FLASH: Flexible Learning of Adaptive Sampling from History in Temporal Graph Neural Networks

Or Feldman[1], Krishna Sri Ipsit Mantri[2], Carola-Bibiane Schönlieb[3]
Chaim Baskin[1], Moshe Eliasof[1,3]
[1]Ben-Gurion University of the Negev, Israel
[2]Purdue University, USA
[3]University of Cambridge, UK
orfel@post.bgu.ac.il, kmantri@uni-bonn.de, cbs31@cam.ac.uk, chaimbaskin@bgu.ac.il
eliasof@bgu.ac.il

Abstract

Aggregating temporal signals from historic interactions is a key step in future link prediction on dynamic graphs. However, incorporating long histories is resource-intensive. Hence, temporal graph neural networks (TGNNs) often rely on historical neighbors sampling heuristics such as uniform sampling or recent neighbors selection. These heuristics are static and fail to adapt to the underlying graph structure. We introduce FLASH, a learnable and graph-adaptive neighborhood selection mechanism that generalizes existing heuristics. FLASH integrates seamlessly into TGNNs and is trained end-to-end using a self-supervised ranking loss. We provide theoretical evidence that commonly used heuristics hinder TGNNs performance, motivating our design. Extensive experiments across multiple benchmarks demonstrate consistent and significant performance improvements for TGNNs equipped with FLASH.

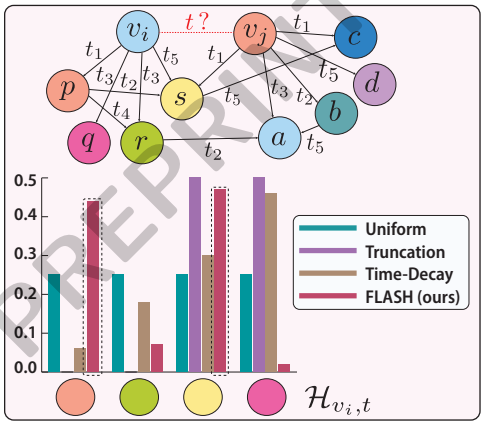

Figure 1: Illustration of different neighborhood selection strategies for predicting a link between v_i and v_j. Circles represent nodes and their colors indicate each node's feature. The bar chart on the right shows how each strategy scores the neighbors. Static heuristics (truncation or uniform sampling) either discard them or fail to prioritize important neighbors, while FLASH assigns higher scores to these key neighbors.

1 Introduction

Dynamic graphs provide a natural framework for modeling real-world systems where entities and their interactions evolve over time. They underpin a wide range of applications, including social and communication networks [Shetty and Adibi, 2004; Kumar *et al.*, 2019], user-item recommendation systems [Kumar *et al.*, 2019], and financial or knowledge-intensive platforms [Ni *et al.*, 2019; Shamsi *et al.*, 2022]. Predicting future interactions in these settings has emerged as a central learning task, leading to the development of Temporal Graph Neural Networks (TGNNs) – models specifically designed to learn from sequences of timestamped events represented by dynamic graphs.

TGNNs process dynamic graphs by encoding temporal interaction patterns into node representations, enabling them to predict future links. A common challenge in these models is how to efficiently aggregate information from a node's history of interactions, which can grow unbounded over time. Processing complete histories quickly becomes computationally prohibitive, especially in high-frequency interaction settings. To address this, existing models such as TGN [Rossi *et al.*, 2020], TGAT [Xu *et al.*, 2020], DyGFormer [Yu *et al.*, 2023], GraphMixer [Cong *et al.*, 2023], FreeDyG [Tian *et al.*, 2023], and others [Zhou *et al.*, 2022; Zou *et al.*, 2024; Gravina *et al.*, 2024; Xu *et al.*, 2024] adopt memory-efficient heuristics. These typically include strategies like uniform sampling, time-decay weighting, or truncating to the k most recent interactions. While effective at reducing computational overhead, these approaches are static and fail to adapt to the local graph structure or task-specific temporal signals. As shown in Figure 1, such heuristics apply uniform or truncated selection schemes that overlook potentially informative neighbors, whereas adaptive strategies can prioritize structurally meaningful interactions.

Static heuristics like uniform sampling [Rossi *et al.*, 2020], truncation [Cong *et al.*, 2023] or hybrid approaches [Luo and Li, 2024] are appealing due to their simplicity, but they treat all interactions as equally informative or rely solely on recency. This ignores the fact that some neighbors may be more relevant than others due to their position in the graph or their interaction patterns. Moreover, the optimal sampled

neighborhood may vary across time, nodes, and tasks, making fixed strategies fundamentally limited. These shortcomings are further amplified in heterogeneous or rapidly evolving graphs, where structural context can shift dramatically over time. This motivates the need for a learnable, structure-aware neighborhood selection mechanism that can adaptively prioritize informative past interactions.

To address these limitations, we propose FLASH – **F**lexible **L**earning of **A**daptive **S**election from **H**istory for **Temporal Graph Neural Networks**, a learnable and graph-adaptive neighborhood selection mechanism for TGNNs. FLASH replaces static heuristics with a data-driven approach that learns to prioritize historically informative neighbors based on their structural and temporal context. Crucially, because the true importance of neighbors is not known a priori, our method is trained using a self-supervised ranking objective that encourages selecting neighbors most predictive of future interactions. FLASH is lightweight, general-purpose, and integrates seamlessly into a wide range of existing TGNN architectures, including TGNNs with non-differentiable feature extractors [Yu *et al.*, 2023; Tian *et al.*, 2023]. This allows it to improve predictive performance without requiring architectural changes. Our key contributions are as follows:

- We propose FLASH, a novel graph-adaptive, learnable neighborhood selection mechanism that seamlessly integrates with any TGNN.
- We design a self-supervised training objective based on ranking loss, enabling our method to learn informative neighbor selection without access to ground-truth labels.
- We provide a theoretical analysis showing that FLASH is provably more expressive than the common heuristics of recent neighbors selection and uniform sampling.
- We conduct extensive experiments across multiple dynamic graph benchmarks, demonstrating consistent performance gains across diverse TGNN backbones compared to common neighbor sampling baselines.

2 Related Work

Neighborhood selection in static graphs Existing sampling techniques for large static graphs often rely on substructure sampling (e.g., nodes or edges), as employed by GraphSAGE [Hamilton *et al.*, 2017] and FastGCN [Chen *et al.*, 2018], or utilize random walks, as in PinSage [Ying *et al.*, 2018]. Other methods, such as GraphSAINT [Zeng *et al.*, 2020] and Cluster-GCN [Chiang *et al.*, 2019], are specifically designed to facilitate efficient training on large graphs. However, these approaches typically do not address inference or the temporal nature of dynamic graphs. In many TGNNs, uniform sampling can be viewed as dynamic extension of GraphSAGE-like methods, in which each neighbor in the historical neighborhood is sampled with equal probability. On the other hand, truncation sampling retains only the most recent neighbors, and its stochastic variant of time-weighting, reduces the sampling probability of older neighbors over time, effectively treating recency as a measure of importance. This parallels importance sampling in static graphs (e.g., FastGCN), where recent interactions are explicitly prioritized. However, our experiments show that recency alone is insufficient to capture the complexities of temporal interactions. Instead, incorporating node-specific contextual information at each interaction point, as done by FLASH, proves crucial for achieving robust and accurate performance in dynamic graph settings.

