Neighbor-Aware Snapshot-Based Temporal Graph Learning

Jinxi Yang[*] Wuhan University
He Li[*] Wuhan University
Wei Yu Wuhan University
Xingyu Gao Chinese Academy of Sciences

Mang Ye[†] Wuhan University

Abstract

Dynamic graph learning models the temporal evolution of structural interactions. Existing methods present a stark trade-off between performance and efficiency. Event-based models achieve strong accuracy because processing graphs edge-by-edge allows them to continuously update and leverage fine-grained joint neighborhood information. Conversely, snapshot-based methods offer high computational efficiency by processing entire snapshots simultaneously. However, maintaining and updating dynamic neighborhood states for all nodes simultaneously is computationally prohibitive, limiting their ability to exploit rich neighborhood information. Consequently, current snapshot methods fail to utilize joint neighborhood information, leading to significantly weaker performance. To bridge this gap, we propose the Neighbor-Aware Graph Neural Network (NAGNN), a snapshot-based architecture that represents each node's evolving neighborhood as a feature distribution and compresses it into a fixed-dimensional representation. Specifically, NAGNN combines a Locality-Sensitive Hashing (LSH)-based reservoir with degree-scaled statistical moments to capture complementary fine-grained and global properties of the neighborhood distribution, enabling efficient tensorized computation while preserving informative structural signals. A Gated Recurrent Unit (GRU) further models the temporal evolution of these compressed neighborhood representations. Experiments on six temporal graph benchmarks show that NAGNN consistently improves over strong snapshot-based baselines and substantially narrows the performance gap to event-based models while retaining the computational advantages of snapshot processing. Code is available at `https://github.com/Jinxi-Yang-WHU/NAGNN.git`.

1 Introduction

Dynamic graphs naturally model evolving interactions in domains like social networks, financial services, traffic management, and recommender systems [1–4]. Learning from these graphs requires capturing the temporal evolution of node semantics, edge dynamics, and local structural patterns. A primary downstream task is dynamic link prediction, which forecasts the formation of future edges by leveraging these learned temporal patterns.

The landscape of temporal graph learning is broadly divided into snapshot-based [5–8] and event-based [9–13] paradigms. Recent benchmarking under the Unified Temporal Graph (UTG) [14] has formalized a trade-off between the two: event-based methods achieve superior predictive performance,

[*]Equal contribution.
[†]Corresponding author: `yemang@whu.edu.cn`.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

whereas snapshot-based methods are at least an order of magnitude faster. Event-based models process interactions sequentially as discrete, timestamped events. This sequential nature allows recent state-of-the-art architectures, such as DyGFormer [10] and NAT [9], to maintain and efficiently update memory banks for the two specific endpoints of an edge at each timestamp. By doing so, they can extract and leverage fine-grained *joint neighborhood information*, which UTG [14] identifies as the primary driver of their strong performance. However, this event-by-event processing creates a severe computational bottleneck, restricting their scalability in large-scale, real-world networks.

Conversely, snapshot-based networks process entire graph snapshots simultaneously, enabling highly efficient and scalable matrix operations. However, this simultaneous processing introduces a critical architectural flaw: it is computationally prohibitive to maintain and update a dynamic neighborhood memory bank for all nodes concurrently at each timestamp. As a result, current snapshot-based methods fail to leverage joint neighborhood information, leading to significantly weaker predictive performance compared to their event-based counterparts. Crucially, UTG [14] explicitly highlights this gap, noting that a snapshot-based method capable of capturing joint neighborhood dynamics does not yet exist in the literature. Motivated by this gap, our primary contribution is to overcome this fundamental limitation by seamlessly integrating the computational efficiency of snapshots with the structural fidelity of joint neighborhood modeling.

To overcome this fundamental limitation, we propose the Neighbor-Aware Graph Neural Network (NAGNN), a novel, scalable snapshot-based paradigm that seamlessly integrates joint neighborhood information without sacrificing computational efficiency. To enable simultaneous processing of neighborhood states across all nodes, we view the variable-sized multiset of neighbor features as an empirical feature distribution and encode it into a fixed-dimensional representation. Specifically, we introduce a Locality-Sensitive Hashing [15] (LSH)-based reservoir that partitions neighborhood features into a fixed number of similarity-aware slots, preserving fine-grained structure within the distribution. In parallel, we compute degree-scaled first- and second-order statistics to summarize its global characteristics while retaining neighborhood-size information. Together, these complementary encodings transform irregular neighborhood distributions into compact, fixed-dimensional features amenable to efficient tensorized processing. Crucially, we theoretically prove that this LSH-based encoding scheme captures local structural information with a rigorously bounded approximation error, providing strong mathematical guarantees for its representational fidelity. Finally, a Gated Recurrent Unit (GRU) [16] tracks these enriched structural embeddings across successive snapshots to model their temporal evolution. By synthesizing these components, NAGNN achieves a deep, contextualized understanding of graph dynamics while retaining the computational tractability of snapshot-based processing.

In summary, our main contributions are threefold:

- **Novel Framework:** We introduce NAGNN, the first snapshot-based architecture capable of leveraging joint neighborhood information as effectively as advanced event-based methods. This fundamentally bridges the longstanding gap between snapshot efficiency and event-level predictive performance.
- **Theoretical Guarantees:** We provide a formal theoretical analysis of our LSH-based neighborhood encoding scheme. We mathematically prove that our approach approximates the true joint neighborhood distribution with a strictly bounded error, ensuring robust and reliable structural representations.
- **Improved Performance–Efficiency Trade-off:** Through extensive experiments, we demonstrate that NAGNN outperforms existing state-of-the-art snapshot-based methods. Furthermore, it achieves accuracy competitive with strong event-based approaches while operating at nearly an order of magnitude lower computational cost.

2 Related Work

2.1 Snapshot-based Graph Learning

Building on the success of static Graph Neural Networks (GNNs) [17–21], snapshot-based models operate on Discrete-Time Dynamic Graphs (DTDGs) [22–24], which represent graph evolution as a sequence of static snapshots [25]. The canonical architecture couples a static GNN to encode spatial topology with a recurrent module, such as a GRU, to model temporal dynamics [5]. This modularity

seamlessly integrates rapid advancements in static GNNs [26, 27]. Crucially, processing entire graph snapshots in a single batch affords massive parallelization and high computational efficiency.

2.2 Event-based Graph Learning

Event-based methods process Continuous-Time Dynamic Graphs (CTDGs), treating graph evolution as a continuous stream of timestamped interactions [25]. This incremental, event-by-event processing captures highly fine-grained temporal and causal dependencies [12]. Recent state-of-the-art models, such as DyGFormer [10] and NAT [9], excel at dynamic link prediction [9, 10, 14] precisely because this sequential formulation allows them to continuously maintain and update precise representations of joint neighborhood information. These methods maintain explicit memories of individual historical neighbors and retrieve these neighbor-level states for the endpoints involved in each event. In contrast, our snapshot-based framework summarizes neighborhood information into compact fixed-size representations that can be updated for all nodes in parallel.

2.3 Unified Temporal Graph Learning

To rigorously benchmark these divergent paradigms, the Unified Temporal Graph (UTG) framework [14] was recently introduced. UTG provides a principled methodology for bidirectional format conversion. Specifically, it aggregates continuous CTDG events within a fixed interval into DTDG snapshots, or conversely, disaggregates DTDG snapshots into CTDG streams by treating every edge as an individual event sharing the timestamp of its parent snapshot. This standardized conversion enables fair and direct performance comparisons.

3 Preliminary

3.1 Snapshot-based Methods

A DTDG can be represented by:

$$\mathcal{G} = \{\mathcal{G}_0, \mathcal{G}_1, \ldots, \mathcal{G}_T\}$$

At each snapshot t, for a graph $\mathcal{G}^{(t)} = (\mathcal{V}, \mathcal{E}^{(t)})$ and node features $\mathbf{X}^{(t)}$, a snapshot-based model generates a comprehensive node representation $\mathbf{z}_v^{(t)}$ for each node $v \in \mathcal{V}$. For the link prediction task, the likelihood of an edge forming between node u and node v at the subsequent snapshot $t+1$ is determined by a scoring function applied to their current embeddings $\mathbf{z}_u^{(t)}$ and $\mathbf{z}_v^{(t)}$. Specifically, the probability is computed as

$$p_{u,v}^{(t+1)} = \sigma(\text{MLP}(\mathbf{z}_u^{(t)} \odot \mathbf{z}_v^{(t)})) \tag{1}$$

where $\odot$ and σ denote the Hadamard product and sigmoid activation function, respectively.

