Topology-monitorable Contrastive Learning on Dynamic Graphs

Zulun Zhu
Nanyang Technological University
Singapore
ZULUN001@ntu.edu.sg

Kai Wang
Nanyang Technological University
Singapore
kai_wang@ntu.edu.sg

Haoyu Liu
Nanyang Technological University
Singapore
haoyu.liu@ntu.edu.sg

Jintang Li
Sun Yat-sen University
Guangzhou, China
lijt55@mail2.sysu.edu.cn

Siqiang Luo*
Nanyang Technological University
Singapore
siqiang.luo@ntu.edu.sg

ABSTRACT

Graph contrastive learning is a representative self-supervised graph learning that has demonstrated excellent performance in learning node representations. Despite the extensive studies on graph contrastive learning models, most existing models are tailored to static graphs, hindering their application to real-world graphs which are often dynamically evolving. Directly applying these models to dynamic graphs brings in severe efficiency issues in repetitively updating the learned embeddings. To address this challenge, we propose IDOL, a novel contrastive learning framework for dynamic graph representation learning. IDOL conducts the graph propagation process based on a specially designed Personalized PageRank algorithm which can capture the topological changes incrementally. This effectively eliminates heavy recomputation while maintaining high learning quality. Our another main design is a topology-monitorable sampling strategy which lays the foundation of graph contrastive learning. We further show that the design in IDOL achieves a desired performance guarantee. Our experimental results on multiple dynamic graphs show that IDOL outperforms the strongest baselines on node classification tasks in various performance metrics.

CCS CONCEPTS

• **Information systems** → **Web mining**; • **Theory of computation** → **Dynamic graph algorithms**.

KEYWORDS

Dynamic graph, contrastive learning, PageRank

ACM Reference Format:
Zulun Zhu, Kai Wang, Haoyu Liu, Jintang Li, and Siqiang Luo. 2024. Topology-monitorable Contrastive Learning on Dynamic Graphs. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24), August 25–29, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3637528.3671777

*Siqiang Luo is the corresponding author.

KDD '24, August 25–29, 2024, Barcelona, Spain

ACM ISBN 979-8-4007-0490-1/24/08.
https://doi.org/10.1145/3637528.3671777

1 INTRODUCTION

Graph representation learning has gained significant attention with the burgeoning of graph data across various applications, such as social networks [24, 28], online transactions [10, 23], recommendation systems [21, 77] and time-series traffic forecasting [16, 61]. Given the vast presence of unlabeled graph data, self-supervised learning on graphs has surged in popularity, which fosters the success of graph contrastive learning acting as a representative self-learning methodology [25, 29, 47, 48, 65]. By maximizing the similarity among positive samples and dissimilarity among negative samples, graph contrastive learning has demonstrated excellent performance in pretraining graph neural networks (GNNs) without manual labeling, thereby markedly boosting the model performance in the downstream tasks [39, 52].

Many real-world graphs are intrinsically dynamic, characterized by the frequent updates of nodes, edges, and attributes [76]. For example, Pinterest [74] and Tencent [26] use the graph structure to measure the real-time proximity among users, where each user is a node and the interaction between users is represented by an edge. The graph data can be updated thousands of times per second due to the extensive user base. In such dynamic scenarios, existing research on graph contrastive learning, originally designed for static graphs, becomes largely inapplicable [44, 69, 75], because the pretrained node representations lack the adaptability to accommodate new topological changes, leading to a decline in effectiveness.

Limitations of Existing Methods. Recent studies [3, 34, 56] start to explore graph contrastive learning approaches considering temporal dynamics. They segment the spatial structure of an evolving graph into multiple snapshots given a time window, which is called the discrete-time dynamic graph (DTDG). Unfortunately, there are two significant limitations of this category of approaches: (i) First, these models still *lack efficiency*, particularly when applied to large-scale graphs, since the node embeddings have to be regenerated via the GNN encoder (e.g., GCN [30], TGAT [63]) when updates occur. (ii) Second, the general positive/negative sampling strategy based on graph augmentation is not tailored to the dynamic graphs and has been proven *insufficient* for dynamic graph contrastive learning [13, 27, 54, 67]. For example, one recent method [56] assumes that the evolution of a graph unfolds smoothly and thus directly crafts positive and negative samples from the historical graphs with short and long timespans. Nonetheless, this sampling strategy exhibits limited effectiveness because it falls short in observing the topological changes and has the potential to erroneously classify stationary nodes as negative samples or evolving nodes as

positive samples, which goes against the intuition of contrastive objectives.

We contend that the aforementioned issues can be alleviated by the topology-monitorable capacity, i.e., taking full advantage of topological changes in dynamic graphs, which is overlooked by existing contrastive learning studies based on DTDG. Such a design effectively addresses the limitations mentioned earlier. Firstly, by continuously monitoring the topological structure, we can identify evolving nodes and subsequently perform local updates on their embeddings, eliminating the need for repetitive and global calculations. Secondly, by quantifying the extent of these changes, we can use this measure to inform and refine our sampling strategies. Therefore, to endow graph contrastive learning with the topology-monitorable capacity, we first employ the paradigm of continuous-time dynamic graphs (CTDG) [43], which are represented as a series of temporal edges with continuous dynamics and can exhibit more evolving locality than DTDG [34]. Then, inspired by recent GNN simplification works based on Personalized PageRank (PPR) [8, 31, 36], we propose to measure the topological changes of dynamic graphs in both graph propagation and pairwise sampling processes with efficient PPR-based algorithms.

Contributions. Based on this insight, we propose IDOL [1], a pioneering approach for node representation learning in continuous-time dynamic graphs (CTDG). IDOL aims to improve the efficiency of graph contrastive learning in a dynamic scenario with a paradigm of incremental update. For this purpose, we suggest to adopt an efficient graph propagation method based on Personalized PageRank (PPR) [9], which enables the incremental embedding update based on historical embeddings and eliminates repetitive embedding recomputation. In our evaluation, IDOL can achieve up to 3x faster pretraining compared with baselines. Moreover, we design a topology-monitorable sampling strategy in self-supervised learning of dynamic graphs, avoiding repetitive computation for augmentation as well as relaxing the evolving assumption of graphs in recent literature [6, 56, 64]. From a model design perspective, we are pioneers in implementing the *decoupled architecture* (e.g., decoupling propagation and training) within the realm of contrastive learning. Our approach conveys a crucial insight: the effectiveness in downstream tasks is closely linked to the performance achieved in our decoupled model design of contrastive learning. This could potentially spark further innovative investigations within the graph community. Our contributions are summarized as follows:

•We identify the limitations of existing contrastive learning methods in the dynamic graph scenario and introduce IDOL as an innovative self-supervised learning solution, crafted to excel in continuous-time dynamic graphs (CTDG).

•We adopt a PPR-based technique for incremental embedding update and present a topology-monitorable sampling method to generate contrastive pairs, thereby significantly boosting training efficiency and effectively enhancing the embedding quality in contrastive learning, respectively.

• To the best of our knowledge, IDOL is a pioneering algorithm that applies decouple propagation and training in contrastive learning, and in the meanwhile attaining a desired complexity (see Table 1). Within this innovative framework, we theoretically establish a performance guarantee for downstream tasks by linking the contrastive loss with the downstream task loss.

•We conduct comprehensive experiments across various real datasets to demonstrate the efficiency and effectiveness of our approach in dynamic graph scenarios. Remarkably, IDOL outperforms in prediction accuracy for dynamic node classification, while entailing notably less pretraining time.

[1] ***I***ncremental ***D***ynamic Graph C***o***ntrastive ***L***earning

2 PRELIMINARY AND RELATED WORKS

In this section, we begin by introducing the problem definition of this work and review the basic graph contrastive learning framework and Personalized PageRank (PPR) algorithms related to our design. We list the frequently used notations in Appendix A.1. We leave the detailed proofs of this paper in our technique report [12].

2.1 Problem Definition

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{X})$ be a directed and attributed graph, where $\mathcal{V} = \{v_1, v_2, ..., v_n\}$ is the set of n nodes, $\mathcal{E} = \mathcal{V} \times \mathcal{V}$ is the set of m edges and $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_F\}$ is the set of node attribute matrix with $\boldsymbol{x}_i \in \mathbb{R}^{n\times 1}$ representing the attribute vector in i-th dimension. For each node $v \in \mathcal{V}$, $\mathcal{N}_{out}(v)$ stands for the out-neighbors of v and $\mathcal{N}_{in}(v)$ stands for the in-neighbors.

