ADADGL: ADAPTIVE DISTRIBUTIONAL NEGATIVE SAMPLING FOR ROBUST DYNAMIC GRAPH LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Dynamic graph models are commonly trained and evaluated with sampled negative edges, making their performance inherently dependent on the underlying negative sampling distribution. However, existing methods typically rely on a fixed distribution, such as random, historical, or inductive sampling. This creates a negative sampling distribution shift: models trained under one distribution can degrade substantially when evaluated under another, while evaluation with easy random negatives can overestimate model performance.

We propose **AdaDGL**, an adaptive distributional negative sampling framework for robust dynamic graph learning under such shifts. During training, AdaDGL samples similarity-aware negatives from broadly covered candidates to balance hardness and coverage. At inference, it constructs hard negatives for reliable evaluation when the target distribution is unknown, and efficiently adapts the prediction head when the target distribution is known. Experiments across multiple datasets and backbones show consistent gains under different negative sampling distributions, with up to **7.41%** average precision improvement and **2.98**$\times$ training acceleration when supporting multiple target distributions.

1 INTRODUCTION

Robust dynamic graph learning aims to learn temporal representations that remain reliable as graph structures, node semantics, and interaction patterns evolve over time, supporting edge- and node-level temporal prediction tasks in real-world systems such as social media (Alvarez-Rodriguez et al., 2021; Kumar et al., 2019; Song et al., 2019), recommender systems (Fan et al., 2021; Yi et al., 2025; Zhang et al., 2023), communication networks, and traffic forecasting (Sharma et al., 2023; Yu et al., 2018; Jiang et al., 2023). Among these tasks, dynamic link prediction is a prevalent objective for training and testing dynamic graph models (Kumar et al., 2019; Xu et al., 2020; Rossi et al., 2020; Wang et al., 2021; Cong et al., 2023b; Yu et al., 2023; Tian et al., 2024; Zou et al., 2024; Lu et al., 2024; Peng et al., 2025).

Negative sampling is essential in dynamic link prediction (Trivedi et al., 2019; Yang et al., 2020; Wang & Isola, 2020) because positive edges are observed, whereas negative edges are not explicitly labeled and must be constructed. Consequently, the resulting negative sampling distribution is therefore not merely an implementation choice: it defines the contrastive self-supervision during training, shapes the learned temporal representations, and determines the hardness of negative samples at inference. Robust dynamic graph learning therefore requires models to remain reliable across different negative sampling distributions during training and inference.

However, **existing works typically enforce a single, fixed negative sampling distribution**, such as random, historical, or inductive sampling. These strategies suffer from a coverage–hardness tradeoff, as shown in Figure 1. Random sampling operates without prior knowledge of the target distribution[1] and broadly covers candidate negative nodes, but it often fails to provide sufficiently hard negatives for discriminative learning (Poursafaei et al., 2022; Tabassum et al., 2022). Prior-based sampling, including historical and inductive sampling, leverages assumptions about the target distribution and can produce harder negatives, but the introduced prior knowledge may bias negative sample selection and restrict robustness to other distributions (Geweke, 1989; Rawat et al., 2021). This limitation is further amplified in dynamic graphs, where structural proximity, semantic similarity, node activity, and interaction patterns evolve over time. As a result, a model optimized under one negative distribution may fail to generalize when the negative distribution changes.

[1] Prior knowledge excludes certain edges by assigning them zero probability based on past assumptions.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

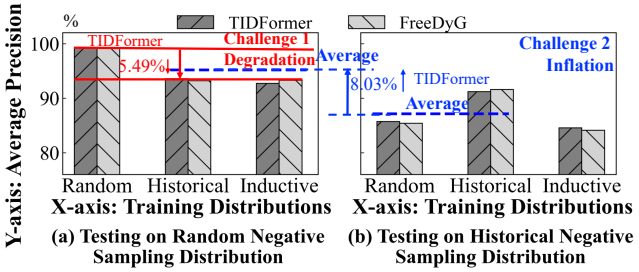


Figure 1: Average precision of TIDFormer (Peng et al., 2025) and FreeDyG (Tian et al., 2024) on Wikipedia (Poursafaei et al., 2022) under different training and test negative distributions (transductive setting). We obtain consistent findings under inductive setting and other dynamic graph models.

Figure 1 empirically confirms this bottleneck of dependence on genagive sampling distributions. First, a mismatch between training and testing distributions can substantially degrade prediction performance. For example, TIDFormer (Peng et al., 2025) trained with random negative sampling shows obtains **5.49%** lower average precision under historical testing than the same model trained with historical negatives. Second, the testing distribution itself can substantially affect the apparent model performance. For instance, TIDFormer obtains an average of **8.03%** higher average precision under random testing than under historical testing, as random sampling often produce easier negatives. We observe consistent patterns across other dynamic graph models and settings (Appendix D.2). These observations reveal two related issues for robust dynamic graph learning: fixed negative distributions limit robustness across training and testing conditions, while easy negative distributions can lead to unreliable evaluation at inference.

These observations raise three challenges for robust dynamic graph learning under a negative sampling distribution shift. **Challenge 1: Robust training under varying negative distributions.** It shows that average precision degrades when training and testing negative sampling distributions are mismatched, suggesting that fixed training distributions fail to provide robust self-supervision. Effective negatives should be hard for discriminability, broad for distributional coverage, and adaptive to evolving semantic and structural similarities. **Challenge 2: Reliable inference with unknown target negative distributions.** When the target distribution is unknown, random sampling is a straightforward and prior-free choice, but it often inflates the average precision with easy negatives, failing to indicate whether the model can distinguish positives from semantically and structurally close hard negatives. **Challenge 3: Efficient adaptation to known target negative distributions.** When the target distribution is known, such as historical or inductive sampling, conventional approaches typically train a separate model for each target distribution, introducing redundant computation. Thus, efficient adaptation is needed without full retraining.

To address these challenges, we propose AdaDGL, an adaptive distributional negative sampling framework for robust dynamic graph learning. For **Challenge 1**, we propose **a dynamic distribution-aware negative sampling method** for dynamic graph training, which models a semantic-and-structure-driven negative sampling distribution. This enables the learned model to maintain generalization ability while improving discriminative capacity, resulting in a general model that can adapt to different negative sampling distributions (Section 3.1). For **Challenge 2**, we propose **a similarity-aware negative sampling strategy** at inference in Section 3.2 which generates more semantically and structurally similar negative samples to enable more reliable evaluation of model performance in situations without prior knowledge. For **Challenge 3**, we propose **an efficient fine-tuning method for prior-knowledge** in Section 3.2, which enhances the ability of a general model to leverage prior knowledge, leading to improved performance without requiring retraining.

Our contributions are summarized as follows: (1) We identify **negative sampling distribution shift** as a critical robustness bottleneck in dynamic graph learning, showing that mismatched training and testing distributions can degrade prediction performance, while easy testing negatives can obscure model discriminability. (2) We propose AdaDGL, a plug-and-play adaptive distributional negative sampling mechanism for existing continuous-time dynamic graph models, supporting similarity-aware training negatives, prior-free hard-negative inference, and lightweight adaptation to known target distributions. (3) We evaluate AdaDGL across multiple datasets, dynamic graph backbones, and both link prediction and node classification tasks, showing consistent gains with up to a **7.41%** average precision improvement and **2.98**$\times$ training acceleration by avoiding redundant full training for different negative distributions.

2 PRELIMINARIES

Dynamic Graphs (Peng et al., 2025; Yu et al., 2023; Lu et al., 2024). A dynamic graph is defined as $G = (V, E, T)$, where V is the node set, E is set of the interactions, T is the timestamp set.

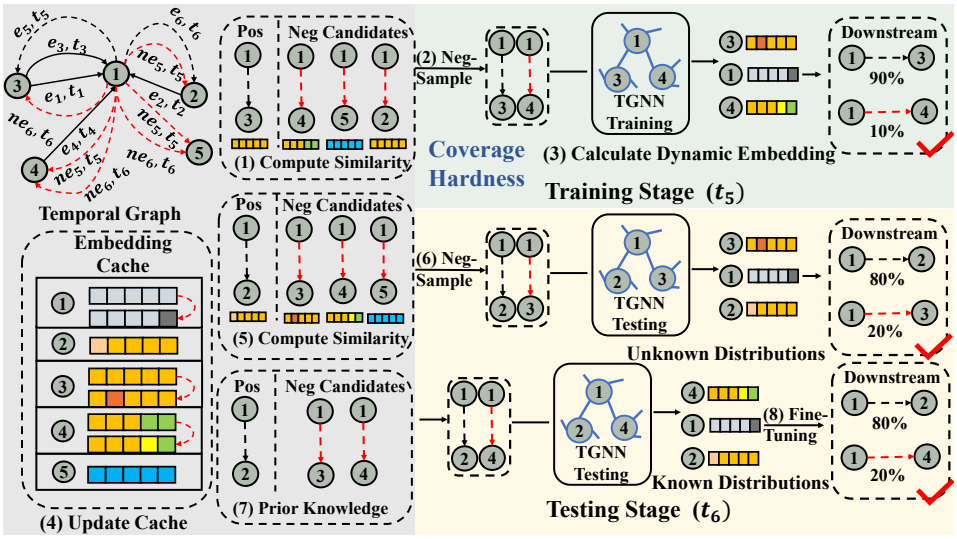


Figure 2: **Overview of AdaDGL**. **A. Training Stage.** (1) Compute similarities between positive nodes and negative candidate nodes; (2) Negative sample according to a similarity probability distribution; (3) Calculate dynamic embeddings of nodes; (4) Update node embedding cache accordingly with the newly calculated embeddings. **B. Inference Stage for w/o prior knowledge situation**. (5) Compute similarities between positive nodes and negative candidate nodes. (6) Negative sample the most similar negative node from candidates, which is used for contrastive inference. **C. Inference Stage for prior knowledge situation**. (7) Use the prior-knowledge negative sampling. (8) Fine-tune multi-layer perceptron predictor on training data. Then we use the fine-tuned model for downstream tasks.

Edge stream $E = \langle e_1, e_2, \dots \rangle$ represents the evolution of the dynamic graph structure. Each edge $e_i = (u_i, v_i, t_i)$ represents an interaction from node u_i to node v_i at time t_i. The features of u_i, v_i, and e_i are denoted as x_{u_i}, x_{v_i}, and x_{e_i}, respectively.

Dynamic Graph Learning. Given an edge stream E, dynamic graph learning aims to learn node representations that summarize historical structural, temporal, and semantic information from the evolving graph. In this work, we consider two common downstream tasks: (1) *Link Prediction*: given an edge candidate $e = (u_e, v_e, t_e)$, predict whether u_e and v_e interact at time t_e; and (2) *Node Classification*: given a node involved in an interaction, predict its label at a specific timestamp.

Related Works. Recent dynamic graph learning methods typically learn node representations from both structural and feature information over continuous time (Cong et al., 2023b; Yu et al., 2023; Tian et al., 2024; Zou et al., 2024; Lu et al., 2024; Peng et al., 2025). These methods can be broadly divided into three categories: RNN-based, MLP-based, and attention-based methods. RNN-based methods, such as JODIE (Kumar et al., 2019), TGN (Rossi et al., 2020), and CAWN (Wang et al., 2021), sample temporal neighborhood information and model time-ordered interactions. JODIE and TGN update node embeddings via recurrent architectures, while CAWN encodes sampled walks with RNNs to generate representations. MLP-based temporal models, such as GraphMixer (Cong et al., 2023b), RepeatMixer (Zou et al., 2024), FreeDyG (Tian et al., 2024), and TPNet (Lu et al., 2024), capture dynamic patterns through feed-forward architectures that mix temporal and neighborhood information without relying on recurrent or attention mechanisms. Attention-based methods, such as TGAT (Xu et al., 2020), DyGFormer (Yu et al., 2023), and TIDFormer (Peng et al., 2025), model temporal dependencies by leveraging self-attention over time-ordered interactions.

3 AdaDGL

This section presents AdaDGL for robust dynamic graph learning under negative sampling distribution shifts. As shown in Figure 2, AdaDGL addresses this shift at both training (Section 3.1) and inference (Section 3.2) stages: it constructs similarity-aware negatives for robust self-supervision during training, performs hard-negative inference when the target distribution is unknown, and adapts only the prediction head when the target distribution is known.

3.1 Dynamic Distribution-Aware Negative Sampling Method during Training

As discussed in Section 1, robust training requires negatives that balance discriminative hardness and candidate coverage. To begin, we construct the candidate set for negative sampling with respect to the observed edges as follows.

Step 1: Candidate Set Construction for Negative Sampling. For an observed edge $e_i = (u_i, v_i, t_i)$, AdaDGL constructs a candidate set $\mathcal{C}$ by randomly sampling k nodes[3] from the valid negative node set $\mathcal{N}_{e_i}$, where $\mathcal{N}_{e_i}$ is the set of all nodes excluding u_i, v_i[4], and the target nodes pointed to by u_i at timestamp t_i. We can call the observed target node v_i the positive node (or denote it as v_i^+) and an unobserved target node $v_j^- \in \mathcal{C}$ from the negative candidate for this edge, while noting that the positive/negative labels are defined over edges rather than nodes.

Random candidate construction provides broad coverage over valid negative nodes, avoiding early commitment to fixed negative sampling rules. Then, for an observed edge, we sample one node v_j^- (which is used as the negative for training) from the negative candidate set, based on a negative distribution calculated by the similarity-aware mechanism, as shown in Step 2.

Step 2: Similarity-Aware Negative Sampling. Firstly, we define the similarity measure as follows.

Similarity Measure. For two nodes $u, v \in V$ with embeddings $\mathbf{h}_u$ and $\mathbf{h}_v$, we define

$$\mathrm{sim}(u, v) = \cos(\mathbf{h}_u, \mathbf{h}_v), \tag{1}$$

where $\cos(\cdot, \cdot)$ denotes cosine similarity. In AdaDGL, this function is used as $\mathrm{sim}(v_i^+, v_j^-)$ to measure the semantic and structural closeness between the positive node and a candidate negative target node. For each candidate negative node $v^- \in \mathcal{C}$, AdaDGL computes $\mathrm{sim}(v^+, v^-)$ using Equation 1. Motivated by (Schroff et al., 2015; Huang et al., 2013), we define the dynamic distribution-aware negative sampling distribution (DNS) based on node similarities.

Definition 1 (**Dynamic Distribution-aware Negative Sampling (DNS) Distribution**). *For a positive node v^+ and candidate negative set $\mathcal{C}$, DNS samples a negative candidate $v^- \in \mathcal{C}$ with probability*

$$p_{\mathrm{DNS}}(v^- \mid v^+; \tau) = \frac{\exp(\mathrm{sim}(v^+, v^-)/\tau)}{\sum_{u \in \mathcal{C}} \exp(\mathrm{sim}(v^+, u)/\tau)}, \tag{2}$$

where τ[5] controls the sharpness of the sampling distribution and stabilizes the early training stage.

Based on Definition 1, AdaDGL assigns higher sampling probabilities to candidate negatives that are more similar to the positive node. This provides harder negatives for discriminative self-supervision. Meanwhile, because candidates are randomly constructed from the valid negative set[6], AdaDGL maintains broad candidate coverage instead of binding the model to fixed negative sampling strategies. For each observed edge, we sample one node from $\mathcal{C}$ according to the probability distribution defined in Equation 2. After sampling, the dynamic graph backbone computes node embeddings from the positive edges and sampled negatives, and updates its parameters through the training objective to better reflect the most recent semantic and structural evoluation in dynamic graph. Node embeddings encode evolving semantic and structural information of the dynamic graph. As shown in Figure 2, Steps 1-4. By increasing the hardness of negative samples, the model learns finer decision boundaries, continuously receiving external information from evolving graph data via updated embeddings.

Embedding Cache. To further enhance the training efficiency, we introduces an embedding cache that maintains the most recent embeddings produced during the previous mini-batch, to avoid the redundant runtime cost produced by the embedding similarity recomputation.

Analysis of Hardness and Coverage during Training. We analyze DNS through the lens of discriminative hardness and candidate coverage. We first define a similarity-aware target distribution, which assigns higher probability to negatives that are more similar to the positive node while preserving full support over candidate nodes.

Definition 2 (**Target Distribution**). *For a positive node v^+, we define the target negative sampling distribution as*

$$p_{\mathrm{target}}(v^- \mid v^+) = \frac{\exp(\mathrm{sim}(v^+, v^-))}{\sum_{u \in V} \exp(\mathrm{sim}(v^+, u))}. \tag{3}$$

[3]For observed edge $e_i = (u_i, v_i, t_i)$, we perform random sampling by obtaining a candidate node v_j^- from node set $V \setminus \{u_i, v_i\}$. Repeating this procedure k times yields k candidates. We use $k = 10$ in our experiments.