Learning from Large Historical Neighborhoods Learning from large historical neighborhoods in dynamic graphs poses significant challenges in computational cost and capturing long-term dependencies. To address the latter, the patching technique introduced in [Yu *et al.*, 2023], adopted in other recent studies [Tian *et al.*, 2023; Ding *et al.*, 2025]. The method splits the historical neighborhood into chronological patches, each linearly projected into a single representative vector. Since historical neighbors must be encoded (e.g., via co-occurrence encoding [Yu *et al.*, 2023]), neighborhood selection precedes it to reduce computational overhead. Thus, as proposed by [Yu *et al.*, 2023] for DyGFormer, patching is a complementary strategy for processing large historical neighborhoods, in future link prediction tasks.

Weighing and Selection Mechanisms. Some approaches, such as TGAT [Xu *et al.*, 2020], employ attention mechanisms to weight neighbors, but only after the neighborhood has been sampled. Other works use reinforcement learning [Wang *et al.*, 2022] to learn selection strategies, but they require task-specific reward design and extensive training. Methods based on hard negative mining or curriculum sampling [Younesian *et al.*, 2024] share our motivation of identifying informative samples, but are not tailored for self-supervised learning and are not readily applicable to TGNNs. In contrast, FLASH learns to sample informative neighbors in a self-supervised, fully differentiable manner.

3 Background

Continuous Time Dynamic Graph (CTDG) is represented as a sequence of time-stamped events $\mathcal{G} = \{(x_1, t_1), (x_2, t_2) \cdots\}$ where $t_1 \leq t_2 \leq \cdots$. Each event (x_i, t_i) represents an action on the graph that occurred at time t_i. Each such action can be either node addition, node removal, edge addition, or edge removal. We denote $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t)$ the snapshot of CTDG at time t, which is the graph received by applying all the events in $\mathcal{G}$ that occurred until time t, where $\mathcal{G}_0 = (\varnothing, \varnothing)$. We denote $F_{\mathcal{V}} : \mathcal{V} \times \mathbb{R}^+ \to \mathbb{R}^{d_{\mathcal{V}}}$ and $F_{\mathcal{E}} : \mathcal{E} \times \mathbb{R}^+ \to \mathbb{R}^{d_{\mathcal{E}}}$ as the functions that map a node or an edge to their features, at a specific point in time. For a given node v and timestamp t, we denote the historic neighborhood $\mathcal{H}_{v,t}$ as the set of all nodes that interacted with v before time t:

$$\mathcal{H}_{v,t} = \{u | (v, u) \in \mathcal{E}\} \tag{1}$$

Certain TGNNs allow the same node u to appear multiple times in $\mathcal{H}_{v,t}$, where each occurrence of u is associated with a distinct timestamp that records the time v and u interacted.

TGNNs construct node representations by sampling a subset $\mathcal{S}_{v,t}(k) \subseteq \mathcal{H}_{v,t}$ of k historical neighbors. Existing approaches employ various static selection strategies:

$$\mathcal{S}^{tru}_{v,t}(k) = \{u | r_u \leq k\} \tag{2}$$

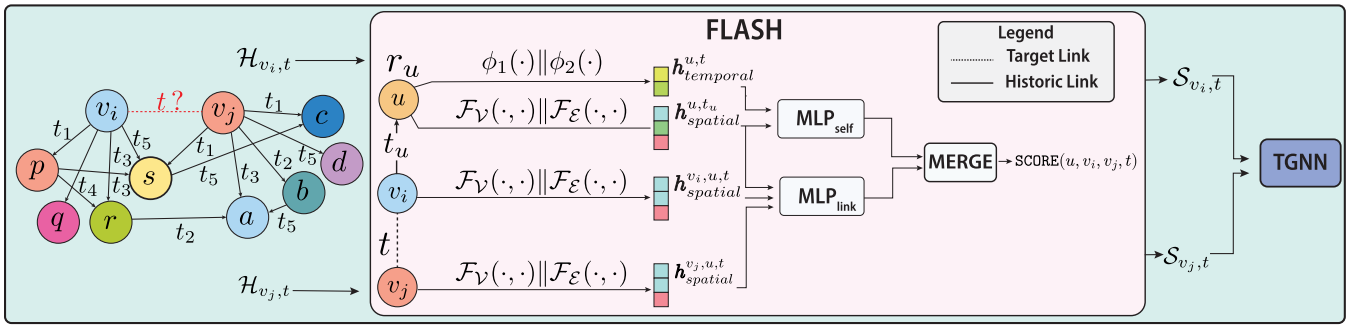

Figure 2: Overview of FLASH. Each historical neighbor u is assigned a score based on its temporal, spatial, and structural relationships with v_i and v_j. The highest-scoring neighbors are selected.

$$S^{\text{uni}}_{v,t}(k) \sim \text{Unif}\{\, S \subseteq \mathcal{H}_{v,t} \mid |S| = k \,\}. \tag{3}$$

where r_u is the rank of u with respect to the sequence of sorted neighbors from $\mathcal{H}_{v,t}$ in a decreasing order by their timestamp. Some TGNNs [Rossi *et al.*, 2020] allow sampling beyond the 1-hop historical neighborhood, either by allowing the sampling technique to sample farther nodes in advance or by applying the sampling technique recursively, i.e., sampling from the neighborhood of the sampled neighbors.

Given the sampled neighborhood $\mathcal{S}_{v,t}(k)$, the representation of node v at time t is computed as:

$$\boldsymbol{z}^t_v = \psi\left(\mathcal{S}_{v,t}(k)\right) \tag{4}$$

where $\psi(\cdot)$ is a parametric function of the TGNN that maps node to vector based on its sampled neighborhood.

For the task of future link prediction, given a pair of nodes (v_i, v_j) and their appropriate representations $\boldsymbol{z}^t_{v_i}$ and $\boldsymbol{z}^t_{v_j}$ at time t, a TGNN assigns a probability to the existence of a future edge between them using a learnable MERGE function:

$$p(v_i, v_j \mid t) = \texttt{MERGE}(\boldsymbol{z}^t_{v_i}, \boldsymbol{z}^t_{v_j}) \tag{5}$$

To train and evaluate TGNNs for future link prediction, the common approach is to split the entire sequence of interactions into two consecutive non-overlapping segments: a training prefix and an evaluation suffix. All interactions within the training prefix serve as positive examples, indicating node pairs that do form an edge at a specific time. For negative examples, random node pairs are sampled. The TGNN parameters are then updated by minimizing a binary classification loss (e.g., cross-entropy) that distinguishes positive from negative edges:

$$\mathcal{L}_{\text{task}} = -\sum_{(v_i,v_j,t)\in\text{train}} \Big[y^t_{ij} \log p(v_i, v_j|t) + (1 - y^t_{ij}) \log\big(1 - p(v_i, v_j|t)\big)\Big] \tag{6}$$

where y^t_{ij} is 1 for observed edges in the training prefix and 0 otherwise. Once trained, the TGNN can be used to predict edges in the evaluation suffix by computing the probability $p(v_i, v_j \mid t)$ for new future node pairs.

