Towards Adaptive Neighborhood for Advancing Temporal Interaction Graph Modeling

SIWEI ZHANG, Fudan University, Shanghai, China

XI CHEN, Fudan University, Shanghai, China

YUN XIONG, Fudan University, Shanghai, China

XIXI WU, Fudan University, Shanghai, China

YAO ZHANG, Fudan University, Shanghai, China

YONGRUI FU, Fudan University, Shanghai, China

View all

Published: 25 August 2024

Citation in BibTeX format

KDD '24: The 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining
August 25 - 29, 2024
Barcelona, Spain

Conference Sponsors:
SIGMOD
SIGKDD

Towards Adaptive Neighborhood for Advancing Temporal Interaction Graph Modeling

Siwei Zhang
swzhang22@m.fudan.edu.cn
Shanghai Key Laboratory of Data Science, School of Computer Science, Fudan University
Shanghai, China

Xi Chen
Yun Xiong*
x_chen21@m.fudan.edu.cn
yunx@fudan.edu.cn
Shanghai Key Laboratory of Data Science, School of Computer Science, Fudan University
Shanghai, China

Xixi Wu
21210240043@m.fudan.edu.cn
Shanghai Key Laboratory of Data Science, School of Computer Science, Fudan University
Shanghai, China

Yao Zhang
Yongrui Fu
yaozhang@fudan.edu.cn
23210240154@m.fudan.edu.cn
Shanghai Key Laboratory of Data Science, School of Computer Science, Fudan University
Shanghai, China

Yinglong Zhao
zhaoyinglong.zyl@antgroup.com
Ant Group
Shanghai, China

Jiawei Zhang
jiawei@ifmlab.org
IFM Lab, Department of Computer Science, University of California, Davis
CA, USA

Abstract

Temporal Graph Networks (TGNs) have demonstrated their remarkable performance in modeling temporal interaction graphs. These works can generate temporal node representations by encoding the surrounding neighborhoods for the target node. However, an inherent limitation of existing TGNs is their reliance on *fixed*, hand-crafted rules for neighborhood encoding, overlooking the necessity for an adaptive and learnable neighborhood that can accommodate both personalization and temporal evolution across different timestamps. In this paper, we aim to enhance existing TGNs by introducing an *adaptive* neighborhood encoding mechanism. We present **SEAN** (**S**elective **E**ncoding for **A**daptive **N**eighborhood), a flexible plug-and-play model that can be seamlessly integrated with existing TGNs, effectively boosting their performance. To achieve this, we decompose the adaptive neighborhood encoding process into two phases: (i) representative neighbor selection, and (ii) temporal-aware neighborhood information aggregation. Specifically, we propose the Representative Neighbor Selector component, which automatically pinpoints the most important neighbors for the target node. It offers a tailored understanding of each node's unique surrounding context, facilitating personalization. Subsequently, we propose a Temporal-aware Aggregator, which synthesizes neighborhood aggregation by selectively determining the utilization of aggregation routes and decaying the outdated information, allowing our model to adaptively leverage both the contextually significant and current information during aggregation. We conduct extensive experiments by integrating SEAN into three representative TGNs, evaluating their performance on four public datasets and one financial benchmark dataset introduced in this paper. The results demonstrate that SEAN consistently leads to performance improvements across all models, achieving SOTA performance and exceptional robustness.

CCS Concepts

• **Information systems → Data mining**; • **Computing methodologies → Learning latent representations**; *Neural networks.*

Keywords

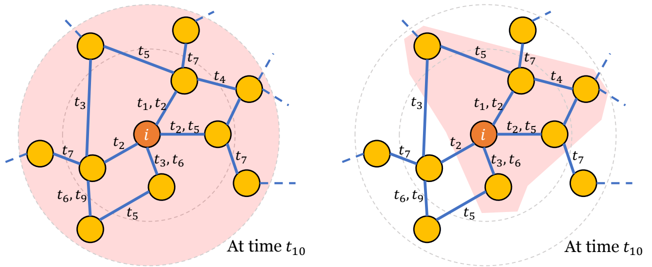
Temporal Graph Networks; Representation Learning; Data Mining

ACM Reference Format:
Siwei Zhang, Xi Chen, Yun Xiong, Xixi Wu, Yao Zhang, Yongrui Fu, Yinglong Zhao, and Jiawei Zhang. 2024. Towards Adaptive Neighborhood for Advancing Temporal Interaction Graph Modeling. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24), August 25–29, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3637528.3671877

1 Introduction

Temporal Interaction Graphs (TIGs) can model the dynamic graph-structured data in many real-world application scenarios, where objects are depicted as nodes and timestamped interactions between them are represented as edges [44]. Unlike static graphs, TIGs exhibit dynamic changes over time. To effectively capture the dynamic nature of TIGs and facilitate representation learning,

*Corresponding author



(a) Existing TGNs encode fixed neighborhoods. **(b) Our model encodes adaptive neighborhoods.**

Figure 1: Comparison between fixed and adaptive neighborhoods during the encoding process. (a) Existing TGNs [19, 37, 44] always adopt fixed rules for neighborhood encoding, *e.g.*, encode 2-hop neighborhoods. (b) We propose adaptive neighborhood encoding to facilitate both the personalized and temporal understanding of a target node.

extensive research has been conducted on developing Temporal Graph Networks (TGNs) [19, 24, 33, 44]. These TGNs can generate temporal node representations by encoding the neighborhoods around the target node, thus enabling downstream predictions.

Despite the remarkable success of existing TGNs, a fundamental weakness inherent in their designs is the reliance on the **fixed**, hand-crafted rules for neighborhood encoding, failing to account for the necessary personalization and the temporal evolution across different timestamps. For example, when generating the representation for a target node at a specific timestamp, as shown in Figure 1a, existing TGNs, like TIGE [44], indiscriminately encode the current 2-hop neighborhood. Different from such existing works, we emphasize that an **adaptive** neighborhood encoding mechanism is crucial for generating more expressive representations.

Personalization. Personalization is essential for TIG representation learning at both graph-scale and node-scale levels. Different TIGs exhibit distinct characteristics [18], and a universal, predefined rule for neighborhood encoding is inflexible [29]. For example, while the k-hop assumption may work well for low-density TIGs, it could lead to indistinguishable representations in denser ones [30]. Additionally, even within a single TIG, the suitable neighborhoods for nodes can differ significantly. This hypothesis is reasonable due to the diverse neighborhood connectivity patterns and potential noise. For instance, in financial networks, transaction patterns of banks arise from different factors [17, 45], and often contain noise irrelevant to their primary financial interests [32, 40].

Temporal evolution. A suitable neighborhood for a target node should adapt to different timestamps to align with the temporal evolution of TIGs, necessitating a temporal-aware design for neighborhood encoding. For example, investors' preferences may change according to economic cycles, such as preferring technology stocks during technological booms while consumer staples during economic downturns. Fixed rules for neighborhood encoding cannot accommodate to such preference shifts in the temporal dimension, leading to suboptimal representations for effectively capturing these changing preferences.

Based on the aforementioned motivations, as depicted in Figure 1b, it becomes critical to allow for a more flexible, scalable, and robust algorithm that performs adaptive neighborhood encoding obeying both personalization and temporal awareness.

In this paper, we aim to enhance existing TGNs by adaptively encoding personalized and temporal-aware neighborhoods through the introduction of a convenient plug-and-play model, referred to as **SEAN** (**S**elective **E**ncoding for **A**daptive **N**eighborhood). Our SEAN can significantly boost existing TGNs, and it comprises two main components: (i) Representative Neighbor Selector. To effectively select personalized neighborhoods for nodes within TIGs, we introduce Representative Neighbor Selector. It refines the neighborhood by choosing representative neighbors who are important to the target node. However, an overemphasis on these neighbors can result in a homogeneous neighborhood that lacks the necessary diversity [6]. Therefore, we incorporate a penalty mechanism that penalizes the over-concentration of neighbors, maintaining a balanced and personalized neighborhood for the target node. (ii) Temporal-aware Aggregator. To achieve temporal-aware neighborhood encoding, we further propose the Temporal-aware Aggregator that automatically aggregates neighborhoods with the temporal understanding of the target node. Specifically, it takes charge of neighborhood aggregation by explicitly determining the utilization of each aggregation route, facilitating the adaptive route aggregation or pruning as needed. Meanwhile, our outdated-decay mechanism strategically de-emphasizes those outdated routes, allowing our model to concentrate on more fresh and up-to-date information.

In summary, our main contributions are:

- We focus on adaptive neighborhood encoding for temporal interaction graph (TIG) modeling and propose SEAN. SEAN is the first model proposed to automate neighborhood encoding in TIG modeling, and it can be easily adopted to boost existing TGNs.
- We develop a Representative Neighbor Selector, which enables the model to effectively choose representative neighbors for the target node, achieving personalized neighborhood encoding.
- We propose a Temporal-aware Aggregator, which aggregates neighborhoods by explicitly determining the utilization of aggregation routes and decaying the outdated information, facilitating temporal-aware neighborhood encoding.
- We conduct extensive experiments on several TIG benchmark datasets to validate SEAN's effectiveness. Furthermore, we introduce **TemFin**, a new financial transaction TIG benchmark dataset in this paper. This benchmark represents a more challenging environment due to its complexity in the financial domain.