Then, we define the continuous-time dynamic graph (CTDG). Consider an initial graph of CTDG $\mathcal{G}_0 = (\mathcal{V}_0, \mathcal{E}_0)$, the set of update events Γ upon the graph consists of inserts and deletes of edges[2], represented as $\Gamma = \{e_1, e_2, ..., e_i, ...e_p\}$. After the i-th edge update $e_i = \{u_i, v_i\}$ arrives the system, the current CTDG $\mathcal{G}_{i-1}$ is transferred to $\mathcal{G}_i$. If e_i already exists in $\mathcal{G}_{i-1}$, e_i is treated as a delete from (u_i, v_i); otherwise, e_i is treated as an insert. Different with CTDG, a discrete-time dynamic graph (DTDG) is represented as a series of snapshot $\{\mathcal{G}_0, \mathcal{G}_1, ..., \mathcal{G}_\tau, ..., \mathcal{G}_T\}$, where $\mathcal{G}_\tau$ signifies the status of graph at time τ. Since the evolutionary process (e.g., update events) between two snapshots (e.g., $\mathcal{G}_{\tau_1}$ and $\mathcal{G}_{\tau_2}$) is not captured, CTDG offers a more comprehensive representation than DTDG.

The research problem of this paper, representation learning on CTDG, is to learn node representation for each node s at any timestamp. Assume there are p update events based on $\mathcal{G}_i$, our target is to calculate the F-dimensional node embeddings for the graph $\mathcal{G}_{i+p}$ incrementally. The learned embeddings capture the dynamic evolution of the CTDG and can be effectively applied to subsequent graph tasks, such as node classification.

2.2 Graph Contrastive Learning

The representative scheme of Graph Contrastive Learning is *Pretraining and Evaluation*, where the node representations are pretrained in a self-supervised manner first and then evaluated in downstream tasks [39, 71]. The main ingredients of the pretraining include: (i) *Sampling Strategy* based on graph augmentation producing positive and negative sample pairs, and (ii) *Loss Function* supervising the model to determine the similarity of each specific representation pair [39, 62]. Detailed related works of Graph Contrastive Learning are described in Appendix A.2.

In graph-related scenarios, the common sampling strategy generates two or more augmentation views of the original graph with

[2] The inserts and deletes of vertices can be replaced by adding and removing relevant incident edges, hence we only discuss edge updates in this paper.

Table 1: Comparison of pretraining complexity. K stands for the number of convolution layers, T means the number of temporal views, and F is the dimension of node features.

Method	Scenarios	Encoder	Complexity for Initial Graph		Complexity for Updating p Edges
			Encoder	Projector&Loss	
DGI [58]	static	GCN [30]	$O(4KnF^2 + 4KmF)$	$O(4nF^3)$	$O(4KnF^2 + 4KmF + 4nF^3)$
GRACE [75]	static	GCN	$O(4KnF^2 + 4KmF$	$O(4nF^2 + 2n^2F + 2nF)$	$O(4KnF^2 + 4KmF + 2n^2F)$
BGRL [55]	static	GCN	$O(4KnF^2 + 4KmF)$	$O(4nF^2 + 4nF)$	$O(4KnF^2 + 4KmF + 4nF^2)$
GGD [72]	static	GCN	$O(4KnF^2 + 4KmF)$	$O(4nF^2 + 2nF)$	$O(4KnF^2 + 4KmF + 4nF^2)$
DDGCL [56]	dynamic	TGAT [63]	$O(4Kn^2F + 4KmF)$	$O(2nF^3 + 2n^2F^3)$	$O(4Kn^2F + 4KmF + 2n^2F^3)$
CLDG [64]	dynamic	GCN	$O(4KTnF^2 + 4KTmF)$	$O(4TnF^2 + 2n^2T^2F)$	$O(4KTnF^2 + 4KTmF + 2n^2T^2F)$
IDOL (ours)	dynamic	PPR+MLP	$O(4nF^2 + 2KmF)$	$O(nF)$	$O(4nF^2 + KmF + p \log n)$

various augmentation functions such as random node dropping and feature masking [27, 67]. According to the connectivity similarity, one popular mechanism is generally treating the representations of the same node in two different views as a positive sample pair, otherwise a negative one [56, 75]. The loss function of contrastive learning denotes the negative estimated mutual information, where the commonly used formats include Jensen–Shannon divergence [22], NCE [19] and InfoNCE [46].

Specifically, for one node s in the graph, the positive sample set $\{^+\boldsymbol{z}_s\}$ and negative sample set $\{^-\boldsymbol{z}_s\}$ are extracted via the sampling strategy and we denote the node embedding of node s as $\boldsymbol{z}_s$ [3]. Then, in the representation learning model, a GNN encoder $f_\theta(\cdot)$ parameterized by θ is utilized to generate node representations, accompanied by a projector $p(\cdot)$ (e.g., dot product) measures the similarity of each representation pair. Finally, the pretraining goal is to obtain the optimal encoder θ via the contrastive learning loss, such as the following InfoNCE loss used by [49, 62, 75]:

$$\mathcal{L}_{InfoNCE} = -\frac{1}{n}\sum_{s=1}^{n} \log \frac{e^{p(\boldsymbol{z}_s, {}^+\boldsymbol{z}_s)}}{e^{p(\boldsymbol{z}_s, {}^+\boldsymbol{z}_s)} + \sum_{i=1}^{M} e^{p(\boldsymbol{z}_s, {}^-\boldsymbol{z}_i)}}, \tag{1}$$

where $p(\boldsymbol{z}_1, \boldsymbol{z}_2) = f_\theta(\boldsymbol{z}_1) f_\theta(\boldsymbol{z}_2)^\top / \tau$ and τ is a temperature hyperparameter. The InfoNCE loss estimates the mutual information between each node representation $\boldsymbol{z}_s$ with one positive sample $^+\boldsymbol{z}_s$ and M negative samples $\{^-\boldsymbol{z}_i\}$.

2.3 Personalized PageRank

Given a source node s and a target node t in a graph, the Personalized PageRank (PPR) [9] $\pi(s,t)$ is a topology-based measure to reflect the probability that a random walk starting from s ends at t. The single-source PPR (SSPPR), which computes $\pi(s,t)$ for any t in a graph given the source node s, has been a significant building block of various applications [18, 37, 38, 41, 42, 78]. Given the source node s, the SSPPR vector $\boldsymbol{\pi}(s) \in \mathbb{R}^{1\times n}$ aims to obtain the solution of the following equation:

$$\boldsymbol{\pi}(s) = \sum_{i=0}^{\infty} \alpha(1-\alpha)^i \cdot \left(\boldsymbol{A}\boldsymbol{D}^{-1}\right)^i \cdot \boldsymbol{e}_s, \tag{2}$$

where $\boldsymbol{A}$ is the adjacent matrix, $\boldsymbol{D}$ is the degree matrix, α is the decay factor of random walk, and $\boldsymbol{e}_s$ is a one-hot vector with $\boldsymbol{e}_s(s) = 1$, respectively. The SSPPR computation needs to extract eigenvalues of an $n \times n$ matrix and is expensive on a large-scale graph [59]. In order to compute SSPPR efficiently, *Forward Push* algorithm [5] is developed to approximate the value $\pi(s,t)$ given the source node s and $t \in \mathcal{V}$, which achieves an underestimate with an error bound. A detailed description can be found in our technique report [12].

PPR-based Node Embedding. Since PPR can reflect the topological relationship between nodes, recent studies of simplified GNNs [36] utilize the SSPPR algorithm combined with the attribute matrix to calculate the embedding of nodes. Such PPR-based node embedding is more efficient and scalable compared to graph propagation via GNNs, and it facilitates incremental updates through local modifications to the termination probability of random walks. Specifically, SCARA [36] adopts the *Forward Push* algorithm to enhance the scalability of the graph propagation and efficiency. Instant [73] and DynAnom [17] utilize a model structure consisting of PPR-based embedding and refreshing rules to achieve the incremental update for the node embeddings. The hallmark of methods employing *Forward Push* is their propagation process, which offers a fundamental approximation guarantee. Interestingly, our empirical findings reveal that prioritizing efficiency over this guarantee can actually enhance accuracy in contrastive learning. Therefore, we simply aggregate node information from a fixed number of neighboring hops to not only accelerate embedding computation but also preserve high-quality results.

3 METHODOLOGY

3.1 Overview

In this section, we introduce our proposed IDOL, which is divided into three key components as depicted in Figure 1. Firstly, different from most of the existing contrastive learning methods utilizing GCN [30] or TGAT[63] as graph encoder, we decouple the graph propagation with training, and utilize PPR-based embeddings and multi-layer perceptron (MLP) to incorporate the structural information in the initial graph (Sec. 3.2). Secondly, given upcoming update events, we incrementally update the node embedding efficiently to avoid the propagation from scratch (Sec. 3.3). Moreover, instead of generating two different augmented graphs from scratch, we propose a novel topology-monitorable sampling strategy to avoid the repetitive computation and directly select positive and negative samples from the pre-existing historical embeddings (Sec. 3.4). Finally, we freeze the encoder parameters after pretraining and output the node embeddings for evaluation in the downstream tasks (e.g., node classification). By exploring the connectivity between classification accuracy and the convergence of our contrastive loss, we prove that our decoupled and simplified framework IDOL provides

[3] In the later sections, we call that $\boldsymbol{z}_s$ and $^+\boldsymbol{z}_s$ (resp. $^-\boldsymbol{z}_s$) can form a positive (resp. negative) pair for a clear presentation.