3.1 Theoretical analysis of Heuristic Neighborhood Samplings

The lack of flexibility in common neighborhood sampling techniques, such as k recent selection (truncation) or uniform sampling, hinder the performance of TGNNs. Specifically, we show that there exist dynamic graphs such that any TGNN relying on these common heuristics cannot learn from them (i.e., it cannot overfit on this graph).

Theorem 1. *For any k there exists a dynamic graph on which a TGNN that applies k recent neighbors sampling cannot learn.*

Theorem 2. *For any k, there exists a dynamic graph on which a TGNN that applies uniform sampling of k historical neighbors cannot learn.*

To prove Theorem 1, we find a graph that the $k + 1$ recent neighbor is required to learn the dynamic behavior of the graph. To prove Theorem 2, we find a dynamic graph that requires consistently selecting the same recent neighbor to learn its dynamic behavior. We provide the full proofs and assumptions for Theorem 1 and Theorem 2 in Appendix D.

Implication. These common sampling heuristics discard potentially crucial historical interactions, limiting the performance of TGNNs and reduce their expressive power. We aim to develop a learnable and adaptive sampling technique that not only generalizes these heuristics but also enables TGNNs to become more expressive.

4 Method

In the previous section, we showed that current neighborhood sampling strategies do not fully account for the graph structure and its features, preventing TGNNs from capturing even simple evolving dynamics. Another limitation of these sampling heuristics stems from the fact that different TGNNs learn temporal dynamics in distinct ways due to their diverse designs. For example, TGN [Rossi *et al.*, 2020] uses memory states, whereas DyGFormer [Yu *et al.*, 2023] employs a Transformer-based architecture [Vaswani *et al.*, 2017] and jointly processes $\mathcal{S}v_i, t$ and $\mathcal{S}v_j, t$. Thus, enforcing the same fixed neighborhood sampling scheme across various TGNNs may undermine their performance. A learnable sampling strategy can adapt not only to the structure of the dynamic graph but also to the way each TGNN architecture exploits temporal signals.

4.1 Desiderata for Adaptive Neighborhood Sampling in TGNNs

We propose an adaptive neighborhood sampling mechanism that satisfies the following desiderata:

(1) **Adapt to CTDG Dynamics:** We seek a mechanism $\texttt{SAMPLE}(\cdot)$ that accounts for node and edge attributes as well as their timestamps, i.e., the learnable parameters $\boldsymbol{\theta}$ should be informed by the CTDG's interaction patterns. Formally, for a node v with historical neighborhood $\mathcal{H}_{v,t}$ and a candidate interaction partner v',

$$\mathcal{S}_{v,t} = \texttt{SAMPLE}(v, v', \mathcal{H}_{v,t}; \boldsymbol{\theta}, k)\,, \tag{7}$$

where $|\mathcal{S}_{v,t}| = k$.

(2) **Generalize Existing Heuristics:** $\texttt{SAMPLE}(\cdot)$ should subsume commonly used heuristics (e.g., truncation, uniform sampling) as special cases via appropriate choices of $\boldsymbol{\theta}$.

(3) **Seamless Integration with TGNNs:** Since TGNN backbones encode temporal dynamics differently (e.g., memory modules, attention), the relevance of sampled neighbors may depend on the backbone. Accordingly, the TGNN should inform the sampling procedure and the learning of $\boldsymbol{\theta}$. Moreover, the mechanism should integrate seamlessly with a broad range of TGNNs, including models with non-differentiable feature extractors.

(4) **Self-Supervised Learning of Neighbor Importance:** Because ground-truth neighbor "importance" is unavailable, the mechanism should be learnable in a self-supervised manner that rewards neighbor subsets conducive to accurate link prediction.

4.2 FLASH

To predict a future link (v_i, v_j) at time t, we learn how informative each historical neighbor $u \in \mathcal{H}_{v_i,t}$ is for this prediction. Concretely, $\texttt{SAMPLE}(\cdot; \boldsymbol{\theta})$ assigns a score to each historical neighbor and selects the k neighbors with the highest score.

Learning the scoring function. We first construct an informative representation for each historical neighbor that captures both **structural** (interaction context) and **temporal** (time-dependent) information (**D1**). For a neighbor $u \in \mathcal{H}_{v_i,t}$, let t_u denote the time at which u and v_i interacted, and let r_u denote the rank of u when sorting $\mathcal{H}_{v_i,t}$ in decreasing order of interaction time. We define:

$$\boldsymbol{h}^{u}_{\text{spatial}} = \big[F_{\mathcal{V}}(u, t_u) \,\|\, F_{\mathcal{E}}(v_i, u, t_u) \,\|\, \mathcal{M}(u)\big] \tag{8}$$

$$\boldsymbol{h}^{v_i,u,t}_{\text{spatial}} = \big[F_{\mathcal{V}}(v_i, t_u) \,\|\, F_{\mathcal{V}}(v_i, t) \,\|\, \mathcal{M}(v_i)\big] \tag{9}$$

$$\boldsymbol{h}^{v_j,u,t}_{\text{spatial}} = \big[F_{\mathcal{V}}(v_j, t_u) \,\|\, F_{\mathcal{V}}(v_j, t) \,\|\, \mathcal{M}(v_j)\big] \tag{10}$$

$$\boldsymbol{h}^{u,t}_{\text{temporal}} = [\phi_1(t - t_u) \,\|\, \phi_2(r_u)], \tag{11}$$

where $\phi_1(\cdot)$ and $\phi_2(\cdot)$ are Time2Vec representations kazemi2019time2vec, $\mathcal{F}_{\mathcal{V}}$ maps nodes and timestamps to their corresponding features, $F_{\mathcal{E}}$ maps edges and timestamps to their corresponding features (as defined in Section 3), $\mathcal{M}(\cdot)$ denotes learnable node features, and $\|$ denotes concatenation. Following the standard TGNN pipeline and prior work deng2024taser, we project the neighbor and link representations to a common dimensionality using MLPs, and then combine them via a $\texttt{MERGE}$ operator:

$$\begin{aligned}\texttt{SCORE}_{u,v_i,v_j,t} = \texttt{MERGE}\Big(&\texttt{MLP}_{\texttt{self}}(\boldsymbol{h}^{u}_{spatial}, \boldsymbol{h}^{u,t}_{temporal}),\\ &\texttt{MLP}_{\texttt{link}}(\boldsymbol{h}^{u}_{spatial},\\ &\quad \boldsymbol{h}^{v_i,u,t}_{spatial}, \boldsymbol{h}^{v_j,u,t}_{spatial})\Big)\end{aligned} \tag{12}$$

where $\texttt{MERGE}$ is an MLP with a single hidden layer, as suggested for combining source–destination information yu2023towards,tian2023freedyg. $\texttt{SCORE}_{u,v_i,v_j,t}$ measures the utility of neighbor u for predicting the link (v_i, v_j) at time t, where larger values indicate higher importance. FLASH then selects the k neighbors with the highest scores. Figure 2 provides a detailed schematic of FLASH.