2 Related Work

2.1 Temporal Graph Networks (TGNs)

Temporal Graph Networks (TGNs) are designed to generate temporal node representations of TIGs by encoding the neighborhoods for the target node at any given timestamp [3, 40, 41, 43]. They typically encode their neighborhoods based on a fixed, pre-defined rule. According to how these neighborhoods are pre-defined, existing TGNs can be categorized into two types: Breadth-First Search TGNs (BFS-TGNs) and Depth-First Search TGNs (DFS-TGNs).

BFS-TGNs [4, 11, 13, 19, 24, 33, 40, 41, 44] prioritize selecting neighborhoods close to the target node. They then encode these neighborhoods via various aggregation functions, generating representations for the target node. Different from BFS-TGNs, DFS-TGNs [20, 27] utilize some temporal walks starting from the target node as their neighborhoods, and encode these walks through the frequency of node occurrences. Currently, DyGFormer [37], one of BFS-TGNs, encodes the first-order neighborhoods through Transformer blocks, achieving SOTA results among existing TGNs.

Different from existing TGNs, our model goes beyond fixed rules for neighborhood encoding. It adaptively encodes the neighborhoods that contribute most to the representations, thus enhancing the model's overall effectiveness.

2.2 Adaptive Neighborhood for Graph Learning

Graph learning with adaptive neighborhoods aims to encode undetected or enhanced neighborhoods for better representation learning, which has achieved remarkable performance on static graphs.

Early works [7, 14, 39, 46] encode adaptive neighborhoods by utilizing Reinforcement Learning (RL). However, these works face significant limitations due to the immense search space and the non-differentiability of models, resulting in substantial computational overhead. To address this issue, GeniePath [15] employs a differentiable module for adaptive neighborhood encoding, reducing the model complexity and achieving better performance. Subsequently, an increasing number of adaptive neighborhood encoding methods [10, 16, 22, 28, 29, 31, 35, 47] have been proposed, significantly empowering the representation learning on static graphs.

These existing methods cannot be directly adopted in TIGs, primarily because TIGs exhibit significant temporal differences compared to static graphs. For instance, TIGs are consistently characterized by a sequence of timestamped interactions, rather than by adjacency matrices that are typical in static graphs, due to the dynamic nature of TIGs.

3 Preliminaries

3.1 Problem Formulation

Definition 3.1. **Temporal Interaction Graph.** Given[1] a node set $\mathcal{V} = \{1, ..., |\mathcal{V}|\}$, a temporal interaction graph can be represented as a sequence of edge set $\mathcal{E} = \{(i, j, t)\}$, where $i, j \in \mathcal{V}$ and $t > 0$. Each edge (i, j, t) denotes an interaction between node i and node j that happened at time t with the feature $\mathbf{e}_{ij}(t)$. The edge feature vector can depict various information about the interactions among nodes, *e.g.*, interaction type. Any edge $(i, j, t) \in \mathcal{E}$ only has access to its historical data before time t, *i.e.*, $\{(i, j, \tau) \in \mathcal{E} | \tau < t\}$.

Definition 3.2. **Temporal Interaction Graph Modeling.** Given an edge $(i, j, t) \in \mathcal{E}$ and its historical data $\{(i, j, \tau) \in \mathcal{E} | \tau < t\}$, temporal interaction graph modeling aims to learn a mapping function $f : (i, j, t) \mapsto \mathbf{z}_i(t), \mathbf{z}_j(t)$, where $\mathbf{z}_i(t), \mathbf{z}_j(t) \in \mathbb{R}^d$ represent the representations of nodes i and j at time t, respectively, and d is the representation vector dimension.

[1]For simplicity, in this paper, we can just represent a node by its index, *i.e.*, an integer.

3.2 Temporal Embedding Module

As a crucial component within most existing TGNs, temporal embedding module [44] is responsible for generating temporal representations for any given node i at time t, $\mathbf{z}_i(t)$, which involves a K-layer temporal attention network to encode i's current neighborhood. In this section, we generalize various temporal embedding modules employed in different TGNs into a cohesive framework. This unification allows us to provide a comprehensive overview of the neighborhood encoding process within TGNs.

For the target node i at time t of the k-th layer, $k \in \{1, ..., K\}$, temporal embedding module aggregates i's neighborhood information $\widetilde{\mathbf{h}}_i^{(k)}(t)$ and i's representation from the previous layer $\mathbf{h}_i^{(k-1)}(t)$ using a Multi-Layer Perceptron (MLP) as the aggregator:

$$\mathbf{h}_i^{(k)}(t) = \mathrm{MLP}^{(k)}\left(\mathbf{h}_i^{(k-1)}(t) \| \, \widetilde{\mathbf{h}}_i^{(k)}(t)\right), \tag{1}$$

where $\|$ denotes the concatenation operation. The neighborhood information $\widetilde{\mathbf{h}}_i^{(k)}$ is obtained through an attention mechanism [33], which involves assigning attention scores to i's neighborhood:

$$\widetilde{\mathbf{h}}_i^{(k)}(t) = \mathrm{Softmax}\left(\mathbf{a}_i^{(k)}(t)\right) \cdot \mathbf{V}_i^{(k)}(t), \tag{2}$$

where $\mathbf{a}_i^{(k)}(t) = [a_{ij}^{(k)}(t)]_{j \in \mathcal{N}_i(t)}$ represents the attention vector of node i, and the matrix $\mathbf{V}_i^{(k)}(t) = [\mathbf{v}_{ij}^{(k)}(t)]_{j \in \mathcal{N}_i(t)}$ encapsulates the messages from i's neighborhood. Each element $a_{ij}^{(k)}(t)$ is the attention score from node i to its neighboring node $j \in \mathcal{N}_i(t)$, which is computed as follows:

$$a_{ij}^{(k)}(t) = \frac{f_{\mathrm{q}}\left(\mathbf{h}_i^{(k-1)}(t)\right) f_{\mathrm{k}}\left(\mathbf{h}_j^{(k-1)}(t)\right)^T}{\sqrt{d^{(k)}}}, \tag{3}$$

and each row $\mathbf{v}_{ij}^{(k)}(t)$ is the message carried from neighbor j, which is determined by $\mathbf{v}_{ij}^{(k)}(t) = f_{\mathrm{v}}(\mathbf{h}_j^{(k-1)}(t))$. In the above equations, $f_*(\cdot)(* \in \{\mathrm{q}, \mathrm{k}, \mathrm{v}\})$ represents the encoding functions for queries, keys, and values, respectively [25, 36]. These functions may have different specific representations for different TGNs [19, 33, 44]. For simplicity, we consider a single-head attention formula in this paper. By incorporating personalized and temporal-aware designs into the neighborhood attention mechanism (Equation 3) and the aggregation process (Equation 1), we can elevate this module to achieve adaptive neighborhood encoding. This innovative design enables our model to serve as a seamlessly integratable plug-and-play enhancement for various existing TGNs, broadening their flexibility and effectiveness.

4 Methodology

As we have mentioned, adaptive neighborhood encoding requires both personalized and temporal considerations. Therefore, we decompose this process into two phases, *i.e.*, a Representative Neighbor Selector first assigns distinctive attention to select important neighbors for personalization, and our Temporal-aware Aggregator then aggregates the neighborhood information to ensure temporal relevance. We will introduce these components in the following subsections.

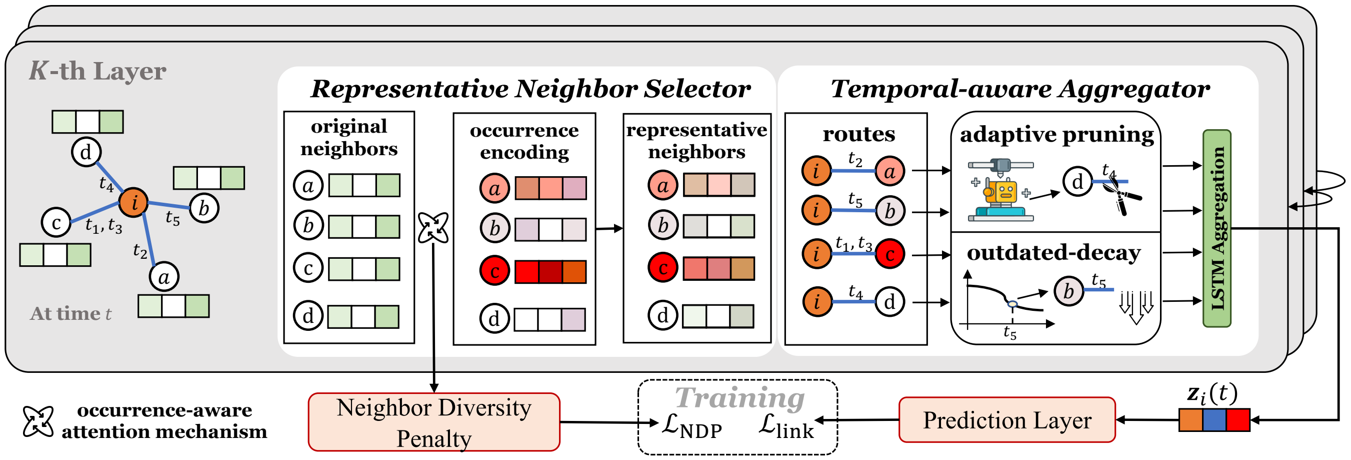


Figure 2: Framework of the proposed plug-and-play model. Our Representative Neighbor Selector can empower the model to pinpoint the important neighbors, who then act as the personalized representatives for the target node. Meanwhile, we propose the neighbor diversity penalty to penalize the over-concentration of these neighbors, thus maintaining a more balanced neighborhood. Furthermore, we conduct our Temporal-aware Aggregator by LSTM aggregation, where we propose an adaptive pruning module that explicitly determines whether to aggregate information from the given route or to prune it as needed, and an outdated-decay mechanism that de-emphasizes the outdated information. Finally, we generate the temporal node representations for downstream tasks by extracting the encoded neighborhood information from the K-th layer of SEAN.