3.2 Joint Neighborhood Information.

For a node v, let $\mathcal{N}_k^{(\tau)}(v)$ denote its k-hop neighborhood at snapshot τ. We define the historical k-hop neighborhood of v up to time t as

$$\mathcal{H}_{v,k}^{(\leq t)} = \left\{ (\tau, w, \mathbf{h}_w^{(\tau)}) \;\middle|\; 1 \leq \tau \leq t,\ w \in \mathcal{N}_k^{(\tau)}(v) \right\}, \qquad k \in \{1, 2\}.$$

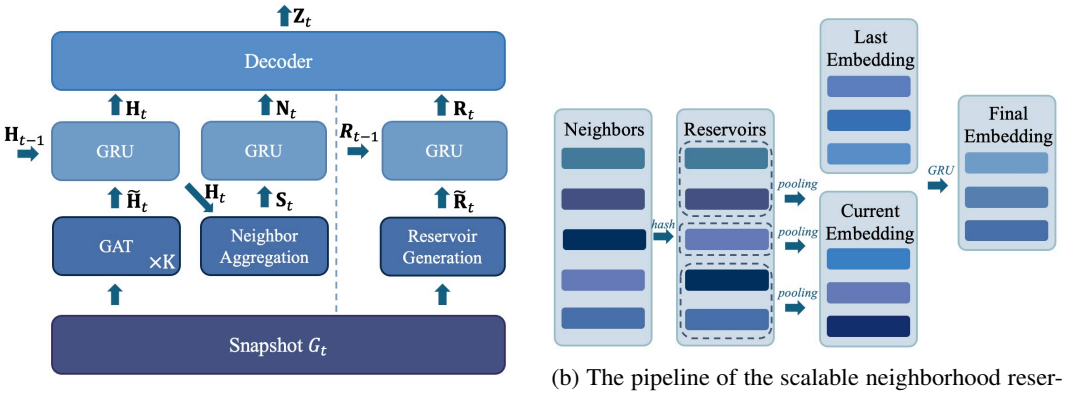
For a candidate interaction (u, v) at time $t+1$, we define its *joint neighborhood information* as

$$\mathcal{J}_{u,v}^{(\leq t)} = \left(\{\mathcal{H}_{u,k}^{(\leq t)}\}_{k=1}^{2}, \{\mathcal{H}_{v,k}^{(\leq t)}\}_{k=1}^{2} \right),$$

which contains the historical neighborhood states of both endpoints over all preceding timestamps.

Event-based methods such as NAT model this information explicitly through neighborhood memories: NAT maintains node-wise caches containing representations of individual historical neighbors and retrieves these neighbor-level memories when processing an interaction. In contrast, NAGNN does not explicitly retain every historical neighbor. At each snapshot τ, it views the neighborhood feature multiset $\{\mathbf{h}_w^{(\tau)} : w \in \mathcal{N}_k^{(\tau)}(v)\}$ as an empirical feature distribution

$$\mu_{v,k}^{(\tau)} = \frac{1}{|\mathcal{N}_k^{(\tau)}(v)|} \sum_{w \in \mathcal{N}_k^{(\tau)}(v)} \delta_{\mathbf{h}_w^{(\tau)}}.$$



(a) The pipeline of the NAGNN model.

(b) The pipeline of the scalable neighborhood reservoir.

Figure 1: Illustration of model structure. **(a)** NAGNN. **(b)** LSH-based reservoir.

NAGNN then encodes each $\mu_{v,k}^{(\tau)}$ using complementary degree-scaled statistical moments and LSH-based reservoir summaries, and recurrently propagates these distributional representations across snapshots using GRUs. Consequently, the endpoint representations $\mathbf{z}_u^{(t)}$ and $\mathbf{z}_v^{(t)}$ provide compact temporal encodings of $\mathcal{J}_{u,v}^{(\leq t)}$, which are combined by the link predictor in Equation (1).

4 Method

We propose a snapshot-based temporal graph learning framework called Neighbor-Aware Graph Neural Network (NAGNN), which can capture joint neighborhood information efficiently. Our model enriches ROLAND [5] architecture by integrating two specialized modules: a **Temporally-Smoothed Neighborhood Aggregator** that tracks the evolving statistical moments of local neighborhoods, and a **Reservoir-based Structural Hasher** that maintains a compact, temporally-aware summary of the feature distribution within multi-hop neighborhoods. The notations in this section are shown in the Table 2. The pipeline is illustrated by Figure 1a.

At each snapshot t, for a graph $\mathcal{G}^{(t)} = (\mathcal{V}, \mathcal{E}^{(t)})$ and node features $\mathbf{X}^{(t)}$, our model generates a comprehensive node representation $\mathbf{z}_v^{(t)}$ for each node $v \in \mathcal{V}$. This representation is a concatenation of five distinct components:

$$\mathbf{z}_v^{(t)} = \left[\mathbf{h}_v^{(t)} \,\|\, \mathbf{n}_{v,1}^{(t)} \,\|\, \mathbf{n}_{v,2}^{(t)} \,\|\, \mathbf{r}_{v,1}^{(t)} \,\|\, \mathbf{r}_{v,2}^{(t)}\right], \tag{2}$$

where $\mathbf{h}_v^{(t)}$ is the core temporal node embedding, $\mathbf{n}_{v,k}^{(t)}$ represents the k-hop temporally-smoothed neighborhood aggregate, and $\mathbf{r}_{v,k}^{(t)}$ is the k-hop reservoir-based structural summary. We detail each component below.

4.1 Temporal Node Feature Encoding

This component operates as a spatial-temporal encoder block. It first employs stacked GAT [19] layers for spatial feature aggregation. The resulting sequence of node embeddings is then processed by a Gated Recurrent Unit (GRU)[16] to model their temporal evolution.

$$\widetilde{\mathbf{H}}_{(t)} = \text{ReLU}\left(\text{GAT}(\mathbf{X}^{(t)}, \mathcal{E}^{(t)})\right), \tag{3}$$

where $\widetilde{\mathbf{H}}(t)$ is the matrix of all intermediate node features.

To capture temporal dependencies, these spatially aggregated features are used to update the hidden states of the nodes using GRU cells. This process maintains a memory of each node's past states, allowing the model to learn complex evolutionary patterns.

$$\mathbf{h}_v^{(t)} = \text{GRU}\left(\widetilde{\mathbf{h}}_v^{(t)}, \mathbf{h}_v^{(t-1)}\right), \tag{4}$$

The final output, $\mathbf{h}_v^{(t)} \in \mathbb{R}^{d_h}$, serves as the core node embedding, capturing the primary features of a node in its spatial and temporal context.

4.2 Temporally-Smoothed Neighborhood Aggregation

To effectively capture the dynamic characteristics of a node's local neighborhood, our aggregation module is designed to effectively integrate and summarize information from adjacent nodes. Rather than explicitly storing the historical neighborhood sets defined in Section 3.2, NAGNN conceptualizes the multiset of feature vectors from a node's neighbors, $\{\mathbf{h}_u^{(t)} \mid u \in \mathcal{N}(v)\}$, as an empirical probability distribution and encodes it into fixed-dimensional vectors.

This distribution can be powerfully characterized by its statistical moments. The first moment (mean) describes its central tendency, while the second moment is essential for calculating its variance, which measures feature dispersion. Together, these two moments can uniquely determine a Gaussian distribution, making them highly descriptive statistics. However, a direct computation of these moments via mean aggregation would obscure vital structural information by normalizing by the degree $|\mathcal{N}(v)|$.

To retain this crucial degree information while still capturing the shape of the feature distribution, our module instead computes unnormalized sums. This approach is equivalent to scaling the moments by the degree. It can be intuitively understood as characterizing a sample of size $|\mathcal{N}(v)|$ drawn from the neighborhood's feature distribution. Consequently, the resulting aggregates encode both the shape of the distribution (via the relative values of the sums) and its scale (via their overall magnitude, which is proportional to the degree).

Specifically, for the 1-hop neighborhood $\mathcal{N}_1(v)$, we compute two statistics:

$$\mathbf{a}_{v,1}^{(t)} = \sum_{u \in \mathcal{N}_1(v)} \mathbf{h}_u^{(t)}, \quad \mathbf{a}_{v,1}^{\text{sq}(t)} = \sum_{u \in \mathcal{N}_1(v)} \left(\mathbf{h}_u^{(t)} \odot \mathbf{h}_u^{(t)} \right) \tag{5}$$

where $\odot$ denotes the element-wise product. Here, $\mathbf{a}_{v,1}^{(t)}$ corresponds to the degree-scaled first moment (the sum of features), while $\mathbf{a}_{v,1}^{\text{sq}(t)}$ is the degree-scaled second raw moment (the sum of squared features). These vectors are concatenated, $\mathbf{s}_{v,1}^{(t)} = [\mathbf{a}_{v,1}^{(t)} \parallel \mathbf{a}_{v,1}^{\text{sq}(t)}]$, and the process is repeated to obtain the 2-hop aggregate, $\mathbf{s}_{v,2}^{(t)}$.

Finally, to ensure temporal smoothness and mitigate noise from transient structural changes, these raw statistics are updated via a shared GRU cell:

$$\mathbf{n}_{v,k}^{(t)} = \text{GRU}_{\text{neigh}} \left(\mathbf{s}_{v,k}^{(t)}, \mathbf{n}_{v,k}^{(t-1)} \right) \quad \text{for } k \in \{1, 2\}. \tag{6}$$

The resulting vectors, $\mathbf{n}_{v,1}^{(t)}$ and $\mathbf{n}_{v,2}^{(t)}$, represent a stable, evolving signature of the node's local neighborhood that is rich in both feature and structural information.

4.3 Reservoir of Neighborhood Information

The most significant innovation of NAGNN is the use of a reservoir-based mechanism to efficiently summarize the feature distributions of multi-hop neighborhoods. Our approach is designed for the snapshot-based setting, where neighborhood information and node representations are computed for the entire graph simultaneously at each snapshot, rather than being updated incrementally for specific nodes as individual events occur. The core idea is to hash neighbors into a fixed number of slots based on their features and maintain a temporally evolving representation for each slot. The overview of this module is shown as Figure 1b.