[4]When multiple edges sharing the same source node are observed at the same timestamp, we exclude all their endpoints from the set of nodes eligible for negative sampling to prevent false negatives from introducing noise and degrading model performance (Chen et al., 2022; Xia et al., 2022). For example, given observed temporal edges $e_i = (u_i, v_i, t_i)$ and $e_j = (u_i, v_j, t_i)$, we exclude u_i, v_i, and v_j from the eligible nodes.

[5]Here, $\tau \in \{2, 1\}$. Training starts with $\tau = 2$, and when validation loss stops decreasing, τ is lowered to 1.

[6]During negative sampling, to avoid false negative noise, our strategy needs to exclude all negative samples. We conduct a static analysis in Appendix D.8. This analysis shows we completely exclude all false negative nodes.

Based on (Huang et al., 2013; Schroff et al., 2015; van den Oord et al., 2018), to yield stronger contrastive signals for enhanced model performance, negative nodes similar to the positive ones should be sampled with a higher frequency. Definition 2 formalizes this intuition as a similarity-aware target distribution. By assigning higher probability to hard negatives, i.e., negatives that are semantically or structurally close to positives, this distribution encourages the model to learn fine-grained decision boundaries among similar nodes during training.

Definition 3 (Coverage Objective for All Positives). *Let $\mathcal{P}_{\text{train}}$ be the set of positive nodes during training. We define the overall coverage objective over a negative sampling distribution p as*

$$\text{Coverage}(p) = \sum_{v^+ \in \mathcal{P}_{\text{train}}} \frac{|\{u \in V \mid p(u \mid v^+) = 0\}|}{|V|}. \tag{4}$$

This objective measures the fraction of nodes that cannot be sampled as negatives for each positive nodes. Following the coverage perspective in (Geweke, 1989), a smaller value indicates broader candidate coverage, because fewer nodes are assigned zero sampling probability. Conversely, a larger value means that more nodes are excluded from negative sampling, leading to poorer coverage.

Theorem 1. *Let p_{target} be the negative sampling distribution as defined in Definition 2. Then,* $\text{Coverage}(p_{\text{target}}) = 0$.

We provide a detailed proof of Theorem 1[7] Based on Theorem 1 and $\text{Coverage}(p) \geq 0$, we find that p_{target} yields the minimum coverage objective. Thus, target distribution could balance hardness and coverage during training.

Lemma 1. *For a positive node v^+ and a corresponding negative node v^-, let $p_{DNS}(v^+, v^-)$ be the probability of the DNS distribution and $p_{\text{target}}(v^+, v^-)$ be the target distribution. Then, as the number of training epochs increases, and we can obtain that* $\text{KL}\big(p_{\text{DNS}} \,\|\, p_{\text{target}}\big) \to 0$*, where* $\text{KL}(\cdot, \cdot)$ *is the Kullback-Leibler divergence.*

We provide a detailed proof of Lemma 1 in Appendix A.2. Based on Lemma 1, as the training proceeds, the negative sample distribution can closely approximate target distribution, finally converges toward the target distribution p_{target}.

Lemma 2. *Let $p_{\text{static}}(v^+, v^-)$ denote a static negative sampling distribution, such as random, historical, or inductive sampling. Then,* $\text{KL}\big(p_{\text{static}} \,\|\, p_{\text{target}}\big) > 0$.

We provide a detailed proof of Lemma 2 in Appendix A.2. Based on Lemma 2, as the negative sample distribution can not approximate target distribution.

Theorem 2. *DNS achieves a smaller KL divergence from the target distribution than that of existing random methods, i.e.,* $\text{KL}(p_{\text{DNS}} \,\|\, p_{\text{target}}) < \text{KL}(p_{\text{random}} \,\|\, p_{\text{target}})$.

We provide a detailed proof of Theorem 2 in Appendix A.2. Based on Definition 2 and Theorem 2, DNS better aligns with the similarity-aware target distribution than static negative sampling methods, as reflected by a smaller Kullback–Leibler divergence. This alignment enables DNS to provide harder negatives while preserving candidate coverage under the defined coverage objective.

Complexity Analysis. For a training batch of m edges, DNS computes similarities between each positive sample and a constant-size candidate set. Since the candidate size k is fixed, DNS introduces $O(m)$ additional computation and storage for negative sampling, matching the complexity of static negative sampling methods. In contrast, directly sampling from the target distribution over all nodes would require comparing each positive sample against all candidate nodes, incurring $O(|V| \cdot m)$ time and space complexity. Detailed analysis is provided in Appendix B.1.

3.2 Similarity-Aware Negative Sampling and Adaptation Method at Inference

For the distribution of negative nodes at inference, we consider two categories: (1) scenarios with unknown target negative distributions and (2) scenarios with known target negative distributions. The former corresponds to the prior-free setting, where the inference distribution is unknown and no predefined historical or inductive prior is available. Recent distribution adaptation methods (Liao et al., 2026; Lee et al., 2023) would like to adapt to known distributions. However, adaptation to known negative sampling distributions remains underexplored in dynamic graph learning.

Similarity-Aware Negative Sampling for Unknown Target Negative Distributions (SNS). For unknown target negative distributions, we propose the SNS method to enable more reliable prior-free evaluation of model performance. Random sampling is prior-free but often yields overly easy

[7]For Theorems 1 to 5, We defer the assumptions of theorems in Appendix A. These assumptions are commonly adopted in theorem analysis. Meanwhile, we further empirically validate the effectiveness of the theoretical results through experiments in Section D.8.

negatives. In contrast, SNS considers semantic and structural information encoded by dynamic graph embeddings, and therefore constructs harder negatives for inference.

The proposed SNS method consists of two steps. First, given an observed edge e_i, we sample $\epsilon \times |V|$[8] candidate nodes randomly from $\mathcal{N}_{e_i}$, where $\mathcal{N}_{e_i}$ is the node set with respect to e_i, defined in Step 1 of Section 3.1. Second, we compute the similarity between each candidate $v_j^- \in \mathcal{N}$ and the positive node v_i based on Equation 1, and select the candidate with the largest similarity[9] as the negative sample. Thus, SNS constructs semantically and structurally close hard negatives without relying on predefined historical or inductive priors.

Adaptation for Known Target Negative Distributions. We provide an adaptation method for known target negative distributions, enabling a general model to efficiently leverage prior knowledge. When the target distribution is known, such as historical or inductive sampling, directly retraining a model for each target distribution introduces redundant computation. To avoid this, our adaptation method freezes the parameters of the dynamic graph backbone and updates only the lightweight multilayer perceptron-based prediction head on training data sampled from the known target distribution. This allows the model to adapt to the known downstream distribution without full retraining.

Analysis of Hardness and Coverage at Inference. We analyze why SNS achieves a better coverage-hardness tradeoff than random sampling, which is the common prior-free baseline for unknown target negative distributions.

Definition 4 (**Node Hardness Rank**). *When inferring an edge $e_i = (u_i, v_i^+, t_i)$, given source node u_i and positive node v_i^+, we compute the similarity between each candidate $v_j \in \mathcal{N}$ and v_i^+ as $s_{v_j^-} = \mathrm{sim}(v_i^+, v_j^-)$. Let $f(v_j^-)$ denote the position of v_j^- in the descending order of similarity scores. The hardness rank of a selected negative node v_j^- is defined as $R_{v_j^-} = \frac{f(v_j^-)}{|\mathcal{N}|}$. A smaller hardness rank indicates a harder negative sample.*

Definition 5 (**Method Hardness Rank**). *For each test edge e_k, let $R_{e_k} = \mathbb{E}_{v^- \in \mathcal{N}_{e_k}}[R_{v^-}]$, where $\mathcal{N}_{e_k}$ is the node set with respect to e_k, defined in Step 1 of Section 3.1. The method hardness rank is defined as the average hardness over all test edges: $R_{\text{method}} = \frac{1}{K}\sum_{k=1}^{K} R_{e_k}$, where K is the number of test edges.*

Theorem 3. *Let R_{SNS} and R_{random} denote the expected normalized hardness ranks produced by similarity-aware negative sampling and uniform random sampling, respectively. Then,*

$$R_{\text{SNS}} < R_{\text{random}}. \tag{5}$$

We provide a detailed proof of Theorem 3 in Appendix A.3. Since $R_{\text{SNS}} \leq R_{\text{random}}$ for $\epsilon \in (\frac{1}{|V|}, 1)$, SNS selects harder negative samples than random sampling under the hardness-rank metric. Therefore, SNS provides a stronger prior-free inference protocol for evaluating model discriminability.

Definition 6 (**Coverage Objective for Inference**). *Let $\mathcal{P}_{\text{infer}}$ be the set of positive nodes at inference. We define the overall coverage objective over a negative sampling method as:*

$$\text{Coverage}(\text{method}) = \sum_{v^+ \in \mathcal{P}_{\text{infer}}} \frac{|\{v_i \in V \mid p(v_i, v^+) = 0\}|}{|V|}. \tag{6}$$

This objective measures the fraction of nodes that cannot be sampled as negative nodes at inference. A smaller value indicates broader candidate coverage.[10]

Theorem 4. *The coverage of our similarity-aware negative sampling is 0.*

We provide a detailed proof of Theorem 4 in Appendix A.4. Theorem 4 shows that SNS can potentially sample all nodes as negative examples across positive nodes at inference. Since $\text{Coverage}(\text{method}) \geq 0$, SNS achieves full support under the defined coverage objective.

Based on Theorems 3 and 4, SNS provides harder negative samples than random sampling while maintaining broad candidate coverage in the prior-free inference setting.

Analysis of Adaptation Method. We analyze why the adaptation method can improve performance when the target negative distribution is known.

[8] ϵ is a sampling ratio, which is set to 0.1 in our experiments, $\epsilon \in (\frac{1}{|V|}, 1]$.

[9] During inference, we select the most similar nodes as negatives to evaluate model performance under a hard test distribution. In contrast, during training, we sample negatives from a similarity-based probability distribution rather than always selecting the most similar nodes, making the model robust across different test distributions.

[10] Where $p(v_i, v^+)$ denotes the probability of selecting v_i as a negative sample for v^+.

Theorem 5 (**Empirical Risk Analysis of Adaptation Method**). *Let $f_w(x)$ denote the prediction function after adapting to the known distribution, and let $f_{w/o}(x)$ denote the prediction function without adapting. $\mathcal{L}$ is a loss function, p_{infer} is the inference distribution. Under standard empirical risk minimization, optimizing on the known distribution yields no worse expected risk:* $\mathbb{E}_{p_{\text{infer}}}[\mathcal{L}(f_w(x), y)] \leq \mathbb{E}_{p_{\text{infer}}}[\mathcal{L}(f_{w/o}(x), y)]$.

We provide a detailed proof of Theorem 5 in Appendix A.5. Theorem 5 shows that, under the same target inference distribution and fixed representations, adapting the prediction head can reduce the expected risk compared to the non-adapted model.

Complexity Analysis. Let n denote the total node number. Both conventional static negative sampling and AdaDGL have linear time and space complexity, $O(n)$, ensuring that AdaDGL introduces no additional overhead compared to static approaches at inference. Details are provided in Appendix B.2.

4 EVALUATION

We conduct extensive experiments to answer the following research questions: $\mathbf{Q_1}$: Can AdaDGL improve the prediction performance in the testing stage? Does our similarity-aware sampling (SNS) strategy of AdaDGL could better reflect the models' ability? $\mathbf{Q_2}$: How efficient is AdaDGL in terms of training and testing time? $\mathbf{Q_3}$: What is the effect of the proposed optimization techniques and corresponding theorems? $\mathbf{Q_4}$: How do the parameters of AdaDGL impact its overall performance and effectiveness? $\mathbf{Q_5}$: How does AdaDGL perform on other downstream tasks (e.g., node classification) and in long-term datasets(e.g., tgbl-comment)?

4.1 EXPERIMENTS SETUP

Datasets. We utilize seven publicly available real-world datasets: Wikipedia, REDDIT, MOOC, LastFM, Enron, Social Evo, and UCI, which are collected by (Poursafaei et al., 2022). Appendix C.1 provides a detailed description and the statistics of the datasets. For model training, validation and testing, the datasets were divided chronologically into three subsets at a 70%–15%–15% split ratio.

Baselines. To evaluate the performance of AdaDGL, we conduct experiments comparing it with previous state-of-the-arts, including TGAT, TGN, GraphMixer (GM), DyGFormer (DGF), FreeDyG (FDG), RepeatMixer (RM), TPNet (TPN), TIDFormer (TIDF). Detailed descriptions of these baselines can be found in Appendix C.2. For fairness, all parameters follow the defaults. All the above methods can be categorized into three groups: Memory-based, MLP-Mixer-based, Attention-based. We choose one representative model from each category in our AdaDGL: TGN (AdaTGN), FreeDyG (AdaFDG) and TIDFormer (AdaTIDF). The results show that our method improves prediction accuracy on different groups. Our code is available in the supplementary material.

Evaluation Tasks and Implementation Details. We follow (Kumar et al., 2019; Xu et al., 2020; Rossi et al., 2020; Wang et al., 2021; Cong et al., 2023b; Yu et al., 2023; Tian et al., 2024; Zou et al., 2024; Lu et al., 2024; Peng et al., 2025) to evaluate models for link prediction. This task has two settings: the transductive setting and the inductive setting. For traditional sampling method. We adopt random (rnd), historical (hist), inductive (ind) negative sampling strategies for evaluation. For each strategy, the baseline model is trained under random same negative sampling distribution and evaluated under the corresponding negative sampling distribution (random (rnd), historical (hist), inductive (ind) and SNS negative sampling strategies. Our method is trained under the dynamic distribution-aware negative sampling distribution and evaluated under different negative sampling strategies at testing time. The model performance is measured by Average Precise (AP) and Area Under the Receiver Operating Characteristic Curve (AUC-ROC). We use AP to evaluate performance of link prediction, and we use AUC-ROC to evaluate performance of node classification. In our tables, the best and second-best results are emphasized by **bold** and underlined fonts.

4.2 PERFORMANCE ($\mathbf{Q_1}$)

We evaluate AdaDGL (AdaTIDF, AdaTGN, AdaFDG) on 7 datasets. We report the performance of different methods on the AP metric for transductive dynamic link prediction with three negative sampling strategies in Table 1.

Specifically, **AdaDGL outperforms existing state-of-the-art methods by up to 7.41%.** Meanwhile, according to the average rankings across all datasets, In the hist, ind, and SNS settings, AdaTIDF occupies first place with average ranks of 1.57, 1.00 and 1.14, respectively, while AdaFDG consistently ranks second at 2.57, 2.29 and 2.29. AdaTIDF and AdaFDG consistently achieve the best and second-best average performance. Taking TIDFormer as a representative example, our AdaDGL framework yields remarkable performance improvements. with an overall average gain of 3.17%.

Furthermore, **AdaDGL can be flexibly applied to various existing models and consistently enhance their performance,** for the classical method TGN, AdaDGL improves its average rank from

Table 1: AP (%) for Transductive Link Prediction with Different Negative Nampling Strategies. Results of TGAT, GM, and DGF are Provided in Appendix D.3 .

