,103	172	172	30 days
Wikipedia [18]	9,227	157,474	172	172	30 days
Reddit [18]	10,984	672,447	172	172	30 days
AskUbuntu [34]	159,316	964,437	172	172	2613 days
SuperUser [34]	194,085	1,443,339	172	172	2773 days
Wiki-Talk [19, 34]	1,140,149	7,833,140	172	172	2320 days

4.2 Main experiments

4.2.1 Effectiveness. We evaluate the T-GNN baselines and EAGLE on seven real-world graphs using three metrics, including AP, MRR, and HR@10. As shown in Table 3, our proposed EAGLE consistently achieves superior or comparable performance against all T-GNN baselines across all datasets on these three metrics, demonstrating its superior ability to balance both effectiveness and efficiency in temporal link prediction tasks. Specifically, JODIE [18], TGAT [51], and GraphMixer [5] achieve unsatisfactory performance, as they fail to effectively capture the critical combination of short-term temporal recency and long-term structural dependencies. Similarly, GraphMixer [5] uses shallow MLPs to encode node and link features among neighbors but lacks the capacity to capture global patterns effectively. In contrast, TGN [39], DyGFormer [55], and Zebra [24] show improved performance by incorporating advanced attention mechanisms and memory modules. However, these models often come with significant computational overhead and fail to explicitly balance the influence of recent interactions and global structural contexts. For instance, Zebra [24] performs well in leveraging global structural dependencies using T-PPR but overlooks the importance of temporal recency. DyGFormer [55], while utilizing transformers to encode node and edge features, suffers from high time and space complexity, which limits its scalability to large graphs. Our EAGLE framework addresses these limitations by explicitly integrating short-term temporal recency and long-term global structural patterns, making it robust across diverse temporal graph datasets.

4.2.2 Efficiency. As shown in Table 4, EAGLE demonstrates significantly faster training and inference speeds while maintaining a low memory footprint, highlighting its superior scalability and practicality. Specifically, JODIE [18], TGAT [51], and GraphMixer [5] exhibit inefficiencies in both training and inference. JODIE relies on inefficient RNNs, and TGAT incorporates inefficient temporal attention mechanisms, Similarly, GraphMixer requires substantial resources to process temporal and structural information, making it less efficient than EAGLE. DyGFormer [55] uses transformer blocks, which

Table 3: Effectiveness Evaluation. OOM denotes out-of-GPU memory. and TLE indicates exceeding the two-day training limit. In particular, EAGLE-Struc. does not have standard deviation, as it is a training-free and deterministic algorithm in Section 3.2.2.