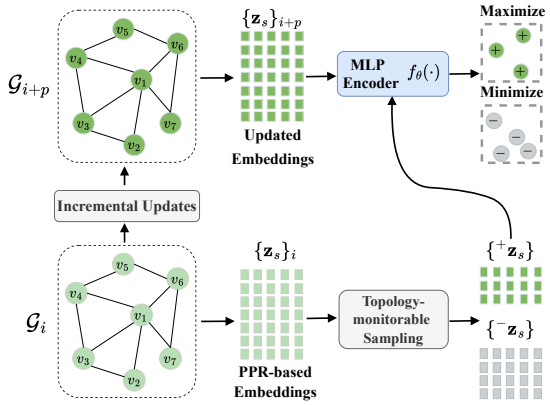


Figure 1: Graphical illustration of the IDOL architecture. $\{\mathrm{z}_s\}_i$ denotes the node embeddings in $\mathcal{G}_i$. $\{^+\mathrm{z}_s\}$ and $\{^-\mathrm{z}_s\}$ denote the positive and negative sample sets, respectively.

a performance guarantee of the downstream task, demonstrating the theoretical feasibility and effectiveness (Sec. 3.5).

3.2 PPR-based Embedding by *K-hop Push*

To achieve scalable node embeddings and update them incrementally in dynamic graphs, we utilize PPR-based algorithms to pre-calculate node embeddings before processing them through the MLP layer. The dominant paradigm of existing PPR-based embedding derives from the insight of *Forward Push* based on a residue threshold (e.g., [36]). Nonetheless, *Forward Push*, which aims to approximate PPR scores with an accuracy guarantee, falls short in effectively showcasing the expressive potential of all nodes during graph propagation. This limitation can lead to an uneven distribution of smoothness in the node embedding space, ultimately resulting in a decline in performance [45, 70]. The experiments in Sec. 4.5 indicate that this limitation leads to reduced prediction accuracy compared with our method under the same propagation time, alongside a heightened sensitivity to hyperparameter settings.

To solve the aforementioned issue, we incorporate the intuition of APPNP [31] aggregating information of K-hop neighbors, and employ a *K-hop Push* algorithm to pre-compute the PPR-base node embeddings. Specifically, we deprecate the approximation paradigm of *Forward Push* and adopts the K-hop truncated terms of Eq. 2 as follows:

$$\boldsymbol{h}_i = \sum_{l=0}^{K-1} \alpha(1-\alpha)^l \cdot \left(\boldsymbol{A}\boldsymbol{D}^{-1}\right)^l \cdot \boldsymbol{x}_i. \tag{3}$$

By employing this design, we ensure that each node can acquire a balanced smoothness distribution, aligning precisely with a simplified complexity of $O(KmF)$ as demonstrated in SGC [60], given the hop number K and feature dimension F.

Initial Graph Embedding. To calculate the node embeddings in initial graph $\mathcal{G}_0$, we conduct the *K-hop Push* algorithm to calculate the PPR values of all nodes for each dimension of the attribute matrix $\boldsymbol{X}$, which is shown in Algorithm 1. Specifically, we extract the attribute vector (column vector) $\boldsymbol{x}_i \in \mathbb{R}^{n\times 1}$ $(1 \le i \le F)$ for each dimension of $\boldsymbol{X}$ and assign it as the initial value of *residue vector* $\boldsymbol{r}_i^0$. Meanwhile, the *reserve vector* $\boldsymbol{h}_i \in \mathbb{R}^{n\times 1}$ is set as the all-zero vector. Here in the PPR procedure, $\boldsymbol{r}_i^l$ represents the unpropagated mass

Algorithm 1: K-hop Push Algorithm

Input : Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, teleport probability α, feature matrix $\boldsymbol{X}$, number of propagation layer K.
Output : Reserve vector $\boldsymbol{h}_i$ and residue vector $\boldsymbol{r}_i^K$ $(1 \le i \le F)$

```
foreach x_i ∈ X do
    r_i^0 = x_i; h_i = 0;
    for l = 0 to K − 1 do
        r_i^{l+1} = 0;
        foreach s ∈ V do
            foreach t ∈ N_out(s) do
                r_i^{l+1}(t) += (1 − α) · r_i^l(s) / |N_out(s)|;
            h_i(s) += α · r_i^l(s); r_i^l(s) = 0;
        clear r_i^l;
```

of attribute vector $\boldsymbol{x}_i$, and $\boldsymbol{h}_i$ represents the propagated part. Then, we iteratively repeat the following steps for $l = \{0, 1, ..., K-1\}$. For each node $s \in \mathcal{V}$, $(1-\alpha)$ fraction of $\boldsymbol{r}_i^l(s)$ will be propagated into the out-neighbors $t \in \mathcal{N}_{out}(s)$, and hence each out-neighbor t iteratively receives $(1-\alpha)\cdot \boldsymbol{r}_i^l(s)/|\mathcal{N}_{out}(s)|$ and add it to the residue $\boldsymbol{r}_i^{l+1}(t)$. Additionally, α fraction of $\boldsymbol{r}_i^l(s)$ will be transferred into the reserve vector $\boldsymbol{h}_i(s)$ and $\boldsymbol{r}_i^l(s)$ is reset as 0. At the end of l-th iteration, we clear the residue vector $\boldsymbol{r}_i^l$ to save the space.

After we conduct the propagation and obtain the K-hop reserve vector $\boldsymbol{h}_i$ $(1 \le i \le F)$ via *K-hop Push* for all dimensions in $\boldsymbol{X}$, the final node embedding matrix can be concatenated as:

$$\boldsymbol{Z} = \mathrm{Concat}(\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_F), \tag{4}$$

where each node embedding $\boldsymbol{z}_s \in \mathbb{R}^{1\times F}$. Compared with existing contrastive learning methods utilizing GCN [30] or TGAT [63] for graph propagation, the above PPR-based embedding process is more efficient and scalable. We will further discuss the computational complexity in the Sec. 3.6.

3.3 Incremental Embedding Update

In our decoupled contrastive learning approach, we initially pre-calculate node embeddings for the initial graph using the previously described steps. However, the graphs incur frequent updates upon the nodes and edges in the dynamic scenarios, which requires us to incorporate the updated information and reform the node embedding. Instead of recalculating embedding from scratch or requiring repetitive convolution operations such as the works in [6, 56], we focus on transforming partial embeddings incrementally triggered by the update events. Due to the characteristics of the PPR algorithm, one can locally adjust the reserve vector $\boldsymbol{h}_i$ and residue vector $\boldsymbol{r}_i$ for value updates provided that the invariant property between them is maintained [17, 73]. Following the updating streamline of the vanilla PPR methods employed in Instant [73] and DynAnom [17], we adopt a similar approach to refresh the embeddings while holding the invariant property. For completeness, we refer interested readers to our technical report [12] for a complete proof and references. Given the invariant property $\boldsymbol{h}_i(u) + \alpha \boldsymbol{r}_i(u) = \alpha \boldsymbol{x}_i(u) + \sum_{v\in\mathcal{N}_{out}(u)} \frac{(1-\alpha)\boldsymbol{h}_i(v)}{|\mathcal{N}_{out}(v)|}$ $(u \in \mathcal{V})$, we summarize the heart of updating rules in the following lemma:

Lemma 1. *When adding an edge (u, v), performing the following rules can maintain this invariant property:* $\boldsymbol{r}_i(u) \leftarrow \boldsymbol{r}_i(u) - \frac{\boldsymbol{h}_i(u)}{\alpha|\mathcal{N}_{out}(u)|}, \boldsymbol{r}_i(v) \leftarrow \boldsymbol{r}_i(v) + \frac{(1-\alpha)\boldsymbol{h}_i(u)}{\alpha|\mathcal{N}_{out}(u)|}, \boldsymbol{h}_i(u) \leftarrow \boldsymbol{h}_i(u) \cdot \frac{|\mathcal{N}_{out}(u)|+1}{|\mathcal{N}_{out}(u)|}$.