Training FLASH. For each node pair (v_i, v_j) at time t, let $y^t_{ij} \in \{0, 1\}$ indicate whether (v_i, v_j) is a *positive* (observed) edge ($y^t_{ij} = 1$) or a *negative* (unobserved) edge ($y^t_{ij} = 0$). Let a TGNN compute edge probabilities using the subsets selected by FLASH, $(\mathcal{S}_{v_i,v_j,t}, \mathcal{S}_{v_j,v_i,t})$, and using uniformly sampled subsets $(\mathcal{S}^{\text{uni}}_{v_i,t}, \mathcal{S}^{\text{uni}}_{v_j,t})$. We define the probability difference:

$$\begin{aligned}\Delta^t_{ij} = &\,p\big(v_i, v_j | t; \mathcal{S}_{v_i,v_j,t}, \mathcal{S}_{v_j,v_i,t}\big)\\ &- p\Big(v_i, v_j | t; \mathcal{S}^{\text{uni}}_{v_i,t}, \mathcal{S}^{\text{uni}}_{v_j,t}\Big)\,.\end{aligned} \tag{13}$$

Let $\overline{s_{v_i}}, \overline{s_{v_j}}$ denote the average FLASH scores over nodes in $\mathcal{S}_{v_i,v_j,t}$ and $\mathcal{S}_{v_j,v_i,t}$, respectively, and let $\overline{s^{\text{uni}}_{v_i}}, \overline{s^{\text{uni}}_{v_j}}$ denote the corresponding averages over $\mathcal{S}^{\text{uni}}_{v_i,t}$ and $\mathcal{S}^{\text{uni}}_{v_j,t}$. For positive edges, we prefer $\Delta^t_{ij} > 0$ (our subsets increase the predicted probability), while for negative edges we prefer $\Delta^t_{ij} < 0$ (our subsets decrease it). We enforce this preference using a pairwise RankNet burges2005learning logistic loss:

$$\mathcal{L}^t_{ij} = \begin{cases} -\Big[\log\sigma\big(\overline{s_{v_j}} - \overline{s^{\text{uni}}_{v_j}}\big) \\ \quad + \log\sigma\big(\overline{s_{v_i}} - \overline{s^{\text{uni}}_{v_i}}\big)\Big] & \text{if } (y^t_{ij} - \frac{1}{2})\Delta^t_{ij} > 0, \\ -\Big[\log\sigma\big(\overline{s^{\text{uni}}_{v_j}} - \overline{s_{v_j}}\big) \\ \quad + \log\sigma\big(\overline{s^{\text{uni}}_{v_i}} - \overline{s_{v_i}}\big)\Big] & \text{otherwise.} \end{cases} \tag{14}$$

Minimizing $\mathcal{L}^t_{ij}$ encourages the score averages to move in the direction consistent with y^t_{ij} without requiring any ground-truth importance supervision (**D3**). When the condition in Equation (14) holds, the neighborhoods selected by FLASH improve over uniform sampling, and the loss encourages higher scores for selected neighbors over the uniformly sampled neighbors. Otherwise, uniform sampling performs better and the loss correspondingly increases the scores of uniformly sampled neighbors relative to those selected by