Structural Hasher. We first define a structural hasher, which maps a d_h-dimensional feature vector to a discrete slot index in $[0, R-1]$, where R is the reservoir capacity. This is achieved via signed random projections:

$$\text{slot}(\mathbf{h}) = \left(\sum_{i=0}^{k-1} 2^i \cdot \mathbb{I}(\mathbf{h}^T \mathbf{p}_i > 0) \right) \pmod R, \tag{7}$$

where $\mathbf{P} = [\mathbf{p}_0, \ldots, \mathbf{p}_{k-1}] \in \mathbb{R}^{d_h \times k}$ is a fixed random Gaussian matrix and $\mathbb{I}(\cdot)$ is the indicator function. This hash function groups nodes with similar feature vectors into the same slot.

Reservoir Update. For each node v, we maintain two reservoirs, $\mathbf{R}_{v,1}, \mathbf{R}_{v,2} \in \mathbb{R}^{R \times d_h}$, for its 1-hop and 2-hop neighborhoods, respectively. At each snapshot t, we compute a temporary pooling for each slot by averaging the features of all neighbors that hash to that slot. The 1-hop reservoir can be generated by:

$$\tilde{\mathbf{R}}_{v,1,s}^{(t)} = \text{Mean}_{\{u \in \mathcal{N}_1(v) | \text{slot}(\mathbf{h}_u^{(t)}) = s\}} \left(\mathbf{h}_u^{(t)} \right). \tag{8}$$

A similar aggregation approach is performed for the 2-hop neighbors to obtain $\tilde{\mathbf{R}}_{v,2,s}^{(t)}$.

These temporary poolings are then used to update the persistent reservoir states via GRU [16] cells, allowing each slot to learn the temporal evolution of the specific feature-group it represents:

$$\mathbf{R}_{v,k,s}^{(t)} = \text{GRU}_{\text{res}} \left(\tilde{\mathbf{R}}_{v,k,s}^{(t)}, \mathbf{R}_{v,k,s}^{(t-1)} \right). \tag{9}$$

Reservoir Pooling. Finally, to obtain a fixed-size representation from the R slots, we employ an attention-based pooling mechanism. The features in each slot are first transformed by a small MLP, and attention weights are computed to aggregate them into a single vector:

$$\mathbf{e}_{v,k,s}^{(t)} = \text{MLP}_{\text{pool}}(\mathbf{R}_{v,k,s}^{(t)}), \quad \gamma_{v,k,s}^{(t)} = \text{softmax}_s \left(\mathbf{w}_a^T \mathbf{e}_{v,k,s}^{(t)} \right), \quad \mathbf{r}_{v,k}^{(t)} = \sum_{s=0}^{R-1} \gamma_{v,k,s}^{(t)} \mathbf{e}_{v,k,s}^{(t)} \tag{10}$$

The vectors, $\mathbf{r}_{v,1}^{(t)}$ and $\mathbf{r}_{v,2}^{(t)}$, provide a powerful summary of the neighborhood information.

4.4 Link Prediction

The final node representation $\mathbf{z}_v^{(t)}$ is a concatenation of five distinct components:

$$\mathbf{z}_v^{(t)} = \left[\mathbf{h}_v^{(t)} \parallel \mathbf{n}_{v,1}^{(t)} \parallel \mathbf{n}_{v,2}^{(t)} \parallel \mathbf{r}_{v,1}^{(t)} \parallel \mathbf{r}_{v,2}^{(t)} \right], \tag{11}$$

The probability of an edge existing between two nodes (u, v) at snapshot t is predicted by Equation (1). In essence, our method makes the prediction not only based on the node's own features and a rich, but also multi-faceted summary of its neighborhood, encoding local structure explicitly, and achieving strong performance.

5 Theoretical Analysis of the Reservoir

5.1 Setup and Notation

Let $\mathcal{N} = \{\mathbf{h}_1, \ldots, \mathbf{h}_m\} \subset \mathbb{R}^{d_h}$ denote the multiset of neighbor feature vectors for a given target node, bounded by $\|\mathbf{h}_i\| \leq B$. For SimHash, we draw k independent Gaussian projections $\mathbf{p}_\ell \sim \mathcal{N}(0, I)$ to form k-bit signatures $\mathbf{y}_i \in \{\pm 1\}^k$, where the ℓ-th bit ($\ell = 0, \ldots, k-1$) is $y_i^{(\ell)} = \text{sign}(\langle \mathbf{h}_i, \mathbf{p}_\ell \rangle)$. For analytical tractability, we analyze collisions induced by the underlying k-bit SimHash signatures, while the finite-capacity reservoir used in practice may introduce additional collisions when these signatures are mapped to R slots. This induces a collision indicator $X_{ij} = \mathbf{1}\{\mathbf{y}_i = \mathbf{y}_j\}$ for $1 \leq i < j \leq m$, and a total collision count $S = \sum_{i<j} X_{ij}$. For each slot s containing n_s items, the reservoir stores the local feature mean $\mathbf{r}_s = \frac{1}{n_s} \sum_{i:\, s(i)=s} \mathbf{h}_i$, resulting in a Sum-of-Squared-Errors $\text{SSE} = \sum_s \sum_{i:\, s(i)=s} \|\mathbf{h}_i - \mathbf{r}_s\|^2$. Finally, defining the global means $\bar{\mathbf{h}} = \frac{1}{m} \sum_i \mathbf{h}_i$ and $\bar{\mathbf{r}} = \sum_s \frac{n_s}{m} \mathbf{r}_s$, the empirical neighbor covariance $\mathbf{\Sigma}_\mathcal{N}$ and reservoir covariance $\mathbf{\Sigma}_\mathcal{R}$ are formulated as:

$$\mathbf{\Sigma}_\mathcal{N} = \frac{1}{m} \sum_{i=1}^{m} (\mathbf{h}_i - \bar{\mathbf{h}})(\mathbf{h}_i - \bar{\mathbf{h}})^\top, \qquad \mathbf{\Sigma}_\mathcal{R} = \sum_s \frac{n_s}{m} (\mathbf{r}_s - \bar{\mathbf{r}})(\mathbf{r}_s - \bar{\mathbf{r}})^\top.$$

A summary of the notation used in the theoretical analysis is provided in Table 3.

5.2 Theoretical Superiority of Reservoir

Theorem 5.1 (Exact mean preservation)**.** *For any hashing mechanism (deterministic or random),*

$$\bar{\mathbf{r}} = \bar{\mathbf{h}}.$$

This theorem indicates that our method preserves the empirical mean exactly and provides a faithful summary, localizing all approximation error to higher-order statistics.

Theorem 5.2 (High-probability SSE / covariance-gap (SimHash))**.** *Under Gaussian sign projections and $\|\mathbf{h}_i\| \le B$, let $\bar{S} = \mathbb{E}[S]$ be the mean collision count. For any $\delta \in (0,1)$, with probability at least $1-\delta$,*

$$\mathrm{SSE} \le 4B^2\Big(\bar{S} + m^2\sqrt{\frac{k\ln(1/\delta)}{32}}\Big), \tag{12}$$

$$\big\|\mathbf{\Sigma}_{\mathcal{N}} - \mathbf{\Sigma}_{\mathcal{R}}\big\|_{\mathrm{op}} \le \frac{4B^2}{m}\Big(\bar{S} + m^2\sqrt{\frac{k\ln(1/\delta)}{32}}\Big). \tag{13}$$

If pairwise angles are (empirically) "uniform" so that $\bar{S} = \dfrac{m(m-1)}{2(k+1)}$.

This theorem demonstrates that that the fidelity of the reservoir approximation is governed by the collision behavior of the hash function. A higher hash resolution can better separate dissimilar neighborhood features and reduce collision-induced distortion, thereby yielding a more faithful approximation of the neighborhood structure.

Theorem 5.3 (Distributional Approximation Bound – Informal)**.** *Let $\mu_{\mathcal{N}}$ and $\mu_{\mathcal{R}}$ denote the empirical feature distributions of a node's true joint neighborhood and its LSH-compressed representation, respectively. Let K be a positive-definite kernel with RKHS feature map ϕ, and assume that ϕ is L_K-Lipschitz, i.e.,*

$$\|\phi(\mathbf{x}) - \phi(\mathbf{y})\|_{\mathcal{H}} \le L_K\|\mathbf{x}-\mathbf{y}\|_2.$$

Then, in expectation over the random LSH projections,

$$\mathbb{E}\big[\mathrm{MMD}_K^2(\mu_{\mathcal{N}}, \mu_{\mathcal{R}})\big] \le \frac{L_K^2}{m}\sum_{i<j} p_{ij}\|\mathbf{h}_i - \mathbf{h}_j\|_2^2 \le \frac{4L_K^2B^2}{m}\sum_{i<j} p_{ij},$$

where m is the neighborhood size and $p_{ij} = \Pr(s(i) = s(j))$ is the probability that neighbor features i and j collide in the same hash bucket. The formal theorem is provided in Section B.

Remark: The bound characterizes how the approximation error of the compressed neighborhood representation depends on pairwise feature dissimilarity and hash collisions. In particular, collisions between dissimilar neighborhood features contribute more strongly to the approximation error, motivating the use of locality-sensitive hashing to preferentially group similar features.