Compatibility with DTDG. By adhering to these principles, edges are incorporated into the CTDG based on their chronological sequence. DTDG, a specific instance of CTDG, can also be managed using the update rules by leveraging the timestamps of edges across various snapshots. Given that the degree of node u at time τ is $\mathcal{N}^{\tau}_{out}(u)$ and the degree change from τ_1 to τ_2 is $\Delta_{\tau_1,\tau_2}(u) = \mathcal{N}^{\tau_2}_{out}(u) - \mathcal{N}^{\tau_1}_{out}(u)$, we can update the reserve and residue vector from snapshot $\mathcal{G}_{\tau_1}$ to $\mathcal{G}_{\tau_2}$ utilizing the following derivative rules: $\boldsymbol{r}_i(u) \leftarrow \boldsymbol{r}_i(u) - \frac{\Delta_{\tau_1,\tau_2}(u)\boldsymbol{h}_i(u)}{\alpha|\mathcal{N}^{\tau_1}_{out}(u)|}, \boldsymbol{r}_i(v) \leftarrow \boldsymbol{r}_i(v) + \frac{(1-\alpha)\Delta_{\tau_1,\tau_2}(u)\boldsymbol{h}_i(u)}{\alpha|\mathcal{N}^{\tau_1}_{out}(t)|}, \boldsymbol{h}_i(u) \leftarrow \boldsymbol{h}_i(u) \cdot \frac{|\mathcal{N}^{\tau_2}_{out}(t)|}{|\mathcal{N}^{\tau_1}_{out}(t)|}$.

With the rules above, we incrementally update the vectors of nodes affected by the update event, accompanied by updating their node embeddings. We provide the pseudo-code of updating the node embeddings in our technique report [12]. With such embedding update approach, we next discuss how to generate contrastive pairs for the updated embeddings using a topology-based approach as detailed in Section 3.4.

3.4 Topology-monitorable Sampling

The concepts of graph augmentation are central to the paradigm of contrastive learning. Recent works [56] emphasize the assumption that the evolution of graphs unfolds smoothly, and the overall properties remain relatively stable despite occasional edge updates. However, this constraint is impractical and too strong in real-world applications, as a node might undergo substantial updates in a short interval while remaining unchanged over an extended one. We experimentally prove that employing this assumption is suboptimal to capture the practical dynamics in Sec. 4.5.

In order to break this strong constraint, we propose a topology-monitorable sampling strategy to generate positive and negative samples from the topological views in the dynamic graph. Formally, given a snapshot $\mathcal{G}_{i+p}$ to be predicted, we explicitly exploit the topological transformation caused by p update events and adopt a PPR-based approach to distinguish the positive and negative pairs from the historical embeddings of $\mathcal{G}_i$.

Topological measurement by PPR. As a measurement of topological information, the difference in SSPPR values between two snapshots $\mathcal{G}_i$ and $\mathcal{G}_{i+p}$ reflect the topological change on a node caused by p updates. Since SSPPR contains $\pi(s,t)$ for any t in a graph given the source node s, we utilize the maximum value of $\left|\pi\left(\mathcal{G}_{i+p}, s, t\right) - \pi\left(\mathcal{G}_i, s, t\right)\right|$ to describe the impact degree of the node s caused by p updates. Then we define an *impact vector* $\boldsymbol{I}_{\mathcal{G}_i,\mathcal{G}_{i+p}} \in \mathbb{R}^{n\times 1}$ representing the impact degrees on all graph nodes, where each element of $\boldsymbol{I}_{\mathcal{G}_i,\mathcal{G}_{i+p}}$ is defined as:

$$\boldsymbol{I}_{\mathcal{G}_i,\mathcal{G}_{i+p}}(s) = \max_{t\in\mathcal{V}_i} \left|\pi\left(\mathcal{G}_{i+p}, s, t\right) - \pi\left(\mathcal{G}_i, s, t\right)\right|, s \in \mathcal{V}_i. \quad (5)$$

Upper bound of impact vector. As the graph evolves, the measurement $\boldsymbol{I}_{\mathcal{G}_i,\mathcal{G}_{i+p}}(s)$ can capture the upper bound of topological change on node s and guide us to distinguish positive and negative samples. However, the exact result of $\boldsymbol{I}_{\mathcal{G}_i,\mathcal{G}_{i+p}}$ requires a time-consuming computation with respect to all-pair PPR values,

Algorithm 2: Dynamic Sampling Algorithm

Input : Node embedding $\boldsymbol{z}'_s$ for all $s \in \mathcal{V}_i$ based on graph $\mathcal{G}_i$, residue threshold r^b_{max}, gap threshold λ, update event sequence $\Gamma = \{(u_1, v_1), (u_2, v_2), ..., (u_p, v_p)\}$.

Output : Positive sample ${}^{+}\boldsymbol{z}_s$ or negative sample ${}^{-}\boldsymbol{z}_s$ for each $s \in \mathcal{V}_{i+p}$ in $\mathcal{G}_{i+p}$

1 /* **Reverse Push from** $\{u_1, u_2, ..., u_p\}$ */
2 $\boldsymbol{r}_b = \boldsymbol{0}; \boldsymbol{\pi}_b = \boldsymbol{0}$;
3 **foreach** $(u_j, v_j) \in \Gamma$ **do**
4 $\quad \boldsymbol{r}_b(u_j) = \frac{\left||\mathcal{N}_{out}(\mathcal{G}_{i+p}, u_j)| - |\mathcal{N}_{out}(\mathcal{G}_i, u_j)|\right|}{|\mathcal{N}_{out}(\mathcal{G}_{i+p}, u_j)|}$;
5 **while** $\exists s \in V$ *s.t.* $\boldsymbol{r}_b(s) > r^b_{max}$ **do**
6 $\quad$ **foreach** $t \in \mathcal{N}_{in}(\mathcal{G}_i, s)$ **do**
7 $\quad\quad \boldsymbol{r}_b(t) += (1-\alpha) \cdot \frac{\boldsymbol{r}_b(s)}{|\mathcal{N}_{out}(\mathcal{G}_i, t)|}$;
8 $\quad \boldsymbol{\pi}_b(s) += \alpha \cdot \boldsymbol{r}_b(s)$; $\boldsymbol{r}_b(s) = 0$;
9 /* **Sampling Contrastive Pairs** */
10 **foreach** $s \in \mathcal{V}$ **do**
11 $\quad$ **if** $\frac{r^b_{max} + \boldsymbol{\pi}_b(s)}{\alpha} \geq \lambda$ **then**
12 $\quad\quad {}^{-}\boldsymbol{z}_s = \boldsymbol{z}'_s, {}^{-}Z_{\mathcal{G}_{i+p}}$.add$({}^{-}\boldsymbol{z}_s)$
13 $\quad$ **else**
14 $\quad\quad {}^{+}\boldsymbol{z}_s = \boldsymbol{z}'_s, {}^{+}Z_{\mathcal{G}_{i+p}}$.add$({}^{+}\boldsymbol{z}_s)$

which refers to the calculation of $\pi\left(\mathcal{G}_{i+p}, s, t\right)$ and $\pi\left(\mathcal{G}_i, s, t\right)$ for all $s, t \in \mathcal{V}_i$. Instead of the redundant calculation, we estimate the upper bound of $\boldsymbol{I}_{\mathcal{G}_i,\mathcal{G}_{i+p}}(s)$ in the following lemma:

Lemma 2. *Given the update sequence* $\Gamma = \{(u_1, v_1), (u_2, v_2), ..., (u_p, v_p)\}$, *we have:*

$$\boldsymbol{I}_{\mathcal{G}_i,\mathcal{G}_{i+p}}(s) \leq \sum_{j=1}^{p} \frac{\left||\mathcal{N}_{out}(\mathcal{G}_{i+p}, u_j)| - |\mathcal{N}_{out}(\mathcal{G}_i, u_j)|\right| \pi(\mathcal{G}_i, s, u_j)}{\alpha|\mathcal{N}_{out}(\mathcal{G}_{i+p}, u_j)|}.$$

Based on Lemma 2, the impact on any $s \in \mathcal{V}_i$ caused by the update sequence Γ is related to the PPR value $\pi\left(\mathcal{G}_i, s, u_j\right)$ and the degree $|\mathcal{N}_{out}\left(\mathcal{G}_i, u_j\right)|$ and $|\mathcal{N}_{out}\left(\mathcal{G}_{i+p}, u_j\right)|$, where u_j is the starting node of the updated edge in Γ. The calculation of $\pi\left(\mathcal{G}_i, s, u_j\right)$ for all $s \in \mathcal{G}_i$ can be implemented by *Reverse Push* algorithm focusing on single-target PPR [4]. Nonetheless, the *Reverse Push* method can only assist in estimating a single update, making it inefficient when dealing with batch updates (e.g., update sequence Γ).