Method ↓ / Dataset →	WIKIPEDIA AP ↑	REDDIT AP ↑	MOOC AP ↑	LASTFM AP ↑	SOCIAL EVO. AP ↑	ENRON AP ↑	UCI AP ↑
TGAT + Trunc.	$94.05_{\pm 0.06}$	$93.63_{\pm 0.16}$	$79.69_{\pm 0.24}$	$65.64_{\pm 0.34}$	$85.36_{\pm 0.25}$	$72.91_{\pm 0.58}$	$79.74_{\pm 0.37}$
TGAT + Uni.	$62.60_{\pm 0.36}$	$87.94_{\pm 0.03}$	$60.03_{\pm 0.11}$	$50.66_{\pm 0.11}$	$53.22_{\pm 0.79}$	$51.34_{\pm 0.32}$	$60.42_{\pm 0.22}$
TGAT + NLB	$91.49_{\pm 0.25}$	$92.78_{\pm 0.11}$	$77.37_{\pm 0.17}$	$65.64_{\pm 0.34}$	$83.19_{\pm 0.07}$	$70.69_{\pm 0.93}$	$77.80_{\pm 1.25}$
TGAT + TASER	$67.78_{\pm 0.99}$	$89.14_{\pm 0.59}$	$67.10_{\pm 0.56}$	$52.28_{\pm 0.52}$	$70.40_{\pm 0.45}$	$60.55_{\pm 0.91}$	$70.62_{\pm 0.33}$
TGAT + FLASH	$\mathbf{94.67}_{\pm 0.38}$	$\mathbf{95.92}_{\pm 0.13}$	$\mathbf{80.24}_{\pm 0.54}$	$\mathbf{74.94}_{\pm 1.49}$	$\mathbf{92.17}_{\pm 0.25}$	$\mathbf{79.04}_{\pm 0.94}$	$\mathbf{87.84}_{\pm 0.12}$
TGN + Trunc.	$98.55_{\pm 0.05}$	$98.61_{\pm 0.03}$	$90.13_{\pm 0.64}$	$82.62_{\pm 2.09}$	$91.63_{\pm 0.51}$	$86.51_{\pm 2.29}$	$93.34_{\pm 0.25}$
TGN + Uni.	$98.49_{\pm 0.08}$	$98.59_{\pm 0.01}$	$83.08_{\pm 1.11}$	$67.60_{\pm 5.66}$	$65.52_{\pm 6.06}$	$85.47_{\pm 2.00}$	$93.34_{\pm 0.25}$
TGN + NLB	$98.31_{\pm 0.09}$	$98.61_{\pm 0.04}$	$89.81_{\pm 0.60}$	$80.45_{\pm 1.94}$	$91.19_{\pm 0.38}$	$86.51_{\pm 2.29}$	$92.91_{\pm 0.43}$
TGN + TASER	$98.15_{\pm 0.07}$	$98.63_{\pm 0.04}$	$90.57_{\pm 1.02}$	$75.58_{\pm 4.07}$	$92.24_{\pm 0.62}$	$86.87_{\pm 0.081}$	$89.86_{\pm 2.61}$
TGN + FLASH	$\mathbf{98.73}_{\pm 0.06}$	$\mathbf{99.06}_{\pm 0.03}$	$\mathbf{91.19}_{\pm 0.51}$	$\mathbf{90.54}_{\pm 0.62}$	$\mathbf{93.43}_{\pm 0.11}$	$\mathbf{89.90}_{\pm 0.76}$	$\mathbf{95.17}_{\pm 0.16}$
GRAPHMIXER + Trunc.	$96.23_{\pm 0.24}$	$95.17_{\pm 0.03}$	$80.71_{\pm 0.11}$	$72.98_{\pm 0.07}$	$87.09_{\pm 0.12}$	$81.63_{\pm 0.47}$	$93.14_{\pm 0.44}$
GRAPHMIXER + Uni.	$77.06_{\pm 0.16}$	$89.80_{\pm 0.05}$	$64.75_{\pm 0.39}$	$63.96_{\pm 0.09}$	$55.69_{\pm 0.12}$	$55.65_{\pm 3.04}$	$71.27_{\pm 2.79}$
GRAPHMIXER + NLB	$95.09_{\pm 0.12}$	$95.17_{\pm 0.03}$	$78.61_{\pm 0.09}$	$72.98_{\pm 0.07}$	$85.66_{\pm 0.09}$	$81.01_{\pm 0.30}$	$92.51_{\pm 0.60}$
GRAPHMIXER + TASER	$93.27_{\pm 1.43}$	$94.78_{\pm 0.17}$	$78.97_{\pm 0.66}$	$72.24_{\pm 0.93}$	$87.57_{\pm 2.53}$	$77.90_{\pm 0.48}$	$88.59_{\pm 1.10}$
GRAPHMIXER + FLASH	$\mathbf{97.51}_{\pm 0.25}$	$\mathbf{96.62}_{\pm 0.11}$	$\mathbf{80.85}_{\pm 0.52}$	$\mathbf{82.68}_{\pm 0.74}$	$\mathbf{92.84}_{\pm 0.11}$	$\mathbf{85.74}_{\pm 0.70}$	$\mathbf{93.21}_{\pm 0.61}$
DYGFORMER + Trunc.	$96.91_{\pm 0.05}$	$95.15_{\pm 0.07}$	$82.47_{\pm 0.07}$	$74.81_{\pm 0.09}$	$85.73_{\pm 0.06}$	$82.35_{\pm 0.66}$	$89.61_{\pm 0.22}$
DYGFORMER + Uni.	$96.87_{\pm 0.07}$	$95.15_{\pm 0.07}$	$82.47_{\pm 0.07}$	$74.81_{\pm 0.09}$	$85.71_{\pm 0.06}$	$82.35_{\pm 0.66}$	$89.61_{\pm 0.22}$
DYGFORMER + NLB	$96.74_{\pm 0.07}$	$95.03_{\pm 0.09}$	$81.09_{\pm 0.05}$	$74.52_{\pm 0.18}$	$85.40_{\pm 0.10}$	$81.86_{\pm 0.45}$	$89.12_{\pm 0.19}$
DYGFORMER + TASER	$95.47_{\pm 0.32}$	$93.83_{\pm 0.33}$	$81.56_{\pm 0.67}$	$72.72_{\pm 1.73}$	$90.09_{\pm 0.11}$	$81.74_{\pm 0.64}$	$88.15_{\pm 0.59}$
DYGFORMER + FLASH	$\mathbf{98.17}_{\pm 0.04}$	$\mathbf{98.11}_{\pm 0.02}$	$\mathbf{82.96}_{\pm 0.42}$	$\mathbf{86.09}_{\pm 0.13}$	$\mathbf{92.41}_{\pm 0.11}$	$\mathbf{88.70}_{\pm 0.17}$	$\mathbf{92.96}_{\pm 0.16}$
FREEDYG + Trunc.	$98.35_{\pm 0.01}$	$97.53_{\pm 0.02}$	$85.02_{\pm 0.02}$	$80.19_{\pm 0.04}$	$90.88_{\pm 0.05}$	$88.77_{\pm 0.20}$	$95.21_{\pm 0.33}$
FREEDYG + Uni.	$98.33_{\pm 0.02}$	$95.98_{\pm 0.03}$	$67.52_{\pm 0.18}$	$67.72_{\pm 0.10}$	$89.16_{\pm 0.03}$	$88.46_{\pm 0.08}$	$95.21_{\pm 0.33}$
FREEDYG + NLB	$98.33_{\pm 0.01}$	$97.59_{\pm 0.02}$	$83.56_{\pm 0.12}$	$80.02_{\pm 0.09}$	$90.40_{\pm 0.05}$	$88.77_{\pm 0.20}$	$95.17_{\pm 0.26}$
FREEDYG + TASER	$97.94_{\pm 0.35}$	$97.11_{\pm 0.30}$	$76.97_{\pm 3.87}$	$78.54_{\pm 1.90}$	$90.84_{\pm 1.09}$	$87.24_{\pm 1.47}$	$93.15_{\pm 1.46}$
FREEDYG + FLASH	$\mathbf{98.94}_{\pm 0.04}$	$\mathbf{98.72}_{\pm 0.02}$	$\mathbf{85.69}_{\pm 0.27}$	$\mathbf{88.46}_{\pm 0.07}$	$\mathbf{93.62}_{\pm 0.05}$	$\mathbf{91.31}_{\pm 0.01}$	$\mathbf{95.86}_{\pm 0.16}$

Table 1: Comparison of various neighborhood sampling methods on transductive future edge prediction using 2 historical neighbors on different datasets from DyGLib. The best performing method is in bold.

FLASH. We train FLASH jointly with a given TGNN by optimizing the sum of Equation (14) and Equation (6). Gradients from Equation (6) update only the TGNN parameters, while for Equation (14) we stop gradients through Δ_{ij}^t so that updates flow only to FLASH parameters (**D4**). To ensure constant runtime, FLASH first applies a recent-neighbors finder deng2024taser to restrict the candidate set to a fixed size. It then computes scores only within this reduced neighborhood rather than over the full historical neighborhood. In Appendix G, we provide a throughout analysis of FLASH and TASER.

4.3 Theoretical Analysis of FLASH

The components of FLASH adapt to the evolution of the dynamic graph over time and to the interactions being predicted, by incorporating both spatiotemporal features and potentially interacting nodes when constructing each sampled neighborhood. We show that FLASH can replicate both k-recent neighbor selection and uniform sampling. Furthermore, we show that the graphs used in the proofs of Theorem 1 and Theorem 2 can be learned by a TGNN that utilizes FLASH. Therefore:

Theorem 3. FLASH *is more expressive than k-recent neighbor selection and uniform sampling.*

We provide the full proof of Theorem 3 in Appendix D.

5 Experiments

We evaluate FLASH across multiple dynamic graph benchmarks and compare it with recent state-of-the-art baselines. Section 5.1 presents our key empirical findings, and Section 5.2 provides additional ablation studies. Additional results are reported in Appendix C. Our experiments aim to answer the following research questions:

- **(RQ1)** How does our proposed strategy compare to established sampling baselines in predictive accuracy?
- **(RQ2)** Does our method generalize across varying graph sizes, sparsity levels, and temporal granularities?
- **(RQ3)** What is the computational overhead of our approach relative to existing methods?

Experimental Setup. We test our method on five common TGNNs: TGAT [Xu *et al.*, 2020], TGN [Rossi *et al.*, 2020], GRAPHMIXER [Cong *et al.*, 2023], DYGFORMER [Yu *et al.*, 2023], and FREEDYG [Tian *et al.*, 2023]. Detailed descriptions of each model are provided in Appendix B. Each model is trained with four neighbor sampling strategies: Truncation [Cong *et al.*, 2023; Yu *et al.*, 2023], Uniform [Rossi *et al.*, 2020], No-Look-Back (NLB) [Luo and Li, 2024], TASER with the recent-neighbors finder and a linear predictor [Deng *et al.*, 2024], and our proposed method. Following standard practice, we use a chronological 70%-15%-15% train-validation-test split. Additional parameters regarding the training scheme and implementation-specific details are provided in Appendix E.