Model	Metric	JODIE	TGAT	TGN	GMixer	Zebra	DyGFormer	EAGLE		
								Time	Struc.	Hybrid
Contacts	*AP*	OOM	42.30±0.63	OOM	TLE	44.95±0.13	TLE	14.61±0.16	54.20	**54.24±0.11**
	MRR	OOM	68.19±0.41	OOM	TLE	70.07±0.05	TLE	60.70±0.12	78.51	**78.59±0.06**
	HR@10	OOM	96.28±0.04	OOM	TLE	95.89±0.02	TLE	78.24±0.09	98.18	**98.25±0.05**
LastFM	*AP*	OOM	13.15±0.55	17.73±0.48	TLE	20.45±0.07	TLE	20.43±0.68	31.49	**31.54±0.20**
	MRR	OOM	23.32±0.38	28.05±0.32	TLE	32.68±0.04	TLE	31.36±0.42	48.05	**48.11±0.14**
	HR@10	OOM	36.11±0.05	43.60±0.05	TLE	47.92±0.01	TLE	45.07±0.55	67.40	**67.52±0.07**
Wikipedia	*AP*	67.07±0.62	66.84±1.30	78.96±1.21	69.77±0.41	82.32±0.06	86.96±0.22	69.18±0.63	86.08	**87.02±0.29**
	MRR	71.88±0.25	71.62±1.13	81.79±0.43	73.75±0.67	85.56±0.04	87.79±0.04	72.75±0.49	88.09	**88.19±0.11**
	HR@10	85.16±0.10	84.80±0.21	91.21±0.05	87.30±0.11	92.10±0.02	**92.61±0.03**	84.59±0.09	92.04	92.22±0.02
Reddit	*AP*	69.34±0.21	72.91±0.25	73.37±0.91	70.07±0.11	74.05±0.07	77.92±0.33	31.96±0.78	74.85	**78.09±0.66**
	MRR	75.30±0.12	77.40±0.17	78.33±0.33	76.75±0.09	78.89±0.05	85.57±0.26	53.39±0.30	84.23	**86.15±0.25**
	HR@10	91.56±0.07	92.14±0.04	92.99±0.12	92.02±0.05	93.23±0.01	93.34±0.06	78.45±0.08	92.53	**94.25±0.03**
AskUbuntu	*AP*	OOM	49.28±1.47	69.10±0.59	71.92±0.63	69.02±0.17	69.55±0.35	72.58±0.33	48.37	**72.80±0.19**
	MRR	OOM	54.83±0.69	73.15±0.44	76.91±0.38	72.55±0.09	74.80±0.16	79.44±0.25	57.58	**79.82±0.11**
	HR@10	OOM	65.75±0.44	84.45±0.19	86.54±0.15	84.13±0.02	85.29±0.11	87.60±0.08	67.55	**87.72±0.02**
SuperUser	*AP*	OOM	49.95±0.89	67.50±0.45	TLE	68.24±0.23	TLE	68.63±0.91	44.89	**69.47±0.16**
	MRR	OOM	50.92±0.38	72.20±0.33	TLE	74.45±0.21	TLE	78.09±0.77	52.96	**78.85±0.10**
	HR@10	OOM	56.57±0.10	81.14±0.15	TLE	82.10±0.07	TLE	86.02±0.29	60.69	**87.17±0.04**
WikiTalk	*AP*	OOM	58.03±0.79	OOM	OOM	61.23±0.12	OOM	66.71±0.93	46.63	**67.34±0.23**
	MRR	OOM	61.88±0.56	OOM	OOM	64.43±0.09	OOM	78.04±0.76	58.61	**79.15±0.17**
	HR@10	OOM	70.33±0.17	OOM	OOM	74.78±0.05	OOM	89.34±0.27	69.95	**90.28±0.05**

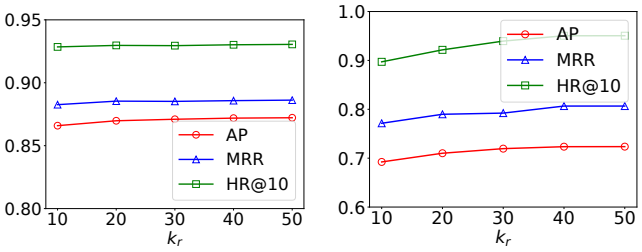
Table 4: Efficiency evaluation. OOM indicates out-of-GPU memory, and TLE indicates exceeding the two-day training limit.

Model	Metric	JODIE	TGAT	TGN	GMixer	Zebra	DyGFormer	EAGLE		
								Time	Struc.	Hybrid
Contacts	*T-train (s)*	OOM	128457	OOM	TLE	9059	TLE	751	**72**	823
	T-infer (s)	OOM	40278	OOM	TLE	5948	TLE	240	**210**	467
	M-train (MB)	OOM	7056	OOM	TLE	1336	TLE	4223	**675**	4223
	M-infer (MB)	OOM	7928	OOM	TLE	2050	TLE	5012	**1103**	5012
Wikipedia	*T-train (s)*	5468	4509	2941	10038	630	6750	71	**3**	74
	T-infer (s)	612	541	500	1184	411	2681	21	**7**	29
	M-train (MB)	1810	2714	3856	1440	724	1544	1120	**340**	1120
	M-infer (MB)	4180	4772	5226	3896	2045	3880	3624	**560**	3624
WikiTalk	*T-train (s)*	OOM	155328	OOM	OOM	12773	OOM	1208	**37**	1245
	T-infer (s)	OOM	67103	OOM	OOM	8102	OOM	539	**118**	682
	M-train (MB)	OOM	14994	OOM	OOM	2224	OOM	5129	**980**	5129
	M-infer (MB)	OOM	16045	OOM	OOM	2308	OOM	6067	**1528**	6067

are highly inefficient due to their complex self-attention mechanisms. Although Zebra [24] improves efficiency with data management strategies, it still faces significant computational overhead due to its reliance on multiple T-PPR matrices. EAGLE addresses these limitations with its lightweight time-aware and structure-aware modules and an efficient hybrid scoring mechanism. As a result, EAGLE delivers a speedup of over 50× compared to DyGFormer [55]. Peak GPU memory usage during inference is higher than during training because inference uses 99 negative samples compared to only 1 during training. Additionally, as the test progresses, the increasing graph size and expanded neighborhoods require more memory to process the growing number of nodes and neighbors.