Proposition 5.4 (Comparison vs uniform sampling)**.** *Let $\widehat{\mu}_s$ be the empirical measure formed by sampling s neighbors uniformly without replacement. Then for any kernel K with $K(\mathbf{x},\mathbf{x}) \le K_0$,*

$$\mathbb{E}\big[\mathrm{MMD}_K^2(\mu_{\mathcal{N}}, \widehat{\mu}_s)\big] \le \frac{4K_0}{s}.$$

Together with Theorem 5.3, this provides a direct comparison between the derived approximation bounds of LSH-based compression and uniform sampling. In particular, the reservoir admits a tighter derived bound whenever

$$\frac{L_K^2}{m}\sum_{i<j} p_{ij}\|\mathbf{h}_i - \mathbf{h}_j\|_2^2 \le \frac{4K_0}{s}.$$

This proposition shows that the advantage of the LSH-based reservoir depends on the distance-weighted collision term $\sum_{i<j} p_{ij}\|\mathbf{h}_i - \mathbf{h}_j\|_2^2$: by preferentially grouping similar features and separating dissimilar ones, locality-sensitive hashing can yield a tighter distributional approximation bound than uniform sampling. We discuss this comparison in detail in Appendix C.5.

5.3 Remarks and Practical Guidance

- Exact mean preservation is an immediate and robust advantage for any downstream task that depends on first-moment neighborhood features.
- The key quantity controlling reservoir error is the pairwise collision probability sum $\sum_{i<j} p_{ij}$ (or $p_{\max}$). Designing the hash to make p_{ij} small for cross-mode / dissimilar pairs directly reduces the bounds.
- Sampling must increase s proportionally to $1/p_{\max}$ (in the worst-case mixing scenario) to match reservoir fidelity. Thus, the reservoir is particularly advantageous when neighborhoods contain rare but predictive sub-populations or multiple well-separated modes.

Table 1. Test MRR comparison between our method and state-of-the-art methods on different benchmarks. Results reported from 5 runs. The best event-based methods and snapshot-based methods are **bold**. OOM denotes the out of memory. We use the result of event-based methods in UTG [14] directly, as training these models from scratch is too time-consuming.

	Method	Venue	UCI	Enron	Contacts	Social Evo.	MOOC	tgbl-wiki
event	TGN [11]	ICML20	0.091	0.191	0.153	0.283	0.174	0.396
	DyGFormer [10]	NeurIPS23	0.334	**0.331**	**0.283**	**0.366**	OOM	**0.798**
	NAT [9]	LoG22	**0.356**	0.276	0.245	0.258	**0.283**	0.749
	GraphMixer [28]	ICLR23	0.105	0.296	0.055	0.157	OOM	0.118
snapshot	HTGN [6]	KDD21	0.090	0.263	0.159	0.226	0.091	0.563
	GCLSTM [7]	Appl. Intell22	0.077	0.166	0.129	0.278	0.121	0.321
	EGCNo [8]	AAAI20	0.115	0.218	0.191	0.248	0.136	0.413
	GCN [17]	ICLR17	0.068	0.159	0.099	0.282	0.072	0.280
	ROLAND [5]	KDD22	0.100	0.252	0.143	0.239	0.119	0.426
	Hawkes-GAT [29]	KDD25	0.097	0.213	**0.234**	**0.367**	0.074	0.317
	Ours	-	**0.150**	**0.315**	0.207	0.263	**0.179**	**0.629**

Organization Note. All lengthy proofs and derivations are presented in Appendix C to keep the main text concise.

6 Experiments

Dataset We evaluate our model on a diverse set of dynamic graph benchmarks, including several Discrete-Time Dynamic Graphs (DTDGs), namely, UCI, Enron, Contacts, Social Evo., MOOC and one Continuous-Time Dynamic Graph (CTDG), tgbl-wiki. Our entire experimental setup, encompassing data partitioning, temporal granularity, and other preprocessing steps, is identical to that of UTG [14] to ensure reproducibility and fair comparison. The detailed statistics of all the datasets are shown in Table 4.

Implementation Our model architecture aggregates information from both 1-hop and 2-hop neighborhoods. The key hyperparameters are specifically configured for each data paradigm. For DTDG benchmarks, we set the number of reservoirs to 16 per hop as a result of the grid search, train for a fixed 200 epochs without early stopping, and use a learning rate of 0.001. For the CTDG benchmark, these settings are adjusted to 4 reservoirs per hop, 40 training epochs, and a learning rate of 0.0002. All other unspecified hyperparameters adhere to the configuration in UTG [14] to ensure a fair comparison.

Compared Methods and Evaluation To rigorously evaluate our method, we conduct a comparative analysis against a diverse suite of state-of-the-art models. Our baselines are composed of six snapshot-based models (HTGN [6], GCLSTM [7], EvolveGCN [8], GCN [17], ROLAND [5], Hawkes-GAT [29]) and four leading event-based models (TGN [11], DyGFormer [10], NAT [9], GraphMixer [28]). Performance is quantified using Mean Reciprocal Rank (MRR), and for fairness and reproducibility, we adhere to the negative sampling strategy proposed in UTG [14].

6.1 Comparison with SOTA

We benchmark our proposed model, NAGNN, against a comprehensive suite of state-of-the-art methods from both the snapshot-based and event-based paradigms. The evaluation is conducted across two key dimensions: predictive accuracy and computational efficiency.

Performance Analysis. As demonstrated in Table 1, NAGNN consistently and significantly outperforms nearly all snapshot-based baselines across the diverse set of datasets. Notably, the only cases where some snapshot-based methods

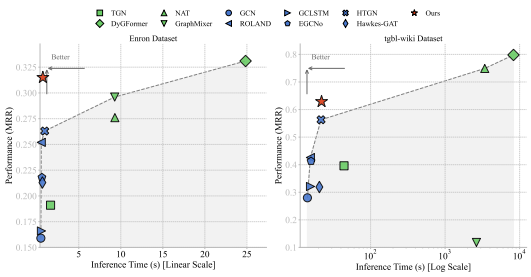
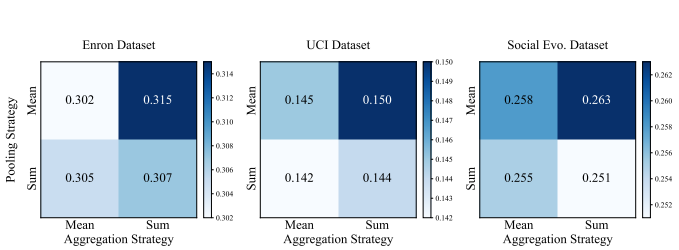
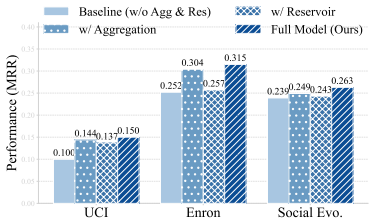


Figure 2: Performance-efficiency trade-off on Enron (left) and tgbl-wiki (right). Each unique marker corresponds to a specific model. The dashed line denotes the Pareto Frontier[30], calculated from all baseline methods.



(a) Ablation results on aggregation/pooling strategies.



(b) Ablation results on model components

Figure 3: Ablation results. **(a)** The result of performance comparison on different aggregation in Equation (5) and pooling in Equation (8) strategies. **(b)** The result of ablation analysis on the neighborhood aggregation (Section 4.2) and the reservoirs (Section 4.3).

surpass our model are on the Contacts and Social Evo datasets. This underperformance can be attributed to the highly predictable nature of the datasets, as evidenced by their low surprise index [31]. This performance gap underscores the critical limitation of traditional snapshot methods: their inability to capture the fine-grained evolution of neighborhood structures between snapshots. Our neighbor-aware design directly rectifies this weakness by encoding rich, dynamic structural information, leading to more accurate predictions. For instance, on Enron, our model achieves an MRR of 0.315, substantially outperforming the strongest snapshot-based rival, HTGN (0.267). Simultaneously, it matches or exceeds the performance of leading event-based models like GraphMixer and NAT. This efficiency advantage is particularly stark on the large-scale MOOC dataset, where powerful event-based methods like DyGFormer fail due to Out-of-Memory (OOM) errors, while NAGNN operates robustly.

The Balance between Inference Time and MRR. To evaluate computational efficiency, we measure the inference time of all snapshot-based methods and TGN, the most efficient event-based baseline, on the same NVIDIA GeForce RTX 3090 GPU, averaging over five independent runs on Enron and tgbl-wiki. For the remaining advanced event-based baselines, we report the inference times provided by UTG [14], as reproducing these methods under the same setting would incur substantially higher computational cost. The results, presented in Fig. 2, clearly illustrate the performance-efficiency trade-off inherent to temporal graph learning. As shown, event-based methods typically occupy the high-performance, high-cost region, while snapshot-based methods are faster but often less accurate. Our model, NAGNN, distinguishes itself by achieving accuracy competitive with the top-performing event-based models (e.g., DyGFormer, NAT) while operating at an inference speed that is orders of magnitude lower and on par with the most efficient snapshot-based approaches.

6.2 Ablation Analysis

Aggregation and Pooling Strategy Our framework incorporates two critical operators: a neighborhood aggregation strategy to combine features from all neighbors (Equation (5)), and a reservoir pooling strategy to summarize features within each slot in the reservoir (Equation (8)). For both, we investigate two canonical choices: mean and sum. We conduct an ablation study on the Enron, UCI, and Social Evo. datasets to determine the optimal combination, with results presented in Figure 3a.