4.1 Representative Neighbor Selector

We propose the occurrence-aware attention and neighbor diversity penalty mechanisms to select personalized representative neighborhoods for the target node.

4.1.1 Occurrence-aware Attention Mechanism. In Equation 3, the attention score is calculated based on the semantic correlation between the target node and its neighboring nodes. However, this semantics-based attention computation may be suboptimal to select representative neighborhoods due to its lack of personalization and adaptability [33, 36]. Here, we propose the occurrence-aware attention mechanism to enhance the traditional semantics-based attention mechanism for selecting important neighbors as personalized representatives. Intuitively, neighbor occurrence, which quantifies how frequently a neighbor occurs, is a significant signal in identifying the importance of each neighbor for a given target node. Frequently occurring neighbors are more likely to be important and possess more relevant information, thus making them more effective and convincing representatives.

Formally, for node i at time t and its historical neighbors $\mathcal{N}_i(t)$, we first count how often each neighbor has occurred historically and derive it to a one-dimensional occurrence frequency feature, which is represented by $\mathbf{f}_i(t) \in \mathbb{R}^{|\mathcal{N}_i(t)|\times 1}$. To balance the raw counts, we then normalize this feature by dividing its temporal node degree $\mathbf{d}_i(t) \in \mathbb{R}^{|\mathcal{N}_i(t)|\times 1}$, which is denoted as:

$$\hat{\mathbf{f}}_i(t) = \mathbf{f}_i(t) \odot \mathbf{d}_i^{-1}(t) \in \mathbb{R}^{|\mathcal{N}_i(t)|\times 1}, \tag{4}$$

where $\odot$ denotes the element-wise product operation. The normalized occurrence frequency feature vector both considers the interaction frequency and mitigates the unexpected adverse impacts from the hub nodes (*i.e.*, those with large degrees), which are known as "degree-related biases" [23, 42]. To enhance the expressiveness of the occurrence information, we apply a function $f_e(\cdot)$ to encode our normalized occurrence frequency feature $\hat{\mathbf{f}}_i(t)$ into a high-dimensional occurrence encoding as follows:

$$\mathbf{R}_i(t) = f_e\left(\hat{\mathbf{f}}_i(t)\right) \in \mathbb{R}^{|\mathcal{N}_i(t)|\times d}. \tag{5}$$

Similar to DyGFormer [37], we implement $f_e(\cdot)$ by a two-layer MLP whose input and output dimensions are 1 and d, respectively.

To further enrich the occurrence-aware attention computation, we incorporate both edge attributes and temporal information. For each neighbor node $j \in \mathcal{N}_i(t)$, we retrieve its occurrence encoding as $\mathbf{r}_{ij}(t) = \mathbf{R}_{i,j:}(t) \in \mathbb{R}^d$. Our occurrence-aware attention score from node i to node j at time t, $\tilde{a}_{ij}(t)$, can be represented by:

$$\tilde{a}_{ij}(t) = \tanh\left(\mathbf{w}_i \cdot [\mathbf{r}_{ij}(t) \| \mathbf{e}_{ij}(t) \| \phi(t-t_j)]\right), \tag{6}$$

where $\phi(\cdot)$ is the commonly used time encoding function in TGNs [33, 44], and $\mathbf{w}_i \in \mathbb{R}^{1\times 3d}$ represents the trainable reshaping vector.

Finally, we update the traditional attention score $a_{ij}(t)$ with our new occurrence-aware attention score $\tilde{a}_{ij}(t)$ to obtain the final attention score as $\hat{a}_{ij}(t) = a_{ij}(t) + \tilde{a}_{ij}(t)$. This combined score integrates both semantic relevance and occurrence frequency, enabling our model to select important neighbors as representatives.

4.1.2 Neighbor Diversity Penalty. Although our occurrence-aware attention greatly helps the model in identifying the most important neighbors as the representatives for the target node, an overemphasis on these neighbors can result in a homogeneous neighborhood that lacks the necessary diversity. For example, in the world of finance, an abundance of homogeneous investment choices may lead to the undesirable phenomenon known as the "Financial Echo Chamber" [8], analogous to the "Information Cocoon" in Recommender Systems [21, 34]. To address this issue, we propose an additional neighbor diversity penalty mechanism to encourage the selection of more diverse neighbors. By leveraging this mechanism, our approach aims to strike a balance between prioritizing important neighbors and ensuring their diversity, thereby enhancing the overall effectiveness of our model.

Let $\mathbb{I}(\cdot)$ be the indication function [9], and $\hat{a}_{ij}^{(k)}$ represents the occurrence-aware attention score from node i to its neighbor $j \in \mathcal{N}_i(t)$ in the k-th layer[2]. To ensure that our model penalizes the over-important neighbors with excessively high attention scores, we first employ a score-filtering technique in the attention scores based on a threshold $\tau \in [0, 1]$, beyond which the scores are preserved. We have:

$$\hat{a}_{ij}^{(k)} = \mathbb{I}\left(\hat{a}_{ij}^{(k)} \geq \tau\right) \cdot \hat{a}_{ij}^{(k)}. \tag{7}$$

Subsequently, we calculate the average score within i's neighbors by $\bar{a}_i^{(k)} := \frac{1}{N}\sum_j \hat{a}_{ij}^{(k)}$ where $N = |\mathcal{N}_i(t)|$ is the total number of neighbors. Then, we compute the product similarity between i's neighbors' attention scores and their corresponding average score by $s_i^{(k)} = \frac{1}{N}\sum_j \hat{a}_{ij}^{(k)} \cdot \bar{a}_i^{(k)}$. Finally, we minimize the dissimilarity across nodes and layers, promoting alignment and coherence within our attention mechanism as follows:

$$\mathcal{L}_{\text{NDP}} = -\frac{1}{K}\sum_{k=1}^{K}\mathcal{L}_{\text{NDP}}^{(k)}, \quad \mathcal{L}_{\text{NDP}}^{(k)} = \frac{1}{|\mathcal{V}|}\sum_{i\in\mathcal{V}} s_i^{(k)}. \tag{8}$$

$\mathcal{L}_{\text{NDP}}$ will serve as a controllable auxiliary loss in our model. This penalty method helps our model to prevent the over-concentration of just a few neighbors, ensuring a balanced neighborhood.

4.2 Temporal-aware Aggregator

To incorporate temporal understanding, we introduce a Temporal-aware Aggregator to replace the initial aggregator $\text{MLP}^{(k)}(\cdot)$ in Equation 1. Specifically, we implement this module using LSTM cells [38], chosen for its well-known strengths in temporal awareness:

$$\mathbf{h}^{(k)} = \text{AGG}^{(k)}\left(\mathbf{h}^{(k-1)}\right) = \text{LSTM}^{(k)}\left(\mathbf{h}^{(k-1)}, \widetilde{\mathbf{h}}^{(k)}, \mathbf{c}^{(k)}\right). \tag{9}$$

Here, we fetch the hidden state from the k-th LSTM cell $\mathbf{h}^{(k)}$ as our **output** representation of the k-th aggregation layer $\text{AGG}^{(k)}(\cdot)$. The **inputs** are three folds: the output representation (*i.e.*, hidden state) from the previous layer $\mathbf{h}^{(k-1)}$, the refined neighborhood information $\widetilde{\mathbf{h}}^{(k)}$ that incorporates our proposed attention mechanism, and the cell state from the k-th LSTM $\mathbf{c}^{(k)}$. To fully achieve temporal-aware encoding, we enhance our LSTM aggregator through an adaptive pruning module to facilitate selective aggregates of $\mathbf{h}^{(k)}$, and an outdated-decay module to enhance the temporal sensitivity of the cell state $\mathbf{c}^{(k)}$. Note that we simplify our notation by omitting the subscript term i and the time t. In practice, for node i at time t, we determine its temporal representation by extracting the output representation from the last layer, *i.e.*, $\mathbf{z}_i(t) = \mathbf{h}_i^{(K)}(t)$.

4.2.1 *Adaptive Pruning Module.*

This module not only encourages our model to focus on incorporating more crucial information for performance improvement (Section 5.7) but also enhances the model interpretability, which explicitly specifies the model's decision-making process in seeking adaptive neighborhoods (Section 5.6).

[2]For clarity, we omit the timestamp term t in the remaining part of our paper, unless specified otherwise.