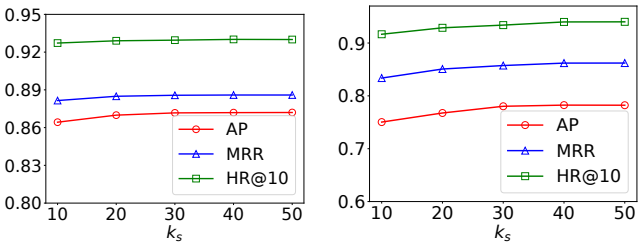
4.3 Ablation Study

We evaluated its **time-aware module** (EAGLE-Time), **structure-aware module** (EAGLE-Struc.), and their **hybrid model** (EAGLE-Hybrid) on both effectiveness and efficiency. Specifically, regarding effectiveness, as shown in Table 3, EAGLE-Time captures short-term temporal recency by aggregating the top-k_r most recent neighbors, excelling in datasets like AskUbuntu and SuperUser, where recent interactions heavily influence future behavior. Conversely, EAGLE-Struc., which leverages T-PPR to capture long-term structural patterns, performs well in structurally dominated datasets but fails to account for immediate temporal trends in highly dynamic graphs. EAGLE-Hybrid combines the strengths of both modules using an adaptive weighting mechanism, dynamically balancing short-term recency and long-term structure based on the dataset. This integration consistently outperforms the standalone modules in effectiveness. In terms of efficiency, as shown in Table 4, the variants of EAGLE are highly lightweight, as they process only the most recent neighbors and influential nodes, resulting in faster training and inference times. In particular, EAGLE-Struc is extremely efficient, with faster inference and lower memory usage, because it makes predictions directly from the T-PPR scores in Equation (10).



(a) Wikipedia. (b) AskUbuntu.

Figure 4: Recent neighbor number k_r.



(a) Wikipedia. (b) Reddit.

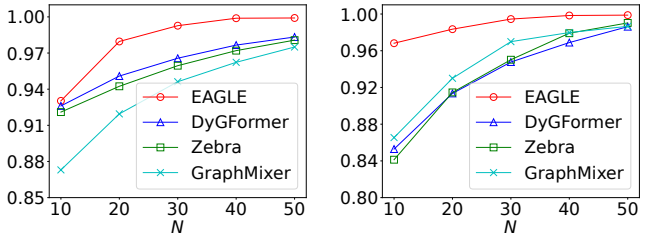
Figure 5: Global influential node number k_s.

4.4 Parameter sensitivity

4.4.1 Most Recent Neighbor Number k_r. We analyze the impact of the number of most recent neighbors k_r in Section 3.2.1. We evaluate k_r on two representative datasets, i.e., Wikipedia and AskUbuntu, vary $k_r \in \{10, 20, 30, 40, 50\}$, and report the metrics AP, MRR, and HR@10. As shown in Figure 4 (a) and (b), the increasing k_r generally improves model performance up to a certain threshold, beyond which the performance stabilizes. For instance, on the Wikipedia shown in Figure 4 (a), AP, MRR, and HR@10 steadily increase as k_r grows from 10 to 40, indicating that a larger number of recent neighbors provides more relevant temporal information. However, further increasing k_r beyond 40 yields diminishing returns, as the added neighbors contribute little additional information.

4.4.2 Global Influential Node Number k_s. We analyze the impact of the number of global influential nodes k_s in Section 3.2.2 for EAGLE. We evaluate the effect of k_s on two representative datasets, Wikipedia and Reddit, with varying $k_s \in \{10, 20, 30, 40, 50\}$. The used metrics are AP, MRR, and HR@10. As shown in Figure 5 (a) and (b), increasing k_s generally improves model performance up to a certain threshold, after which the performance stabilizes. For example, on the Wikipedia, AP, MRR, and HR@10 improve significantly as k_s grows from 10 to 30, indicating that incorporating more globally influential nodes enriches the structural context and enhances link prediction. However, when k_s exceeds 30, the performance gains diminish, suggesting that the most impactful structural information has already been captured.