The experiments consistently reveal that sum aggregation paired with mean pooling yields superior performance across all datasets. We hypothesize a two-fold explanation for this phenomenon. First, sum aggregation implicitly preserves the node degree, a crucial structural statistic that characterizes the neighborhood's scale and distribution. Conversely, within the LSH reservoirs, the number of features in the same reservoir is inherently stochastic due to the random projections. In this context, mean pooling provides a stable, normalized summary, whereas sum pooling would introduce significant feature variance tied to the random feature count, thereby degrading model performance.

Ablation Study on Model Components. To isolate the impact of our two components, our ablation study (Figure 3b) reveals that compared to the ROLAND baseline [5], the neighborhood aggregation module alone yields substantial performance gains, whereas the LSH reservoir module offers only a marginal improvement in isolation. However, the true benefit emerges from their synergy. The

full model significantly outperforms either in isolation, confirming a complementary relationship. We attribute this to a two-stage encoding process: the aggregation module first captures a robust, degree-aware global context, which the reservoir module then enriches with high-resolution structural details. This synergy validates our core hypothesis that fusing global and fine-grained neighborhood information is essential for state-of-the-art performance.

Organization Note. More ablation and sensitivity analysis is shown in Section E.

7 Conclusion

In this paper, we resolve the fundamental trade-off between structural fidelity and computational efficiency in dynamic graph learning. We present NAGNN, the first snapshot-based framework capable of leveraging fine-grained joint neighborhood information, a critical capability previously exclusive to computationally expensive event-based models. By introducing a novel encoding scheme that combines LSH-based reservoir memory with degree-scaled statistical moments, NAGNN compactly preserves complex topological semantics. Crucially, we theoretically prove that this mechanism rigorously bounds the structural approximation error. When integrated with a GRU for temporal evolution, NAGNN establishes a new performance-efficiency frontier. Extensive experiments demonstrate that NAGNN achieves competitive predictive performance against strong event-based methods while retaining the substantial computational efficiency of snapshot-based processing. Overall, NAGNN offers an effective and scalable approach for incorporating rich neighborhood information into snapshot-based temporal graph learning, providing a practical foundation for large-scale dynamic graph applications.

Limitations and Future Work. While NAGNN demonstrates strong performance across most benchmarks, a noticeable gap remains relative to computationally intensive event-based methods (e.g., NAT [9] and DyGFormer [10]) on dense datasets such as UCI and MOOC. One possible explanation is that dense snapshots may aggregate a large number of interactions within each interval, resulting in more complex neighborhood changes that are harder for snapshot-based models to capture at a fine-grained level. A finer temporal granularity could potentially produce sparser snapshots and simplify the structural evolution within each interval, thereby facilitating more effective joint neighborhood modeling. Exploring adaptive temporal granularity and granularity-aware snapshot construction therefore represents a promising direction for further improving the performance–efficiency trade-off of snapshot-based temporal graph models.

Acknowledgments

This work was supported in part by the National Key Research and Development Program of China (Grant No. 2024YFC3308403), in part by the National Natural Science Foundation of China (Grant No. T254100759, 62506269). This work was also supported by the WHU-Kingsoft Joint Lab. The supercomputing system at the Supercomputing Center of Wuhan University supported the numerical calculations in this paper.

References

[1] Yanbang Wang, Pan Li, Chongyang Bai, and Jure Leskovec. Tedic: Neural modeling of behavioral patterns in dynamic social interaction networks. In *Proceedings of the Web Conference 2021*, pages 693–705, 2021.

[2] Manuel Dileo, Matteo Zignani, and Sabrina Gaito. Temporal graph learning for dynamic link prediction with text in online social networks. *Machine learning*, 113(4):2207–2226, 2024.

[3] Yejin Kim, Youngbin Lee, Vincent Yuan, Annika Lee, and Yongjae Lee. A temporal graph network framework for dynamic recommendation. *arXiv preprint arXiv:2403.16066*, 2024.

[4] Amirhossein Nadiri and Frank W Takes. A large-scale temporal analysis of user lifespan durability on the reddit social media platform. In *Companion Proceedings of the Web Conference 2022*, pages 677–685, 2022.

[5] Jiaxuan You, Tianyu Du, and Jure Leskovec. Roland: graph learning framework for dynamic graphs. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, pages 2358–2366, 2022.

[6] Menglin Yang, Min Zhou, Marcus Kalander, Zengfeng Huang, and Irwin King. Discrete-time temporal network embedding via implicit hierarchical learning in hyperbolic space. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pages 1975–1985, 2021.

[7] Jinyin Chen, Xueke Wang, and Xuanheng Xu. Gc-lstm: Graph convolution embedded lstm for dynamic network link prediction. *Applied Intelligence*, 52(7):7513–7528, 2022.

[8] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao Schardl, and Charles Leiserson. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 5363–5370, 2020.

[9] Yuhong Luo and Pan Li. Neighborhood-aware scalable temporal network representation learning. In *Learning on Graphs Conference*, pages 1–1. PMLR, 2022.

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

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

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

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

[14] Shenyang Huang, Farimah Poursafaei, Reihaneh Rabbany, Guillaume Rabusseau, and Emanuele Rossi. Utg: Towards a unified view of snapshot and event based models for temporal graphs. *arXiv preprint arXiv:2407.12269*, 2024.

[15] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. Similarity search in high dimensions via hashing. In *Vldb*, volume 99, pages 518–529, 1999.

[16] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[17] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[18] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[19] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[20] Claudio DT Barros, Matheus RF Mendonça, Alex B Vieira, and Artur Ziviani. A survey on embedding dynamic graphs. *ACM Computing Surveys (CSUR)*, 55(1):1–37, 2021.

[21] Qionghao Huang and Jili Chen. Enhancing academic performance prediction with temporal graph networks for massive open online courses. *Journal of Big Data*, 11(1):52, 2024.

[22] Jiasheng Zhang, Jialin Chen, Menglin Yang, Aosong Feng, Shuang Liang, Jie Shao, and Rex Ying. Dtgb: A comprehensive benchmark for dynamic text-attributed graphs. *Advances in Neural Information Processing Systems*, 37:91405–91429, 2024.

[23] Shenyang Huang, Farimah Poursafaei, Jacob Danovitch, Matthias Fey, Weihua Hu, Emanuele Rossi, Jure Leskovec, Michael Bronstein, Guillaume Rabusseau, and Reihaneh Rabbany. Temporal graph benchmark for machine learning on temporal graphs. *Advances in Neural Information Processing Systems*, 36: 2056–2073, 2023.

[24] ZhengZhao Feng, Rui Wang, TianXing Wang, Mingli Song, Sai Wu, and Shuibing He. A comprehensive survey of dynamic graph neural networks: Models, frameworks, benchmarks, experiments and challenges. *IEEE Transactions on Knowledge and Data Engineering*, 2025.

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

[26] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In *Proceedings of the web conference 2020*, pages 2704–2710, 2020.

[27] Jinwoo Kim, Dat Nguyen, Seonwoo Min, Sungjun Cho, Moontae Lee, Honglak Lee, and Seunghoon Hong. Pure transformers are powerful graph learners. *Advances in Neural Information Processing Systems*, 35: 14582–14595, 2022.

[28] Weilin Cong, Si Zhang, Jian Kang, Baichuan Yuan, Hao Wu, Xin Zhou, Hanghang Tong, and Mehrdad Mahdavi. Do we really need complicated model architectures for temporal networks? *arXiv preprint arXiv:2302.11636*, 2023.

[29] QingGuo Qi, Hongyang Chen, Minhao Cheng, and Han Liu. Input snapshots fusion for scalable discrete-time dynamic graph neural networks. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pages 1138–1149, 2025.

[30] Vilfredo Pareto. *Manuale di economia politica con una introduzione alla scienza sociale*, volume 13. Società editrice libraria, 1919.

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

[32] Colin McDiarmid et al. On the method of bounded differences. *Surveys in combinatorics*, 141(1):148–188, 1989.

[33] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The journal of machine learning research*, 13(1):723–773, 2012.

[34] Vladimir S Korolyuk and Yu V Borovskich. *Theory of U-statistics*, volume 273. Springer Science & Business Media, 2013.

Table 2: Summary of key notations used for Section 4.

Symbol	Description
$\mathcal{G}, \mathcal{G}^{(t)}$	A dataset, and a graph snapshot at time t
$\mathcal{V}, \mathcal{E}^{(t)}$	The set of nodes (vertices) and edges in the snapshot at time t
$\mathbf{x}_v^{(t)} \in \mathbb{R}^d$	The d-dimensional input feature vector for node v at snapshot t
$\mathbf{h}_v^{(t)} \in \mathbb{R}^{d_h}$	The d_h-dimensional learned embedding for node v at snapshot t by the GAT
$\mathbf{H}^{(t)} \in \mathbf{R}^{N \times d_h}$	The matrix made by node features in snapshot t
$\mathbf{z}_v^{(t)} \in \mathbb{R}^{5*d_h}$	The final representation for node v at snapshot t
$\mathbf{P} \in \mathbb{R}^{d_h \times k}$	The random projection matrix in the LSH
$\mathbf{R}_{v,k} \in \mathbb{R}^{R \times d_h}$	The embedding matrices of the reservoir of the k-hop neighborhood
$\mathbf{a}_{v,k}^{(t)}, \mathbf{a}_{v,k}^{\mathrm{sq}(t)} \in \mathbb{R}^{d_h}$	The degree-scaled first moment (the sum of features) and second raw moment (the sum of squared features) of the k-hop neighborhood
$\mathbf{n}_{v,k}^{(t)} \in \mathbb{R}^{d_h}$	the k-hop temporally-smoothed neighborhood aggregate
$\mathbf{r}_{v,k}^{(t)} \in \mathbb{R}^{d_h}$	the k-hop reservoir-based structural summary

Table 3: Summary of key notations used for Section 5 and Section C.