NS	Datasets	TGN	FDG	RM	TPN	TIDF	AdaTIDF	AdaTGN	AdaFDG
hist	Wikipedia	86.86 ± 0.33	91.59 ± 0.57	90.20 ± 1.04	81.55 ± 4.10	91.21 ± 0.78	92.25 ± 0.76	88.11 ± 0.29	**92.31 ± 0.42**
	Reddit	81.22 ± 0.61	85.67 ± 1.01	83.02 ± 1.20	81.02 ± 1.31	83.27 ± 1.01	84.07 ± 0.99	82.34 ± 0.56	**87.61 ± 1.04**
	MOOC	87.06 ± 1.93	86.71 ± 0.81	92.19 ± 0.58	92.69 ± 0.95	95.37 ± 1.09	**96.87 ± 1.01**	89.08 ± 1.62	90.62 ± 0.76
	LastFM	76.87 ± 4.64	79.71 ± 0.51	86.73 ± 0.34	87.74 ± 0.50	88.68 ± 0.57	**89.97 ± 0.55**	78.32 ± 2.52	82.71 ± 0.56
	Enron	73.91 ± 1.76	78.87 ± 0.82	87.38 ± 0.18	80.79 ± 1.68	81.59 ± 1.17	**87.62 ± 1.19**	76.22 ± 1.66	81.64 ± 0.75
	Social	94.45 ± 0.56	97.79 ± 0.23	97.40 ± 0.15	96.80 ± 0.41	97.39 ± 0.67	98.17 ± 0.92	96.71 ± 0.42	**98.62 ± 0.31**
	UCI	80.43 ± 2.12	86.10 ± 1.19	87.23 ± 0.23	86.34 ± 0.80	89.57 ± 1.17	**93.34 ± 0.72**	83.76 ± 0.83	90.13 ± 1.01
	Avg. Rank	7.57	4.86	4.00	5.43	3.43	**1.57**	6.57	2.57
ind	Wikipedia	85.62 ± 0.44	90.05 ± 0.79	88.86 ± 0.97	79.35 ± 5.52	91.15 ± 1.17	**92.34 ± 1.14**	88.64 ± 0.33	92.10 ± 1.07
	Reddit	88.10 ± 0.24	90.74 ± 0.17	91.11 ± 0.73	88.19 ± 0.33	91.57 ± 0.50	**92.46 ± 0.62**	88.16 ± 0.26	91.66 ± 0.43
	MOOC	77.50 ± 2.91	83.01 ± 0.87	83.11 ± 1.28	88.18 ± 0.97	84.53 ± 1.11	**88.49 ± 1.06**	76.92 ± 1.86	87.62 ± 1.05
	LastFM	65.95 ± 5.98	72.19 ± 0.24	75.46 ± 0.78	77.99 ± 1.30	75.69 ± 1.17	**85.40 ± 0.96**	75.33 ± 2.85	80.25 ± 0.57
	Enron	70.89 ± 2.72	77.81 ± 0.65	83.17 ± 0.50	75.36 ± 1.81	79.67 ± 0.66	**83.24 ± 0.62**	74.78 ± 1.32	80.46 ± 0.54
	Social	95.13 ± 0.56	97.57 ± 0.15	97.52 ± 0.28	97.35 ± 0.32	97.70 ± 0.36	**98.50 ± 0.24**	95.92 ± 0.44	98.14 ± 0.17
	UCI	70.94 ± 0.71	82.35 ± 0.73	84.20 ± 0.34	77.26 ± 1.57	85.81 ± 1.39	**87.78 ± 0.86**	73.11 ± 0.71	86.34 ± 0.17
	Avg. Rank	7.57	5.14	4.29	5.29	3.43	**1.00**	7.00	2.29
SNS	Wikipedia	66.71 ± 0.83	75.81 ± 0.83	74.73 ± 0.76	77.81 ± 0.65	75.31 ± 0.83	78.33 ± 0.82	72.42 ± 0.75	**78.96 ± 0.76**
	Reddit	60.67 ± 0.33	69.71 ± 0.92	64.58 ± 0.72	66.86 ± 0.53	67.16 ± 0.98	**72.15 ± 0.91**	63.44 ± 0.31	72.04 ± 0.83
	MOOC	65.84 ± 2.21	71.77 ± 0.42	70.36 ± 0.58	74.63 ± 0.58	73.66 ± 1.83	**79.28 ± 1.31**	70.62 ± 1.76	76.47 ± 1.34
	LastFM	67.86 ± 2.71	80.62 ± 0.75	73.62 ± 1.45	80.45 ± 0.71	80.82 ± 0.81	**85.14 ± 0.76**	72.03 ± 2.85	83.69 ± 0.62
	Enron	67.62 ± 0.35	81.36 ± 0.61	83.96 ± 0.78	79.66 ± 0.71	80.63 ± 1.82	**84.87 ± 1.22**	73.70 ± 0.21	82.98 ± 0.83
	Social	76.52 ± 0.42	84.72 ± 0.81	85.41 ± 0.34	84.11 ± 0.75	85.66 ± 1.08	**87.65 ± 1.06**	84.18 ± 0.32	85.34 ± 0.64
	UCI	69.21 ± 0.96	79.89 ± 0.71	81.92 ± 0.46	81.65 ± 0.74	81.72 ± 0.83	**86.77 ± 0.82**	75.91 ± 0.67	84.92 ± 0.81
	Avg. Rank	8.00	4.43	4.71	4.86	3.86	**1.14**	6.71	2.29

7.57 to 6.57 under the historical sampling strategy. This demonstrates that AdaDGL can consistently enhance performance regardless of backbone capacity. We also conduct experiments under SNS. In Table 1, this setting provides more challenging negative samples with similar semantics and structural information at inference.

These results show DNS's superiority. DNS achieves the smallest KL divergence from the ideal distribution among different strategies, providing harder negatives while maintaining full coverage, which explains AdaDGL's consistently superior performance in Table 1. As for SNS, the average AP of all models across all datasets decreases from 87.60 in hist and 84.79 in ind to 76.68 in SNS. The results of our SNS tests reflect that we indeed provide a more challenging evaluation protocol.

Moreover, we extend our evaluation to larger datasets, as shown in Table 18 we conduct additional experiments on tgbl-comment and tgbl-coin-v2 from the TGB (Huang et al., 2023), AdaDGL shows improvements on weaker backbones (TGN: 37.90 → ADATGN: 46.27, +8.37) while also achieving SOTA performance on strong backbones (AdaFDG) in tgbl-comment. Please refer to Appendix D.3 for the results of AP for inductive dynamic link prediction and AP for random NSS.

4.3 Efficiency ($\mathbf{Q_2}$)

As shown in Tables 15 and 16. In training stage, AdaDGL can adapt to multiple distributions with a single once-trained model, whereas traditional methods must be retrained separately for each distribution, resulting in significant extra time overhead. On the Wikipedia and Reddit datasets, AdaDGL reduces the time[11] overhead by 0.6563 and 0.6453, respectively. These results demonstrate that our method achieves training acceleration. In testing stage, AdaDGL exhibits comparable time overhead with other baseline models, which is consistent with our previous analysis in Section 3.2. We also implement the time and memory overhead in Appendix D.6.

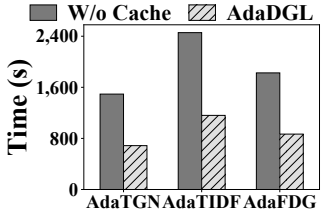


Figure 3: Ablation Study on Embedding Cache. (10 epochs, Wikipedia)

4.4 Ablation Study ($\mathbf{Q_3}$)

DNS Strategy. We conduct an ablation study by removing key components of AdaDGL: dynamic distribution-aware negative sampling distribution (DNS) and prior-knowledge fine-tune method. As shown in Table 2, removing either of these two methods leads to significant performance degradation in different negative sampling strategy. Notably, even without fine-tuning, our model still outperforms existing models, which further validates the effectiveness of our proposed DNS negative sampling and prior-knowledge-based fine-tuning strategy.

Embedding Cache. We conduct an ablation study by removing a key component of AdaDGL, the Embedding Cache. As shown in Figure 3, the Embedding Cache reduces the embedding recomputation overhead by 49.06% on average, demonstrating its effectiveness in avoiding recomputation.

4.5 Parameter Analysis ($\mathbf{Q_4}$)

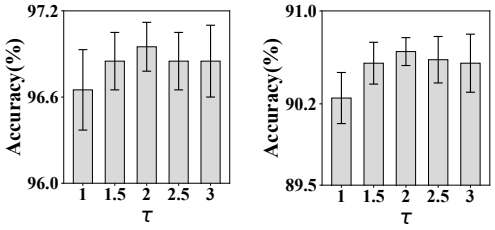
We conduct experiments by varying parameter τ and k introduced in Section 3.1, parameter ϵ introduced in Section 3.2. All experiments are conducted under hist NSS in transductive settings. More parameter experiments on inductive settings, are provided in Appendix D.3.

The parameter τ controls the initial negative sampling distribution and influences training stability.

[11]Timecost includes fine-tune time cost.

Table 2: Ablation Study of Different Components in AdaDGL under Transductive Settings

NS	Model	Method	Wikipedia	Reddit	MOOC	LastFM	Enron	Social	UCI
his	AdaTIDF	AdaDGL	**92.25±0.76**	**84.07±0.99**	**96.87±1.01**	**89.97±0.55**	**87.62±1.19**	**98.17±0.92**	**93.34±0.72**
		w/o DNS	91.21±0.78	83.27±1.01	95.37±1.09	88.68±0.57	81.59±1.17	97.39±0.67	89.57±1.17
		w/o Fine-tune	92.01±0.75	83.89±1.07	96.24±1.06	89.63±0.61	87.41±1.25	98.05±1.03	92.45±1.03
	AdaFDG	AdaDGL	**92.31±0.42**	**87.61±1.04**	**90.62±0.76**	**82.71±0.56**	**81.64±0.75**	**98.62±0.31**	**90.13±1.01**
		w/o DNS	91.59±0.57	85.67±1.01	86.71±0.81	79.71±0.51	78.87±0.82	97.79±0.23	86.10±1.19
		w/o Fine-tune	92.03±0.46	87.25±1.08	90.18±0.78	82.24±0.53	80.03±0.81	98.32±0.42	88.63±1.08
ind	AdaTIDF	AdaDGL	**92.34±1.14**	**92.46±0.62**	**88.49±1.06**	**85.40±0.96**	**83.24±0.62**	**98.50±0.24**	**87.78±0.86**
		w/o DNS	91.15±1.17	91.57±0.50	84.53±1.11	75.69±1.17	79.67±0.66	97.70±0.36	85.81±1.39
		w/o Fine-tune	91.83±1.15	91.92±0.67	86.18±1.09	82.19±1.03	81.63±0.64	98.13±0.27	86.93±0.94
	AdaFDG	AdaDGL	**92.10±1.07**	**91.66±0.43**	**87.62±1.05**	**80.25±0.57**	**80.46±0.54**	**98.14±0.17**	**86.34±0.17**
		w/o DNS	90.05±0.79	90.74±0.17	83.01±0.87	72.19±0.24	77.81±0.65	97.57±0.15	82.35±0.73
		w/o Fine-tune	91.03±1.09	91.38±0.49	85.93±0.93	78.09±0.32	79.34±0.61	98.03±0.19	85.97±0.33



(a) AdaTIDF (b) AdaFDG

Figure 4: Analysis on the τ in MOOC

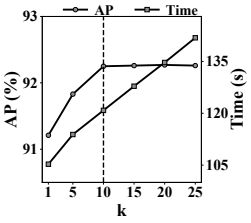


Figure 5: Analysis on k in Wikipedia (AdaTIDF)

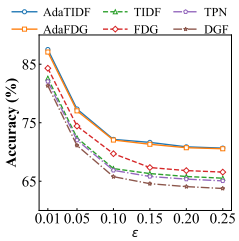


Figure 6: Analysis on ϵ in Reddit

Table 3: Performance Comparison on Dynamic Node Classification.

Dataset	JODIE	DyRep	TGAT	TGN	CAWN	GM	DGF	TIDF	AdaTIDF	AdaTGN
Wikipedia	**88.99 ± 1.05**	86.39 ± 0.98	84.09 ± 1.27	86.38 ± 2.34	84.88 ± 1.33	86.80 ± 0.79	87.44 ± 1.08	87.53 ± 1.12	88.34 ± 1.14	87.31 ± 2.46
Reddit	60.37 ± 2.58	63.72 ± 1.32	70.04 ± 1.09	63.27 ± 0.90	66.34 ± 1.78	64.22 ± 3.32	68.00 ± 1.74	69.59 ± 1.70	**70.23 ± 0.35**	64.21 ± 0.83
Avg. Rank	5.5	7.5	6.0	8.5	7.0	6.0	4.0	3.0	**1.5**	6.0

Specifically, τ acts as an experimental variable during the initial phase, and is ultimately fixed to 1 to match the target distribution. As shown in Figure 4, both AdaTIDF and AdaFDG achieve their lowest standard deviations (1.01 and 0.76) at $\tau = 2$, respectively. Therefore, we set $\tau = 2$ by default.

The parameter k controls the size of the negative candidates during training. As shown in Figure 5, (time represents training time cost per epoch), we observe that AP score increases with larger k, accompanied by longer training time. When $k = 10$, the model is already sufficient to approximate the target distribution. Considering both accuracy and efficiency, we set $k = 10$.

The parameter ϵ controls the size of negative candidates at inference. As shown in Figure 6, the model trained with AdaDGL consistently outperforms other methods across different ϵ settings.

Fine-tune Data Proportions. To examine the practical relevance of Theorem 5, we gradually increase proportions of fine-tuning data from 0% to 100% on MOOC under historical negative sampling. [12] As shown in Table 21, AP improves monotonically from 88.32 ± 1.54 to 89.08 ± 1.62 for TGN and from 95.48 ± 1.03 to 96.86 ± 1.01 for TIDFormer. The stable standard deviations and consistent gains on temporally held-out test data suggest that head-only adaptation is stable under the considered data budgets, providing support for the practical relevance of Theorem.

4.6 More Downstream Tasks ($\mathbf{Q_5}$)

We conduct experiments on dynamic node classification in Table 3. AdaTIDF outperforms most previous baselines, with an average ranking of 1.5, Meanwhile, the average ranking of TGN is improved from 8.5 to 6.0. Results demonstrate that AdaDGL achieves superior and stable performance.

5 Conclusion

This paper presents AdaDGL, an adaptive distributional negative sampling framework for robust dynamic graph learning under negative sampling distribution shifts. During training, AdaDGL learns a generalizable model through dynamic distribution-aware negative sampling, balancing negative hardness and candidate coverage. At inference, AdaDGL handles two practical settings: when the target negative distribution is unknown, SNS constructs similarity-aware hard negatives for more discriminative evaluation; when the target distribution is known, lightweight prediction-head adaptation efficiently specializes the model without full retraining. Experiments across multiple datasets and dynamic graph backbones demonstrate consistent improvements under different negative sampling distributions, while reducing the training cost of supporting multiple target distributions.

[12] Fine-tuning data are chronologically selected from the latest portion of the training period, while validation and test data remain unseen.

AI USE STATEMENT

In this work, we used generative AI tools for grammar checking, improving sentence clarity, and polishing the organization and presentation of the manuscript. We have not used generative AI tools for developing the core research ideas, formulating the problem, designing the methodology, collecting or analyzing data, conducting experiments, interpreting results, generating scientific claims, preparing figures or tables, or selecting and verifying references. These uses are not applicable to this work. Additionally, we used generative AI tools for suggesting alternative wording and improving the readability of selected passages. We have reviewed all AI-assisted work. We manually checked and revised every AI-assisted passage, independently verified all technical statements, mathematical expressions, citations, experimental descriptions, and reported results, and removed any suggestions that were inaccurate or unsupported.

ETHICS STATEMENT

This work does not involve human participants, animal subjects, or the collection of new personal or sensitive data. We use publicly available benchmark datasets solely for evaluating the scalability and performance of temporal graph neural network systems. These datasets were obtained from theiroriginal sources and used in accordance with the associated licenses and usage terms. We do not attempt to identify individuals or infer sensitive attributes from the data. To the best of our knowledge, this work does not introduce additional ethical risks beyond those inherent in the public datasets and computational methods used.

REPRODUCIBILITY STATEMENT

We have made efforts to facilitate the independent reproduction of the results presented in this work. The paper provides detailed descriptions of the proposed approach, experimental setup, datasets, evaluation protocols, and implementation details. The assumptions and derivations required to understand the theoretical results are included in the main paper and supplementary materials, where applicable. All datasets used in the experiments are publicly available, and the corresponding preprocessing procedures and data splits are described in the paper or supplementary materials. We also provide the implementation, configuration files, and instructions required to reproduce the reported experiments as supplementary materials.

REFERENCES

Unai Alvarez-Rodriguez, Federico Battiston, Guilherme Ferraz de Arruda, Yamir Moreno, Matjaž Perc, and Vito Latora. Evolutionary dynamics of higher-order interactions in social networks. *Nature Human Behaviour*, 5(5):586–595, 2021.

Tsai-Shien Chen, Wei-Chih Hung, Hung-Yu Tseng, Shao-Yi Chien, and Ming-Hsuan Yang. Incremental false negative detection for contrastive learning. In *ICLR*, 2022.

Weilin Cong, Yanhong Wu, Yuandong Tian, Mengting Gu, Yinglong Xia, Chun-cheng Jason Chen, and Mehrdad Mahdavi. Dyformer : A scalable dynamic graph transformer with provable benefits on generalization ability. In *SDM*, pp. 442–450, 2023a.