4.4.3 Hit Ratio@N. We compare the EAGLE model with the most effective baselines, including GMixer [5], Zebra [24], and DyGFormer [55] on HR@N by varying $N \in \{10, 20, 30, 40, 50\}$. Due to similar observations, we present results on the Wikipedia and AskUbuntu datasets. As shown in Figure 6 (a) and (b), the results



(a) Wikipedia. (b) AskUbuntu.

Figure 6: Hit Ratio@N.

demonstrate that EAGLE consistently achieves higher HR@N values compared to the baselines, highlighting its superior ability to prioritize true positive nodes in its predictions. EAGLE can balance short-term temporal recency and long-term global structural patterns to deliver accurate and efficient predictions.

5 CONCLUSION

We proposed EAGLE, an efficient and effective framework for temporal link prediction in dynamic graphs. EAGLE integrates short-term temporal recency and long-term global structural patterns through a time-aware module and a structure-aware module. Extensive experiments on seven real-world datasets show that EAGLE outperforms various T-GNNs and is over 50 times faster than state-of-the-art transformer-based models.

ACKNOWLEDGMENTS

Prof. Lei Chen's work is supported by National Key Research and Development Program of China Grant No. 2023YFF0725100, National Science Foundation of China (NSFC) under Grant No. U22B2060, Guangdong-Hong Kong Technology Innovation Joint Funding Scheme Project No. 2024A0505040012, the Hong Kong RGC GRF Project 16213620, RIF Project R6020-19, AOE Project AoE/E-603/18, Theme-based project TRS T41-603/20R, CRF Project C2004-21G, Key Areas Special Project of Guangdong Provincial Universities 2024ZDZX1006, Guangdong Province Science and Technology Plan Project 2023A0505030011, Guangzhou municipality big data intelligence key lab, 2023A03J0012, Hong Kong ITC ITF grants MHX/078/21 and PRP/004/22FX, Zhujiang scholar program 2021JC02X170, Microsoft Research Asia Collaborative Research Grant, HKUST-Webank joint research lab and 2023 HKUST Shenzhen-Hong Kong Collaborative Innovation Institute Green Sustainability Special Fund, from Shui On Xintiandi and the InnoSpace GBA. Prof. Qing Li is supported by GRF Project 15200023 and RIF Project R1015-23. Prof. Zhang Chen is supported by P0045948 and P0046453 (Accel Group Holding Limited and Minshang Creative Technology Holdings Limited Donations), P0046701 and P0046703 (PolyU internal funding), P0048183 and P0048191 (Research Matching Grant funded by UGC), P0048887 (ITF-ITSP, ITS/028/22FP), P0048984 (Postdoc Matching Fund), P0051906 (RGC Early Career Scheme, 25600624), and P0054482 (Two Square Capital Limited Donation). Dr. Haoyang Li is supported by research funding P0052504, P0057770, and P0053707.

REFERENCES

[1] Xin-Wei Cai, Xiangyu Ke, Kai Wang, Lu Chen, Tianming Zhang, Qing Liu, and Yunjun Gao. 2023. Efficient Temporal Butterfly Counting and Enumeration on Temporal Bipartite Graphs. Proc. VLDB Endow. 17, 4 (2023), 657–670. https://doi.org/10.14778/3636218.3636223

[2] Chaoyi Chen, Dechao Gao, Yanfeng Zhang, Qiange Wang, Zhenbo Fu, Xuecang Zhang, Junhua Zhu, Yu Gu, and Ge Yu. 2023. NeutronStream: A Dynamic GNN Training Framework with Sliding Window for Graph Streams. Proc. VLDB Endow. 17, 3 (Nov. 2023), 455–468. https://doi.org/10.14778/3632093.3632108

[3] Huiyuan Chen and Jing Li. 2018. Exploiting structural and temporal evolution in dynamic link prediction. In Proceedings of the 27th ACM International conference on information and knowledge management. 427–436.

[4] Xin Chen, Jieming Shi, You Peng, Wenqing Lin, Sibo Wang, and Wenjie Zhang. 2024. Minimum Strongly Connected Subgraph Collection in Dynamic Graphs. Proceedings of the VLDB Endowment 17, 6 (2024), 1324–1336.

[5] Weilin Cong, Si Zhang, Jian Kang, Baichuan Yuan, Hao Wu, Xin Zhou, Hanghang Tong, and Mehrdad Mahdavi. 2023. Do We Really Need Complicated Model Architectures For Temporal Networks?. In The Eleventh International Conference on Learning Representations.