Considering the *batch* update events included in Γ, we incorporate the insight of *Reverse Push* to efficiently estimate the upper bound of $\boldsymbol{I}_{\mathcal{G}_i,\mathcal{G}_{i+p}}(s)$ after p update events for each $s \in \mathcal{V}_i$, which is shown in Algorithm 2 (lines 3-8). Similar to *K-hop Push*, we maintain an auxiliary *residue vector* $\boldsymbol{r}_b$. We assign the intial value of $\boldsymbol{r}_b$ at u_j as $\frac{\left||\mathcal{N}_{out}(\mathcal{G}_{i+p}, u_j)| - |\mathcal{N}_{out}(\mathcal{G}_i, u_j)|\right|}{|\mathcal{N}_{out}(\mathcal{G}_{i+p}, u_1)|}$, where u_j is the starting point of the update event in Γ (line 4), and otherwise set it as 0. The difference is that we set a residue threshold r^b_{max} to early stop the iteration process (line 5) and the residue value (e.g., $\boldsymbol{r}_b(s)$) is propagated along the in-neighbor set $\mathcal{N}_{in}(\mathcal{G}_i, s)$ (line 6). Notice that for *Dynamic Sampling* we replace the manner utilized in *K-hop Push* since setting the threshold r^b_{max} can provide us with an intermediate upper bound of $\boldsymbol{I}_{\mathcal{G}_i,\mathcal{G}_{i+p}}(s)$ demonstrated as: $\boldsymbol{I}_{\mathcal{G}_i,\mathcal{G}_{i+p}}(s) \leq \frac{\boldsymbol{\pi}_b(s) + r^b_{max}}{\alpha}$. To simplify our presentation, we directly introduce this crucial intermediate upper bound here. It plays a pivotal role in sampling contrastive

pairs, as elaborated in the upcoming Lemma 3. The detailed proof of this upper bound can be found in the technique report [12].

Sampling with impact vector. After the iteration in *Reverse Push* ends, we compared the value of $\frac{r^b_{max}+\pi_b(s)}{\alpha}$ with an empirical gap threshold λ to filter the positive samples from $\mathcal{G}_i$ satisfying $\boldsymbol{I}_{\mathcal{G}_i,\mathcal{G}_{i+p}}(s) \leq \lambda$, as presented in the following lemma:

Lemma 3. *Given the update event sequence* $\Gamma = \{(u_1, v_1), (u_2, v_2), ..., (u_p, v_p)\}$*, the result returned by Algorithm 2 guarantees that for any node index s in* $\{^+\boldsymbol{z}_s\}$*, we have* $\boldsymbol{I}_{\mathcal{G}_i,\mathcal{G}_{i+p}}(s) \leq \lambda$.

Based on Lemma 3, the nodes experiencing minor impacts, as determined by λ, are selected. Then we treat the corresponding historical embeddings in $\mathcal{G}_i$ as the positive sample set $\{^+\boldsymbol{z}_s\}$. Meanwhile, we retain other nodes that incur *significant* topological changes and treat their historical embeddings as negative samples (line 10-14). Here we also denote the positive and negative samples of $\mathcal{G}_{i+p}$ as $^+\boldsymbol{Z}_{\mathcal{G}_{i+p}} = \{^+\boldsymbol{z}_s\}$ and $^-\boldsymbol{Z}_{\mathcal{G}_{i+p}} = \{^-\boldsymbol{z}_s\}$ respectively. The concrete value of $n_p = |^+\boldsymbol{Z}_{\mathcal{G}_{i+p}}|$ and $n_g = |^-\boldsymbol{Z}_{\mathcal{G}_{i+p}}|$ can be coordinated by setting different threshold λ and we can easily get $n_p + n_g = n$, where n is the number of nodes.

3.5 Theoretical Performance Guarantee

After pre-calculating the node embeddings and building contrastive pairs, we form our contrastive loss within a decoupled architecture. Generally, the contrastive loss enforces the embeddings to be discriminative between the positive pairs from the joint distribution $p(\boldsymbol{z}_s, ^+\boldsymbol{z}_s)$ and the negative pairs from the marginal distribution $p(\boldsymbol{z}_s)$ and $p(^-\boldsymbol{z}_s)$. Given the positive sample set $^+\boldsymbol{Z}_{\mathcal{G}_{i+p}}$ and negative sample set $^-\boldsymbol{Z}_{\mathcal{G}_{i+p}}$, the node representations after our MLP encoder is denoted as $f_\theta(\boldsymbol{z}_s)$, $f_\theta(^+\boldsymbol{z}_s)$ and $f_\theta(^-\boldsymbol{z}_s)$. Here $f_\theta(\boldsymbol{z}_s) = \boldsymbol{z}_s \cdot \boldsymbol{w}_\theta \in \mathbb{R}^{1\times F'}$, where $\boldsymbol{w}_\theta \in \mathbb{R}^{F\times F'}$ is the shared trainable parameters and F' is the hidden dimension of the representation. Then we build the contrastive objective of IDOL as follows:

$$\mathcal{L}_{IDOL} = \frac{1}{n}\left(\sum_{s=1}^{n_g} f_\theta(\boldsymbol{z}_s) f_\theta(^-\boldsymbol{z}_s)^\top - \sum_{s=n_g+1}^{n} f_\theta(\boldsymbol{z}_s) f_\theta(^+\boldsymbol{z}_s)^\top\right), \quad (6)$$

where we denote the index of nodes which have the negative samples as $\{1, 2, ...n_g\}$, and the index with respect to positive samples as $\{n_g + 1, n_g + 2, ...n_g + n_p\}$. The overall number of positive and negative samples is identical to n, which is significantly smaller than that of previous loss functions such as InfoNCE in Eq. 1. Note that since there is no historical embedding to augment the initial graph $\mathcal{G}_0$, we follow the work in [58, 72] and shuffle the node order in $\boldsymbol{X}$ to generate negative samples, where we have $n_p = 0, n_g = n$.

We further provide a theoretical perspective to prove the capability of topology-monitorable graph contrastive learning in downstream tasks. Considering the classical node classification task, we use $\boldsymbol{Y} = \{y_1, y_2, ..., y_n\}$ to denote the label set and we adopt the Cross-Entropy (CE) loss [11] as:

$$\mathcal{L}_{CE} = -\frac{1}{n}\sum_{s=1}^{n} \log \frac{\exp\left(f_\theta(\boldsymbol{z}_s) w_{y_s}^\top\right)}{\sum_{i=1}^{C} \exp\left(f_\theta(\boldsymbol{z}_s) w_i^\top\right)} \quad (7)$$

where w_i is the trainable weights for i-th class in the linear classifier utilized in downstream tasks and the weight matrix is denoted as $\boldsymbol{W}_C = \{w_1, w_2, ..., w_C\}$ given there are C classes.

The performance in the downstream tasks and the contrastive pretraining are demonstrated by the downstream task loss $\mathcal{L}_{CE}$ and the contrastive loss $\mathcal{L}_{IDOL}$ respectively. Then, we have the following upper bound of the downstream task loss $\mathcal{L}_{CE}$:

Lemma 4. *The loss* $\mathcal{L}_{CE}$ *can be bounded by the loss* $\mathcal{L}_{IDOL}$*:*

$$\mathcal{L}_{CE} \leq \mathcal{L}_{IDOL} + \log C + \frac{2}{3}a^2 \quad (8)$$

holds with probability at least $1 - O(1/n)$*. Here we assume* $\boldsymbol{X} \in [-1, 1]$ *after the feature normalization, and our graph encoder* $f_\theta(\cdot)$ *is Xavier [14] initialized using a uniform distribution* $\boldsymbol{w}_\theta \sim U(-a, a)$.

Regarding IDOL's decoupled architecture during graph propagation and pretraining, Lemma 4 hints a crucial point: the extent of pretraining considerably impacts the performance of the downstream tasks. Given that the value of $(\log C + \frac{2}{3}a^2)$ is relatively minor compared to the loss value, reducing $\mathcal{L}_{IDOL}$ can effectively enhance downstream task performance. This observation supports the fact that even a *single epoch* of pretraining with IDOL can yield high accuracy in downstream tasks. Additionally, the improved convergence of the contrastive loss further boosts the classification accuracy. This observation is validated in our hyperparameter analysis, as shown in Sec. 4.6 where we varied the number of training epochs in the encoder. Moreover, while this direct association between pretraining and downstream task performance is specific to IDOL, it could also shed light on the rationale behind the need for sufficient pretraining epochs in existing graph contrastive learning methods [44, 55, 58] in improving downstream task performance.

3.6 Complexity Analysis

Compared with the contrastive methods using GCN-based encoder such as [55, 58, 72, 75], IDOL takes less time to pretrain in the dynamic scenarios. Taking a K-layer [30] GCN encoder as an example, these methods need to take $O(4KnF^2 + 4KmF)$ time[4] per training step including forward propagation and backward propagation of the network. On the initial graph, IDOL directly computes the node embeddings by our *K-hop* algorithm, which only consumes $O(2KmF)$ time ($O(KmF)$ for negative sampling). In addition to the MLP encoder (optional as one layer), the complexity of IDOL on the initial graph per training step is $O(4nF^2 + 2KmF)$.