Symbol	Description
$\mathbf{h}_i \in \mathbb{R}^{d_h}$	The feature of i-th neighbor nodes
$\mathbf{r}_s \in \mathbb{R}^{d_h}$	The feature of the s-th slot in the reservoir
$\mathcal{N}$	The set of neighbor features
$\mathcal{R}$	The set of features in reservoir
$\mathbf{p}_l$	The l-th hash projection vector
$\mathbf{P}$	The hash projection matrix
$k \in \mathbb{N}$	The number of projection vectors
SSE	Sum-of-Squared Errors
$\mathbf{\Sigma}_{\mathcal{N}}$	Empirical covariance of the neighbor set
$\mathbf{\Sigma}_{\mathcal{R}}$	Empirical covariance of the reservoir
$p_{i,j}$	The collision probability between feature $\mathbf{h}_i$ and $\mathbf{h}_j$
MMD	Maximum Mean Discrepancy
$\mu_{\mathcal{N}}$	Dirac measure of $\mathcal{N}$
$\mu_{\mathcal{R}}$	Dirac measure of $\mathcal{R}$

A Notation

A.1 Notations for Section 4

The notations used in Section 4 are shown in Table 2.

A.2 Notations for Section 5 and Section C

The notation used in Section 5 and Section C are shown in Table 3.

B Formal Theorem

Theorem B.1 (Distributional approximation: MMD bound)**.** *Let $\{\mathbf{h}_i\}_{i=1}^m \subset \mathbb{R}^{d_h}$ satisfy $\|\mathbf{h}_i\|_2 \leq B$. Let K be a positive-definite kernel with RKHS $\mathcal{H}$ and feature map ϕ, and assume that ϕ is L_K-Lipschitz:*

$$\|\phi(\mathbf{x}) - \phi(\mathbf{y})\|_{\mathcal{H}} \leq L_K \|\mathbf{x} - \mathbf{y}\|_2.$$

We define the following quantities:

1. ***Dirac Measure and Empirical Measures:*** *The empirical measure of the neighborhood is defined as*

$$\mu_{\mathcal{N}} \triangleq \frac{1}{m} \sum_{i=1}^{m} \delta_{\mathbf{h}_i},$$

where δ_x denotes the ***Dirac measure*** *which places a unit mass at the point x. For a measurable set A, $\delta_x(A) = 1$ if $x \in A$ and 0 otherwise.*

Similarly, given a random hash function $s(\cdot)$ and corresponding reservoir slot embeddings $\{\mathbf{r}_s\}$, the empirical measure of the reservoir is defined as

$$\mu_{\mathcal{R}} \triangleq \sum_{s} \frac{n_s}{m} \delta_{\mathbf{r}_s},$$

where $n_s = |\{i \mid s(i) = s\}|$ is the number of items hashed to slot s.

2. ***Squared Maximum Mean Discrepancy (MMD):*** *The squared MMD between two probability measures μ and ν on $\mathcal{X}$ is defined as the squared distance between their mean embeddings in the RKHS associated with K:*

$$\begin{aligned}\mathrm{MMD}_K^2(\mu, \nu) \triangleq & \mathbb{E}_{\mathbf{x},\mathbf{x}' \sim \mu}[K(x, x')] \\ & - 2\mathbb{E}_{\mathbf{x} \sim \mu, \mathbf{y} \sim \nu}[K(x, y)] \\ & + \mathbb{E}_{\mathbf{y},\mathbf{y}' \sim \nu}[K(y, y')].\end{aligned}$$

Then, for any fixed hash assignment,

$$\mathrm{MMD}_K^2(\mu_{\mathcal{N}}, \mu_{\mathcal{R}}) \leq \frac{L_K^2}{m} \mathrm{SSE}\,.$$

Moreover, taking expectation over the random hash function,

$$\mathbb{E}\big[\mathrm{MMD}_K^2(\mu_{\mathcal{N}}, \mu_{\mathcal{R}})\big] \leq \frac{L_K^2}{m} \sum_{i<j} p_{ij} \|\mathbf{h}_i - \mathbf{h}_j\|_2^2 \leq \frac{4L_K^2 B^2}{m} \sum_{i<j} p_{ij},$$

where

$$p_{ij} \coloneqq \Pr(s(i) = s(j)).$$

This theorem reveals a fundamental advantage of our approach by linking the distributional error directly to the interplay between data similarity and the hash function's behavior. The error bound is a sum of pairwise distances (Δ_{ij}^K), each weighted by the probability of collision (p_{ij}). This structure shows that the error is inherently minimized by a hash function that separates dissimilar items (large Δ_{ij}^K) while grouping similar ones. Our use of SimHash, a Locality-Sensitive Hash, is expressly designed for this purpose, performing an intelligent, error-aware summarization that adapts to the local geometry of the neighborhood data.

C Proofs and Full Derivations

C.1 Proof of Theorem 5.1

Proof. By definition,

$$\bar{\mathbf{r}} = \sum_{s} \frac{n_s}{m} \mathbf{r}_s \tag{14}$$

$$= \frac{1}{m} \sum_{s} \sum_{i:\, s(i)=s} \mathbf{r}_s \tag{15}$$

$$= \frac{1}{m} \sum_{s} \sum_{i:\, s(i)=s} \frac{1}{n_s} \sum_{j:\, s(j)=s} \mathbf{h}_j. \tag{16}$$

Interchanging sums yields $\bar{\mathbf{r}} = \frac{1}{m} \sum_{j=1}^{m} \mathbf{h}_j = \bar{\mathbf{h}}$. □

C.2 Proof of Theorem 5.2 (SimHash Concentration)

We give the complete McDiarmid-based derivation and the deterministic inequalities used to pass from collision counts to SSE and operator-norm.

Step 1: Deterministic SSE Upper Bound via Collisions. For each occupied slot s define $n_s = \#\{i : s(i) = s\}$ and slot mean $\mathbf{r}_s$. Using the identity

$$\sum_{i:\, s(i)=s} \|\mathbf{h}_i - \mathbf{r}_s\|^2 = \frac{1}{2n_s} \sum_{i,j:\, s(i)=s} \|\mathbf{h}_i - \mathbf{h}_j\|^2,$$

sum over s and use $\|\mathbf{h}_i - \mathbf{h}_j\| \le 2B$ to obtain

$$\mathrm{SSE} = \frac{1}{2} \sum_s \frac{1}{n_s} \sum_{i,j:\, s(i)=s} \|\mathbf{h}_i - \mathbf{h}_j\|^2 \tag{17}$$

$$\le \frac{1}{2} \sum_s \sum_{i,j:\, s(i)=s} \|\mathbf{h}_i - \mathbf{h}_j\|^2 \tag{18}$$

$$= \frac{1}{2} \sum_{i \ne j} \mathbf{1}\{s(i) = s(j)\} \|\mathbf{h}_i - \mathbf{h}_j\|^2. \tag{19}$$

Using $\|\mathbf{h}_i - \mathbf{h}_j\|^2 \le (2B)^2 = 4B^2$, then we can obtain:

$$\mathrm{SSE} \le 2B^2 \sum_{i \ne j} \mathbf{1}\{s(i) = s(j)\} = 4B^2 S.$$

Step 2: McDiarmid Bound on S. Let $S = f(\mathbf{p}_0, \ldots, \mathbf{p}_{k-1})$ with independent Gaussian vectors $\mathbf{p}_\ell$. Change a single projection $\mathbf{p}_\ell \mapsto \mathbf{p}'_\ell$. Let A be the set of indices i whose sign at coordinate ℓ flips. Then the number of unordered crossing pairs whose ℓ-bit-equality may change is at most $|A|(m-|A|) \le \lfloor m^2/4 \rfloor$. Thus $|S' - S| \le m^2/4$ deterministically. Apply McDiarmid Ineuqality [32]:

$$\Pr(S - \mathbb{E}[S] \ge t) \le \exp\Big(- \frac{2t^2}{\sum_{\ell=0}^{k-1} (m^2/4)^2} \Big) = \exp\Big(- \frac{32t^2}{km^4} \Big).$$

Set the RHS to δ and solve for t to obtain the deviation term

$$t = m^2 \sqrt{\frac{k \ln(1/\delta)}{32}}.$$

Thus $S \le \bar{S} + m^2 \sqrt{\frac{k \ln(1/\delta)}{32}}$ with probability $\ge 1 - \delta$.