Importance of Small Neighborhoods As discussed in Section 1, processing large neighborhoods is computationally prohibitive in both time and space. Figure 4 reports the throughput of state-of-the-art TGNNs for varying neighborhood sizes. The results show that increasing the neighborhood processed in parallel substantially degrades model

Method ↓ / Dataset →	WIKIPEDIA AP ↑	REDDIT AP ↑	MOOC AP ↑	LASTFM AP ↑	SOCIAL EVO. AP ↑	ENRON AP ↑	UCI AP ↑
TGAT + Trunc.	$94.26_{\pm 0.09}$	$90.82_{\pm 0.18}$	$78.12_{\pm 0.18}$	$72.35_{\pm 0.42}$	$82.73_{\pm 0.29}$	$70.25_{\pm 1.30}$	$80.95_{\pm 0.29}$
TGAT + Uni.	$62.77_{\pm 0.43}$	$82.85_{\pm 0.13}$	$58.38_{\pm 0.14}$	$50.75_{\pm 0.07}$	$53.06_{\pm 0.40}$	$51.63_{\pm 0.72}$	$59.65_{\pm 0.36}$
TGAT + NLB.	$91.46_{\pm 0.24}$	$89.70_{\pm 0.15}$	$75.42_{\pm 0.22}$	$72.35_{\pm 0.42}$	$80.66_{\pm 0.12}$	$68.77_{\pm 1.03}$	$78.47_{\pm 0.68}$
TGAT + TASER	$67.62_{\pm 1.08}$	$84.51_{\pm 0.75}$	$63.60_{\pm 0.52}$	$53.53_{\pm 0.85}$	$67.22_{\pm 0.59}$	$58.46_{\pm 1.38}$	$64.94_{\pm 0.90}$
TGAT+FLASH	$\mathbf{94.70}_{\pm 0.35}$	$\mathbf{94.24}_{\pm 0.10}$	$\mathbf{78.85}_{\pm 0.61}$	$\mathbf{79.56}_{\pm 1.57}$	$\mathbf{90.23}_{\pm 0.25}$	$\mathbf{76.70}_{\pm 1.13}$	$\mathbf{87.77}_{\pm 0.17}$
TGN + Trunc.	$97.84_{\pm 0.05}$	$97.28_{\pm 0.08}$	$90.02_{\pm 1.30}$	$87.23_{\pm 1.18}$	$89.08_{\pm 1.42}$	$79.66_{\pm 2.23}$	$89.36_{\pm 0.52}$
TGN + Uni.	$97.82_{\pm 0.12}$	$97.13_{\pm 0.15}$	$82.47_{\pm 0.41}$	$76.10_{\pm 6.58}$	$57.15_{\pm 0.35}$	$77.91_{\pm 2.34}$	$89.36_{\pm 0.52}$
TGN + NLB.	$97.61_{\pm 0.08}$	$97.23_{\pm 0.09}$	$89.19_{\pm 0.48}$	$85.79_{\pm 2.00}$	$86.51_{\pm 2.51}$	$79.66_{\pm 2.23}$	$88.34_{\pm 0.52}$
TGN + TASER	$97.39_{\pm 0.17}$	$97.33_{\pm 0.12}$	$89.90_{\pm 1.18}$	$81.97_{\pm 3.73}$	$90.31_{\pm 1.50}$	$78.99_{\pm 2.56}$	$85.28_{\pm 1.22}$
TGN+FLASH	$\mathbf{98.08}_{\pm 0.12}$	$\mathbf{98.24}_{\pm 0.07}$	$\mathbf{90.50}_{\pm 0.61}$	$\mathbf{92.04}_{\pm 0.40}$	$\mathbf{92.06}_{\pm 0.48}$	$\mathbf{82.95}_{\pm 1.02}$	$\mathbf{92.40}_{\pm 0.27}$
GraphMixer + Trunc.	$95.80_{\pm 0.24}$	$92.21_{\pm 0.07}$	$79.38_{\pm 0.18}$	$79.52_{\pm 0.11}$	$84.83_{\pm 0.14}$	$76.17_{\pm 0.43}$	$91.60_{\pm 0.19}$
GraphMixer + Uni.	$78.72_{\pm 0.12}$	$84.29_{\pm 0.10}$	$65.91_{\pm 0.30}$	$73.74_{\pm 0.07}$	$51.13_{\pm 0.15}$	$52.46_{\pm 3.33}$	$71.69_{\pm 1.43}$
GraphMixer + NLB	$94.34_{\pm 0.10}$	$92.21_{\pm 0.07}$	$76.89_{\pm 0.10}$	$79.52_{\pm 0.11}$	$83.09_{\pm 0.08}$	$74.99_{\pm 0.34}$	$90.89_{\pm 0.23}$
GraphMixer +TASER	$92.42_{\pm 1.54}$	$91.25_{\pm 0.30}$	$76.99_{\pm 0.59}$	$79.43_{\pm 1.02}$	$85.10_{\pm 2.73}$	$70.70_{\pm 0.44}$	$86.78_{\pm 1.76}$
GraphMixer + FLASH	$\mathbf{97.06}_{\pm 0.29}$	$\mathbf{94.77}_{\pm 0.21}$	$\mathbf{79.44}_{\pm 0.55}$	$\mathbf{86.70}_{\pm 0.74}$	$\mathbf{90.74}_{\pm 0.11}$	$\mathbf{80.61}_{\pm 1.41}$	$\mathbf{91.66}_{\pm 0.24}$
DyGFormer + Trunc.	$97.02_{\pm 0.07}$	$93.40_{\pm 0.08}$	$81.27_{\pm 0.09}$	$79.97_{\pm 0.11}$	$82.86_{\pm 0.07}$	$80.46_{\pm 0.91}$	$89.99_{\pm 0.18}$
DyGFormer + Uni.	$97.01_{\pm 0.06}$	$93.40_{\pm 0.08}$	$81.27_{\pm 0.09}$	$79.86_{\pm 0.12}$	$82.80_{\pm 0.07}$	$80.46_{\pm 0.91}$	$89.81_{\pm 0.16}$
DyGFormer + NLB	$96.86_{\pm 0.08}$	$93.27_{\pm 0.09}$	$79.62_{\pm 0.11}$	$79.66_{\pm 0.23}$	$82.68_{\pm 0.10}$	$80.09_{\pm 0.38}$	$89.52_{\pm 0.22}$
DyGFormer + TASER	$95.70_{\pm 0.26}$	$91.05_{\pm 0.51}$	$80.37_{\pm 0.65}$	$77.93_{\pm 1.60}$	$88.20_{\pm 0.25}$	$79.28_{\pm 1.20}$	$84.98_{\pm 0.58}$
DyGFormer + FLASH	$\mathbf{97.91}_{\pm 0.04}$	$\mathbf{97.36}_{\pm 0.04}$	$\mathbf{81.42}_{\pm 0.51}$	$\mathbf{88.55}_{\pm 0.15}$	$\mathbf{90.56}_{\pm 0.28}$	$\mathbf{84.88}_{\pm 0.66}$	$\mathbf{92.73}_{\pm 0.15}$
FreeDyG + Trunc.	$98.03_{\pm 0.03}$	$96.34_{\pm 0.04}$	$84.04_{\pm 0.06}$	$85.09_{\pm 0.07}$	$89.06_{\pm 0.09}$	$84.59_{\pm 0.33}$	$93.98_{\pm 0.15}$
FreeDyG + Uni.	$98.01_{\pm 0.02}$	$93.86_{\pm 0.04}$	$68.53_{\pm 0.24}$	$76.41_{\pm 0.09}$	$86.92_{\pm 0.09}$	$83.96_{\pm 0.21}$	$93.98_{\pm 0.15}$
FreeDyG + NLB.	$97.99_{\pm 0.01}$	$96.34_{\pm 0.04}$	$82.13_{\pm 0.10}$	$84.98_{\pm 0.13}$	$88.38_{\pm 0.17}$	$84.59_{\pm 0.33}$	$93.81_{\pm 0.16}$
FreeDyG +TASER	$97.67_{\pm 0.34}$	$95.63_{\pm 0.38}$	$74.93_{\pm 4.60}$	$84.07_{\pm 1.25}$	$88.87_{\pm 1.17}$	$81.31_{\pm 2.04}$	$92.02_{\pm 1.32}$
FreeDyG + FLASH	$\mathbf{98.59}_{\pm 0.03}$	$\mathbf{98.21}_{\pm 0.04}$	$\mathbf{84.38}_{\pm 0.25}$	$\mathbf{90.73}_{\pm 0.09}$	$\mathbf{91.53}_{\pm 0.63}$	$\mathbf{87.13}_{\pm 0.62}$	$\mathbf{94.60}_{\pm 0.13}$