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? In *ICLR*, 2023b.

Ziwei Fan, Zhiwei Liu, Jiawei Zhang, Yun Xiong, Lei Zheng, and Philip S. Yu. Continuous-time sequential recommendation with temporal graph collaborative transformer. In *CIKM*, pp. 433–442, 2021.

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. *TKDE*, 38(1):26–46, 2026.

Kuang Gao, Chuang Liu, Jia Wu, Bo Du, and Wenbin Hu. Towards a better negative sampling strategy for dynamic graphs. *Neural Networks*, 173:106175, 2024.

John Geweke. Bayesian inference in econometric models using monte carlo integration. *Econometrica: Journal of the Econometric Society*, pp. 1317–1339, 1989.

Shubham Gupta and Srikanta Bedathur. Robust training of temporal gnns using nearest neighbour based hard negatives. In *COMAD/CODS*, pp. 54–63, 2024.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry P. Heck. Learning deep structured semantic models for web search using clickthrough data. In *CIKM*, pp. 2333–2338, 2013.

Shenyang Huang, Farimah Poursafaei, Jacob Danovitch, Matthias Fey, Weihua Hu, Emanuele Rossi, Jure Leskovec, Michael M. Bronstein, Guillaume Rabusseau, and Reihaneh Rabbany. Temporal graph benchmark for machine learning on temporal graphs. In *NeurIPS*, 2023.

Renhe Jiang, Zhaonan Wang, Jiawei Yong, Puneet Jeph, Quanjun Chen, Yasumasa Kobayashi, Xuan Song, Shintaro Fukushima, and Toyotaro Suzumura. Spatio-temporal meta-graph learning for traffic forecasting. In *AAAI*, pp. 8078–8086, 2023.

Srijan Kumar, Xikun Zhang, and Jure Leskovec. Predicting dynamic embedding trajectory in temporal interaction networks. In *KDD*, pp. 1269–1278, 2019.

Yoonho Lee, Annie S. Chen, Fahim Tajwar, Ananya Kumar, Huaxiu Yao, Percy Liang, and Chelsea Finn. Surgical fine-tuning improves adaptation to distribution shifts. In *ICLR*, 2023.

Xinyuan Liao, Xinyue Zhang, Xing Wei, Junwei Liu, Shuai Zhao, Siqi Bu, and Yi Zhang. Pe-tsfm: Self-supervised time-series learning for generalizable power converter health monitoring under unseen conditions. *IEEE Transactions on Industrial Electronics*, 2026.

Xiaodong Lu, Leilei Sun, Tongyu Zhu, and Weifeng Lv. Improving temporal link prediction via temporal walk matrix projection. In *NeurIPS*, 2024.

Guillaume Papa, Aurélien Bellet, and Stéphan Clémençon. On graph reconstruction via empirical risk minimization: Fast learning rates and scalability. In *NeurIPS*, pp. 694–702, 2016.

Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao B. Schardl, and Charles E. Leiserson. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *AAAI*, pp. 5363–5370, 2020.

Jie Peng, Zhewei Wei, and Yuhang Ye. Tidformer: Exploiting temporal and interactive dynamics makes A great dynamic graph transformer. In *KDD*, pp. 2245–2256, 2025.

James W Pennebaker, Martha E Francis, Roger J Booth, et al. Linguistic inquiry and word count: Liwc 2001. *Mahway: Lawrence Erlbaum Associates*, 71(2001):2001, 2001.

Farimah Poursafaei, Shenyang Huang, Kellin Pelrine, and Reihaneh Rabbany. Towards better evaluation for dynamic link prediction. In *NeurIPS*, 2022.

Ankit Singh Rawat, Aditya Krishna Menon, Wittawat Jitkrittum, Sadeep Jayasumana, Felix X. Yu, Sashank J. Reddi, and Sanjiv Kumar. Disentangling sampling and labeling bias for learning in large-output spaces. In *ICML*, pp. 8890–8901, 2021.

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

Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, pp. 815–823, 2015.

Amit Sharma, Ashutosh Sharma, Polina Nikashina, Vadim Gavrilenko, Alexey Tselykh, Alexander Bozhenyuk, Mehedi Masud, and Hossam Meshref. A graph neural network (gnn)-based approach for real-time estimation of traffic speed in sustainable smart cities. *Sustainability*, 15(15):11893, 2023.

Weiping Song, Zhiping Xiao, Yifan Wang, Laurent Charlin, Ming Zhang, and Jian Tang. Session-based social recommendation via dynamic graph attention networks. In *WSDM*, pp. 555–563, 2019.

Afrina Tabassum, Muntasir Wahed, Hoda Eldardiry, and Ismini Lourentzou. Hard negative sampling strategies for contrastive representation learning. *CoRR*, abs/2206.01197, 2022.

Yuxing Tian, Yiyan Qi, and Fan Guo. Freedyg: Frequency enhanced continuous-time dynamic graph model for link prediction. In *ICLR*, 2024.

Rakshit S. Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. Dyrep: Learning representations over dynamic graphs. In *ICLR*, 2019.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748, 2018.

Vladimir Vapnik. Statistical learning theory. 1998. *NY Wiley*, 1998.

Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *ICML*, pp. 9929–9939, 2020.

Yanbang Wang, Yen-Yu Chang, Yunyu Liu, Jure Leskovec, and Pan Li. Inductive representation learning in temporal networks via causal anonymous walks. In *ICLR*, 2021.

Yuxia Wu, Yuan Fang, and Lizi Liao. On the feasibility of simple transformer for dynamic graph modeling. In *WWW*, pp. 870–880, 2024.

Jun Xia, Lirong Wu, Ge Wang, Jintao Chen, and Stan Z. Li. Progcl: Rethinking hard negative mining in graph contrastive learning. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (eds.), *ICML*, volume 162, pp. 24332–24346, 2022.

Da Xu, Chuanwei Ruan, Evren Körpeoglu, Sushant Kumar, and Kannan Achan. Inductive representation learning on temporal graphs. In *ICLR*, 2020.

Zhen Yang, Ming Ding, Chang Zhou, Hongxia Yang, Jingren Zhou, and Jie Tang. Understanding negative sampling in graph representation learning. In *KDD*, pp. 1666–1676, 2020.

Lu Yi, Jie Peng, Yanping Zheng, Fengran Mo, Zhewei Wei, Yuhang Ye, Yue Zixuan, and Zengfeng Huang. Tgb-seq benchmark: Challenging temporal gnns with complex sequential dynamics. In *ICLR*, 2025.

Jiaxuan You, Tianyu Du, and Jure Leskovec. ROLAND: graph learning framework for dynamic graphs. In *KDD*, pp. 2358–2366, 2022.

Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *IJCAI*, pp. 3634–3640, 2018.

Le Yu, Leilei Sun, Bowen Du, and Weifeng Lv. Towards better dynamic graph learning: New architecture and unified library. In *NeurIPS*, 2023.

Mengqi Zhang, Shu Wu, Xueli Yu, Qiang Liu, and Liang Wang. Dynamic graph neural networks for sequential recommendation. *IEEE Trans. Knowl. Data Eng.*, 35(5):4741–4753, 2023.

Xin Zhang, Fei Cai, Jianming Zheng, Zhiqiang Pan, Wanyu Chen, Honghui Chen, and Chonghao Chen. Triangle matters! topdyg: Topology-aware transformer for link prediction on dynamic graphs. In *WWW*, pp. 3607–3617, 2025.

Yanping Zheng, Lu Yi, and Zhewei Wei. A survey of dynamic graph neural networks. *Frontiers Comput. Sci.*, 19(6):196323, 2025.

Tao Zou, Yuhao Mao, Junchen Ye, and Bowen Du. Repeat-aware neighbor sampling for dynamic graph learning. In *KDD*, pp. 4722–4733, 2024.

CONTENTS

1 Introduction **1**

2 Preliminaries **2**

3 AdaDGL **3**
- 3.1 Dynamic Distribution-Aware Negative Sampling Method during Training 3
- 3.2 Similarity-Aware Negative Sampling and Adaptation Method at Inference 5

4 Evaluation **7**
- 4.1 Experiments setup 7
- 4.2 Performance ($\mathbf{Q_1}$) 7
- 4.3 Efficiency ($\mathbf{Q_2}$) 8
- 4.4 Ablation Study ($\mathbf{Q_3}$) 8
- 4.5 Parameter Analysis ($\mathbf{Q_4}$) 8
- 4.6 More Downstream Tasks ($\mathbf{Q_5}$) 9

5 Conclusion **9**

A Proofs **15**
- A.1 AdaDGL Improves Coverage during Training. 15
- A.2 AdaDGL improves Hardness during Training. 15
- A.3 AdaDGL Improves Hardness at Inference. 17
- A.4 AdaDGL Improves Coverage at Inference. 18
- A.5 AdaDGL Improves Model Performance by Adaptation. 18

B Time and Space Complexity Computation **19**
- B.1 Time and Space Complexity Computation of AdaDGL during Training 19
- B.2 Time and Space Complexity Computation of AdaDGL at Inference. 19

C Additional Datasets and Baselines Discussion **20**
- C.1 Statistics of Datasets 20
- C.2 Statistics of Baselines 21
- C.3 Discussion about DTDG 21

D Experimental details **22**
- D.1 Hardware and Parameters Setup. 22
- D.2 AP of Different Models on Different Datasets under Inductive and Transductive Settings. 22
- D.3 AP for Link Prediction 22
- D.4 More Parameter Analysis 23
- D.5 Static Analysis of False Negative Samples 23
- D.6 Analysis of Time and Memory Overhead. 25
- D.7 More Ablation Stucy 25
- D.8 Analysis on theorem 26
- D.9 Analysis on Other Negative Sampling Strategy. 27

D.10 Analysis on Effect of Adaption. 28

E Limitations and Broader Impacts **28**

E.1 Limitations 28

E.2 Broader Impacts 29

A PROOFS

A.1 ADADGL IMPROVES COVERAGE DURING TRAINING.

Assumption. (1). The exponential function is positive (Xu et al., 2020).

Proof. From Definition 2, the target distribution is defined as a softmax over all nodes in $\mathcal{V}$:

$$p_{\text{target}}(v^- \mid v) = \frac{\exp(\text{sim}(v^+, v^-))}{\sum_{u \in \mathcal{V}} \exp(\text{sim}(v^+, u))}.$$

Since the exponential function satisfies $\exp(x) > 0$ for any $x \in \mathbb{R}$, we have:

$$\exp(\text{sim}(v^+, u)) > 0, \quad \forall u \in \mathcal{V}.$$

Thus, for every positive node $v^+ \in \mathcal{P}_{\text{train}}$:

$$\sum_{v^+ \in \mathcal{P}_{\text{train}}} \frac{|\{u \in V \mid p(u \mid v^+) = 0\}|}{|V|}.$$

Summing over all positive nodes $\mathcal{P}_{\text{train}}$ yields:

$$\text{Coverage}(p_{\text{target}}) = \sum_{v^+ \in \mathcal{P}_{\text{train}}} 0 = 0.$$

This completes the proof. □

A.2 ADADGL IMPROVES HARDNESS DURING TRAINING.

Assumption. (1). Negative nodes are sampled independently and identically distributed (i.i.d.) from the full node set. (Gupta & Bedathur, 2024)
(2). By the law of large numbers, the empirical average of the exponentiated similarity scores over the candidate set converges almost surely (a.s.) to the theoretical expectation over the full node set. (Gao et al., 2024)

Proof. We analyze the asymptotic behavior of DNS and random negative sampling with respect to the target distribution p_{target}.

Step 1: Convergence of DNS to target distribution.

We consider the target negative sampling distribution defined in Definition 2:

$$p_{\text{target}}(v^- \mid v^+) = \frac{\exp(\text{sim}(v^+, v^-))}{\sum_{u \in \mathcal{V}} \exp(\text{sim}(v^+, u))}.$$

DNS approximates the full node set $\mathcal{V}$ using a candidate set $\mathcal{C} \subset \mathcal{V}$, sampled i.i.d. with full support over $\mathcal{V}$.

We define the empirical partition function:

$$\hat{Z}(v^+) = \frac{1}{|\mathcal{C}|} \sum_{u \in \mathcal{C}} \exp(\text{sim}(v^+, u)).$$

(i) Law of Large Numbers.

Since $\mathcal{C}$ is i.i.d. sampled from a full-support distribution over $\mathcal{V}$, by the Law of Large Numbers (Here, π represents the node sampling distribution, and the expectation $\mathbb{E}_{v \sim \pi}$ is taken over all candidate nodes.) :

$$\hat{Z}(v^+) \xrightarrow{a.s.} \mathbb{E}_{u \sim \pi}[\exp(\text{sim}(v^+, u))].$$

where a.s. abbreviates almost surely. This convergence denotes $\hat{Z}(v^+)$ is an unbiased estimator of $\sum_{u \in \mathcal{V}} \exp(\text{sim}(v^+, u))$.

(ii) Embedding convergence.

Under dynamic graph model training dynamics:

$$\mathrm{sim}(v^+, u) \to \mathrm{sim}^*(v^+, u), \quad \forall u \in \mathcal{V}.$$

Thus:

$$\hat{Z}(v^+) \to Z^*(v^+), \quad Z^*(v^+) = \sum_{u \in \mathcal{V}} \exp(\mathrm{sim}(v^+, u)).$$

(iii) Distribution convergence.

Therefore, for every node:

$$p_{\text{DNS}}(v^-, v^+) \to p_{\text{target}}(v^-, v^+).$$

Hence DNS converges pointwise to the target distribution.

Step 2: Limitation of uniform random negative sampling.

For a positive node v^+, let $\mathcal{V}^-(v^+)$ denote its valid negative-node set and let $N_{v^+} = |\mathcal{V}^-(v^+)|$. Uniform random negative sampling is defined as

$$p_{\text{random}}(v_i^- \mid v^+) = \frac{1}{N_{v^+}}, \qquad \forall v_i^- \in \mathcal{V}^-(v^+).$$

Let

$$s_i = \mathrm{sim}(v^+, v_i^-),$$

and recall that the target distribution is

$$p_{\text{target}}(v_i^- \mid v^+) = \frac{\exp(s_i)}{\sum_{j=1}^{N_{v^+}} \exp(s_j)}.$$

Then, the KL divergence between uniform random sampling and the target distribution is

$$\begin{aligned}
&\mathrm{KL}\big(p_{\text{random}}(\cdot \mid v^+) \,\big\|\, p_{\text{target}}(\cdot \mid v^+)\big) \\
&= \sum_{i=1}^{N_{v^+}} \frac{1}{N_{v^+}} \log \frac{1/N_{v^+}}{\exp(s_i)/\sum_{j=1}^{N_{v^+}} \exp(s_j)} \\
&= \log\left(\frac{1}{N_{v^+}} \sum_{j=1}^{N_{v^+}} \exp(s_j)\right) - \frac{1}{N_{v^+}} \sum_{i=1}^{N_{v^+}} s_i.
\end{aligned}$$

By Jensen's inequality,

$$\log\left(\frac{1}{N_{v^+}} \sum_{j=1}^{N_{v^+}} \exp(s_j)\right) \geq \frac{1}{N_{v^+}} \sum_{i=1}^{N_{v^+}} s_i.$$

Therefore,

$$\mathrm{KL}\big(p_{\text{random}}(\cdot \mid v^+) \,\big\|\, p_{\text{target}}(\cdot \mid v^+)\big) \geq 0.$$

Moreover, equality holds if and only if

$$\mathrm{sim}(v^+, v_1^-) = \mathrm{sim}(v^+, v_2^-) = \cdots = \mathrm{sim}(v^+, v_{N_{v^+}}^-).$$

Thus, if the similarity scores of valid negative nodes are non-uniform, then

$$\mathrm{KL}\big(p_{\text{random}}(\cdot \mid v^+) \,\big\|\, p_{\text{target}}(\cdot \mid v^+)\big) > 0.$$

Step 3: Conclusion.

Combining Step 2 and Step 3:

$$\mathrm{KL}(p_{\text{DNS}} \,\|\, p_{\text{target}}) < \mathrm{KL}(p_{\text{random}} \,\|\, p_{\text{target}}).$$

This completes the proof. □

A.3 AdaDGL Improves Hardness at Inference.

Assumption. (1). The exponential function is positive (Xu et al., 2020).