Formally, we introduce a layer state $\tilde{u}^{(k)} \in [0, 1]$ for each k-th layer, which represents the probability that the current route will either be aggregated or pruned. We have:

$$u^{(k)} = f_{\text{round}}\left(\tilde{u}^{(k)}\right), \tag{10}$$

where $f_{\text{round}}(\cdot)$ denotes the binarization function derived from the rounding operation and $u^{(k)} \in \{0, 1\}$. Then, $u^{(k)}$ is utilized to determine whether the current route information will be aggregated ($u^{(k)} = 1$) or copied from the previous layer ($u^{(k)} = 0$). We have:

$$\mathbf{h}^{(k)} = u^{(k)} \cdot \mathbf{h}^{(k)} + \left(1 - u^{(k)}\right) \cdot \mathbf{h}^{(k-1)}, \tag{11}$$

Afterward, we update the following layer state $\tilde{u}^{(k+1)}$ using the aggregated/copied information from the k-th layer $\mathbf{h}^{(k)}$ as follows:

$$\Delta\tilde{u}^{(k)} = \sigma\left(\mathbf{W}_p \cdot \mathbf{h}^{(k)} + \mathbf{b}_p\right), \tag{12}$$

$$\tilde{u}^{(k+1)} = u^{(k)} \cdot \Delta\tilde{u}^{(k)} + \left(1 - u^{(k)}\right) \cdot \left(\tilde{u}^{(k)} + \max\left(\Delta\tilde{u}^{(k)}, 1 - \tilde{u}^{(k)}\right)\right), \tag{13}$$

where $\mathbf{W}_p \in \mathbb{R}^{d\times d}$ is the learnable matrix, $\mathbf{b}_p$ is the learnable vector, and $\sigma(\cdot)$ is the sigmoid function. During aggregation, the aggregated neighborhood progressively becomes closer to the central node as the number of aggregation layers increases. It means that higher aggregation layers aggregate more central neighborhoods that have been proven to hold more vital information [37]. This formulation encodes the observation that the likelihood of aggregation increases with the number of aggregation layers. Whenever the k-th layer aggregation is omitted, the pre-activation of the layer state for the following layer $\tilde{u}^{(k+1)}$ is incremented by $\Delta\tilde{u}^{(k)}$. Conversely, if the k-th layer aggregation is performed, the accumulated value will be reset and $\tilde{u}^{(k+1)} = \Delta\tilde{u}^{(k)}$. In this way, our model can explicitly determine the route aggregation or pruning as needed.

4.2.2 *Outdated-decay Module.*

Outdated-decay module aims to de-emphasize those outdated aggregation routes, ensuring the incorporation of more fresh and up-to-date information.

We believe that the long-term cell state retains important and more enduring information, while the short-term cell state is utilized for processing immediate information. Therefore, we first decompose the k-th layer cell sate $\mathbf{c}^{(k)}$ into two elements, *i.e.*, the short-term cell state $\mathbf{c}_s^{(k)}$ and the long-term cell state $\mathbf{c}_l^{(k)}$. We have:

$$\mathbf{c}_s^{(k)} = \tanh\left(\mathbf{W}_d \cdot \mathbf{c}^{(k)} + \mathbf{b}_d\right), \tag{14}$$

$$\mathbf{c}_l^{(k)} = \mathbf{c}^{(k)} - \mathbf{c}_s^{(k)}, \tag{15}$$

where matrix $\mathbf{W}_d \in \mathbb{R}^{d\times d}$ and vector $\mathbf{b}_d$ are learnable parameters. The short-term cell state $\mathbf{c}_s^{(k)}$ is initially produced by a neural network with the activation function. Subsequently, the long-term cell state $\mathbf{c}_l^{(k)}$ can be distinguished from the original cell state $\mathbf{c}^{(k)}$.

To prevent from forgetting too rapidly, we keep our long-term cell state $\mathbf{c}_l^{(k)}$ untouched and forget the short-term cell state $\mathbf{c}_s^{(k)}$ according to the time interval $\Delta_t = t - t^-$ between the current time t and the last interaction time of the neighbor in the corresponding route t^-. We apply a decay mechanism as follows:

$$\hat{\mathbf{c}}_s^{(k)} = \mathbf{c}_s^{(k)} \cdot g\left(\Delta_t\right). \tag{16}$$

Algorithm 1: Traning SEAN (one epoch).

```
input  : Temporal interaction graph edge set $\mathcal{E}$;
         Aggregation layer $K$; NDP controlling parameter $\lambda$.
1  Initialize all model parameters;
2  foreach $batch \subseteq \mathcal{E}$ do
3      foreach $k = 1, 2, ..., K$ do
4          Retrieve neighborhood for target node $i$;
5          Compute occurrence-aware attention score by Equation 6;
6          Compute the final score $\hat{\mathbf{a}}_i^{(k)}$ and refine neighborhood information $\widetilde{\mathbf{h}}_i$;
7          Apply outdated-decay by Equation 14-17;
8          Round layer state $\hat{u}^{(k)}$ to $u^{(k)}$ as Equation 10;
9          Aggregate neighborhood by Equation 11;
10         Update $\hat{u}^{(k+1)}$ by Equation 13;
11     end
12     Compute NDP loss $\mathcal{L}_{\text{NDP}}$ by Equation 8;
13     Compute loss $\mathcal{L}$ with $\lambda$ by Equation 19 and backward;
14 end
```

The decay function $g(\Delta_t)$ is defined by $\exp\{-2 \cdot \Delta_t / t_{max}\}$, where t_{max} represents the maximum value among all time intervals. Finally, the forgotten short-term cell state $\hat{\mathbf{c}}_s^{(k)}$ and the untouched long-term cell state $\mathbf{c}_l^{(k)}$ are combined to generate the final decayed cell state $\mathbf{c}_*^{(k)}$ by:

$$\mathbf{c}_*^{(k)} = \mathbf{c}_l^{(k)} + \hat{\mathbf{c}}_s^{(k)}, \tag{17}$$

which is the adjusted input cell state to the k-th LSTM cell, generating the output representation for the k-th aggregation layer.

4.3 Training

4.3.1 Error Gradients. The entire model is differentiable except for the round process $f_{\text{round}}(\cdot)$ in Equation 10. In this paper, we employ the straight-through estimator [1, 2] to allow all parameters to be trained efficiently without the need for any extra supervision signals. This involves approximating the step process with identity during gradient computation in the backward pass: $\frac{\partial f_{\text{round}}(x)}{\partial x} = 1$.

4.3.2 Loss Function. We employ temporal link prediction as our self-supervised task for training. Specifically, for each link (i, j, t), we calculate its occurrence probability $\hat{p}_{ij}(t)$ with the concatenated temporal representations of interaction nodes, *i.e.*, $\mathbf{z}_i(t)$ and $\mathbf{z}_j(t)$, through a two-layer MLP [44]. Then, we compute the loss function by the cross entropy of the link prediction task as follows:

$$\mathcal{L}_{\text{link}} = -\sum_{(i,j,t)\in\mathcal{E}} \left[\log \hat{p}_{ij}(t) + \log\left(1 - \hat{p}_{ik}(t)\right)\right], \tag{18}$$

where k is the negative destination node by random sampling. Finally, we consider our NDP loss $\mathcal{L}_{\text{NDP}}$ in Equation 8 for regularizing the learned attention mechanisms in our final training loss:

$$\mathcal{L} = \mathcal{L}_{\text{link}} + \lambda \mathcal{L}_{\text{NDP}}, \tag{19}$$

where $\lambda \in \mathbb{R}^+$ is a controlling hyper-parameter for our penalty method.

4.4 Complexity Analysis

We provide a detailed complexity analysis of SEAN where we show the complexity of both components. The Representative Neighbor Selector module assigns distinct attention scores to nodes and also provides the neighbor diversity penalty loss, leading to the computational complexity of $O(K|\mathcal{V}|d)$, where $K, |\mathcal{V}|, d$ refer to the total number of stacking layers, nodes, and the embedding dimension, respectively. The Temporal-aware Aggregator is implemented based on the LSTM module, therefore leading to a complexity of $O(K|\mathcal{V}|d^2)$. Therefore, the overall complexity of SEAN is $O(K|\mathcal{V}|d + K|\mathcal{V}|d^2)$, maintaining efficiency as it scales *linearly* with the number of nodes.

We want to emphasize that the temporal embedding module mentioned in Section 3.2 in most existing TGNs also incorporates an explicit attention assignment stage. In this case, integrating SEAN *only* leads to an additional overall complexity of $O(K|\mathcal{V}|d^2)$.

5 Experiments

5.1 Experimental Settings

5.1.1 Datasets. We conduct experiments with five datasets, including four public datasets [13] and one dataset introduced in our paper. The four public datasets - Wikipedia, Reddit, MOOC, and LastFM - are widely used in temporal interaction graph modeling. Meanwhile, recognizing that these available datasets are social or event networks, we release TemFin, a new TIG benchmark dataset from the complicated financial domain. TemFin consists of half a month of transactions sampled from a private financial transfer transaction network in the Ant Finance Group.[3] Details of all datasets are described in Table 5 in the Appendix due to the page limitations. All datasets are sequentially split according to the edge timestamp order by 70%, 15%, and 15% for training, validation, and testing, respectively [44].

5.1.2 Baselines. For comparison, we choose nine existing TGNs as our baselines, and summarize the detailed description of our baselines as follows:

- **DyRep** [24]: It is a notable implementation of the temporal point process in its neighborhoods and introduces a projection layer that estimates user representation in the future.
- **JODIE** [13]: This method considers the dynamic representation of nodes changing over time and utilizes two RNNs for temporal interaction graph modeling.
- **TGAT** [33]: TGAT aggregates its neighborhoods by applying a temporal embedding module based on the temporal attention mechanism.
- **TGN** [19]: TGN summarizes the former models and proposes a memory module to record the historical behaviors of nodes, significantly improving the performance of temporal interaction graph modeling.
- **TIGE** [44]: TIGE enhances the TGN framework by incorporating a dual memory module, effectively addressing the issue of staleness and building upon TGN's foundational concepts.
- **GraphMixer** [5]: This approach employs a simple MLP-Mixer for neighborhood encoding and has demonstrated impressive

[3]The dataset is sampled properly only for experiment purposes and does not imply any commercial information. All personal identity information (PII) has been removed.