Table 2: Comparison of our suggested sampling strategy with other sampling strategies (Truncation, Uniform and NLB) in the inductive setting. Results are reported in AP for future edge prediction with random negative sampling over five different seeds. The significantly best result for each benchmark appears in bold font. 2 historical neighbors are used by each method.

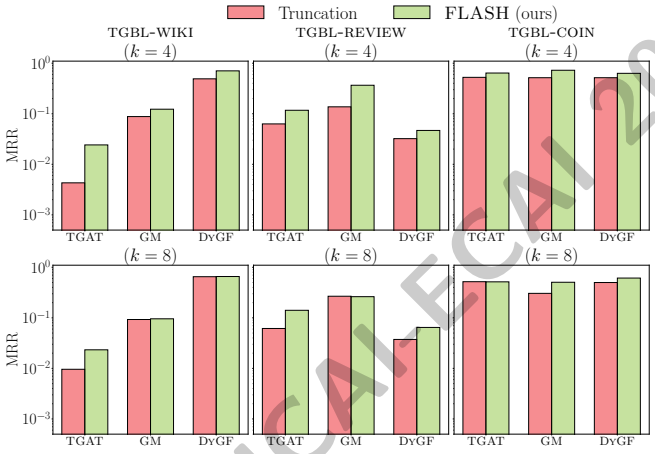

Figure 3: FLASH vs. Truncation on the TGB benchmark. Results are reported in MRR (dynamic link prediction) with random negative sampling over three runs, using $k = 4$ and $k = 8$ historical neighbors. Here GM refers to GRAPHMIXER and DYGF refers to DYGFORMER.

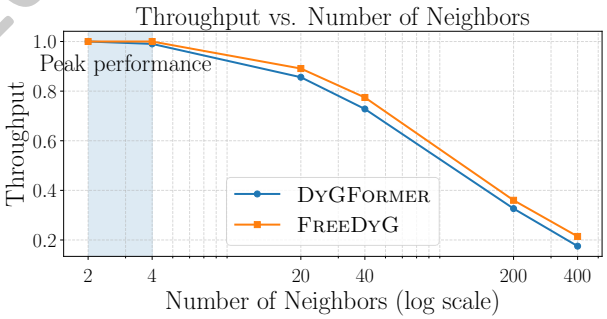

Figure 4: Throughput comparison of TGNNs as a function of selected neighborhood size, reported as the number of edges processed per second. Values are normalized by the throughput at neighborhood size 2 (e.g., if DYGFORMER processes 200 edges/s with neighborhood size 2 and 100 edges/s with neighborhood size 40, the reported value is 0.5).

throughput. Consequently, TGNNs should employ small neighborhoods (2–4) to achieve peak runtime performance. Accordingly, we evaluate neighborhood selection strategies under a small neighborhood budget (2–4) to determine which strategy is optimal for future link prediction while sustaining peak throughput in TGNNs.

5.1 Results

Evaluation with DyGLib. We use the WIKIPEDIA, REDDIT, MOOC, LASTFM, ENRON, SOCIAL EVO., and UCI datasets from the DyGLib benchmark [Yu *et al.*, 2023]. DyGLib includes datasets spanning a wide range of graph sizes, with some containing over a million edges (**RQ2**). Full dataset statistics are provided in Appendix A. We evaluate future edge prediction (Equation (6)) under two settings: (1) *transductive*, where all nodes appear during training, and (2)

ϕ_2	ϕ_1	$\mathcal{M}$	$\texttt{MLP}_{link}$	WIKIPEDIA	ENRON	UCI
✗	✓	✓	✓	$98.73_{\pm 0.07}$	$89.63_{\pm 0.72}$	$95.08_{\pm 0.19}$
✗	✗	✓	✓	$98.20_{\pm 0.06}$	$89.50_{\pm 0.74}$	$95.08_{\pm 0.33}$
✓	✓	✗	✓	$98.58_{\pm 0.06}$	$86.45_{\pm 1.25}$	$93.27_{\pm 0.61}$
✓	✓	✓	✗	$98.54_{\pm 0.09}$	$87.38_{\pm 0.81}$	$93.06_{\pm 0.60}$
✓	✓	✓	✓	$\mathbf{98.73}_{\pm 0.06}$	$\mathbf{89.90}_{\pm 0.76}$	$\mathbf{95.17}_{\pm 0.16}$

Table 3: Ablation of design choices of FLASH. (✓) denotes inclusion and (✗) denotes exclusion of the component. (*transductive*)

inductive, where test nodes are unseen during training. Tables 1 and 2 report results across models and methods. Overall, FLASH consistently outperforms previously suggested neighborhood selection strategies, both in the transductive setting and the inductive setting (**RQ1**). For example, on SOCIAL EVO., FLASH achieves a $\sim$**8%** relative improvement over truncation for TGAT, and on LASTFM, a $\sim$**15%** relative improvement for FREEDYG, demonstrating strong effectiveness even for computationally expensive models.