Step 3: Translate to SSE and Operator Norm. We begin by proving $\mathbf{\Sigma}_\mathcal{N} - \mathbf{\Sigma}_\mathcal{R}$ is positive semidefinite matrix, noting that $\bar{\mathbf{r}} = \sum_s \frac{n_s}{m} \mathbf{r}_s = \frac{1}{m} \sum_s \sum_{i:s(i)=s} \mathbf{h}_i = \bar{\mathbf{h}}$.

$$\begin{aligned}
\mathbf{\Sigma}_\mathcal{N} &= \frac{1}{m} \sum_{i=1}^{m} (\mathbf{h}_i - \bar{\mathbf{h}})(\mathbf{h}_i - \bar{\mathbf{h}})^\top \\
&= \frac{1}{m} \sum_s \sum_{i:s(i)=s} \big((\mathbf{h}_i - \mathbf{r}_s) \\
&\quad + (\mathbf{r}_s - \bar{\mathbf{h}}) \big) \big((\mathbf{h}_i - \mathbf{r}_s) + (\mathbf{r}_s - \bar{\mathbf{h}}) \big)^\top \\
&= \frac{1}{m} \sum_s \\
&\quad \left[\sum_{i:s(i)=s} (\mathbf{h}_i - \mathbf{r}_s)(\mathbf{h}_i - \mathbf{r}_s)^\top + n_s (\mathbf{r}_s - \bar{\mathbf{h}})(\mathbf{r}_s - \bar{\mathbf{h}})^\top \right] \\
&= \frac{1}{m} \sum_s \sum_{i:s(i)=s} (\mathbf{h}_i - \mathbf{r}_s)(\mathbf{h}_i - \mathbf{r}_s)^\top + \mathbf{\Sigma}_\mathcal{R}.
\end{aligned}$$

The cross-terms vanish since $\sum_{i:s(i)=s}(\mathbf{h}_i - \mathbf{r}_s) = \mathbf{0}$. Rearranging gives $\mathbf{A} = \mathbf{\Sigma}_{\mathcal{N}} - \mathbf{\Sigma}_{\mathcal{R}} = \frac{1}{m}\sum_s \sum_{i:s(i)=s}(\mathbf{h}_i - \mathbf{r}_s)(\mathbf{h}_i - \mathbf{r}_s)^\top$. As a sum of outer products (which are PSD matrices), $\mathbf{A}$ is PSD. For any symmetric PSD matrix $\mathbf{A}$, its eigenvalues λ_j are non-negative, thus its operator norm $\|\mathbf{A}\|_{\mathrm{op}} = \lambda_{\max}(\mathbf{A}) \le \sum_j \lambda_j(\mathbf{A}) = \mathrm{tr}(\mathbf{A})$. Therefore, $\|\mathbf{\Sigma}_{\mathcal{N}} - \mathbf{\Sigma}_{\mathcal{R}}\|_{\mathrm{op}} \le \mathrm{tr}(\mathbf{\Sigma}_{\mathcal{N}} - \mathbf{\Sigma}_{\mathcal{R}})$.

Then combine the deterministic SSE bound and the McDiarmid deviation to conclude (12). Use

$$\|\mathbf{\Sigma}_{\mathcal{N}} - \mathbf{\Sigma}_{\mathcal{R}}\|_{\mathrm{op}} \le \mathrm{tr}(\mathbf{\Sigma}_{\mathcal{N}} - \mathbf{\Sigma}_{\mathcal{R}}) = \frac{1}{m}\mathrm{SSE}$$

to obtain (13). This completes the proof.

C.3 Proof of Theorem B.1

Proof. Let ϕ denote the RKHS feature map associated with K. Since

$$\mu_{\mathcal{R}} = \sum_s \frac{n_s}{m}\delta_{\mathbf{r}_s} = \frac{1}{m}\sum_{i=1}^m \delta_{\mathbf{r}_{s(i)}},$$

the squared MMD can be written as

$$\mathrm{MMD}_K^2(\mu_{\mathcal{N}}, \mu_{\mathcal{R}}) = \left\| \frac{1}{m}\sum_{i=1}^m \big(\phi(\mathbf{h}_i) - \phi(\mathbf{r}_{s(i)})\big) \right\|_{\mathcal{H}}^2.$$

By Jensen's inequality,

$$\mathrm{MMD}_K^2(\mu_{\mathcal{N}}, \mu_{\mathcal{R}}) \le \frac{1}{m}\sum_{i=1}^m \big\|\phi(\mathbf{h}_i) - \phi(\mathbf{r}_{s(i)})\big\|_{\mathcal{H}}^2.$$

Using the L_K-Lipschitz property of ϕ,

$$\mathrm{MMD}_K^2(\mu_{\mathcal{N}}, \mu_{\mathcal{R}}) \le \frac{L_K^2}{m}\sum_{i=1}^m \|\mathbf{h}_i - \mathbf{r}_{s(i)}\|_2^2 = \frac{L_K^2}{m}\,\mathrm{SSE}.$$

From the deterministic SSE relation established in Equation (19),

$$\mathrm{SSE} \le \sum_{i<j} \mathbf{1}\{s(i) = s(j)\}\|\mathbf{h}_i - \mathbf{h}_j\|_2^2.$$

Therefore,

$$\mathrm{MMD}_K^2(\mu_{\mathcal{N}}, \mu_{\mathcal{R}}) \le \frac{L_K^2}{m}\sum_{i<j} \mathbf{1}\{s(i) = s(j)\}\|\mathbf{h}_i - \mathbf{h}_j\|_2^2.$$

Taking expectation over the random hash function and using

$$\mathbb{E}[\mathbf{1}\{s(i) = s(j)\}] = p_{ij},$$

we obtain

$$\mathbb{E}\big[\mathrm{MMD}_K^2(\mu_{\mathcal{N}}, \mu_{\mathcal{R}})\big] \le \frac{L_K^2}{m}\sum_{i<j} p_{ij}\|\mathbf{h}_i - \mathbf{h}_j\|_2^2.$$

Finally, since $\|\mathbf{h}_i\|_2 \le B$,

$$\|\mathbf{h}_i - \mathbf{h}_j\|_2 \le \|\mathbf{h}_i\|_2 + \|\mathbf{h}_j\|_2 \le 2B,$$

and hence

$$\mathbb{E}\big[\mathrm{MMD}_K^2(\mu_{\mathcal{N}}, \mu_{\mathcal{R}})\big] \le \frac{4L_K^2 B^2}{m}\sum_{i<j} p_{ij}.$$

This completes the proof. □

C.4 Proof of Proposition 5.4 (Sampling MMD Bound)

Proof. Let $\widehat{\mu}_s = \frac{1}{s}\sum_{t=1}^{s} \delta_{z_t}$ be the empirical measure of s points sampled uniformly without replacement from $\mathcal{N}$. For a kernel with $K(x,x) \le K_0$ the variance of the empirical kernel mean estimator is bounded by $O(K_0/s)$. Concretely (standard U-statistic argument [33, 34]),

$$\mathbb{E}\big[\|\phi_{\mu_\mathcal{N}} - \phi_{\widehat{\mu}_s}\|_\mathcal{H}^2\big] \le \frac{4K_0}{s},$$

where $\phi_\nu = \mathbb{E}_{\mathbf{x}\sim\nu}[K(\cdot,x)]$ is the mean embedding in RKHS. This yields the stated MMD bound. (A short derivation: expand the squared norm, use sampling without replacement covariance formulas and bound diagonal terms by K_0.) □

C.5 General Superiority Comparison (Formal Statement and Derivation)

We formalize the theoretical advantage of the reservoir through its distributional approximation bounds.

Theorem C.1 (Distributional approximation-bound comparison)**.** *Let K be a positive-definite kernel satisfying $K(\mathbf{x},\mathbf{x}) \le K_0$, and let its RKHS feature map ϕ be L_K-Lipschitz. Let $\phi_\mathcal{R}$ and $\phi_\mathcal{N}$ denote the reservoir and neighborhood mean embeddings, respectively, and let μ_{bs} be an s-sample empirical measure. Then*

$$\mathbb{E}\left[\|\phi_\mathcal{N} - \phi_\mathcal{R}\|_\mathcal{H}^2\right] \le \frac{L_K^2}{m}\sum_{i<j} p_{ij}\|\mathbf{h}_i - \mathbf{h}_j\|_2^2,$$

while

$$\mathbb{E}\left[\|\phi_\mathcal{N} - \phi_{\mu_{\mathrm{bs}}}\|_\mathcal{H}^2\right] \le \frac{4K_0}{s}.$$

Consequently, when

$$\frac{L_K^2}{m}\sum_{i<j} p_{ij}\|\mathbf{h}_i - \mathbf{h}_j\|_2^2 < \frac{4K_0}{s},$$

the LSH-based reservoir admits a tighter derived distributional approximation bound than the s-sample.

Proof. The first inequality is Theorem B.1. The second is Proposition 5.4. The comparison follows directly by comparing right-hand sides. □

Implication for Downstream Risk. Let a downstream predictor be a deterministic function $\Psi : \mathcal{M} \to \mathbb{R}$, where $\mathcal{M}$ is the representation space, and assume that Ψ is L-Lipschitz with respect to the RKHS norm: for any two embeddings ϕ, ϕ',

$$|\Psi(\phi) - \Psi(\phi')| \le L\|\phi - \phi'\|_\mathcal{H}.$$

Let the loss $\ell(y,\hat{y})$ be L_ℓ-Lipschitz in its second argument. Then, for any approximate embedding $\widetilde{\phi}$,

$$\begin{aligned}
&\Big|\mathbb{E}[\ell(Y,\Psi(\widetilde{\phi}))] - \mathbb{E}[\ell(Y,\Psi(\phi_\mathcal{N}))]\Big| \\
&\quad\le L_\ell L\,\mathbb{E}\Big[\|\widetilde{\phi} - \phi_\mathcal{N}\|_\mathcal{H}\Big] \\
&\quad\le L_\ell L\sqrt{\mathbb{E}\Big[\|\widetilde{\phi} - \phi_\mathcal{N}\|_\mathcal{H}^2\Big]}.
\end{aligned}$$

Applying Theorem C.1 to the reservoir gives

$$|\mathbb{E}[\ell(Y,\Psi(\phi_\mathcal{R}))] - \mathbb{E}[\ell(Y,\Psi(\phi_\mathcal{N}))]| \le L_\ell L\sqrt{\frac{L_K^2}{m}\sum_{i<j} p_{ij}\|\mathbf{h}_i - \mathbf{h}_j\|_2^2},$$

whereas uniform sampling satisfies

$$|\mathbb{E}[\ell(Y,\Psi(\phi_{\mu_{\mathrm{bs}}}))] - \mathbb{E}[\ell(Y,\Psi(\phi_\mathcal{N}))]| \le 2L_\ell L\sqrt{\frac{K_0}{s}}.$$

Table 4: The statistics of the datasets.