Proof. **Step 1: Reformulation of method testing rank.**

Based on Definition 2, given an observed edge e_k, the method testing rank is defined as:

$$R_{\text{method}} = \frac{1}{K}\sum_{k=1}^{K}\mathbb{E}[R(e_k)].$$

Since all edges are sampled i.i.d. at inference, we can write:

$$R_{\text{method}} = \mathbb{E}[R_{v^-}],$$

where the expectation is taken over the negative sampling distribution induced by the method.

Step 2: Expected rank under uniform random sampling.

For the random baseline, the selected node v^- is uniformly distributed over $\mathcal{N}_{e_k}$.

Since $R_{v^-} = \frac{f(v^-)}{|\mathcal{N}_{e_k}|}$ and all positions in the ranking are equally likely, we have:

$$\mathbb{E}[R_{v^-}] = \frac{1}{|\mathcal{N}_{e_k}|}\sum_{r=1}^{|\mathcal{N}_{e_k}|}\frac{r}{|\mathcal{N}_{e_k}|} = \frac{1}{|\mathcal{N}_{e_k}|^2}\cdot\frac{|\mathcal{N}_{e_k}|(|\mathcal{N}_{e_k}|+1)}{2}.$$

Thus:

$$R_{\text{random}} = \frac{(|\mathcal{N}_{e_k}|+1)}{2|\mathcal{N}_{e_k}|}.$$

Step 3: Expected rank under SNS.

SNS uniformly samples without replacement ϵN_k candidates from $\mathcal{N}_{e_k}$ and selects the candidate with the highest similarity, i.e., the candidate with the minimum rank. For simplicity, we assume that ϵN_k is an integer. Let

$$M_k = \min_{v\in\mathcal{C}_k} f_k(v)$$

denote the minimum rank among the sampled candidates.

For $1 \le r \le N_k - \epsilon N_k + 1$, the event $M_k \ge r$ means that all ϵN_k sampled candidates have ranks in $\{r, r+1, \ldots, N_k\}$. Therefore,

$$\Pr(M_k \ge r) = \frac{\binom{N_k-r+1}{\epsilon N_k}}{\binom{N_k}{\epsilon N_k}}.$$

Using the tail-sum formula and the hockey-stick identity, we obtain

$$\begin{aligned}
\mathbb{E}[M_k] &= \sum_{r=1}^{N_k-\epsilon N_k+1}\Pr(M_k \ge r) \\
&= \frac{\sum_{r=1}^{N_k-\epsilon N_k+1}\binom{N_k-r+1}{\epsilon N_k}}{\binom{N_k}{\epsilon N_k}} \\
&= \frac{\binom{N_k+1}{\epsilon N_k+1}}{\binom{N_k}{\epsilon N_k}} \\
&= \frac{N_k+1}{\epsilon N_k+1}.
\end{aligned}$$

Thus, the expected normalized rank under SNS is

$$\begin{aligned}
R^{(k)}_{\text{SNS}} &= \mathbb{E}\left[\frac{M_k}{N_k}\right] \\
&= \frac{N_k+1}{N_k(\epsilon N_k+1)}.
\end{aligned}$$

Step 4: Comparison.

The difference between the expected normalized ranks of random sampling and SNS is

$$R_{\text{random}}^{(k)} - R_{\text{SNS}}^{(k)} = \frac{N_k + 1}{2N_k} - \frac{N_k + 1}{N_k(\epsilon N_k + 1)} = \frac{(N_k + 1)(\epsilon N_k - 1)}{2N_k(\epsilon N_k + 1)}.$$

Since $\epsilon \in (\frac{1}{|N_k|}, 1]$, we have

$$R_{\text{random}}^{(k)} - R_{\text{SNS}}^{(k)} > 0,$$

which implies

$$R_{\text{SNS}}^{(k)} < R_{\text{random}}^{(k)}.$$

□

A.4 AdaDGL Improves Coverage at Inference.

Assumption. (1). Negative nodes are sampled independently and identically distributed (i.i.d.) from the full node set. (Gupta & Bedathur, 2024)
(2). By the law of large numbers, the empirical average of the exponentiated similarity scores over the candidate set converges almost surely (a.s.) to the theoretical expectation over the full node set. (Gao et al., 2024)

Proof. We prove the theorem by analyzing the support of the SNS sampling distribution.

From Definition 6, we show that for SNS, the sampling distribution satisfies full support over $\mathcal{V}$.

From the construction of SNS (Section 3.2), negative samples are drawn from a candidate set that is dynamically updated and asymptotically covers the entire node set:

$$\mathcal{C} \to \mathcal{V}.$$

Since the exponential function is strictly positive, it follows that:

$$p(u, v^+) > 0, \quad \forall u \in \mathcal{V}.$$

Therefore, the zero-probability set is empty:

$$\{u \in \mathcal{V} \mid p(u, v^+) = 0\} = \emptyset.$$

Thus:

$$\text{Coverage}(\text{SNS}) = 0.$$

This completes the proof. □

A.5 AdaDGL Improves Model Performance by Adaptation.

Assumption. (1) The prediction head is optimized on the target distribution via ERM. (Papa et al., 2016)
(2) The loss function is differentiable and satisfies standard regularity conditions(Huang et al., 2023)
(3) The empirical risk is a consistent estimator of the expected risk. These assumptions are commonly used in statistical machine learning theory(Vapnik, 1998).

Proof. **Step 1: Fixed representation induced by Dynamic Graph Model.**

Let $h(x)$ denote the representation learned by the dynamic graph model backbone. During inference adaptation, the backbone is frozen, and only the prediction head is updated.

Thus, prediction is fully determined by an MLP:

$$f_\theta(x) = \text{MLP}_\theta(h(x)).$$

Step 2: Without adaptation.

Without adaptation, the parameter θ is optimized on a general negative distribution p_{train}, leading to a suboptimal decision function for the downstream distribution p_{infer}:

$$\theta = \arg\min_{\theta} \mathbb{E}_{p_{\text{train}}}[\mathcal{L}(f_\theta(x), y)].$$

Thus, the induced classifier is biased toward the training distribution.

Step 3: With adaptation.

With adaptation, the MLP is further optimized on the downstream distribution:

$$\theta' = \arg\min_{\theta} \mathbb{E}_{p_{\text{infer}}}[\mathcal{L}(f_\theta(x), y)].$$

Since the representation $h(x)$ is fixed, this reduces to learning a better decision boundary in a similar space.

Step 4: Risk comparison.

We define the expected risk under the downstream distribution p_{infer} as

$$\mathcal{R}_{\text{infer}}(\theta) := \mathbb{E}_{p_{\text{infer}}}[\mathcal{L}(f_\theta(x), y)].$$

Since θ' is obtained by optimizing the MLP on the downstream distribution, by the definition of empirical risk minimization we have

$$\mathbb{E}_{p_{\text{infer}}}[\mathcal{L}(f_\theta(x), y)] = \mathcal{R}_{\text{infer}}(\theta') = \min_{\theta} \mathcal{R}_{\text{infer}}(\theta) \leq \mathcal{R}_{\text{infer}}(\theta) = \mathbb{E}_{p_{\text{infer}}}[\mathcal{L}(f_\theta(x), y)],$$

where θ is the parameter learned on the training distribution p_{train}.

For 0-1 loss, lower expected risk directly corresponds to higher accuracy, hence

$$P_w \geq P_{w/o}.$$

This completes the proof.

□

B Time and Space Complexity Computation

In this section, we detail time and space complexity computation in Sections 3.1 and 3.2.

B.1 Time and Space Complexity Computation of AdaDGL during Training

Complexity Analysis. Consider a batch of m edges during training. For each positive edge, DNS constructs a candidate set of negative samples from the historical node pool. Let c denote the number of candidate negatives per positive edge. *Time complexity.* Computing similarity scores between the positive edge and c candidate negatives requires $O(c)$ operations per edge. For m edges in a batch, this results in $O(n \cdot c) = O(m)$ time complexity, since c is treated as a small constant relative to m. Updating the sampling probabilities for each candidate negative also scales linearly with c, hence it does not change the overall $O(m)$ scaling.

Space complexity. Storing embeddings and intermediate scores for c candidates per edge requires $O(c)$ memory per edge, leading to $O(n \cdot c) = O(n)$ total memory for the batch.

In contrast, directly sampling from the full target distribution over all negatives would require computing and storing scores for $O(|m|)$ positives against $O(|V|)$ potential negatives, resulting in $O(|V| \cdot |m|)$ time and space complexity.

B.2 Time and Space Complexity Computation of AdaDGL at Inference.

Complexity Analysis. Let n denote the total number of nodes in the graph. Both conventional random negative sampling and the AdaDGL method have linear time and space complexity, $O(n)$. For each node, we only sample $\epsilon \times n$ negative candidates and compute their similarity scores. ϵ is a small constant. Since the number of computations per node does not depend on n, the total time across all nodes scales linearly with n. Similarly, storing embeddings and intermediate scores also requires $O(n)$ space. Thus the time and space complexity is $O(n)$.

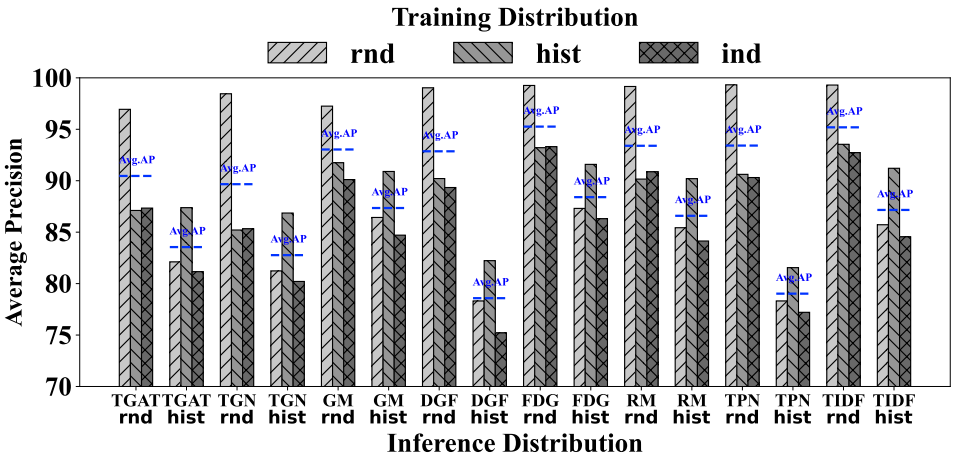


Figure 7: Average Precision on Wikipedia in Transductive Settings across Different Models.

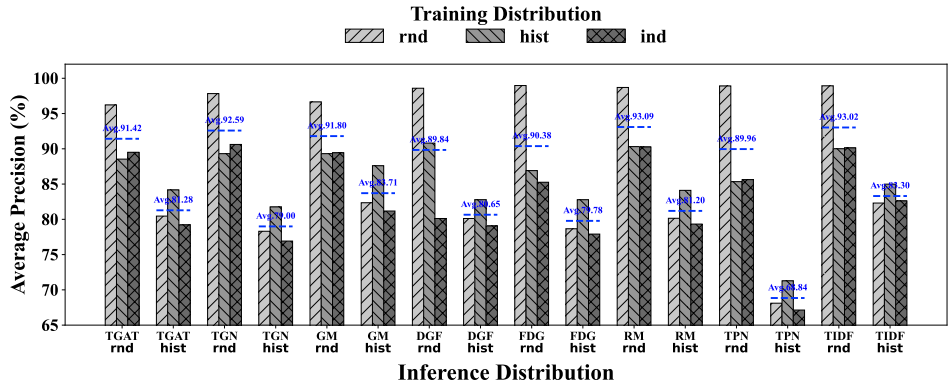


Figure 8: Average Precision on Wikipedia in Inductive Settings across Different Models.

Table 4: Statistics of Seven Real-world Datasets.

Datasets	Domains	Nodes	Edges	Dim. of Node &Edge Feat	Bipartite	Duration	Unique Steps
Wikipedia	Social	9,227	157,474	– & 172	True	1 month	152,757
Reddit	Social	10,984	672,447	– & 172	True	1 month	669,065
MOOC	Interaction	7,144	411,749	– & 4	True	1 month	345,600
LastFM	Interaction	1,980	1,293,103	– & –	True	4.3 years	1,283,614
Enron	Social	184	125,235	– & –	False	3.6 years	22,632
Social Evo.	Proximity	74	2,099,519	– & 2	False	8 months	565,932
UCI	Social	1,899	59,835	– & –	False	196 days	58,911

C Additional Datasets and Baselines Discussion

C.1 Statistics of Datasets

We conduct experiments on seven dynamic graph datasets, the detailed statistics of seven real-world datasets are illustrated in Table 4. The specific dataset details are as follows:

- **Wikipedia**: A bipartite interaction graph capturing user edits on Wikipedia pages over one month. Nodes correspond to users and pages, while edges represent timestamped editing actions. Each edge is annotated with a 172-dimensional feature vector (Pennebaker et al., 2001), and dynamic labels indicate whether users were temporarily banned from editing.

- **Reddit**: A bipartite graph recording user posts in subreddits over one month. Nodes represent users and subreddits, and edges denote timestamped posting events. Each edge includes a 172-dimensional LIWC feature vector. Dynamic labels specify whether users were banned from posting.
- **MOOC**: A bipartite interaction network between students and course content units (e.g., videos or problem sets). Edges indicate student access to specific content and carry a 4-dimensional feature vector.
- **LastFM**: A bipartite graph describing the listening behavior of users for different songs over a period of 4.3 years. Users and songs are nodes, and edges correspond to listening interactions.
- **Enron**: An email communication network capturing exchanges among employees of the Enron energy corporation over a three-year period.
- **Social**: A mobile phone proximity network tracking the daily interactions of an entire undergraduate dormitory for eight months. Each edge includes a 2-dimensional feature vector.
- **UCI**: An online communication network among university students, where nodes represent students and edges denote messages exchanged.

C.2 Statistics of Baselines

We select eight representative models from three categories of dynamic graph learning methods, including Attention-based, RNN-based, and MLP-based approaches:

- **Attention-based Models:**
  - **TGAT**: TGAT computes node representations by aggregating features from temporal neighbors via a self-attention mechanism, and introduces a time encoding function to capture continuous-time dynamics.
  - **DyGFormer (DGF)**: DyGFormer models the correlations between source and target nodes through a neighbor co-occurrence encoding scheme based on historical interaction sequences, and employs a patching mechanism to capture long-range temporal dependencies.
  - **TIDFormer (TIDF)**: TIDFormer incorporates temporal interval encoding and dynamic attention mechanisms to model evolving temporal dependencies.
- **RNN-based Models:**
  - **TGN**: TGN maintains a dynamic memory for each node, which is updated upon interactions via a message function, message aggregator, and memory updater, together with an embedding module for temporal representation learning.
- **MLP-based Models:**
  - **GraphMixer (GM)**: GraphMixer incorporates fixed time encoding into an MLP-Mixer-based architecture to model temporal interactions, along with a node encoder using neighbor mean-pooling.
  - **RepeatMixer (RM)**: RepeatMixer leverages mixer-based token and channel mixing operations to capture repeated interaction patterns and temporal dependencies.
  - **TPNet (TPN)**: TPNet adopts a temporal point process framework to model interaction times and integrates temporal intensity with representation learning.
  - **FreeDyG (FDG)**: FreeDyG is a dynamic graph learning framework that focuses on flexible and efficient temporal representation learning without relying on heavy memory modules, enabling scalable modeling of evolving node interactions.

Due to space constraints, we focus on relatively recent and representative models, omitting some earlier approaches (e.g., JODIE and DyRep) to better reflect current research trends.

C.3 Discussion about DTDG

DTDG models are designed for analyzing discrete-time dynamic graphs (i.e., snapshot-based graphs), where the graph evolution is represented as a sequence of discrete snapshots. Recent representative

Table 5: Searched Ranges of Hyperparameters and the Related Methods.

Hyperparameters	**Searched Ranges**
Number of Sampled First-order Neighbors	[16, 24, 32, 40, 48, 56, 64]
Neighbor Sampling Strategies	[recent, uniform]
Dropout Rate	[0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6]
Weight Decay	[0.0, 1e-6, 1e-4]
Learning Rate	[1e-3, 1e-4, 1e-5, 1e-6]
Dimension of Hidden Features	[8, 16, 32, 64, 128,256]

works include EvolveGCN (Pareja et al., 2020), Roland (You et al., 2022), DyFormer (Cong et al., 2023a), SimpleDyG (Wu et al., 2024), TopDyG (Zhang et al., 2025). Nevertheless, compared with CTDG methods, DTDG modeling fails to capture fine-grained temporal information (Zheng et al., 2025; Feng et al., 2026) inherent in dynamic graphs.