Evaluation with TGB. We evaluate FLASH on the TGB benchmark [Huang *et al.*, 2023] using TGBL-WIKI, TGBL-REVIEW, and TGBL-COIN. In contrast to DyGLib, TGB samples multiple random negative edges for each positive edge and is evaluated using Mean Reciprocal Rank (MRR). From Figure 3, FLASH yields performance that is consistently on par with or better than baselines across TGNNs and neighborhood sizes. Notably, on TGBL-REVIEW, our method improves performance by $\sim 2.67\times$ for GRAPHMIXER when $k = 4$, highlighting robustness across diverse dynamic graph scenarios (**RQ2**).

5.2 Ablation Study

FLASH Design Choices. We ablate key components of FLASH, namely the historical position embedding ϕ_2, temporal embedding ϕ_1, learnable node embedding $\mathcal{M}$, and the link-awareness module $\texttt{MLP}_{link}$ in Equation (12), to assess the contribution of temporal, structural, and interaction context for future edge prediction (**RQ2**). Table 3 shows that the positional embedding ϕ_2 is particularly important on ENRON and UCI, indicating that relying solely on temporal information is suboptimal. The interaction context module, $\texttt{MLP}_{link}$, is consistently beneficial across all datasets, providing reliable performance improvements.

Computational Overhead. Table 4 reports a throughput comparison for FLASH, measured as the number of processed edges per second, against the baselines **(RQ3)**. For a fair comparison, both TASER and FLASH employ the same neighbor finder, which constrains their neighborhoods to an identical size, this benefits both methods relative to uniform sampling. Because all experiments use a unified batch size, model latency is inversely proportional to throughput. Appendix F further provides a time-complexity analysis of FLASH and the baselines.

Model	Uniform	NLB	TASER	FLASH
TGAT	51%	44%	59%	75%
TGN	67%	80%	82%	82%
GRAPHMIXER	42%	40%	59%	64%
DYGFORMER	97%	87%	86%	94%
FREEDYG	84%	85%	91%	92%

Table 4: Throughput comparison of neighbor sampling strategies. Values denote the number of edges processed per second by the TGNN relative to the Truncation baseline. For example, if TGAT with Uniform processes 300 edges/s over a full run while TGAT with Truncation processes 600 edges/s, the reported throughput for TGAT combined with Uniform is 50%.

6 Conclusion

In this work, we introduce FLASH, a flexible, end-to-end learnable neighborhood selection framework for dynamic graph learning. FLASH factorizes the neighbor scoring function into spatial and temporal components, ensuring that neighbor importance reflects both topological structure and temporal context. Our training objective couples the learned scores with downstream predictive performance, enabling effective optimization of the selection policy even when ground-truth importance scores are unavailable and TGNN operations are non-differentiable. This design allows FLASH to integrate seamlessly with a broad range of TGNN architectures, providing an efficient mechanism for selecting the most relevant historical neighbors. We further evaluate FLASH across multiple dynamic graph benchmarks and show that it matches or outperforms the considered baselines.

Contribution Statement

O.F. contributed to the design of the method, implementation, experiments, figure/plot design, and manuscript writing. K.S.I.M. contributed to the design of the method, figure/plot design, and manuscript writing. C.B.S. contributed to manuscript writing. C.B. and M.E. contributed to manuscript writing and equally supervised the work.

References

[Chen *et al.*, 2018] Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: Fast learning with graph convolu-tional networks via importance sampling. In *International Conference on Learning Representations*. International Conference on Learning Representations, ICLR, 2018.

[Chiang *et al.*, 2019] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 257–266, 2019.

[Cong *et al.*, 2023] 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? *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5*, 2023.

[Deng *et al.*, 2024] Gangda Deng, Hongkuan Zhou, Hanqing Zeng, Yinglong Xia, Christopher Leung, Jianbo Li, Rajgopal Kannan, and Viktor Prasanna. Taser: Temporal adaptive sampling for fast and accurate dynamic graph representation learning. In *2024 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 926–937. IEEE, 2024.

[Ding *et al.*, 2025] Zifeng Ding, Yifeng Li, Yuan He, Antonio Norelli, Jingcheng Wu, Volker Tresp, Michael M Bronstein, and Yunpu Ma. Dygmamba: Efficiently modeling long-term temporal dependency on continuous-time dynamic graphs with state space models. *Transactions on Machine Learning Research*, 2025.

[Gravina *et al.*, 2024] Alessio Gravina, Giulio Lovisotto, Claudio Gallicchio, Davide Bacciu, and Claas Grohnfeldt. Long range propagation on continuous-time dynamic graphs. In *International Conference on Machine Learning*, pages 16206–16225. PMLR, 2024.

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

[Huang *et al.*, 2023] 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, 2023.

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

[Luo and Li, 2024] Yuhong Luo and Pan Li. Scalable and efficient temporal graph representation learning via forward recent sampling. In *The Third Learning on Graphs Conference*, 2024.

[Ni *et al.*, 2019] Jianmo Ni, Jiacheng Li, and Julian McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 188–197, 2019.

[Rossi *et al.*, 2020] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal graph networks for deep learning on dynamic graphs. *CoRR*, abs/2006.10637, 2020.

[Shamsi *et al.*, 2022] Kiarash Shamsi, Friedhelm Victor, Murat Kantarcioglu, Yulia Gel, and Cuneyt G Akcora. Chartalist: Labeled graph datasets for utxo and account-based blockchains. *Advances in Neural Information Processing Systems*, 35:34926–34939, 2022.

[Shetty and Adibi, 2004] Jitesh Shetty and Jafar Adibi. The enron email dataset database schema and brief statistical report. *Information sciences institute technical report, University of Southern California*, 4(1):120–128, 2004.

[Tian *et al.*, 2023] Yuxing Tian, Yiyan Qi, and Fan Guo. Freedyg: Frequency enhanced continuous-time dynamic graph model for link prediction. In *The Twelfth International Conference on Learning Representations*, 2023.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[Wang *et al.*, 2022] Kaixin Wang, Cheng Long, Da Yan, Jie Zhang, and H. V. Jagadish. Reinforcement learning enhanced weighted sampling for accurate subgraph counting on fully dynamic graph streams, 2022.

[Xu *et al.*, 2020] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Inductive representation learning on temporal graphs. *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30*, 2020.

[Xu *et al.*, 2024] Yuanyuan Xu, Wenjie Zhang, Ying Zhang, Maria Orlowska, and Xuemin Lin. Timesgn: Scalable and effective temporal graph neural network. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pages 3297–3310. IEEE, 2024.

[Ying *et al.*, 2018] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 974–983, 2018.

[Younesian *et al.*, 2024] Taraneh Younesian, Daniel Daza, Emile van Krieken, Thiviyan Thanapalasingam, and Peter Bloem. Grapes: Learning to sample graphs for scalable graph neural networks, 2024.

[Yu *et al.*, 2023] 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.

[Zeng *et al.*, 2020] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graphsaint: Graph sampling based inductive learning method. In *International Conference on Learning Representations*, 2020.

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

[Zou *et al.*, 2024] Tao Zou, Yuhao Mao, Junchen Ye, and Bowen Du. Repeat-aware neighbor sampling for dynamic graph learning. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4722–4733, 2024.