Dataset	# Nodes	# Edges	# Unique Edges	Surprise	Time Granularity	# Snapshots
UCI	1,899	26,628	20,296	0.535	Weekly	29
Enron	184	10,472	3,125	0.253	Monthly	45
Contact	694	463,558	79,531	0.098	Hourly	673
Social Evo.	74	87,479	4,486	0.005	Daily	244
MOOC	7,144	236,808	178,443	0.718	Daily	31
`tgb1-wiki`	9,227	157,474	18,257	0.108	UNIX timestamp	745 (Hourly)

Therefore, whenever

$$\frac{L_K^2}{m} \sum_{i<j} p_{ij} \|\mathbf{h}_i - \mathbf{h}_j\|_2^2 \leq \frac{4K_0}{s},$$

the LSH-based reservoir also admits a no-larger derived upper bound on downstream excess risk than uniform sampling. Importantly, the reservoir bound is governed by a distance-weighted collision term, which directly reflects the locality-sensitive nature of SimHash: similar features can be compressed together with limited distortion, while dissimilar features are preferentially separated. Hence, the theoretical benefit of LSH-based neighborhood compression extends from preserving the neighborhood distribution to controlling its impact on downstream prediction.

D Statistics of Datasets

The statistics of the datasets used in this paper are shown in Table 4.

E Ablation and Sensitivity Analysis

E.1 Effectiveness of the LSH-Based Reservoir Mechanism

To isolate the contribution of our LSH-based memory from the general snapshot-based architecture, we evaluate NAGNN against two structural alternatives: (1) **ROLAND**, a standard state-of-the-art snapshot GNN that lacks joint neighborhood memory, and (2) **Uniform Sampling (US)**, a variant of our model that replaces the LSH hashing mechanism with a naive, random uniform sampling of neighbors into the reservoir slots.

As shown in Table 5, incorporating joint neighborhood information—even via naive uniform sampling—yields substantial improvements over the base ROLAND architecture on most datasets. This validates our core hypothesis that joint neighborhood awareness is a missing, critical component for snapshot-based dynamic link prediction.

Furthermore, the full NAGNN model consistently and significantly outperforms the US variant across all benchmarks. This performance gap highlights the necessity of the LSH mechanism. While uniform sampling blindly aggregates neighbors, LSH systematically hashes topologically similar features into the same reservoir slots. Consequently, NAGNN successfully preserves fine-grained local structural distributions (as theoretically guaranteed in Theorem **??**). Conversely, random sampling homogenizes and destroys these subtle structural signals. This vulnerability is especially evident on dense, complex graphs like `tgbl-wiki`, where naive uniform sampling actually degrades performance below the ROLAND baseline, proving that how neighborhood information is compressed is just as important as having it.

Table 5: Ablation study on the reservoir update mechanism. Performance is measured by Mean Reciprocal Rank (MRR). US denotes the NAGNN variant using Uniform Sampling instead of LSH.

Method	UCI	Enron	Contacts	Social Evo.	MOOC	tgbl-wiki
ROLAND	0.100	0.252	0.143	0.239	0.119	0.426
US (Ours w/o LSH)	0.142	0.307	0.206	0.252	0.172	0.272
NAGNN (Ours)	**0.150**	**0.315**	**0.207**	**0.263**	**0.179**	**0.629**

Table 6: Sensitivity analysis of the reservoir size (R) on the Enron dataset. Performance is measured by Mean Reciprocal Rank (MRR). The optimal balance between fidelity and sparsity is achieved at $R = 16$.

Reservoir Size (R)	4	8	12	16	20	24	28	32	48
MRR	0.303	0.300	0.309	**0.315**	0.311	0.302	0.301	0.304	0.292

Table 7: Test MRR with standard deviation over five independent runs. The best results within the event-based and snapshot-based categories are highlighted in bold. OOM denotes out of memory.

Type	Method	Venue	UCI	Enron	Contacts	Social Evo.	MOOC	tgbl-wiki
event	TGN	ICML20	0.091 ± 0.002	0.191 ± 0.027	0.153 ± 0.007	0.283 ± 0.009	0.174 ± 0.009	0.396 ± 0.060
	DyGFormer	NeurIPS23	0.334 ± 0.024	$\mathbf{0.331 \pm 0.010}$	$\mathbf{0.283 \pm 0.006}$	$\mathbf{0.366 \pm 0.004}$	OOM	$\mathbf{0.798 \pm 0.004}$
	NAT	LoG22	$\mathbf{0.356 \pm 0.048}$	0.276 ± 0.014	0.245 ± 0.015	0.258 ± 0.036	$\mathbf{0.283 \pm 0.058}$	0.749 ± 0.010
	GraphMixer	ICLR23	0.105 ± 0.008	0.296 ± 0.019	0.055 ± 0.003	0.157 ± 0.005	OOM	0.118 ± 0.002
snapshot	HTGN	KDD21	0.090 ± 0.006	0.263 ± 0.010	0.159 ± 0.003	0.226 ± 0.004	0.091 ± 0.004	0.563 ± 0.012
	GCLSTM	Appl. Intell22	0.077 ± 0.009	0.166 ± 0.005	0.129 ± 0.002	0.278 ± 0.006	0.121 ± 0.010	0.321 ± 0.010
	EGCNo	AAAI20	0.115 ± 0.008	0.218 ± 0.006	0.191 ± 0.004	0.248 ± 0.007	0.136 ± 0.008	0.413 ± 0.009
	GCN	ICLR17	0.068 ± 0.011	0.159 ± 0.012	0.099 ± 0.001	0.282 ± 0.003	0.072 ± 0.005	0.280 ± 0.008
	ROLAND	KDD22	0.100 ± 0.016	0.252 ± 0.011	0.143 ± 0.002	0.239 ± 0.007	0.119 ± 0.002	0.426 ± 0.016
	Hawkes-GAT	KDD25	0.097 ± 0.005	0.213 ± 0.007	$\mathbf{0.234 \pm 0.002}$	$\mathbf{0.367 \pm 0.009}$	0.074 ± 0.004	0.317 ± 0.007
	Ours	–	$\mathbf{0.150 \pm 0.012}$	$\mathbf{0.315 \pm 0.015}$	0.207 ± 0.001	0.263 ± 0.006	$\mathbf{0.179 \pm 0.012}$	$\mathbf{0.629 \pm 0.004}$

E.2 Sensitivity to Reservoir Size (R)

The number of reservoir slots, R, dictates the compression ratio of the joint neighborhood information. To understand its impact on predictive accuracy, we conduct a sensitivity analysis on the Enron dataset by varying R from 4 to 48.

As illustrated in Table 6, the performance exhibits a clear inverted-U trajectory, peaking at $R = 16$. This demonstrates a classic trade-off between structural fidelity and statistical stability. When R is too small (e.g., $R \leq 8$), the model is forced to compress highly diverse neighbor features into very few slots. This over-compression leads to severe hash collisions between topologically dissimilar nodes, increasing the structural approximation error and washing out critical fine-grained details.

Conversely, when R is excessively large (e.g., $R \geq 32$), the reservoir becomes overly sparse. Many slots may contain only a single neighbor or remain completely empty. This sparsity prevents the model from computing meaningful local feature centroids, effectively introducing noise and increasing the risk of overfitting. Therefore, $R = 16$ provides the optimal balance, offering sufficient capacity to resolve complex neighborhood substructures while maintaining robust, dense statistical aggregations within each slot. Notably, fixed-size weighted neighborhood aggregation, a simple and widely adopted strategy akin to the aforementioned uniform sampling, naturally emerges as a degenerate, highly compressed special case of our framework when $R = 1$.

E.3 Stability Across Random Seeds

To evaluate the robustness of the reported results to stochastic training variation, we repeat each experiment with five different random seeds and report the mean and standard deviation of test MRR. As shown in Table 7, NAGNN exhibits consistently small standard deviations across all six benchmarks, ranging from 0.001 to 0.015. These results indicate that the performance of NAGNN is stable across random initializations and that the reported predictive performance is not sensitive to a particular random seed.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction in this paper accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitation in Section 7.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: The proofs of the theorems in the paper are shown in Section C.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The implementation details are shown in Section 6, and all the results in this paper can be reproduced.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code will be made publicly available upon the acceptance of the paper.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The implementation details are shown in Section 6.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: All results are calculated by averaging results from 5 runs under different random seeds. But we do not calculate the error bars directly.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The computational resources are shown in Section 6.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in the paper conforms, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: Our work is intended to make an academic contribution rather than a social impact.

    Guidelines:

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The creators or original owners of assets used in the paper are properly credited, and the license and terms of use are explicitly mentioned and properly respected.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.