Table 1: Average Precision (%) results on Temporal Link Prediction task under both transductive and inductive settings. Results with the pink background represent the performance and corresponding improvements achieved by integrating our model into three backbones. The best results are highlighted in bold, and * denotes the benchmark dataset introduced in this paper.

	Transductive setting					**Inductive setting**				
	Wikipedia	Reddit	MOOC	LastFM	TemFin*	Wikipedia	Reddit	MOOC	LastFM	TemFin*
JODIE	94.62 ± 0.5	91.11 ± 0.3	76.50 ± 1.8	68.77 ± 3.0	88.02 ± 0.2	93.11 ± 0.4	94.36 ± 1.1	77.83 ± 2.1	82.55 ± 1.9	75.35 ± 1.6
TGAT	95.34 ± 0.1	98.12 ± 0.2	60.97 ± 0.3	53.36 ± 0.1	87.70 ± 0.1	93.99 ± 0.3	96.62 ± 0.3	63.50 ± 0.7	55.65 ± 0.2	79.37 ± 0.2
DyRep	94.59 ± 0.2	97.98 ± 0.1	75.37 ± 1.7	68.77 ± 2.1	87.99 ± 0.1	92.05 ± 0.3	95.68 ± 0.2	78.55 ± 1.1	81.33 ± 2.1	76.53 ± 0.1
PINT	98.78 ± 0.1	99.03 ± .01	85.14 ± 1.2	88.06 ± 0.7	90.79 ± 0.1	98.38 ± .04	98.25 ± .04	85.39 ± 1.0	91.76 ± 0.7	81.18 ± 0.2
iLoRE	98.98 ± 0.3	99.11 ± 0.4	90.44 ± 1.0	91.39 ± 0.1	90.90 ± 0.1	98.60 ± 0.3	98.65 ± 0.3	89.75 ± 0.8	93.29 ± 0.8	84.33 ± 0.2
GraphMixer	97.95 ± .03	97.31 ± .01	82.78 ± 0.2	67.27 ± 2.1	86.85 ± 0.1	96.65 ± .02	95.26 ± .02	81.41 ± 0.2	82.11 ± 0.4	77.47 ± 0.2
TGN	98.46 ± 0.1	98.70 ± 0.1	85.88 ± 3.0	80.69 ± 0.2	90.65 ± 0.1	97.81 ± 0.1	97.55 ± 0.1	85.55 ± 2.9	84.66 ± 0.1	80.67 ± 0.2
TGN+	98.79 ± 0.1	98.73 ± .02	89.91 ± 1.2	84.15 ± 2.5	91.04 ± 0.1	98.18 ± 0.1	97.69 ± 0.1	88.91 ± 1.6	88.60 ± 1.2	81.30 ± 0.6
(***imprv.***)	(*+0.33*)	(*+0.03*)	(*+4.03*)	(*+3.46*)	(*+0.39*)	(*+0.37*)	(*+0.14*)	(*+3.36*)	(*+3.94*)	(*+0.63*)
TIGE	98.38 ± 0.1	99.04 ± 0.1	89.64 ± 0.9	87.85 ± 0.9	90.81 ± .02	98.45 ± 0.1	98.39 ± 0.1	89.51 ± 0.7	90.14 ± 1.0	83.22 ± 0.2
TIGE+	98.95 ± .04	99.13 ± .02	**92.26 ± 0.8**	91.42 ± 0.7	90.92 ± .01	98.61 ± 0.1	98.50 ± .03	**91.53 ± 0.8**	93.14 ± 0.6	83.79 ± 0.1
(***imprv.***)	(*+0.12*)	(*+0.09*)	**(*+2.62*)**	(*+3.79*)	(*+0.11*)	(*+0.16*)	(*+0.11*)	**(*+2.02*)**	(*+3.08*)	(*+0.57*)
DyGFormer	99.03 ± .02	99.22 ± .01	87.52 ± 0.5	93.00 ± 0.1	93.34 ± 0.2	98.59 ± .03	98.84 ± .02	86.96 ± 0.4	94.23 ± .09	86.29 ± 0.4
DyGFormer+	**99.27 ± .02**	**99.59 ± .01**	88.29 ± 0.5	**94.73 ± 0.1**	**95.81 ± 0.3**	**98.63 ± .04**	**98.91 ± 0.1**	87.20 ± 0.5	**94.89 ± 0.1**	**86.91 ± .04**
(***imprv.***)	**(*+0.24*)**	**(*+0.37*)**	(*+0.77*)	**(*+1.73*)**	**(*+2.47*)**	**(*+0.04*)**	**(*+0.07*)**	(*+0.24*)	**(*+0.66*)**	**(*+0.62*)**

Table 2: AUROC (%) results on Evolving Node Classification task. Due to table restrictions, we omit some baseline results that have been shown inferior to our three backbones [19, 37, 44].

	DyRep	PINT	GraghMixer	TGN	**TGN+**	(***imprv.***)	TIGE	**TIGE+**	(***imprv.***)	DyGFormer	**DyGFormer+**	(***imprv.***)
Wikipedia	84.59 ± 2.2	87.59 ± 0.6	86.80 ± .01	87.81 ± 0.3	**87.91 ± 0.8**	**(*+0.10*)**	86.92 ± 0.7	87.16 ± 1.5	(*+0.24*)	87.07 ± 0.8	87.49 ± 0.6	(*+0.42*)
Reddit	62.91 ± 2.4	67.31 ± 0.2	64.22 ± .03	67.06 ± 0.9	68.26 ± 0.8	(*+1.20*)	69.41 ± 1.3	**70.24 ± 2.2**	**(*+0.83*)**	68.30 ± 1.5	69.06 ± 2.5	(*+0.76*)
MOOC	67.76 ± 0.5	68.77 ± 1.1	67.21 ± .02	69.54 ± 1.0	72.96 ± 1.8	(*+2.04*)	72.35 ± 2.3	73.85 ± 2.0	(*+1.50*)	77.89 ± 0.5	**78.88 ± 3.2**	**(*+0.99*)**
TemFin*	78.56 ± 2.9	82.22 ± 1.2	81.17 ± 0.7	80.93 ± 2.2	82.39 ± 2.5	(*+1.46*)	80.52 ± 3.0	**83.17 ± 2.2**	**(*+2.65*)**	72.50 ± 0.4	73.05 ± 2.1	(*+0.55*)

performance in the Temporal Link Prediction task under the transductive setting.

- **PINT** [20]: PINT utilizes injective temporal message passing on neighborhoods and leverages relative positional features to improve model performance.
- **iLoRE** [41]: Focusing on instant long-term modeling and re-occurrence preservation, iLoRE provides a nuanced approach to TIG modeling.
- **DyGFormer** [37]: This model introduces a novel architecture that performs Transformer blocks through the patch technique, achieving SOTA results in TIG modeling.

We select three representative baselines to serve as the backbones for our SEAN: TGN [19], TIGE [44], and DyGFormer [37]. We then integrate our model into these three backbones and evaluate their performance in two downstream tasks, *i.e.*, Temporal Link Prediction and Evolving Node Classification.

5.2 Temporal Link Prediction

We start our experiments with the Temporal Link Prediction task, which aims to predict the probability of a link occurring between two specific nodes at a certain time. We utilize two settings: the transductive setting, which performs link prediction on nodes that have appeared during training, and the inductive setting, which predicts links between unseen nodes. We randomly sample an equal number of negative nodes as detailed in Equation 18, and report the average precision (AP) performance with all three backbones, *i.e.*, TGN [19], TIGE [44], and DyGFormer [37].

The results are shown in Table 1. Clearly, we can observe that all three backbone models show improved performance across all datasets in both transductive and inductive settings after integrating our model. Meanwhile, among the improved models, at least one has achieved SOTA results, surpassing all baseline comparisons. This observation proves the effectiveness of the adaptive neighborhood encoding mechanism in TGNs. Moreover, we find the most significant improvement between the backbones and their corresponding improved models specifically on MOOC and lastFM. This may be owing to their high neighborhood complexity, constraining the backbones' effectiveness. In contrast, our model introduces an adaptive neighborhood encoding mechanism that effectively empowers these backbones to better capture and utilize the critical information in such challenging environments, demonstrating our model's effectiveness and robustness in handling complex TIGs.

5.3 Evolving Node Classification

We also conduct the Evolving Node Classification task as a downstream task to further validate the effectiveness of our learned temporal node representations. Specifically, we input the learned

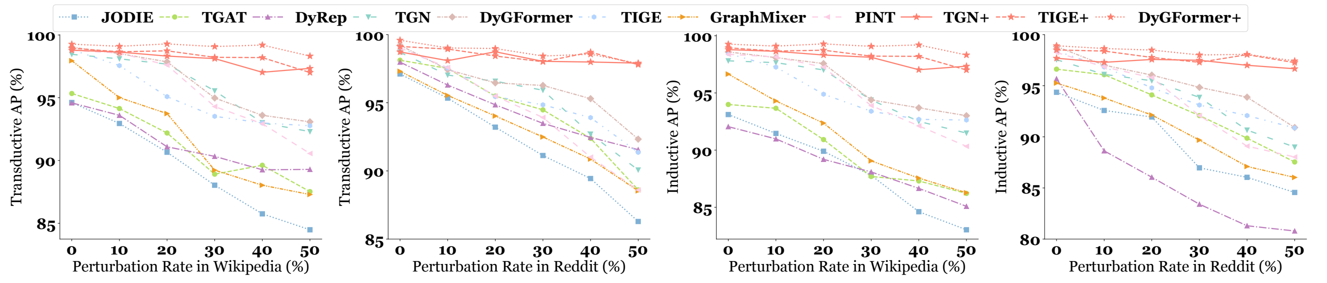


Figure 3: Robustness to noisy neighborhoods on Wikipedia and Reddit in different perturbation rates.