[6] Ariel Debrouvier, Eliseo Parodi, Matías Perazzo, Valeria Soliani, and Alejandro Vaisman. 2021. A model and query language for temporal graph databases. The VLDB Journal 30, 5 (2021), 825–858.

[7] Wenfei Fan, Ruochun Jin, Ping Lu, Chao Tian, and Ruiqi Xu. 2022. Towards event prediction in temporal graphs. Proceedings of the VLDB Endowment 15, 9 (2022), 1861–1874.

[8] Chen Gao, Yu Zheng, Nian Li, Yinfeng Li, Yingrong Qin, Jinghua Piao, Yuhan Quan, Jianxin Chang, Depeng Jin, Xiangnan He, et al. 2023. A survey of graph neural networks for recommender systems: Challenges, methods, and directions. ACM Transactions on Recommender Systems 1, 1 (2023), 1–51.

[9] Shihong Gao, Yiming Li, Yanyan Shen, Yingxia Shao, and Lei Chen. 2024. ETC: Efficient Training of Temporal Graph Neural Networks over Large-scale Dynamic Graphs. Proceedings of the VLDB Endowment 17, 5 (2024), 1060–1072.

[10] Shihong Gao, Yiming Li, Xin Zhang, Yanyan Shen, Yingxia Shao, and Lei Chen. 2024. SIMPLE: Efficient Temporal Graph Neural Network Training at Scale with Dynamic Data Placement. Proceedings of the ACM on Management of Data 2, 3 (2024), 1–25.

[11] Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Predict then propagate: Graph neural networks meet personalized pagerank. arXiv preprint arXiv:1810.05997 (2018).

[12] Julia Gastinger, Shenyang Huang, Mikhail Galkin, Erfan Loghmani, Ali Parviz, Farimah Poursafaei, Jacob Danovitch, Emanuele Rossi, Ioannis Koutis, Heiner Stuckenschmidt, Reihaneh Rabbany, and Guillaume Rabusseau. 2024. TGB 2.0: A Benchmark for Learning on Temporal Knowledge Graphs and Heterogeneous Graphs. Advances in Neural Information Processing Systems (2024).

[13] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. 855–864.

[14] Haixing Huang, Jinghe Song, Xuelian Lin, Shuai Ma, and Jinpeng Huai. 2016. Tgraph: A temporal graph data management system. In Proceedings of the 25th ACM International on Conference on Information and Knowledge Management. 2469–2472.

[15] Qiang Huang, Xin Wang, Susie Xi Rao, Zhichao Han, Zitao Zhang, Yongjun He, Quanqing Xu, Yang Zhao, Zhigao Zheng, and Jiawei Jiang. 2024. Benchtemp: A General Benchmark for Evaluating Temporal Graph Neural Networks. In 2024 IEEE 40th International Conference on Data Engineering (ICDE). 4044–4057.

[16] Qiang Huang, Xiao Yan, Xin Wang, Susie Xi Rao, Zhichao Han, Fangcheng Fu, Wentao Zhang, and Jiawei Jiang. 2024. Retrofitting Temporal Graph Neural Networks with Transformer. arXiv preprint arXiv:2409.05477 (2024).

[17] Shenyang Huang, Farimah Poursafaei, Jacob Danovitch, Matthias Fey, Weihua Hu, Emanuele Rossi, Jure Leskovec, Michael Bronstein, Guillaume Rabusseau, and Reihaneh Rabbany. 2024. Temporal graph benchmark for machine learning on temporal graphs. Advances in Neural Information Processing Systems 36 (2024).

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

[19] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. 2010. Governance in social media: A case study of the Wikipedia promotion process. In Proceedings of the International AAAI Conference on Web and Social Media, Vol. 4. 98–105.

[20] Haoyang Li and Lei Chen. 2021. Cache-based gnn system for dynamic graphs. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management. 937–946.

[21] Haoyang Li and Lei Chen. 2023. Early: Efficient and reliable graph neural network for dynamic graphs. Proceedings of the ACM on Management of Data 1, 2 (2023), 1–28.

[22] Haoyang Li, Yuming Xu, Yiming Li, Hanmo Liu, Li Darian, Zhang Chen Jason, Lei Chen, and Qing Li. 2025. When Speed meets Accuracy: an Efficient and Effective Graph Model for Temporal Link Prediction-Techniqcal Report. Online (2025). https://drive.google.com/drive/folders/1SDvudNBcyQpLdK6cw7-PMfoM_ZpHxtwg?usp=drive_link

[23] Yiming Li, Yanyan Shen, Lei Chen, and Mingxuan Yuan. 2023. Orca: Scalable temporal graph neural network training with theoretical guarantees. Proceedings of the ACM on Management of Data 1, 1 (2023), 1–27.