D Experimental details

D.1 Hardware and Parameters Setup.

Hardware Experiments are conducted on a server equipped with a 2 $\times$ 64 AMD EPYC 7763 64-Core Processor @ 2.45 GHz, 256GB RAM, and 1 $\times$ NVIDIA GeForce RTX 3090.

Parameters Setup We save the model that achieves the best results on the validation set for final testing. Moreover, we conduct a grid search to identify the optimal settings for several critical hyperparameters with the random negative sampling strategy. The search ranges and relevant hyperparameters are detailed in Table 4. To improve training efficiency, we process all datasets using a batch operation with a batch size of 200. To ensure the accuracy of the experimental results, we conduct five runs of experiments using random seeds and report the final average values

D.2 AP of Different Models on Different Datasets under Inductive and Transductive Settings.

As shown in Figures 7 and 8, we further show that constructing robust training negative samples and testing with unknown target negative distributions are challenging. These results show that all existing negative sampling distributions fail to achieve both discriminability and generalization, resulting in limited model capability and poor performance across different negative distributions.

D.3 AP for Link Prediction

Transductive Setting. As shown in Tables 6, Table 7, Table 8 and Table 9. AdaDGL consistently demonstrates strong performance across different negative sampling distributions. Specifically, under the historical distribution, AdaTIDF improves AP over TIDF by 3.77% on UCI and reduces the average rank from 3.43 to 1.57. Under the inductive distribution, AdaTIDF improves LastFM from 75.69% to 85.40% (+9.71%) and achieves the best average rank of 1.00, compared with 3.43 for TIDF. Under SNS, AdaTIDF improves MOOC and UCI by 5.62% and 5.05%, respectively, while reducing the average rank from 3.86 to 1.14. A similar improvement is observed under the random distribution, where AdaTIDF improves MOOC by 1.05% and reduces the average rank from 3.14 to 1.29. For weaker cases, AdaTGN shows relatively smaller gains, e.g., improving from 95.13% to 95.92% on Social under the inductive distribution and from 93.57% to 93.68% under the random distribution. Nevertheless, the overall trend remains positive across different negative sampling distributions, demonstrating the robustness of AdaDGL in the transductive setting.

Inductive Setting. As shown in Table 10, Table 11, Table 12, and Table 13. AdaDGL consistently demonstrates superior performance in inductive settings. Under the historical distribution, AdaTIDF improves over TIDF by 8.94% on Reddit and 6.22% on Enron, while reducing the average rank from 3.43 to 1.14. Under the inductive distribution, the improvements reach 9.07% on Reddit and 6.40% on Enron, with the average rank improving from 3.57 to 1.14. Under the more challenging SNS distribution, AdaTIDF improves Reddit from 65.06% to 70.69% (+5.63%) and UCI from 79.62% to 85.49% (+5.87%), while reducing the average rank from 4.71 to 1.43. In comparison, improvements

on weaker backbones are more modest; for example, AdaTGN improves Reddit from 64.85% to 66.72% under the historical distribution and from 64.84% to 66.71% under the inductive distribution. These results indicate that although the magnitude of improvement varies across backbone models, AdaDGL generally provides consistent gains in inductive link prediction.

SNS provides a more challenging and discriminative evaluation protocol than conventional negative sampling. For example, in the transductive setting, the AP of TIDF decreases from 99.30% to 75.31% on Wikipedia and from 99.38% to 67.16% on Reddit when replacing random sampling with SNS. Similarly, AdaTIDF decreases from 99.34% to 78.33% on Wikipedia and from 99.49% to 72.15% on Reddit. These substantial performance drops indicate that SNS introduces considerably harder negative samples and avoids the overly optimistic performance obtained with easily distinguishable negatives.

Table 6: AP (%) for Transductive Link Prediction with Historical Negative Sampling Strategy. The Best and Second-Best Results are Emphasized by **Bold** and Underlined Fonts.

Methods	Wikipedia	Reddit	MOOC	LastFM	Enron	Social	UCI	Avg. Rank
TGAT	87.38 ± 0.22	79.55 ± 0.20	82.19 ± 0.62	71.59 ± 0.24	64.07 ± 1.05	95.01 ± 0.44	68.27 ± 1.37	10.00
TGN	86.86 ± 0.33	81.22 ± 0.61	87.06 ± 1.93	76.87 ± 4.64	73.91 ± 1.76	94.45 ± 0.56	80.43 ± 2.12	9.14
GM	90.90 ± 0.10	78.44 ± 0.18	77.77 ± 0.92	72.47 ± 0.49	77.98 ± 0.92	94.93 ± 0.31	84.11 ± 1.35	8.71
DGF	82.23 ± 2.54	81.57 ± 0.67	85.85 ± 0.66	81.57 ± 0.48	75.63 ± 0.73	97.38 ± 0.14	82.17 ± 0.82	8.00
FDG	91.59 ± 0.57	85.67 ± 1.01	86.71 ± 0.81	79.71 ± 0.51	78.87 ± 0.82	97.79 ± 0.23	86.10 ± 1.19	5.00
RM	90.20 ± 1.04	83.02 ± 1.20	92.19 ± 0.58	86.73 ± 0.34	87.38 ± 0.18	97.40 ± 0.15	87.23 ± 0.23	4.14
TPN	81.55 ± 4.10	81.02 ± 1.31	92.69 ± 0.95	87.74 ± 0.50	80.79 ± 1.68	96.80 ± 0.41	86.34 ± 0.80	6.14
TIDF	91.21 ± 0.78	83.27 ± 1.01	95.37 ± 1.09	88.68 ± 0.57	81.59 ± 1.17	97.39 ± 0.67	89.57 ± 1.17	3.43
AdaTIDF	92.25 ± 0.76	84.07 ± 0.99	$\mathbf{96.87 \pm 1.01}$	$\mathbf{89.97 \pm 0.55}$	$\mathbf{87.62 \pm 1.19}$	98.17 ± 0.92	$\mathbf{93.34 \pm 0.72}$	**1.57**
AdaTGN	88.11 ± 0.29	82.34 ± 0.56	89.08 ± 1.62	78.32 ± 2.52	76.22 ± 1.66	96.71 ± 0.42	83.76 ± 0.83	7.29
AdaFDG	$\mathbf{92.31 \pm 0.42}$	$\mathbf{87.61 \pm 1.04}$	90.62 ± 0.76	82.71 ± 0.56	81.64 ± 0.75	$\mathbf{98.62 \pm 0.31}$	90.13 ± 1.01	2.57

Table 7: AP (%) for Transductive Link Prediction with Inductive Negative Sampling Strategy. The Best and Second-Best Results are Emphasized by **Bold** and Underlined Fonts.

Methods	Wikipedia	Reddit	MOOC	LastFM	Enron	Social	UCI	Avg. Rank
TGAT	87.00 ± 0.16	89.59 ± 0.24	75.95 ± 0.64	71.13 ± 0.17	63.94 ± 1.36	94.84 ± 0.44	68.67 ± 0.84	9.43
TGN	85.62 ± 0.44	88.10 ± 0.24	77.50 ± 2.91	65.95 ± 5.98	70.89 ± 2.72	95.13 ± 0.56	70.94 ± 0.71	9.71
GM	88.59 ± 0.17	85.26 ± 0.11	74.27 ± 0.92	68.12 ± 0.33	75.01 ± 0.79	94.72 ± 0.33	80.10 ± 0.51	9.14
DGF	78.29 ± 5.38	91.11 ± 0.40	81.24 ± 0.69	73.97 ± 0.50	77.41 ± 0.89	97.68 ± 0.10	72.25 ± 1.71	6.93
FDG	90.05 ± 0.79	90.74 ± 0.17	83.01 ± 0.87	72.19 ± 0.24	77.81 ± 0.65	97.57 ± 0.15	82.35 ± 0.73	5.57
RM	88.86 ± 0.97	91.11 ± 0.73	83.11 ± 1.28	75.46 ± 0.78	83.17 ± 0.50	97.52 ± 0.28	84.20 ± 0.34	4.50
TPN	79.35 ± 5.52	88.19 ± 0.33	88.18 ± 0.97	77.99 ± 1.30	75.36 ± 1.81	97.35 ± 0.32	77.26 ± 1.57	6.43
TIDF	91.15 ± 1.17	91.57 ± 0.50	84.53 ± 1.11	75.69 ± 1.17	79.67 ± 0.66	97.70 ± 0.36	85.81 ± 1.39	3.43
AdaTIDF	$\mathbf{92.34 \pm 1.14}$	$\mathbf{92.46 \pm 0.62}$	$\mathbf{88.49 \pm 1.06}$	$\mathbf{85.40 \pm 0.96}$	$\mathbf{83.24 \pm 0.62}$	$\mathbf{98.50 \pm 0.24}$	$\mathbf{87.78 \pm 0.86}$	**1.00**
AdaTGN	88.64 ± 0.33	89.16 ± 0.26	78.42 ± 1.86	75.33 ± 2.85	74.78 ± 1.32	95.92 ± 0.44	73.11 ± 0.71	7.57
AdaFDG	92.10 ± 1.07	91.66 ± 0.43	87.62 ± 1.05	80.25 ± 0.57	80.46 ± 0.54	98.14 ± 0.17	86.34 ± 0.17	2.29

Table 8: AP (%) for Transductive Link Prediction with SNS Negative Sampling Strategy. The Best and Second-Best Results are Emphasized by **Bold** and Underlined Fonts.

Methods	Wikipedia	Reddit	MOOC	LastFM	Enron	Social	UCI	Avg. Rank
TGAT	65.21 ± 0.66	60.64 ± 0.31	61.72 ± 0.82	65.21 ± 0.72	62.09 ± 0.33	75.91 ± 0.54	62.71 ± 0.52	10.86
TGN	66.71 ± 0.83	60.67 ± 0.33	65.84 ± 2.21	67.86 ± 2.71	67.62 ± 0.35	76.52 ± 0.42	69.21 ± 0.96	9.71
GM	68.61 ± 0.42	60.41 ± 0.57	67.46 ± 1.12	66.34 ± 0.87	70.21 ± 0.44	78.42 ± 0.41	72.14 ± 0.58	9.43
DGF	73.18 ± 0.52	65.78 ± 0.42	68.41 ± 0.52	74.82 ± 0.71	73.81 ± 0.57	79.44 ± 0.71	73.31 ± 0.55	7.14
FDG	75.81 ± 0.83	69.71 ± 0.92	71.77 ± 0.42	80.62 ± 0.75	81.36 ± 0.61	84.72 ± 0.81	79.89 ± 0.71	4.43
RM	74.73 ± 0.76	64.58 ± 0.72	70.36 ± 0.58	73.62 ± 1.45	83.96 ± 0.78	85.41 ± 0.34	81.92 ± 0.46	5.00
TPN	77.81 ± 0.65	66.86 ± 0.53	74.63 ± 0.58	80.45 ± 0.71	79.66 ± 0.71	84.11 ± 0.75	81.65 ± 0.74	4.86
TIDF	75.31 ± 0.83	67.16 ± 0.98	73.66 ± 1.83	80.82 ± 0.81	80.63 ± 1.82	85.66 ± 1.08	81.72 ± 0.83	3.86
AdaTIDF	78.33 ± 0.82	$\mathbf{72.15 \pm 0.91}$	$\mathbf{79.28 \pm 1.31}$	$\mathbf{85.14 \pm 0.76}$	$\mathbf{84.87 \pm 1.22}$	$\mathbf{87.65 \pm 1.06}$	$\mathbf{86.77 \pm 0.82}$	**1.14**
AdaTGN	72.42 ± 0.75	63.44 ± 0.31	70.62 ± 1.76	72.03 ± 2.85	73.70 ± 0.21	84.18 ± 0.32	75.91 ± 0.67	7.29
AdaFDG	$\mathbf{78.96 \pm 0.76}$	72.04 ± 0.83	76.47 ± 1.34	83.69 ± 0.62	82.98 ± 0.83	85.34 ± 0.64	84.92 ± 0.81	2.29

D.4 More Parameter Analysis

We conduct experiments by varying parameters τ and k introduced in Section 3.1 and parameter ϵ introduced in Section 3.2 in inductive settings. We could get similar conclusions as shown in 4.5.

D.5 Static Analysis of False Negative Samples

We conduct an analysis on False Negative Samples in Table 14. This experiment shows our DNS sampling and SNS sampling successfully exclude all false negative nodes.

Table 9: AP (%) for Transductive Link Prediction with Random Negative Sampling Strategy. The Best and Second-Best Results are Emphasized by **Bold** and Underlined Fonts.

Methods	Wikipedia	Reddit	MOOC	LastFM	Enron	Social	UCI	Avg. Rank
TGAT	96.94 ± 0.06	98.52 ± 0.02	85.84 ± 0.15	73.42 ± 0.21	71.12 ± 0.97	93.16 ± 0.17	79.63 ± 0.70	10.71
TGN	98.45 ± 0.06	98.63 ± 0.06	89.15 ± 1.60	77.07 ± 3.97	86.53 ± 1.11	93.57 ± 0.17	92.34 ± 1.04	9.00
GM	97.25 ± 0.03	97.31 ± 0.01	82.78 ± 0.15	75.61 ± 0.24	82.25 ± 0.16	93.37 ± 0.07	93.25 ± 0.57	10.00
DGF	99.03 ± 0.02	99.22 ± 0.01	87.52 ± 0.49	93.00 ± 0.12	92.47 ± 0.12	94.73 ± 0.01	95.79 ± 0.17	6.71
FDG	99.26 ± 0.01	99.48 ± 0.01	89.61 ± 0.19	92.15 ± 0.16	92.51 ± 0.05	94.91 ± 0.01	96.28 ± 0.11	5.29
RM	99.16 ± 0.02	99.22 ± 0.01	92.76 ± 0.10	94.14 ± 0.06	92.66 ± 0.07	94.72 ± 0.02	96.74 ± 0.08	5.50
TPN	99.32 ± 0.03	99.27 ± 0.00	**96.39 ± 0.09**	94.50 ± 0.08	92.90 ± 0.17	94.73 ± 0.02	97.35 ± 0.04	3.43
TIDF	99.30 ± 0.03	99.38 ± 0.03	93.42 ± 0.16	94.60 ± 0.17	93.17 ± 0.06	94.76 ± 0.06	97.44 ± 0.10	3.07
AdaTIDF	**99.34 ± 0.03**	99.49 ± 0.05	94.47 ± 0.16	**94.81 ± 0.15**	**93.31 ± 0.08**	**94.98 ± 0.05**	**97.60 ± 0.66**	**1.29**
AdaTGN	98.61 ± 0.05	98.71 ± 0.07	89.94 ± 0.31	77.17 ± 3.65	86.72 ± 1.05	93.68 ± 0.15	93.12 ± 0.58	7.86
AdaFDG	99.32 ± 0.01	**99.52 ± 0.01**	93.61 ± 0.21	92.35 ± 0.15	93.01 ± 0.03	94.95 ± 0.02	97.31 ± 0.61	3.14

Table 10: AP (%) for Inductive Link Prediction with Historical Negative Sampling Strategy. The Best and Second-Best Results are Emphasized by **Bold** and Underlined Fonts.

Methods	Wikipedia	Reddit	MOOC	LastFM	Enron	Social	UCI	Avg. Rank
TGAT	84.17 ± 0.22	63.47 ± 0.36	76.73 ± 0.29	76.27 ± 0.25	61.40 ± 1.31	93.97 ± 0.54	70.52 ± 0.93	9.29
TGN	81.76 ± 0.32	64.85 ± 0.85	77.07 ± 3.41	66.65 ± 6.11	62.91 ± 1.16	90.66 ± 1.62	70.78 ± 0.78	9.71
GM	87.60 ± 0.30	64.50 ± 0.26	74.00 ± 0.97	76.42 ± 0.22	72.37 ± 1.37	94.01 ± 0.47	81.66 ± 0.49	7.29
DGF	71.42 ± 4.43	65.37 ± 0.60	80.82 ± 0.30	76.35 ± 0.52	67.07 ± 0.62	96.82 ± 0.16	72.13 ± 1.87	7.57
FDG	82.78 ± 0.30	66.02 ± 0.41	81.63 ± 0.33	77.28 ± 0.21	73.01 ± 0.88	96.69 ± 0.14	82.35 ± 0.39	6.00
RM	84.11 ± 1.88	66.14 ± 1.40	83.19 ± 1.72	81.12 ± 0.30	82.86 ± 0.47	96.75 ± 0.31	85.52 ± 0.16	4.29
TPN	71.28 ± 4.33	62.15 ± 1.72	81.85 ± 1.60	82.27 ± 1.22	74.60 ± 1.35	96.38 ± 0.18	78.48 ± 1.18	7.00
TIDF	84.99 ± 1.89	65.62 ± 0.92	84.61 ± 1.17	82.79 ± 0.83	76.99 ± 1.21	96.77 ± 0.19	86.11 ± 1.81	3.43
AdaTIDF	**88.61 ± 1.68**	**74.56 ± 0.64**	**87.69 ± 1.09**	**83.80 ± 0.65**	**83.21 ± 1.19**	97.02 ± 0.17	**87.10 ± 1.65**	**1.14**
AdaTGN	83.62 ± 0.27	66.72 ± 0.82	80.26 ± 2.86	68.76 ± 2.81	68.11 ± 0.93	93.14 ± 1.55	73.61 ± 0.87	7.71
AdaFDG	87.84 ± 0.26	70.81 ± 0.42	84.78 ± 0.31	81.36 ± 0.71	74.86 ± 0.78	**97.16 ± 0.81**	85.62 ± 0.36	2.57

Table 11: AP (%) for Inductive Link Prediction with Inductive Negative Sampling Strategy. The Best and Second-Best Results are Emphasized by **Bold** and Underlined Fonts.