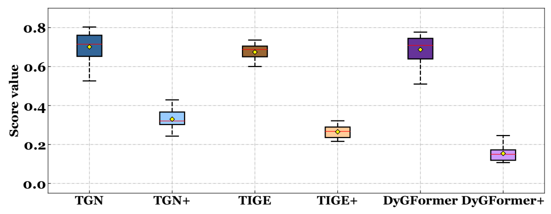
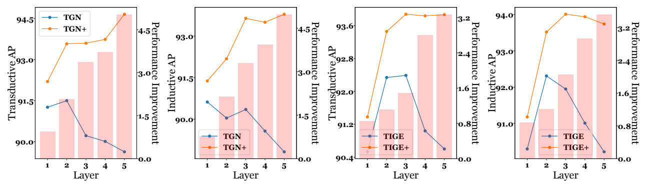


Figure 4: Attention scores assigned from randomly selected nodes to their perturbed neighbors on noisy Wikipedia.



(a) TGN model. **(b) TIGE model.**

Figure 5: Robustness to expanded neighborhoods on MOOC.

representation of node i at time t, $\mathbf{z}_i(t)$, into a two-layer MLP, which maps the representations to the dynamic labels. We carry out the experiments on Wikipedia, Reddit, MOOC, and TemFin. LastFM is excluded due to its lack of node labels. We employ AUROC as our evaluation metric due to the label imbalance issue in these datasets.

We report the results in Table 2. All three improved models demonstrate better performance than the backbones across all datasets, and at least one of them has achieved the SOTA results. This indicates that the learned representations from our model can be more effectively applied to downstream tasks, proving its superiority once again. Notably, the most significant improvement is observed on MOOC and TemFin. This could be due to their relatively less severe label imbalance issue compared to others, thereby facilitating the performance enhancement more readily.

5.4 Robustness to Noisy Neighborhoods

We have mentioned that the suitable neighborhoods for nodes should vary due to the different connectivity patterns and potential noise. To simulate this scenario, we complicate and pollute the nodes' neighborhoods by introducing random noise into TIGs [40]. We then evaluate the model performance (both w/o and w/ SEAN) under these noisy conditions. Specifically, we replace the original links with our randomly sampled noisy links at a certain perturbation rate $p \in \{10\%, 20\%, 30\%, 40\%, 50\%\}$. Meanwhile, at $p = 50\%$, we collect the attention scores assigned from a subset of randomly selected nodes to their perturbed neighbors, visualizing the model's ability to handle noise with box plots. We emphasize that these comparisons are controlled and fair because the noise issue dose exist in TIGs and the introduced noise is prevalent [26, 40].

The results are detailed in Figure 3 and 4. The improved models exhibit exceptional robustness, even when encountering significant noise. This demonstrates the superiority of SEAN in encoding suitable neighborhoods, thus weakening potential noise attacks. As for the perturbed neighbors' scores, we find an obvious reduction and greater focus in the scores when comparing improved models to their corresponding backbones. It suggests that SEAN successfully empowers these backbones to evaluate and mitigate these same-level random noises introduced in nodes' neighborhoods.

5.5 Robustness to Expanded Neighborhoods

We compare the performance across expanded neighborhoods between the backbones and improved models. Specifically, we vary the aggregation layer $K \in \{1, 2, 3, 4, 5\}$ in models and carry out the experiments using TGN [19] and TIGE [44] as the backbones. DyGFormer [37] is not included in this group of experiments due to its limitation to the first-hop neighborhoods in the original design. In Section 1, we keep the neighbor sampling size of TGN [19] at 10 to ensure a fair comparison, which is its default hyper-parameter. However, during these experiments, maintaining the sampling size at 10 will lead to GPU out-of-memory issues rapidly. Therefore, we reduce the size to 5 in practice.

The results are depicted in Figure 5. The improved models also exhibit significant robustness with expanded neighborhoods, and the performance improvement compared to their corresponding backbones progressively widens as the neighborhood expands. It highlights SEAN's capability to adaptively encode the suitable neighborhoods, thus avoiding the over-smoothing issue [12] caused by the expanded neighborhoods. Moreover, fine-tuning the accessible neighborhoods (*i.e.*, the layer number K) is essential for optimizing the backbone performance. In contrast, our improved models reach their peak performance by employing the largest possible accessible neighborhoods, as long as the computational resources permit.

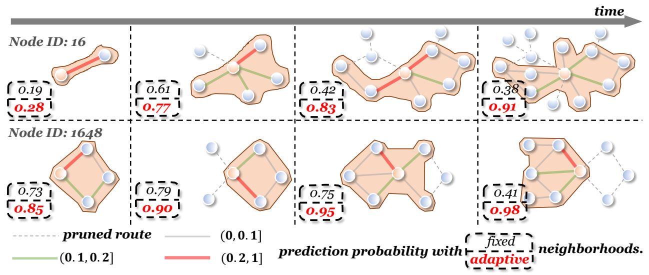


Figure 6: Case study on Wikipedia with TGN model. The introduction of SEAN can lead to both adaptive neighborhood consideration and improved prediction performance.

5.6 Case Study

We conduct the case study to interpret the choice-making process and model effectiveness when SEAN seeks the neighborhoods. As shown in Figure 6, each grid displays the encoded neighborhood for a certain node at a particular time, and dashed lines indicate the aggregation routes pruned by SEAN. The values within those dashed boxes are the Evolving Node Classification prediction probabilities for the backbone and the improved model, respectively. Here, higher values signify better performance. For clarity, we limit the depiction only to 2-hop accessible neighborhoods for each node.

In our observations from Node 16 in Figure 6, the backbone does not consistently show improved performance over time. Conversely, the improved model performance continues to increase, indicating SEAN's ability to manage the increasing neighborhood complexity and progressively stronger potential noise. Another notable aspect is that, even within the same node such as Node 1648 in Figure 6, the improved model shows significant neighborhood differences at distinct timestamps. It suggests that the improved model can effectively encode the most suitable neighborhoods across different timestamps to achieve optimal performance.

5.7 Ablation Study

We conduct the ablation study on the main components of our SEAN, including Representative Neighbor Selector (RNS) in Section 4.1, Temporal-aware Aggregator (TA) in Section 4.2, Neighbor Diversity Penalty (NDP) in Section 4.1.2, Adaptive Pruning Module (APM) in Section 4.2.1, and Outdated-decay Module (OM) in Section 4.2.2. To assess the impact of each component, we remove them individually, resulting in five variants: w/o RNS, w/o TA, w/o NDP, w/o APM, and w/o OM.

We report the performance of the Temporal Link Prediction task using both the improved TGN model and its variants as represented in Figure 7. Note that the green line represents the performance of the TGN backbone. The model performs best when utilizing all components, and the removal of any single component leads to a decrease in performance, highlighting the necessity of each design. Moreover, the performance of all variants surpasses the backbone, validating the effectiveness of our components. Additionally, our Temporal-aware Aggregator component has the most significant contribution to model performance, reaffirming the importance of the temporal-aware design for neighborhood encoding.

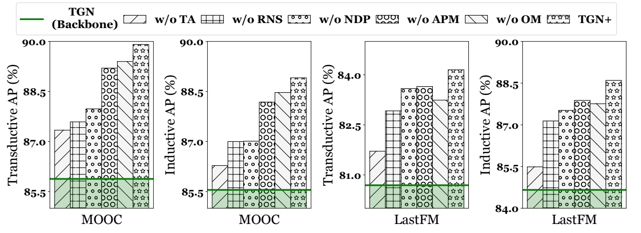


Figure 7: Ablation study on MOOC and LastFM with the improved TGN model. TGN backbone is shown for reference.

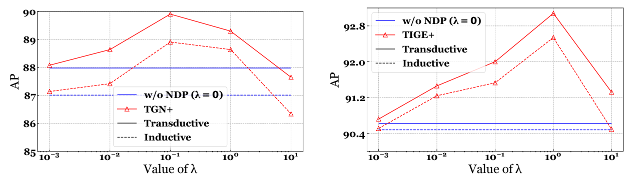


Figure 8: Parameter study on MOOC with improved TGN (left) and TIGE (right) models. w/o NDP is shown for reference.

5.8 Parameter Study

Now, we study how the controlling hyper-parameter of the Neighbor Diversity Penalty (λ in Equation 19) affects the performance. We plot the APs of the improved models with varying values of λ on MOOC in Figure 8. When the penalty method is too strong ($\lambda = 10$), the model may struggle to learn information from those important neighbors, thus leading to a decrease in model performance. Additionally, different models have distinct sensitivities to λ changes. For example, TIGE requires a larger λ for optimal results compared to TGN, suggesting a more severe over-concentration issue within the TIGE model.