When incurring the update events, the GCN-encoder needs to conduct the graph propagation and network training from scratch, In contrast, the embedding update only consumes $O(KmF)$ time[5], since we reuse the historical embeddings as our positive and negative samples. For the complexity of distinguishing positive and negative samples, the *Reverse Push* consume $O(p/r^b_{max})$ time given p update events [4]. Since we set an empirical value as $r^b_{max} = 1/\log n$, the complexity for updating p edges is $O(4nF^2 + KmF + p\log n)$. The summary of the above analysis can be viewed in Table 1.

4 EXPERIMENTS

In this section, we evaluate IDOL to show its effectiveness and efficiency for contrastive learning in dynamic graphs. We conducted

[4]Here we assume for simplicity that the hidden size is F. This complexity includes $O(2KnF^2 + 2KmF)$ for positive encoder and $O(2KnF^2 + 2KmF)$ for negative encoder.

[5]In practice, the complexity of incremental update given p update events is much smaller than $O(KmF)$ when p is not large.

Table 2: Statistics of the datasets. F, C, and S stand for the dimension of attributes, the number of classes, and the number of edge sequences.

Datasets	n	m	F	C	S
Arxiv	169,343	1,157,799	128	40	16
Mag	736,389	10,792,672	128	349	16
Products	2,449,029	61,859,012	100	47	16
Patent	2,738,012	13,960,811	128	6	16

the experiments on a Linux machine with an Intel(R) Xeon(R) Gold 6238R CPU @ 2.20GHz with 160GB RAM and an NVIDIA RTX A5000 with 24GB memory. We provide an open-source implementation of our model IDOL at https://github.com/ZulunZhu/dynamic-contrastive-learning.git. More interesting experiments and analysis can be found in our technique report [12].

4.1 Datasets

We conduct our experiments on four commonly used datasets: *Arxiv*, *Mag*, *Products* [23], and *Patent* [20]. These datasets are chosen to demonstrate the effectiveness of IDOL in various graph application scenarios, such as citation networks, web graphs, and recommendation systems. The statistics of four datasets are introduced in Table 2. For *Arxiv*, *Mag*, *Products*, the partition of training, validation, and testing set follows a chronological split in OGB [23]. For the dynamic dataset *Patent*, the training size is arranged as 70% including 10% for validation. All results shown are the averages from 10 runs.

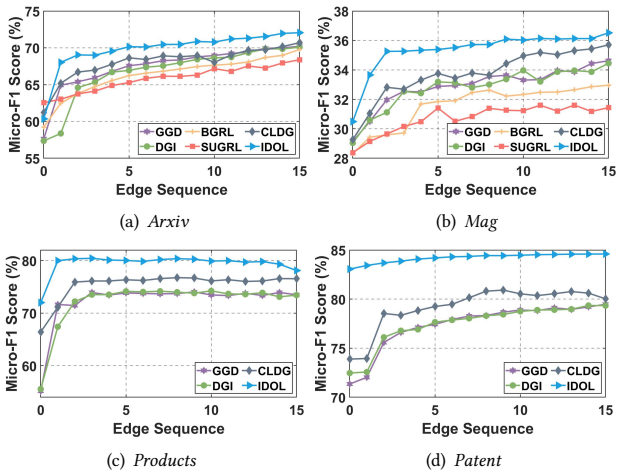
4.2 Baseline Methods

We compare IDOL with the following graph contrastive learning baselines: DGI[58],BGRL [55], GGD[72] and SUGRL [44]. We also compare IDOL with CLDG[64], a state-of-the-art contrastive learning method focusing on dynamic graphs. Furthermore, two classical GNN methods are included, GCN [30] and SGC [60], along with two scalable GNN methods, SCARA [36] and Instant [73]. To ensure reproducibility, we provide a detailed experiment setting in Appendix A.3. Specifically, for *Topology-monitorable Sampling*, we set the threshold $r^b_{max} = 1/\log n$ and $\lambda = 1/n$ by default in IDOL, where n is the node number of datasets. Then, for our two PPR-based algorithms, we set the teleport probability $\alpha = 0.1$ on *Arxiv*, *Mag* *Products*, and $\alpha = 0.2$ on *Patent*.

4.3 Comparison within CTDG settings

In our experiments, we employ the node classification task in dynamic graphs to evaluate the embedding quality of our method. In order to simulate the dynamic scenarios, we follow the experiment setting of CTDG in [73] to segment the whole graph and transform it into evolving states on these datasets. Formally, we divide the whole graph into an initial graph and S partitions of edge sequences for demonstration, where the edge sequences are extracted evenly from the complete graph. During the evaluation process, we progressively update the initial graph with S partitions of removed edges following the format of CTDG and evaluate all models after each addition of these partitions. Notably, the additions of removed edges are ordered according to the true timestamps.

Dynamic accuracy comparison. We first compare the accuracy performance of IDOL with the competitive baselines. For a clear demonstration, we evaluate the Macro-F1 and Micro-F1 scores and report the average values over all moments of evaluation, which is summarized in Table 3. It can be seen that IDOL outperforms previous contrastive learning methods (e.g., DGI, BGRL, SUGRL, GGD, and CLDG) with a non-trivial gap. Compared with the worst model DGI and the best CLDG, our method on average improves by 7.33% and 4.07% on *Products* dataset, respectively. We also note that IDOL surpasses traditional supervised GNN methods such as GCN, SGC, SCARA, and Instant, all of which are trained through an end-to-end supervised process without incorporating the contrastive learning mechanism. This finding reinforces the notion that the integration of the contrastive learning paradigm can significantly enhance the quality of embeddings.



(a) *Arxiv* (b) *Mag*

(c) *Products* (d) *Patent*

Figure 2: Micro-F1 scores in each snapshot of four datasets.

Moreover, we present the detailed performance of multiple contrastive learning methods when adding each edge sequence, which is shown in Figure 2. We adopt the Micro-F1 scores to measure this comparison and we have the following key observation: *The prediction performance of IDOL outperforms the state-of-the-art methods across four datasets.* It is worth noting that, IDOL achieves comparable or even better results on each snapshot of all datasets, where IDOL even outperforms the advanced methods DGI and GGD by over 16.5% on the initial graph of *Products* datasets. Notice that for *Products* and *Patent* datasets, IDOL substantially outperforms the baselines by a large margin, including the competitive method CLDG. The notable performance advantage of IDOL can be credited to its use of an efficient full-graph processing approach for training, in contrast to other methods that require batch processing and neighbor sampling in large graphs. This approach, which fully integrates topology information, significantly enhances performance.

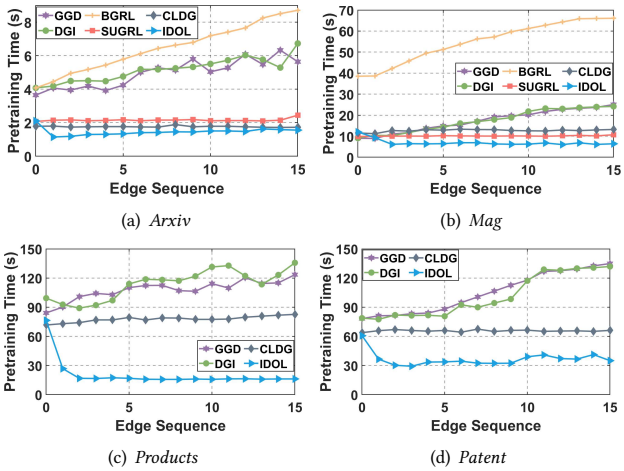
Dynamic efficiency comparison. To demonstrate the efficiency of IDOL, we investigate the time consumption of each contrastive learning method on each dataset. Table 4 reports the average pretraining time (per epoch) over all snapshots. Similarly, we also provide the detailed results on all datasets for each evaluation in Figure 3. For the initial graph, we observe that IDOL shares a close time consumption as baseline methods, which includes the time of embedding calculation and network training. Nevertheless, in the later addition of edge sequences, IDOL drastically

Table 3: The prediction accuracy (%) on four datasets. "OOM" stands for out of memory on a GPU with 24GB memory. The best results are underlined and bold.