[24] Yiming Li, Yanyan Shen, Lei Chen, and Mingxuan Yuan. 2023. Zebra: When temporal graph neural networks meet temporal personalized PageRank. Proceedings of the VLDB Endowment 16, 6 (2023), 1332–1345.

[25] Yiming Li, Yanyan Shen, Lei Chen, and Mingxuan Yuan. 2024. A Caching-based Framework for Scalable Temporal Graph Neural Network Training. ACM Transactions on Database Systems (2024).

[26] Zhao Li, Xin Shen, Yuhang Jiao, Xuming Pan, Pengcheng Zou, Xianling Meng, Chengwei Yao, and Jiajun Bu. 2020. Hierarchical bipartite graph neural networks: Towards large-scale e-commerce applications. In 2020 IEEE 36th International Conference on Data Engineering (ICDE). IEEE, 1677–1688.

[27] Zhao Li, Haishuai Wang, Peng Zhang, Pengrui Hui, Jiaming Huang, Jian Liao, Ji Zhang, and Jiajun Bu. 2021. Live-streaming fraud detection: A heterogeneous graph neural network approach. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. 3670–3678.

[28] Zachary C. Lipton, John Berkowitz, and Charles Elkan. 2015. A Critical Review of Recurrent Neural Networks for Sequence Learning. arXiv:1506.00019 [cs.LG] https://arxiv.org/abs/1506.00019

[29] Antonio Longa, Veronica Lachi, Gabriele Santin, Monica Bianchini, Bruno Lepri, Pietro Lió, Franco Scarselli, and Andrea Passerini. 2023. Graph Neural Networks for temporal graphs: State of the art, open challenges, and opportunities. ArXiv abs/2302.01018 (2023). https://api.semanticscholar.org/CorpusID:256503594

[30] Yunkai Lou, Chaokun Wang, Tiankai Gu, Hao Feng, Jun Chen, and Jeffrey Xu Yu. 2023. Time-topology analysis on temporal graphs. The VLDB Journal 32, 4 (2023), 815–843.

[31] Yuanfu Lu, Xiao Wang, Chuan Shi, Philip S Yu, and Yanfang Ye. 2019. Temporal network embedding with micro-and macro-dynamics. In Proceedings of the 28th ACM international conference on information and knowledge management. 469–478.

[32] Ibomoiye Domor Mienye, Theo G Swart, and George Obaido. 2024. Recurrent neural networks: A comprehensive review of architectures, variants, and applications. Information 15, 9 (2024), 517.

[33] Giang Hoang Nguyen, John Boaz Lee, Ryan A Rossi, Nesreen K Ahmed, Eunyee Koh, and Sungchul Kim. 2018. Continuous-time dynamic network embeddings. In Companion proceedings of the the web conference 2018. 969–976.

[34] Ashwin Paranjape, Austin R Benson, and Jure Leskovec. 2017. Motifs in temporal networks. In Proceedings of the tenth ACM international conference on web search and data mining. 601–610.

[35] 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 Proceedings of the AAAI conference on artificial intelligence, Vol. 34. 5363–5370.

[36] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. 701–710.

[37] Farimah Poursafaei, Shenyang Huang, Kellin Pelrine, and Reihaneh Rabbany. 2022. Towards better evaluation for dynamic link prediction. Advances in Neural Information Processing Systems 35 (2022), 32928–32941.

[38] Hongchao Qin, Rong-Hua Li, Ye Yuan, Guoren Wang, Lu Qin, and Zhiwei Zhang. 2022. Mining bursting core in large temporal graphs. Proceedings of the VLDB Endowment (2022).

[39] 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).

[40] Aneesh Sharma, Jerry Jiang, Praveen Bommannavar, Brian Larson, and Jimmy Lin. 2016. GraphJet: Real-time content recommendations at Twitter. Proceedings of the VLDB Endowment 9, 13 (2016), 1281–1292.

[41] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of graph neural network evaluation. arXiv preprint arXiv:1811.05868 (2018).

[42] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In Proceedings of the 24th international conference on world wide web. 1067–1077.

[43] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. 2019. Dyrep: Learning representations over dynamic graphs. In International conference on learning representations.