6 Conclusion and future work

In this paper, we enhance temporal interaction graph modeling via adaptive neighborhood encoding. We propose SEAN, a plug-and-play model designed to boost existing TGNs. By introducing the Representative Neighbor Selector, we can select the personalized representatives for each target node. The Temporal-aware Aggregator then aggregates neighborhoods in a temporal-aware manner. As for future work, we will consider detecting the hidden routes that do not originally exist during the aggregation process. Additionally, we can also consider other implements for our Temporal-aware Aggregator, such as Transformer blocks.

Acknowledgments

This work is funded in part by the Shanghai Science and Technology Development Fund No.22dz1200704, the National Natural Science Foundation of China Projects No. U1936213, NSF through grants IIS-1763365 and IIS-2106972, and also supported by Ant Group. The first author, Dr. Zhang, also wants to thank Yifeng Wang for his efforts and support in this work.

References

[1] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432* (2013).

[2] Víctor Campos, Brendan Jou, Xavier Giró-i Nieto, Jordi Torres, and Shih-Fu Chang. 2017. Skip rnn: Learning to skip state updates in recurrent neural networks. *arXiv preprint arXiv:1708.06834* (2017).

[3] Xi Chen, Yongxiang Liao, Yun Xiong, Yao Zhang, Siwei Zhang, Jiawei Zhang, and Yiheng Sun. 2023. SPEED: Streaming Partition and Parallel Acceleration for Temporal Interaction Graph Embedding. *arXiv preprint arXiv:2308.14129* (2023).

[4] Xi Chen, Siwei Zhang, Yun Xiong, Xixi Wu, Jiawei Zhang, Xiangguo Sun, Yao Zhang, Yinglong Zhao, and Yulin Kang. 2024. Prompt Learning on Temporal Interaction Graphs. *arXiv preprint arXiv:2402.06326* (2024).

[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? *arXiv preprint arXiv:2302.11636* (2023).

[6] Kaituo Feng, Changsheng Li, Xiaolu Zhang, and Jun Zhou. 2023. Towards Open Temporal Graph Neural Networks. *arXiv preprint arXiv:2303.15015* (2023).

[7] Yang Gao, Hong Yang, Peng Zhang, Chuan Zhou, and Yue Hu. 2021. Graph neural architecture search. In *International joint conference on artificial intelligence*. International Joint Conference on Artificial Intelligence.

[8] Ilaria Gianstefani, Luigi Longo, and Massimo Riccaboni. 2022. The echo chamber effect resounds on financial markets: A social media alert system for meme stocks. *arXiv preprint arXiv:2203.13790* (2022).

[9] Yong Guo, David Stutz, and Bernt Schiele. 2023. Robustifying token attention for vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 17557–17568.

[10] ZHAO Huan, YAO Quanming, and TU Weiwei. 2021. Search to aggregate neighborhood for graph neural network. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 552–563.

[11] Zian Jia, Yun Xiong, Yuhong Nan, Yao Zhang, Jinjing Zhao, and Mi Wen. 2023. MAGIC: Detecting Advanced Persistent Threats via Masked Graph Representation Learning. *arXiv preprint arXiv:2310.09831* (2023).

[12] Nicolas Keriven. 2022. Not too little, not too much: a theoretical analysis of graph (over) smoothing. *Advances in Neural Information Processing Systems* 35 (2022), 2268–2281.

[13] 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*. 1269–1278.

[14] Kwei-Herng Lai, Daochen Zha, Kaixiong Zhou, and Xia Hu. 2020. Policy-gnn: Aggregation optimization for graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 461–471.

[15] Ziqi Liu, Chaochao Chen, Longfei Li, Jun Zhou, Xiaolong Li, Le Song, and Yuan Qi. 2019. Geniepath: Graph neural networks with adaptive receptive paths. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 4424–4431.

[16] Yang Luo, Zehao Gu, Shiyang Zhou, Yun Xiong, and Xiaofeng Gao. 2023. Meteorology-Assisted Spatio-Temporal Graph Network for Uncivilized Urban Event Prediction. In *2023 IEEE International Conference on Data Mining (ICDM)*. IEEE, 468–477.

[17] Dan McGinn, David Birch, David Akroyd, Miguel Molina-Solana, Yike Guo, and William J Knottenbelt. 2016. Visualizing dynamic bitcoin transaction patterns. *Big data* 4, 2 (2016), 109–119.

[18] 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.

[19] 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).

[20] Amauri Souza, Diego Mesquita, Samuel Kaski, and Vikas Garg. 2022. Provably expressive temporal graph networks. *Advances in Neural Information Processing Systems* 35 (2022), 32257–32269.

[21] Cass R Sunstein. 2006. *Infotopia: How many minds produce knowledge*. Oxford University Press.

[22] Qiaoyu Tan, Xin Zhang, Ninghao Liu, Daochen Zha, Li Li, Rui Chen, Soo-Hyun Choi, and Xia Hu. 2023. Bring your own view: Graph neural networks for link prediction with personalized subgraph selection. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*. 625–633.

[23] Xianfeng Tang, Huaxiu Yao, Yiwei Sun, Yiqi Wang, Jiliang Tang, Charu Aggarwal, Prasenjit Mitra, and Suhang Wang. 2020. Investigating and mitigating degree-related biases in graph convoltuional networks. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 1435–1444.

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

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[26] Yiwei Wang, Yujun Cai, Yuxuan Liang, Henghui Ding, Changhu Wang, Siddharth Bhatia, and Bryan Hooi. 2021. Adaptive data augmentation on temporal graphs. *Advances in Neural Information Processing Systems* 34 (2021), 1440–1452.

[27] Yanbang Wang, Yen-Yu Chang, Yunyu Liu, Jure Leskovec, and Pan Li. 2021. Inductive representation learning in temporal networks via causal anonymous walks. *arXiv preprint arXiv:2101.05974* (2021).

[28] Zhili Wang, Shimin Di, and Lei Chen. 2021. Autogel: An automated graph neural network with explicit link information. *Advances in Neural Information Processing Systems* 34 (2021), 24509–24522.

[29] Zhili Wang, Shimin Di, and Lei Chen. 2023. A Message Passing Neural Network Space for Better Capturing Data-dependent Receptive Fields. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2489–2501.

[30] Zhili Wang, Shimin Di, and Lei Chen. 2023. A Message Passing Neural Network Space for Better Capturing Data-dependent Receptive Fields. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2489–2501.

[31] Lanning Wei, Huan Zhao, Quanming Yao, and Zhiqiang He. 2021. Pooling architecture search for graph classification. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2091–2100.

[32] Sheng Xiang, Mingzhi Zhu, Dawei Cheng, Enxia Li, Ruihui Zhao, Yi Ouyang, Ling Chen, and Yefeng Zheng. 2023. Semi-supervised credit card fraud detection via attribute-driven graph representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 14557–14565.

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

[34] Huimin Xu, Zhicong Chen, Ruiqi Li, and Cheng-Jun Wang. 2020. The geometry of information cocoon: Analyzing the cultural space with word embedding models. *arXiv preprint arXiv:2007.10083* (2020).

[35] 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.

[36] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. 2021. Do transformers really perform badly for graph representation? *Advances in Neural Information Processing Systems* 34 (2021), 28877–28888.

[37] Le Yu, Leilei Sun, Bowen Du, and Weifeng Lv. 2023. Towards Better Dynamic Graph Learning: New Architecture and Unified Library. *arXiv preprint arXiv:2303.13047* (2023).

[38] Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. 2019. A review of recurrent neural networks: LSTM cells and network architectures. *Neural computation* 31, 7 (2019), 1235–1270.

[39] Chris Zhang, Mengye Ren, and Raquel Urtasun. 2018. Graph hypernetworks for neural architecture search. *arXiv preprint arXiv:1810.05749* (2018).

[40] Siwei Zhang, Yun Xiong, Yao Zhang, Yiheng Sun, Xi Chen, Yizhu Jiao, and Yangyong Zhu. 2023. RDGSL: Dynamic Graph Representation Learning with Structure Learning. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 3174–3183.

[41] Siwei Zhang, Yun Xiong, Yao Zhang, Xixi Wu, Yiheng Sun, and Jiawei Zhang. 2023. iLoRE: Dynamic Graph Representation with Instant Long-term Modeling and Re-occurrence Preservation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 3216–3225.

[42] Wei Zhang, Xiaogang Wang, Deli Zhao, and Xiaoou Tang. 2012. Graph degree linkage: Agglomerative clustering on a directed graph. In *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part I 12*. Springer, 428–441.

[43] Yao Zhang, Yun Xiong, Dongsheng Li, Caihua Shan, Kan Ren, and Yangyong Zhu. 2021. CoPE: Modeling Continuous Propagation and Evolution on Interaction Graph. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2627–2636.

[44] Yao Zhang, Yun Xiong, Yongxiang Liao, Yiheng Sun, Yucheng Jin, Xuehao Zheng, and Yangyong Zhu. 2023. TIGER: Temporal Interaction Graph Embedding with Restarts. In *ACM Web Conference*.

[45] Fujin Zhou, Thijs Endendijk, and WJ Wouter Botzen. 2023. A review of the financial sector impacts of risks associated with climate change. *Annual Review of Resource Economics* 15 (2023), 233–256.

[46] Kaixiong Zhou, Xiao Huang, Qingquan Song, Rui Chen, and Xia Hu. 2022. Auto-gnn: Neural architecture search of graph neural networks. *Frontiers in big Data* 5 (2022), 1029307.

[47] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. 2020. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in neural information processing systems* 33 (2020), 7793–7804.