Method	Arxiv		Mag		Products		Patent	
	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
GCN	68.42 ± 0.32	47.24 ± 0.29	33.71 ± 0.33	13.72 ± 0.27	74.43 ± 0.17	35.62 ± 0.21	77.60 ± 0.55	78.01 ± 0.52
SGC	68.54 ± 0.37	47.47 ± 0.27	32.66 ± 0.25	14.38 ± 0.26	74.54 ± 0.26	35.82 ± 0.17	77.81 ± 0.49	78.26 ± 0.36
SCARA	68.89 ± 0.26	48.19 ± 0.20	33.36 ± 0.29	14.82 ± 0.17	78.13 ± 0.25	38.42 ± 0.19	81.51 ± 0.73	81.91 ± 0.61
Instant	68.14 ± 0.12	48.01 ± 0.10	33.54 ± 0.21	14.68 ± 0.18	77.86 ± 0.11	38.03 ± 0.16	81.24 ± 0.65	81.48 ± 0.66
DGI	66.69 ± 0.54	43.28 ± 0.31	32.78 ± 0.41	12.94 ± 0.29	72.11 ± 0.29	34.20 ± 0.20	75.45 ± 0.94	76.01 ± 0.31
BGRL	66.31 ± 0.47	43.42 ± 0.24	31.59 ± 0.87	12.58 ± 0.33	OOM	OOM	OOM	OOM
SUGRL	65.81 ± 0.94	42.53 ± 0.88	30.71 ± 0.75	12.14 ± 0.31	OOM	OOM	OOM	OOM
GGD	67.45 ± 0.36	44.10 ± 0.25	32.89 ± 0.29	13.43 ± 0.16	72.35 ± 0.31	35.08 ± 0.13	75.36 ± 0.21	75.93 ± 0.44
CLDG	68.05 ± 0.87	46.15 ± 0.56	33.73 ± 0.75	14.70 ± 0.22	75.37 ± 0.50	37.47 ± 0.81	77.18 ± 0.87	77.64 ± 0.25
IDOL	**69.80 ± 0.21**	**49.44 ± 0.12**	**35.34 ± 0.20**	**16.51 ± 0.15**	**79.44 ± 0.12**	**40.13 ± 0.12**	**84.19 ± 0.87**	**84.11 ± 0.75**

Table 4: The pretraining time (s) per epoch on four datasets.

Method	Arxiv	Mag	Products	Patent
DGI	5.15 ± 0.75	18.42 ± 1.67	113.06 ± 5.52	101.46 ± 2.65
BGRL	6.47 ± 0.50	54.93 ± 5.29	OOM	OOM
SUGRL	2.16 ± 0.04	10.14 ± 1.19	OOM	OOM
GGD	4.94 ± 0.44	17.34 ± 1.56	105.49 ± 3.23	104.96 ± 2.39
CLDG	2.00 ± 0.37	12.70 ± 2.19	77.77 ± 2.74	65.68 ± 3.14
IDOL	**1.76 ± 0.18**	**9.02 ± 1.02**	**20.67 ± 2.15**	**36.57 ± 1.65**



(a) *Arxiv* (b) *Mag*

(c) *Products* (d) *Patent*

Figure 3: Pretraining time comparison.

achieves orders-of-magnitude acceleration, and the gap between IDOL and baseline methods becomes even more pronounced as the graph scale increases. Especially, in the full graph of *Products*, GGD and DGI require over 120 seconds each epoch for pretraining, while IDOL completes this process in just 16.2 seconds. Even faced with suboptimal baseline CLDG, IDOL can still achieve up to $(77.77 - 20.67)/77.77 = 73.42\%$ reduction of pertaining time on average. It's important to highlight that both IDOL and GGD can achieve high prediction accuracy reported in Figure 2 with only one epoch, owing to their rapid convergence. Conversely, DGI, BGRL, SUGRL, and CLDG require over 50 epochs to reach a competitive performance, showing a substantial disparity in training efficiency.

Table 5: The average prediction accuracy (%) on Mooc and Reddit datasets for 10 runs. The average time consumption (s) of pretraining time is enclosed in the bracket. (/) means the corresponding algorithm requires no pretraining stage on this task. The best results are underlined and bold.

Method	Node Classification		Link Prediction	
	Mooc	Reddit	Mooc	Reddit
JODIE [33]	61.53(/)	53.64(/)	94.53(/)	95.78(/)
TGN [51]	**69.03**(65.65s)	**55.53**(128.86s)	98.24(/)	98.76(/)
DDGCL [56]	57.11(59.65s)	52.79(101.89s)	98.11(89.65s)	98.17(189.11s)
IDOL	67.34(**1.64s**)	55.21(**6.64s**)	**99.07**(**3.98s**)	**98.97**(**10.93s**)

The remarkable performance of IDOL can be attributed to two primary factors. First, IDOL efficiently saves pretraining time by incorporating edge updates and implementing incremental embedding refreshment. Additionally, its straightforward MLP structure simplifies the networks and further enhances training efficiency. Secondly, our topology-monitorable sampling method, designed to capture topological changes from an evolving perspective, directly contributes to potential improvements in the quality of positive and negative pairs. Collectively, these two factors underpin IDOL's exceptional efficiency and effectiveness, respectively.

4.4 Comparison within DTDG Settings

In the above experiments, we adopt the CTDG setting to evaluate the performance of IDOL on large-scale datasets. It's worth noting, though, that some DTDG-based algorithms, which don't scale well in these large-scale datasets, can still perform competitively on smaller datasets by leveraging the fine-grained time information. To further establish IDOL's versatility, we conducted a comparative analysis of IDOL against a selection of prominent algorithms on two smaller datasets, Mooc [1] and Reddit [2]. In this experiment, we follow the DTDG setting outlined in existing literature [32, 50, 56], which focuses on predictions for the fully-formed final graph. The results of *node classification* and *link prediction* tasks are summarized in Table 5. Based on this result, IDOL demonstrates broad applicability across various graph types and downstream tasks, achieving comparable prediction accuracy while necessitating shorter pretraining durations. This underscores IDOL's capacity for generalization and efficiency in diverse graph environments.

Table 6: Ablation studies of *K-hop Push* on the *Patent* dataset. $\mathcal{F}(x)$ denotes Forward Push with $r_{max} = x$, and $\mathcal{K}(x), \mathcal{S}(x)$ denotes *K-hop Push* and propagation using SGC with $K = x$ respectively. w/o. means without.

Variants	w/o.	$\mathcal{F}(10^{-5})$	$\mathcal{F}(10^{-7})$	$\mathcal{S}(2)$	$\mathcal{S}(4)$	$\mathcal{K}(2)$	$\mathcal{K}(4)$
Time(s)	0	0.10	37.06	15.14	57.54	7.43	36.57
Accuracy(%)	19.65	29.24	81.24	73.95	77.81	83.85	84.19

Table 7: Ablation studies on sampling strategies. $\tau^+ = i$ and $\tau^- = i$ mean selecting previous i-th snapshot as the positive and negative samples respectively. Corr. means the random corruption. w/o. means without.

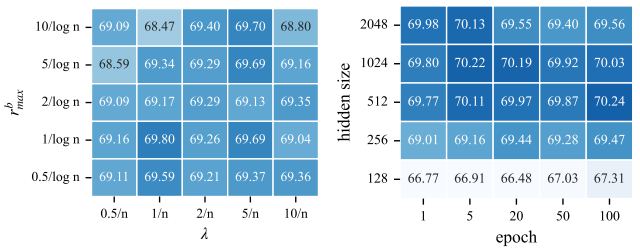
Variants	w/o.	Corr.	$\tau^+ = 1$	$\tau^+ = 5$	$\tau^- = 5$	$\tau^- = 10$	IDOL
Arxiv	69.24	69.36	70.49	69.94	70.26	70.38	72.41
Mag	32.22	33.37	35.64	33.44	34.56	34.74	36.38
Product	73.99	74.87	76.89	75.67	76.26	76.64	78.11

4.5 Ablation Studies

In this section, we conduct ablation studies on the variants of IDOL to verify the effectiveness of two key modules comprehensively: *K-hop Push* and *Topology-monitorable Sampling*.

K-hop Push. On *Patent* datasets, we replace *K-hop Push* module with the conventional *Forward Push* algorithm [8, 73] which utilizes an empirical threshold r_{max} to early stop the iteration of graph propagation. Moreover, we also replace *K-hop Push* with the propagation method in SGC [60] for a comparison. As shown in Table 6, *Forward Push* and *K-hop Push* require more propagation time with a smaller r_{max} and a larger K, respectively. Nevertheless, *K-hop Push* consistently achieves higher accuracy within a comparable time frame. Furthermore, the notable variability in performance with *Forward Push* implies the need for careful selection of the r_{max} threshold, while *K-hop Push* proves to be more stable across various settings of hop number K.

Topology-monitorable Sampling. We compare our sampling strategy with the time-based strategy in [56], which adopts different time windows to sample the positive and negative pairs. To generate different variants, we replace the positive or negative samples in *Topology-monitorable Sampling* by directly sampling from all nodes in the previous τ^+-th or τ^--th snapshot, respectively. Additionally, we also provide a sampling strategy named random corruption for comparison, which disturbs the graph topology for augmentation and is widely used in previous works [55, 58, 72]. As indicated in Table 7, we evaluate different sampling strategies on the final snapshot of three datasets. It is evident that, compared with our method, the utilization of time-based strategies consistently results in a decline in prediction accuracy. Additionally, this decline in accuracy is tied to the incorrect categorization of negative and positive samples. For example, on the *Arxiv* dataset, accuracy falls from 70.49% to 69.94% when τ^+ changes from 1 to 5. This is because, over a longer time window, the graph's structure changes more significantly, causing more samples to be mistakenly identified as positive. This supports the importance of sampling positive and negative pairs based on topological information, as it proves beneficial for enhancing correct sampling.