[44] A Vaswani. 2017. Attention is all you need. Advances in Neural Information Processing Systems (2017).

[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention Is All You Need. arXiv:1706.03762 [cs.CL] https://arxiv.org/abs/1706.03762

[46] Lu Wang, Xiaofu Chang, Shuang Li, Yunfei Chu, Hui Li, Wei Zhang, Xiaofeng He, Le Song, Jingren Zhou, and Hongxia Yang. 2021. Tcl: Transformer-based dynamic graph modelling via contrastive learning. arXiv preprint arXiv:2105.07944 (2021).

[47] Peng Wang, BaoWen Xu, YuRong Wu, and XiaoYu Zhou. 2014. Link prediction in social networks: the state-of-the-art. arXiv preprint arXiv:1411.5118 (2014).

[48] Xuhong Wang, Ding Lyu, Mengjian Li, Yang Xia, Qi Yang, Xinwen Wang, Xinguang Wang, Ping Cui, Yupu Yang, Bowen Sun, et al. 2021. Apan: Asynchronous propagation attention network for real-time temporal graph embedding. In Proceedings of the 2021 international conference on management of data. 2628–2638.

[49] Yanbang Wang, Yen-Yu Chang, Yunyu Liu, Jure Leskovec, and Pan Li. 2022. Inductive Representation Learning in Temporal Networks via Causal Anonymous Walks. arXiv:2101.05974 [cs.LG] https://arxiv.org/abs/2101.05974

[50] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. IEEE transactions on neural networks and learning systems 32, 1 (2020), 4–24.

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

[52] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation learning on graphs with jumping knowledge networks. In International conference on machine learning. PMLR, 5453–5462.

[53] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. 2016. Revisiting semi-supervised learning with graph embeddings. In International conference on machine learning. PMLR, 40–48.

[54] Lu Yi, Jie Peng, Yanping Zheng, Fengran Mo, Zhewei Wei, Yuhang Ye, Yue Zixuan, and Zengfeng Huang. 2025. TGB-Seq Benchmark: Challenging Temporal GNNs with Complex Sequential Dynamics. In The Thirteenth International Conference on Learning Representations. https://openreview.net/forum?id=8e2LirwiJT

[55] Le Yu, Leilei Sun, Bowen Du, and Weifeng Lv. 2023. Towards better dynamic graph learning: New architecture and unified library. Advances in Neural Information Processing Systems 36 (2023), 67686–67700.

[56] Song Yu, Shufeng Gong, Qian Tao, Sijie Shen, Yanfeng Zhang, Wenyuan Yu, Pengxi Liu, Zhixin Zhang, Hongfu Li, Xiaojian Luo, et al. 2024. LSMGraph: A High-Performance Dynamic Graph Storage System with Multi-Level CSR. Proceedings of the ACM on Management of Data 2, 6 (2024), 1–28.

[57] Zihao Yu, Ningyi Liao, and Siqiang Luo. 2024. GENTI: GPU-powered Walk-based Subgraph Extraction for Scalable Representation Learning on Dynamic Graphs. Proceedings of the VLDB Endowment 17, 9 (2024), 2269–2278.

[58] Yalong Zhang, Rong-Hua Li, Qi Zhang, Hongchao Qin, Lu Qin, and Guoren Wang. 2024. Efficient Algorithms for Pseudoarboricity Computation in Large Static and Dynamic Graphs. Proceedings of the VLDB Endowment 17, 11 (2024), 2722–2734.

[59] Yanping Zheng, Zhewei Wei, and Jiajun Liu. 2023. Decoupled Graph Neural Networks for Large Dynamic Graphs. Proc. VLDB Endow. 16, 9 (2023), 2239–2247. https://doi.org/10.14778/3598581.3598595

[60] Yanping Zheng, Lu Yi, and Zhewei Wei. 2025. A survey of dynamic graph neural networks. Frontiers of Computer Science 19, 6 (2025), 1–18.

[61] Hongkuan Zhou, Da Zheng, Israt Nisa, Vasileios Ioannidis, Xiang Song, and George Karypis. 2022. Tgl: A general framework for temporal gnn training on billion-scale graphs. arXiv preprint arXiv:2203.14883 (2022).

[62] Yuan Zuo, Guannan Liu, Hao Lin, Jia Guo, Xiaoqian Hu, and Junjie Wu. 2018. Embedding temporal network via neighborhood formation. In Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining. 2857–2866.