A appendix

A.1 Notations

We summarize relevant notations and their descriptions in this paper as shown in Table 3.

Table 3: Important notations in this paper.

Notations	Description or definition
$\mathbf{f}_i(t)$	Occurrence frequency feature of node i's neighbors at t
$\mathbf{R}_i(t)$	Occurrence encoding of node i's neighbors at t
$\hat{a}_{ij}^{(k)}$	Enhanced attention from nodes i to j in the k-th layer
$u^{(k)}$	The state of the k-th layer
$\mathbf{c}_*^{(k)}$	Outdated-decay cell state in the k-th layer
$\mathcal{L}_{\text{NDP}}$	Loss function of neighbor diversity penalty
$\mathcal{L}_{\text{link}}$	Loss function of temporal link prediction

A.2 Datasets

We employ five datasets, including four public datasets and one dataset introduced in our paper.

The four public datasets[4] used in our paper are provided by JODIE [13]. (i) Wikipedia is a temporal interaction graph capturing edits made by users on Wikipedia pages. (ii) Reddit is a temporal interaction graph recording user posts in different subreddits. In both Wikipedia and Reddit, the edge feature dimension is 172, and user nodes are dynamically labeled to indicate whether they have been banned. (iii) MOOC is a temporal interaction graph that tracks interactions between students and online courses, with dynamic labels on nodes indicating whether students drop out of a course. (iv) LastFM is a temporal interaction graph that logs events between users and songs, but it does not include dynamic labels.

Most existing temporal interaction graph datasets are primarily focused on social or event networks. However, interaction data in the financial domain differs significantly. It typically involves a considerably larger number of participants, resulting in sparser networks. Consequently, learning tasks on such networks tend to be more challenging compared to those on existing available networks. This increased difficulty provides a more rigorous testbed for validating the capabilities of various TGNs, making a more effective benchmark to assess their performance and robustness.

In this paper, we release **TemFin**, a new temporal interaction graph benchmark dataset that records the financial transfer transactions between bank accounts. Within TemFin, bank accounts are represented as nodes, and the financial transactions with timestamp information occurring between two accounts are modeled as timestamped interactions. Additionally, the information about the transactions, *e.g.*, amount and fund channel, is converted into a 154-dimensional vector through one-hot encoding, which then serves as the edge feature for each transaction. Furthermore, the source account nodes are assigned dynamic labels, indicating whether the corresponding account is implicated in money laundering activity. In the TemFin dataset, the Temporal Link Prediction task is designed to forecast whether one account will transfer funds to another at a given future time. Meanwhile, the Evolving Node Classification task concentrates on determining whether a source account in a specific transaction is implicated in money laundering activity. The detailed statistics of all datasets are summarized in Table 5.

[4]https://snap.stanford.edu/jodie/

A.3 Implementation Details

For baselines, all baselines employ the same experimental settings as TIGE [44], thus ensuring comparability in our evaluation process. The detailed default hyper-parameters can be found in its publication.

For our SEAN, we integrate it into three representative TGNs, *i.e.*, TGN [19], TIGE [44], and DyGFormer [37], chosen for their superior performance and widespread recognition. To guarantee fairness in our evaluation, all hyper-parameters in these three backbones remain consistent with their original implementations. We implement SEAN in PyTorch based on their corresponding official implementations. Unless otherwise stated, we use the default hyper-parameters of SEAN summarized in Table 4.

Experiments on all datasets are conducted on a single server with 72 cores, 32GB memory, and one Nvidia Tesla V100 GPU.

Table 4: Default hyper-parameters of SEAN.

Hyper-parameter	Value
Batch size	200
Learning rate	0.0001
Optimizer	Adam
Number of layers	1
Number of attention heads	2
Threshold τ	0.1

A.4 Limitations

We consider two main limitations for our SEAN:

- While SEAN enhances model performance, it inevitably introduces some complexity, which may be unavoidable.
- Our proposed adaptive pruning module serves as a residual link in the aggregation process and improves the overall model performance. However, when the neighborhood is extremely sparse, the pruning mechanism might lose some useful information.

We will address these limitations in our future work by exploring more efficient adaptive neighborhood encoding methods in temporal interaction graph modeling.

A.5 Supplementary Results

Due to space constraints in the main body of our paper, we present the following supplementary results: the robustness study to noisy neighborhoods is depicted in Figure 9 and 10, the robustness study to expanded neighborhoods is shown in Figure 11, additional case study with more other selected nodes are illustrated in Figure 12, and the ablation study is detailed in Figure 13.

Table 5: Detailed statistics of datasets.

Datasets	Domain	# Nodes	# Edges	# Edge Feature	Duration	Label	% Positive Labels
Wikipedia	social	9,227	157,474	172	1 month	editing ban	0.14%
Reddit	social	10,984	672,447	172	1 month	posting ban	0.05%
MOOC	event	7,144	411,749	0	17 month	course dropout	0.98%
LastFM	event	1,980	1,293,103	0	1 month	—	—
TemFin (*new*)	finance	33,245	709,774	154	half a month	money laundering	0.71%

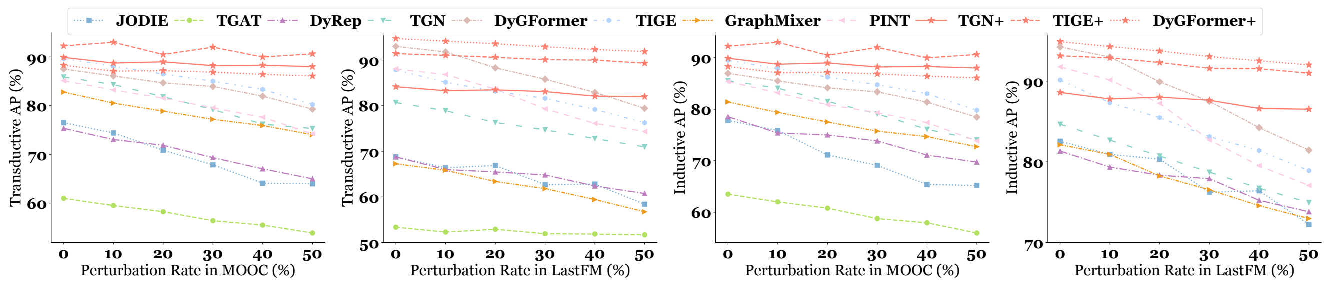


Figure 9: Robustness to noisy neighborhoods on MOOC and LastFM in different perturbation rates. (Supplementary results for Section 5.4)

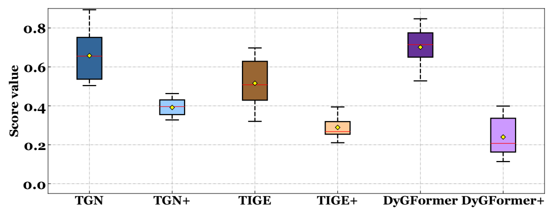


Figure 10: Attention scores assigned from randomly selected nodes to their perturbed neighbors on noisy Reddit. (Supplementary results for Section 5.4)

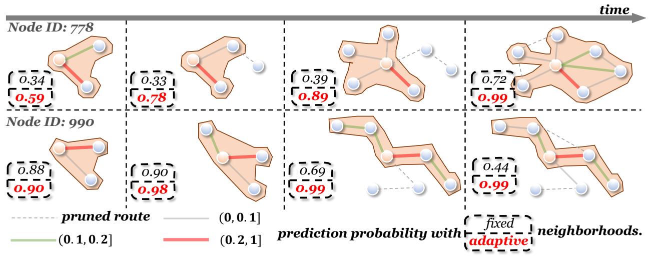
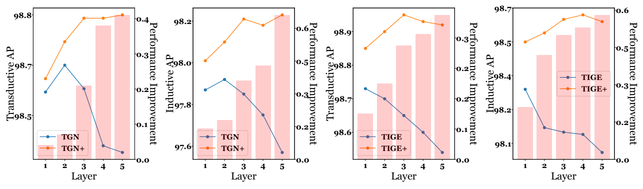


Figure 12: Case study on Wikipedia with TGN model. (Supplementary results for Section 5.6)



(a) TGN model. (b) TIGE model.

Figure 11: Robustness to expanded neighborhoods on Wikipedia with TGN and TIGE models. (Supplementary results for Section 5.5)

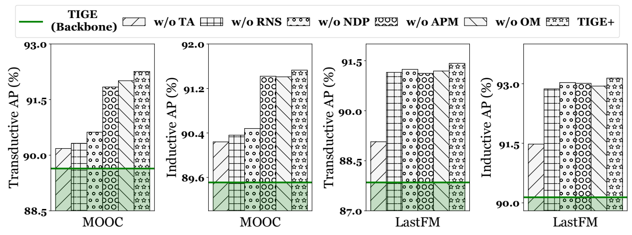


Figure 13: Ablation study on MOOC and LastFM with the improved TIGE model. TIGE backbone is shown for reference. (Supplementary results for Section 5.7)

(i) Robustness Study to Noisy Neighborhoods. Supplementary results for noisy MOOC and LastFM are provided. Additionally, we report the box plot of the attention scores for perturbed neighbors on noisy Reddit. (ii) Robustness Study to Expanded Neighborhoods. Supplementary results for Wikipedia are presented. (iii) Case Study. We present the additional case study derived from other sampled nodes. (iv) Ablation Study. Supplementary ablation results for the improved TIGE model and its variants.