Methods	Wikipedia	Reddit	MOOC	LastFM	Enron	Social	UCI	Avg. Rank
TGAT	84.17 ± 0.22	63.40 ± 0.36	76.72 ± 0.30	76.28 ± 0.25	61.40 ± 1.30	93.97 ± 0.54	70.49 ± 0.93	9.43
TGN	81.77 ± 0.32	64.84 ± 0.84	77.07 ± 3.40	69.46 ± 4.65	62.90 ± 1.16	90.65 ± 1.62	70.73 ± 0.79	9.57
GM	87.60 ± 0.29	64.49 ± 0.25	73.99 ± 0.97	76.42 ± 0.22	72.37 ± 1.38	94.01 ± 0.47	81.64 ± 0.49	7.29
DGF	71.42 ± 4.43	65.35 ± 0.60	80.82 ± 0.30	76.35 ± 0.52	67.07 ± 0.62	96.82 ± 0.17	72.13 ± 1.86	7.57
FDG	87.54 ± 0.26	64.98 ± 0.20	81.41 ± 0.31	77.01 ± 0.43	72.85 ± 0.81	96.91 ± 0.12	82.06 ± 0.58	5.29
RM	84.10 ± 1.88	66.13 ± 1.40	83.19 ± 1.72	81.12 ± 0.30	82.87 ± 0.47	96.75 ± 0.31	85.53 ± 0.16	4.57
TPN	71.29 ± 4.33	62.14 ± 1.72	81.85 ± 1.60	82.27 ± 1.22	74.60 ± 1.35	96.38 ± 0.18	78.50 ± 1.18	7.00
TIDF	84.99 ± 1.89	65.62 ± 0.92	84.61 ± 1.17	82.79 ± 0.83	76.98 ± 1.21	96.78 ± 0.19	86.11 ± 1.81	3.57
AdaTIDF	**88.66 ± 1.68**	**74.69 ± 0.64**	**87.62 ± 1.09**	**83.67 ± 0.67**	**83.38 ± 1.19**	97.08 ± 0.19	**87.16 ± 1.67**	**1.14**
AdaTGN	83.63 ± 0.31	66.71 ± 0.83	80.31 ± 2.76	68.77 ± 2.83	68.62 ± 0.88	93.19 ± 1.62	73.82 ± 0.85	8.00
AdaFDG	87.98 ± 0.21	70.68 ± 0.43	84.76 ± 0.34	81.47 ± 0.68	74.83 ± 0.76	**97.14 ± 0.82**	85.68 ± 0.35	2.57

Table 12: AP (%) for Inductive Link Prediction with SNS Negative Sampling Strategy. The Best and Second-Best Results are Emphasized by **Bold** and Underlined Fonts.

Methods	Wikipedia	Reddit	MOOC	LastFM	Enron	Social	UCI	Avg. Rank
TGAT	63.08 ± 0.64	58.91 ± 0.29	60.72 ± 0.79	64.51 ± 0.77	61.79 ± 0.36	73.68 ± 0.57	60.96 ± 0.47	11.00
TGN	65.72 ± 0.66	59.78 ± 0.42	64.12 ± 2.11	65.37 ± 1.81	65.48 ± 0.44	74.26 ± 0.45	68.87 ± 0.31	9.86
GM	67.26 ± 0.47	61.46 ± 0.55	66.31 ± 1.08	65.06 ± 0.78	68.40 ± 0.41	76.31 ± 0.46	70.62 ± 0.48	9.14
DGF	71.69 ± 0.78	64.01 ± 0.64	67.42 ± 0.63	73.62 ± 0.77	72.18 ± 0.52	77.62 ± 0.67	71.16 ± 0.45	6.86
FDG	75.61 ± 0.84	67.81 ± 0.84	71.08 ± 0.44	79.64 ± 0.71	80.52 ± 0.75	84.61 ± 0.73	78.31 ± 0.67	3.86
RM	72.16 ± 0.83	63.11 ± 0.80	70.42 ± 0.55	72.18 ± 1.34	80.06 ± 0.56	83.61 ± 0.31	79.44 ± 0.36	5.43
TPN	77.06 ± 0.81	65.72 ± 0.64	72.96 ± 1.21	79.66 ± 0.68	78.31 ± 0.64	82.14 ± 0.76	79.12 ± 0.61	4.00
TIDF	73.61 ± 0.86	65.06 ± 0.99	72.62 ± 1.66	78.16 ± 0.71	78.05 ± 1.34	83.61 ± 1.36	79.62 ± 0.54	4.71
AdaTIDF	**76.71 ± 0.86**	**70.69 ± 0.63**	**77.62 ± 0.56**	81.61 ± 0.65	**83.22 ± 0.94**	**85.76 ± 0.83**	**85.49 ± 0.65**	**1.43**
AdaTGN	70.62 ± 0.31	61.61 ± 0.42	66.81 ± 0.55	70.06 ± 2.58	69.43 ± 2.75	76.82 ± 0.31	73.86 ± 0.42	7.86
AdaFDG	78.24 ± 0.65	70.67 ± 0.31	76.54 ± 1.05	**82.31 ± 0.81**	81.79 ± 0.66	83.61 ± 0.57	80.52 ± 0.81	1.86

Table 13: AP (%) for Inductive Link Prediction with Random Negative Sampling Strategy. The Best and Second-Best Results are Emphasized by **Bold** and Underlined Fonts.

Methods	Wikipedia	Reddit	MOOC	LastFM	Enron	Social	UCI	Avg. Rank
TGAT	96.22 ± 0.07	97.09 ± 0.04	85.50 ± 0.19	78.63 ± 0.31	67.05 ± 1.51	91.41 ± 0.16	79.54 ± 0.48	10.43
TGN	97.83 ± 0.04	97.50 ± 0.07	89.04 ± 1.17	81.45 ± 4.29	77.94 ± 1.02	90.77 ± 0.86	88.12 ± 2.05	9.14
GM	96.65 ± 0.02	95.26 ± 0.02	81.41 ± 0.21	82.11 ± 0.42	75.88 ± 0.48	91.86 ± 0.06	91.19 ± 0.42	9.43
DGF	98.59 ± 0.03	98.84 ± 0.02	86.96 ± 0.43	94.23 ± 0.09	89.76 ± 0.34	93.14 ± 0.04	94.54 ± 0.12	6.71
FDG	98.97 ± 0.01	98.91 ± 0.01	87.75 ± 0.62	94.89 ± 0.01	89.69 ± 0.17	94.76 ± 0.05	94.85 ± 0.10	5.00
RM	98.70 ± 0.05	98.85 ± 0.01	93.05 ± 0.12	94.98 ± 0.12	87.97 ± 0.29	93.00 ± 0.11	95.04 ± 0.12	5.71
TPN	98.91 ± 0.01	98.86 ± 0.01	**95.07 ± 0.26**	95.36 ± 0.11	**90.34 ± 0.28**	93.24 ± 0.07	95.74 ± 0.05	3.43
TIDF	98.93 ± 0.02	98.98 ± 0.03	93.29 ± 0.39	95.41 ± 0.17	89.92 ± 0.45	93.28 ± 0.10	95.95 ± 0.12	3.14
AdaTIDF	**99.05 ± 0.05**	**99.07 ± 0.02**	94.52 ± 0.16	**95.77 ± 0.15**	90.11 ± 0.08	93.36 ± 0.05	96.11 ± 0.66	**1.86**
AdaTGN	98.36 ± 0.03	97.61 ± 0.04	89.84 ± 0.31	81.67 ± 3.65	78.34 ± 1.05	90.86 ± 0.15	88.36 ± 0.58	8.14
AdaFDG	99.08 ± 0.02	99.02 ± 0.01	88.36 ± 0.20	95.06 ± 0.10	89.74 ± 0.03	**94.83 ± 0.02**	**97.11 ± 0.61**	3.00

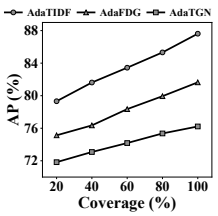


Figure 9: Analysis on Coverage and AP in Enron Dataset during Training (Transductive Settings)

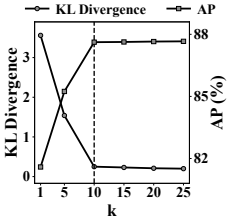


Figure 10: Analysis on KL Divergence and AP in Enron Dataset during Training(AdaTIDF, Transductive Settings)

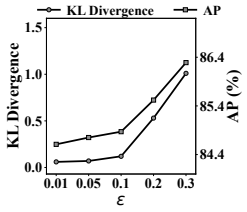


Figure 11: Analysis on KL Divergence and AP in Enron Dataset during Inference (AdaTIDF, Transductive Settings)

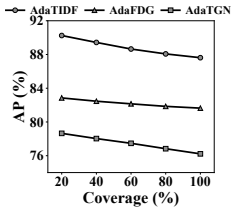
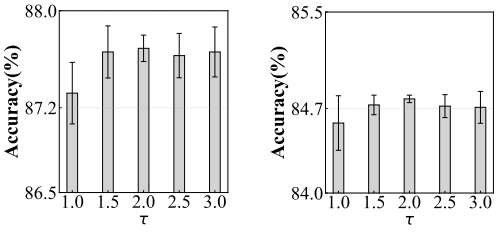


Figure 12: Analysis on Coverage and AP in Enron Dataset during Inference (Transductive Settings)



(a) AdaTIDF (b) AdaFDG

Figure 13: Analysis of τ on the MOOC dataset

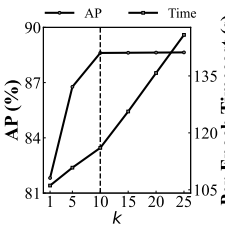


Figure 14: Analysis on k in Wikipedia (AdaTIDF)

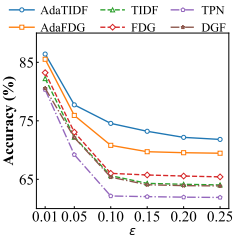


Figure 15: Analysis on ϵ in Reddit

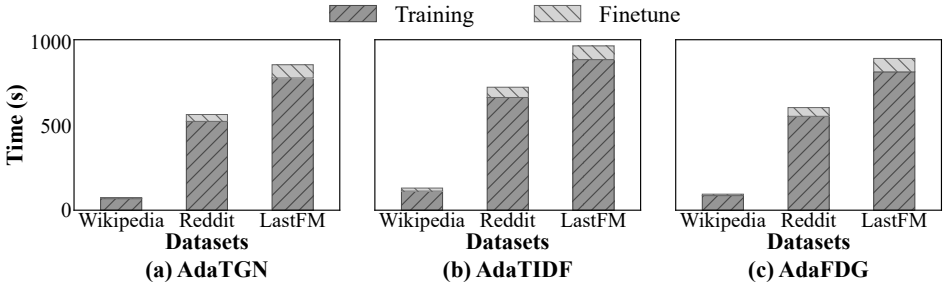


Figure 16: Breakdown of AdaDGL (1 epoch).

D.6 Analysis of Time and Memory Overhead.

The time and memory overhead is small in practice. As stated in Section 3.2, our method computes these embeddings on the fly during training. To further validate the efficiency of DNS, we measure the additional time and memory overhead introduced by DNS sampling. As shown in Table 17, DNS introduces only 3.81% additional time overhead and 0.96% additional memory overhead on average. The per-epoch overhead is therefore limited, as DNS operates on only a small candidate set at each sampling step. Overall, this marginal overhead provides a practical trade-off for the performance improvements achieved by AdaDGL. Moreover, the fine-tuning stage of AdaDGL incurs only a small time cost even when compared with a single training epoch as shown in Figure 16. Since model training typically involves multiple epochs, the relative time cost of fine-tuning becomes even smaller when considering the overall training procedure.

D.7 More Ablation Stucy

We further conduct a study in Tables 19 and 20, where the same MLP-head fine-tuning procedure is applied to all baseline models under the corresponding target distribution.

Under this setting, DNS consistently outperforms random negative sampling. As shown in Tables 19 and 20, DNS with fine-tuning achieves higher AP than random negative sampling with fine-tuning

Dataset	Wikipedia	REDDIT	MOOC	LastFM	Enron	SocialEvo	UCI
Counts of False Negative Samples in Training	0	0	0	0	0	0	0
Counts of False Negative Samples in Testing	0	0	0	0	0	0	0

Table 14: Static Analysis of False Negative Samples

Table 15: Training Time for Multiple Distribution on Different Models (Time/s, 10 epochs)

Dataset	TGAT	TGN	GM	DGF	FDG	RM	TPN	TIDF	AdaTIDF	AdaTGN	AdaFDG
Wikipedia	3976.58	1986.32	2716.78	3462.81	2496.82	2986.14	1898.77	3028.71	1160.51	**686.52**	867.43
Reddit	38946.22	14632.31	17683.44	19732.66	15873.66	19736.54	12796.81	18879.66	6678.31	**5261.71**	5562.61

Table 16: Inference Time on Different Models (Time/s, 1 epoch)

Dataset	TGAT	TGN	GM	DGF	FDG	RM	TPN	TIDF	AdaTIDF	AdaTGN	AdaFDG
Wikipedia	30.62	30.62	28.68	30.12	23.33	30.10	28.87	28.31	28.04	30.36	**22.62**
Reddit	166.85	134.55	135.44	158.61	128.42	148.66	130.21	145.81	140.16	128.62	**124.62**

Table 17: Time and Memory Overhead Introduced by DNS Sampling. (Time proportion is calculated as the DNS time cost divided by the epoch training time, while memory proportion is calculated as the additional DNS memory cost divided by the epoch memory cost.)

Dataset	Model	Time (s)	Time (%)	Memory (MB)	Memory (%)
Wikipedia	AdaTGN	2.45	3.57	6.05	1.19
	AdaFreeDyG	3.52	4.05	6.05	1.07
	AdaTIDFormer	4.62	3.98	6.05	0.83
Reddit	AdaTGN	19.05	3.62	7.21	1.08
	AdaFreeDyG	21.74	3.91	7.21	0.93
	AdaTIDFormer	24.78	3.71	7.21	0.65

Table 18: Performance Comparison on TGB datasets. Metric: MRR.

Datasets	TGAT	TGN	GM	DGF	FDG	RM	TPN	TIDF	AdaTGN	AdaFDG	AdaTIDF
tgbl-comment	56.20 ± 2.11	37.90 ± 2.10	76.17 ± 0.17	67.03 ± 0.14	78.31 ± 0.36	78.86 ± 0.82	$\underline{82.50 \pm 0.61}$	73.38 ± 0.29	46.27 ± 3.91	$\mathbf{83.37 \pm 0.21}$	79.62 ± 0.33
tgbl-coin-v2	60.89 ± 4.15	58.62 ± 3.75	72.53 ± 1.53	75.24 ± 0.48	79.08 ± 0.89	77.23 ± 0.62	83.20 ± 0.19	79.65 ± 0.82	67.81 ± 0.75	$\underline{83.27 \pm 0.63}$	$\mathbf{85.09 \pm 0.24}$

across all datasets and backbones. For example, on LastFM, FreeDyG improves by 2.85% under the historical setting and by 7.73% under the inductive setting.

D.8 Analysis on theorem

Theorem 1. Figure 9 further analyzes the effect of coverage during training, where coverage is defined as the ratio of observed negative samples to all eligible negative samples. As the coverage increases, the model is exposed to a more diverse set of negative samples, which provides richer training signals and consequently improves model performance. This observation validates the analysis in Theorem 1.

Theorem 2. Figure 10 shows that a smaller KL divergence[13] between the training negative distribution and the target distribution generally leads to better model performance. The empirical trend is consistent with the theoretical analysis in Theorem 2, indicating that better alignment with the target negative distribution improves model training.

Theorem 3 Figure 11 shows that a lower KL divergence corresponds to a harder negative distribution used for evaluation, resulting in a more challenging inference setting and lower model performance. This trend is consistent with Theorem 3.

[13]We compute the KL divergence based on the IDs of sampled negative nodes during training. Specifically, we record the negative node IDs sampled in each batch and aggregate them across all epochs. The empirical frequency of each negative node is then used to construct the negative sampling distribution during training, denoted as the *training distribution*. To construct the *target distribution*, for each sampling step, we select the hardest negative node from the candidate set and similarly estimate its distribution according to the empirical frequencies of the selected negative node IDs. We then compute the KL divergence between the training distribution and the target distribution. More detailed experimental settings and the corresponding computation procedure are provided in the appendix.

Table 19: AP Performance under the Historical Negative Sampling Strategy on Transductive Settings.

Dataset	Model	Random + Fine-tune	DNS + Fine-tune
Wikipedia	TGN	86.98 ± 0.38	$\mathbf{88.11 \pm 0.29}$
	FreeDyG	91.65 ± 0.55	$\mathbf{92.31 \pm 0.42}$
	TIDFormer	91.28 ± 0.75	$\mathbf{92.25 \pm 0.76}$
Reddit	TGN	81.28 ± 0.68	$\mathbf{82.34 \pm 0.56}$
	FreeDyG	85.78 ± 1.08	$\mathbf{87.61 \pm 1.04}$
	TIDFormer	83.54 ± 1.05	$\mathbf{84.07 \pm 0.99}$
LastFM	TGN	76.98 ± 3.75	$\mathbf{78.32 \pm 2.52}$
	FreeDyG	79.86 ± 0.58	$\mathbf{82.71 \pm 0.56}$
	TIDFormer	88.68 ± 0.56	$\mathbf{89.97 \pm 0.55}$

Table 20: AP Performance under the Inductive Negative Sampling Strategy on Transductive Settings.

Dataset	Model	Random + Fine-tune	DNS + Fine-tune
Wikipedia	TGN	85.73 ± 0.36	$\mathbf{88.64 \pm 0.33}$
	FreeDyG	90.13 ± 0.77	$\mathbf{92.10 \pm 1.07}$
	TIDFormer	91.26 ± 1.62	$\mathbf{92.34 \pm 1.14}$
Reddit	TGN	88.11 ± 0.43	$\mathbf{88.16 \pm 0.26}$
	FreeDyG	90.76 ± 0.18	$\mathbf{91.66 \pm 0.43}$
	TIDFormer	91.68 ± 0.63	$\mathbf{92.46 \pm 0.62}$
LastFM	TGN	66.03 ± 5.31	$\mathbf{75.33 \pm 2.85}$
	FreeDyG	72.52 ± 0.34	$\mathbf{80.25 \pm 0.57}$
	TIDFormer	75.82 ± 1.21	$\mathbf{85.40 \pm 0.96}$

Table 21: AP (%) on MOOC under Historical Negative Sampling in Transductive Settings with Different Proportions of Target-Distribution Fine-Tuning Data.

Proportions	0%	20%	40%	60%	80%	100%
TGN	88.32 ± 1.54	88.58 ± 1.57	88.81 ± 1.52	88.94 ± 1.53	89.05 ± 1.61	89.08 ± 1.62
TIDFormer	95.48 ± 1.03	95.96 ± 1.04	96.37 ± 1.08	96.69 ± 1.05	96.83 ± 1.01	96.86 ± 1.01

Theorem 4 Figure 12 analyzes the effect of coverage on inference performance in terms of AP. As the coverage increases from 20% to 100%, more candidate negative samples are introduced during evaluation, increasing the probability of including difficult negatives. Consequently, the inference task becomes more challenging and the model performance decreases. This result is consistent with the relationship between coverage and inference difficulty analyzed in Theorem 4.

Theorem 5 Table 21 analyzes the effect of the amount of target-distribution fine-tuning data on AP. As the data proportion increases from 0% to 100%, the AP of TGN increases from 88.32% to 89.08%, while that of TIDFormer increases from 95.48% to 96.86%. The standard deviations remain stable, and no performance degradation is observed. Since only the lightweight MLP prediction head is fine-tuned while the backbone remains frozen, the risk of overfitting is further reduced. These observations support the assumptions of Theorem 5 that greater access to the target distribution improves generalization without causing evident overfitting.

D.9 Analysis on Other Negative Sampling Strategy.

We further compare DNS with ENS (Gao et al., 2024) and nearest-neighbor hard-negative sampling (NHS) (Gupta & Bedathur, 2024). ENS estimates negative-sample difficulty using hand-crafted statistics, including historical existence and temporal proximity, whereas NHS primarily relies on historical interaction frequency. Since both methods operate on fixed statistics extracted from historical observations, they cannot fully capture the evolving semantic and structural relationships between nodes during training. In contrast, DNS estimates negative difficulty directly from the

learned node embeddings, which are continuously updated together with the model. Moreover, the historical existence, temporal proximity, and interaction-frequency information explicitly used by ENS and NHS can also be implicitly reflected in the learned representations, making DNS more comprehensive and adaptive.

As shown in Table 22, DNS consistently outperforms ENS and NHS both with and without adaptation. After adaptation, AdaDGL outperforms the strongest result of ENS and NHS by 0.27–0.85 AP points on Wikipedia, 0.31–1.32 points on Reddit, and 2.13–2.41 points on UCI. Averaged over all nine settings, the improvement over the strongest competing sampler is 1.23 AP points.

Although adaptation improves all three sampling strategies, its average gains are only 0.55 and 0.54 AP points for ENS and NHS, respectively, compared with 0.61 points for AdaDGL. Under the known-distribution setting, adaptation fine-tunes the initial model using samples from the known target distribution. However, the historical statistics employed by ENS and NHS have already been used to select negative samples during the initial training stage. Consequently, fine-tuning exposes the model to largely overlapping negative-sample information and therefore provides limited additional supervision. In contrast, DNS updates negative-sample difficulty according to the evolving learned representations, allowing the adaptation stage to better exploit the known target distribution.

Table 22: Performance Comparison of ENS, NHS, and AdaDGL with and without Adaptation.

Dataset	Model	ENS (w/o Ada)	ENS (w/ Ada)	NHS (w/o Ada)	NHS (w/ Ada)	AdaDGL (w/o Ada)	AdaDGL (Ada)
Wikipedia	TGN	87.03 ± 0.38	87.25 ± 0.41	87.06 ± 0.37	87.34 ± 0.44	87.83 ± 0.31	$\mathbf{88.11 \pm 0.29}$
	FDG	91.74 ± 0.75	91.93 ± 0.91	91.72 ± 0.62	91.98 ± 0.79	92.01 ± 0.75	$\mathbf{92.25 \pm 0.76}$
	TIDF	91.33 ± 0.72	91.45 ± 0.78	91.31 ± 0.81	91.46 ± 0.83	92.03 ± 0.46	$\mathbf{92.31 \pm 0.42}$
Reddit	TGN	81.31 ± 0.76	81.49 ± 0.84	81.27 ± 0.95	81.52 ± 0.91	82.08 ± 0.60	$\mathbf{82.34 \pm 0.56}$
	FDG	85.79 ± 1.12	86.13 ± 0.46	85.71 ± 1.04	86.29 ± 0.48	87.25 ± 1.08	$\mathbf{87.61 \pm 1.04}$
	TIDF	83.41 ± 1.06	83.71 ± 1.45	83.35 ± 1.46	83.76 ± 1.53	83.89 ± 1.07	$\mathbf{84.07 \pm 0.99}$
UCI	TGN	80.61 ± 1.75	81.63 ± 1.72	80.85 ± 1.77	81.53 ± 1.71	82.24 ± 1.67	$\mathbf{83.76 \pm 0.83}$
	FDG	86.26 ± 1.35	87.56 ± 1.29	86.53 ± 1.61	87.72 ± 1.63	88.63 ± 1.08	$\mathbf{90.13 \pm 1.01}$
	TIDF	89.85 ± 1.42	91.15 ± 0.89	89.81 ± 1.43	90.89 ± 0.95	92.45 ± 1.03	$\mathbf{93.34 \pm 0.72}$

D.10 Analysis on Effect of Adaption.

Furthermore, in Tables 23 and 24, we provide additional comparisons by evaluating the contribution of DNS before applying the adaptation method., thereby isolating and evaluating the effectiveness of the proposed adaptation mechanism.

Under historical negative sampling, AdaDGL improves its corresponding backbone in 21 dataset–backbone comparisons. On average, AdaTGN, AdaFDG, and AdaTIDF improve their corresponding backbones by 1.38, 1.75, and 1.80 percentage points, respectively. Particularly notable improvements include AdaTIDF on Enron ($81.59\% \rightarrow 87.41\%$) and UCI ($89.57\% \rightarrow 92.45\%$). Moreover, the AdaDGL variants collectively achieve the best AP on all seven datasets: AdaFDG performs best on Wikipedia, Reddit, and Social, while AdaTIDF performs best on MOOC, LastFM, Enron, and UCI.

Under inductive negative sampling, all 21 dataset–backbone comparisons show performance improvements. AdaTGN, AdaFDG, and AdaTIDF achieve average gains of 1.89, 2.29, and 1.81 percentage points, respectively. The improvements are particularly pronounced on LastFM, where AdaTGN, AdaFDG, and AdaTIDF improve their respective backbones by 6.22, 5.90, and 6.50 percentage points. AdaTIDF achieves the best result on five datasets and the second-best result on the remaining two datasets. These results demonstrate that the training-stage DNS alone improves robustness to different target negative sampling distributions, even before applying the proposed lightweight adaptation mechanism.

E Limitations and Broader Impacts

E.1 Limitations

Although AdaDGL has proven effective in boosting testing performance across several public benchmarks, numerous challenges still remain. For instance, its black-box decision-making process limits its applicability to domains such as social media analysis, where expert interpretability and result traceability are indispensable. Future work can integrate explainability modules together with interactive visualization tools to enable users to better understand the model's prediction mechanism. This will further enhance the model's reliability and practical applicability in real-world scenarios.

Table 23: AP (%) for Transductive Link Prediction with Historical Negative sampling strategy. The Best and Second-Best Results are Emphasized by **Bold** and Underlined Fonts. (AdaDGL do not use adaptation method)

Methods	Wikipedia	Reddit	MOOC	LastFM	Enron	Social	UCI	Avg. Rank
TGAT	87.38 ± 0.22	79.55 ± 0.20	82.19 ± 0.62	71.59 ± 0.24	64.07 ± 1.05	95.01 ± 0.44	68.27 ± 1.37	10.00
TGN	86.86 ± 0.33	81.22 ± 0.61	87.06 ± 1.93	76.87 ± 4.64	73.91 ± 1.76	94.45 ± 0.56	80.43 ± 2.12	9.14
GM	90.90 ± 0.10	78.44 ± 0.18	77.77 ± 0.92	72.47 ± 0.49	77.98 ± 0.92	94.93 ± 0.31	84.11 ± 1.35	8.71
DGF	82.23 ± 2.54	81.57 ± 0.67	85.85 ± 0.66	81.57 ± 0.48	75.63 ± 0.73	97.38 ± 0.14	82.17 ± 0.82	7.86
FDG	91.59 ± 0.57	85.67 ± 1.01	86.71 ± 0.81	79.71 ± 0.51	78.87 ± 0.82	97.79 ± 0.23	86.10 ± 1.19	5.00
RM	90.20 ± 1.04	83.02 ± 1.20	92.19 ± 0.58	86.73 ± 0.34	87.38 ± 0.18	97.40 ± 0.15	87.23 ± 0.23	4.14
TPN	81.55 ± 4.10	81.02 ± 1.31	92.69 ± 0.95	87.74 ± 0.50	80.79 ± 1.68	96.80 ± 0.41	86.34 ± 0.80	6.00
TIDF	91.21 ± 0.78	83.27 ± 1.01	95.37 ± 1.09	88.68 ± 0.57	81.59 ± 1.17	97.39 ± 0.67	89.57 ± 1.17	3.14
AdaTIDF(Wo)	92.01 ± 0.75	83.89 ± 1.07	$\mathbf{96.24 \pm 1.06}$	$\mathbf{89.63 \pm 0.61}$	$\mathbf{87.41 \pm 1.25}$	98.05 ± 1.03	$\mathbf{92.45 \pm 1.03}$	**1.57**
AdaTGN(Wo)	87.83 ± 0.31	82.08 ± 0.60	88.52 ± 1.73	78.05 ± 0.32	75.61 ± 1.60	96.16 ± 0.51	82.24 ± 0.88	7.43
AdaFDG(Wo)	$\mathbf{92.03 \pm 0.46}$	$\mathbf{87.25 \pm 1.08}$	90.18 ± 0.78	82.24 ± 0.53	80.03 ± 0.81	$\mathbf{98.32 \pm 0.42}$	88.63 ± 1.08	3.00

Table 24: AP (%) for Transductive Link Prediction with Inductive Negative Sampling Strategy. The Best and Second-Best Results are Emphasized by **Bold** and Underlined Fonts. (AdaDGL do not use adaptation method)

Methods	Wikipedia	Reddit	MOOC	LastFM	Enron	Social	UCI	Avg. Rank
TGAT	87.00 ± 0.16	89.59 ± 0.24	75.95 ± 0.64	71.13 ± 0.17	63.94 ± 1.36	94.84 ± 0.44	68.67 ± 0.84	9.43
TGN	85.62 ± 0.44	88.10 ± 0.24	77.50 ± 2.91	65.95 ± 5.98	70.89 ± 2.72	95.13 ± 0.56	70.94 ± 0.71	9.71
GM	88.59 ± 0.17	85.26 ± 0.11	74.27 ± 0.92	68.12 ± 0.33	75.01 ± 0.79	94.72 ± 0.33	80.10 ± 0.51	9.00
DGF	78.29 ± 5.38	91.11 ± 0.40	81.24 ± 0.69	73.97 ± 0.50	77.41 ± 0.89	97.68 ± 0.10	72.25 ± 1.71	6.64
FDG	90.05 ± 0.79	90.74 ± 0.17	83.01 ± 0.87	72.19 ± 0.24	77.81 ± 0.65	97.57 ± 0.15	82.35 ± 0.73	5.43
RM	88.86 ± 0.97	91.11 ± 0.73	83.11 ± 1.28	75.46 ± 0.78	$\mathbf{83.17 \pm 0.50}$	97.52 ± 0.28	84.20 ± 0.34	4.36
TPN	79.35 ± 5.52	88.19 ± 0.33	$\mathbf{88.18 \pm 0.97}$	77.99 ± 1.30	75.36 ± 1.81	97.35 ± 0.32	77.26 ± 1.57	6.29
TIDF	91.15 ± 1.17	91.57 ± 0.50	84.53 ± 1.11	75.69 ± 1.17	79.67 ± 0.66	97.70 ± 0.36	85.81 ± 1.39	3.00
AdaTIDF(Wo)	$\mathbf{91.83 \pm 1.15}$	$\mathbf{91.92 \pm 0.67}$	86.18 ± 1.09	$\mathbf{82.19 \pm 1.03}$	81.63 ± 0.64	$\mathbf{98.13 \pm 0.27}$	$\mathbf{86.93 \pm 0.94}$	**1.29**
AdaTGN(Wo)	87.63 ± 0.47	89.03 ± 0.27	78.25 ± 2.03	72.17 ± 3.14	72.83 ± 1.59	95.35 ± 0.49	72.09 ± 0.77	8.14
AdaFDG(Wo)	91.03 ± 1.09	91.38 ± 0.49	85.93 ± 0.93	78.09 ± 0.32	79.34 ± 0.61	98.03 ± 0.19	85.97 ± 0.33	2.71

E.2 BROADER IMPACTS

AdaDGL is a framework for dynamic graph learning that offers strong support for dynamic graph prediction tasks. A key advantage of this framework lies in its adoption of a negative sampling strategy that more closely aligns with real-world characteristics. AdaDGL holds great potential for accelerating discovery in social media analysis, recommendation analysis, and traffic analysis.