(a) r^b_{max} and λ (b) Hidden size and pretraining epoch

Figure 4: Hyperparameter analysis.

4.6 Hyperparameter Analysis

We further analyze the hyperparameter sensitivity of IDOL. The experimental results on *Arxiv* dataset are reported and we have similar observations on the other datasets.

Threshold r^b_{max} and λ in *Dynamic Sampling*. To achieve topology-monitorable sampling, we employ r^b_{max} as the early-stopping threshold in the Reverse Push algorithm and further utilize an empirical gap value λ to distinguish the positive samples from historical embeddings. As shown in Figure 4(a), we present the average accuracy on *Arxiv* by varying the values of r^b_{max} and λ as $\{0.5/\log n, 1/\log n, 2/\log n, 5/\log n, 10/\log n\}$ and $\{0.5/n, 1/n, 2/n, 5/n, 10/n\}$, respectively. It is observed that various settings typically do not significantly affect IDOL, as IDOL consistently delivers its best performance using our default configuration.

Hidden size and pretraining epoch. We investigate the IDOL variants with different hidden sizes and pretraining epochs, as illustrated in Figure 4(b). For a fair comparison, we start this study based on the default setting of the hidden size and epoch number (e.g., 1024 and 1) in the aforementioned experiments. When the hidden size exceeds 512, the results tend to converge closely. However, reducing the hidden size to 128 leads to a notable decline in classification performance. The reason might be that low-dimensional hidden vectors cannot effectively represent large-scale graph information. Furthermore, we can naturally produce a better performance by utilizing more epochs. Since a longer pretraining period will further decrease $\mathcal{L}_{IDOL}$, which enhances the prediction accuracy of the downstream task.

5 CONCLUSION

In this paper, we propose IDOL, a first-ever topology-monitorable contrastive learning framework focusing on dynamic graphs. Given the graph updates, IDOL employs PPR-based techniques to incrementally update node embeddings and uses a strategy that monitors topology changes to select positive and negative pairs. Besides, we employ a simplified training paradigm and provide a performance guarantee by bridging the contrastive loss and the downstream task loss. In terms of time efficiency and prediction accuracy, we conduct extensive experiments on various graph datasets and verify that IDOL is superior to state-of-the-art methods.

ACKNOWLEDGMENTS

This research is supported by the Ministry of Education, Singapore, under its AcRF Tier-2 Grant (T2EP20122-0003). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of the Ministry of Education, Singapore.

REFERENCES

[1] Kdd cup 2015. https://biendata.com/competition/kddcup2015/data/.

[2] Reddit data dump. http://files.pushshift.io/reddit/.

[3] Mohammad Ali Alomrani, Mahdi Biparva, Yingxue Zhang, and Mark Coates. 2022. DyG2Vec: Representation Learning for Dynamic Graphs with Self-Supervision. *arXiv preprint arXiv:2210.16906* (2022).

[4] Reid Andersen, Christian Borgs, Jennifer Chayes, John Hopcraft, Vahab S Mirrokni, and Shang-Hua Teng. 2007. Local computation of pagerank contributions. In *Algorithms and Models for the Web-Graph: 5th International Workshop, WAW 2007, San Diego, CA, USA, December 11-12, 2007. Proceedings 5*. Springer, 150–165.

[5] Reid Andersen, Fan Chung, and Kevin Lang. 2006. Local graph partitioning using pagerank vectors. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*. IEEE, 475–486.

[6] Yuanchen Bei, Hao Xu, Sheng Zhou, Huixuan Chi, Mengdi Zhang, Zhao Li, and Jiajun Bu. 2023. CPDG: A Contrastive Pre-Training Method for Dynamic Graph Neural Networks. *arXiv preprint arXiv:2307.02813* (2023).

[7] Deyu Bo, BinBin Hu, Xiao Wang, Zhiqiang Zhang, Chuan Shi, and Jun Zhou. 2022. Regularizing graph neural networks via consistency-diversity graph augmentations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 3913–3921.

[8] Aleksandar Bojchevski, Johannes Gasteiger, Bryan Perozzi, Amol Kapoor, Martin Blais, Benedek Rózemberczki, Michal Lukasik, and Stephan Günnemann. 2020. Scaling graph neural networks with approximate pagerank. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2464–2473.

[9] Sergey Brin. 1998. The PageRank citation ranking: bringing order to the web. *Proceedings of ASIS, 1998* 98 (1998), 161–172.

[10] Liang Chen, Jiaying Peng, Yang Liu, Jintang Li, Fenfang Xie, and Zibin Zheng. 2020. Phishing scams detection in ethereum transaction network. *ACM Transactions on Internet Technology (TOIT)* 21, 1 (2020), 1–16.

[11] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*. PMLR, 1597–1607.

[12] Zulun Zhu et al. 2024. https://sites.google.com/view/idol-tech/.

[13] Wenzheng Feng, Jie Zhang, Yuxiao Dong, Yu Han, Huanbo Luan, Qian Xu, Qiang Yang, Evgeny Kharlamov, and Jie Tang. 2020. Graph random neural networks for semi-supervised learning on graphs. *Advances in neural information processing systems* 33 (2020), 22092–22103.

[14] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 249–256.

[15] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. 2020. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems* 33 (2020), 21271–21284.

[16] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. 2019. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 922–929.

[17] Xingzhi Guo, Baojian Zhou, and Steven Skiena. 2022. Subset node anomaly tracking over large dynamic graphs. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 475–485.

[18] Pankaj Gupta, Ashish Goel, Jimmy Lin, Aneesh Sharma, Dong Wang, and Reza Zadeh. 2013. Wtf: The who to follow service at twitter. In *Proceedings of the 22nd international conference on World Wide Web*. 505–514.

[19] Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 297–304.

[20] Bronwyn H Hall, Adam B Jaffe, and Manuel Trajtenberg. 2001. The NBER patent citation data file: Lessons, insights and methodological tools.

[21] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 639–648.

[22] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. 2018. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670* (2018).

[23] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems* 33 (2020), 22118–22133.

[24] Yue Hu, Yuhang Zhang, Yanbing Wang, and Daniel Work. 2023. Detecting Socially Abnormal Highway Driving Behaviors via Recurrent Graph Attention Networks. In *Proceedings of the ACM Web Conference 2023*. 3086–3097.

[25] Ziniu Hu, Changjun Fan, Ting Chen, Kai-Wei Chang, and Yizhou Sun. 2019. Pre-training graph neural networks for generic structural feature extraction. *arXiv preprint arXiv:1905.13728* (2019).

[26] Jiawei Jiang, Pin Xiao, Lele Yu, Xiaosen Li, Jiefeng Cheng, Xupeng Miao, Zhipeng Zhang, and Bin Cui. 2020. PSGraph: How Tencent trains extremely large-scale graphs with Spark?. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 1549–1557.

[27] Wei Jin, Tyler Derr, Haochen Liu, Yiqi Wang, Suhang Wang, Zitao Liu, and Jiliang Tang. 2020. Self-supervised learning on graphs: Deep insights and new direction. *arXiv preprint arXiv:2006.10141* (2020).

[28] Xin Ju, Xiaofeng Zhang, and William K Cheung. 2019. Generating synthetic graphs for large sensitive and correlated social networks. In *2019 IEEE 35th international conference on data engineering workshops (ICDEW)*. IEEE, 286–293.

[29] Dongkwan Kim and Alice Oh. 2021. How to Find Your Friendly Neighborhood: Graph Attention Design with Self-Supervision. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net. https://openreview.net/forum?id=Wi5KUNlqWty

[30] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR (Poster)*. OpenReview.net.

[31] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *ICLR (Poster)*. OpenReview.net.

[32] Srijan Kumar, Xikun Zhang, and Jure Leskovec. 2019. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 1269–1278.

[33] Srijan Kumar, Xikun Zhang, and Jure Leskovec. 2019. Predicting Dynamic Embedding Trajectory in Temporal Interaction Networks. In *KDD*. ACM, 1269–1278.

[34] Jong-whi Lee and Jinhong Jung. 2023. Time-aware random walk diffusion to improve dynamic graph learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 8473–8481.

[35] Jintang Li, Zhouxin Yu, Zulun Zhu, Liang Chen, Qi Yu, Zibin Zheng, Sheng Tian, Ruofan Wu, and Changhua Meng. 2023. Scaling Up Dynamic Graph Representation Learning via Spiking Neural Networks. In *AAAI*. AAAI Press, 8588–8596.

[36] Ningyi Liao, Dingheng Mo, Siqiang Luo, Xiang Li, and Pengcheng Yin. 2022. SCARA: Scalable Graph Neural Networks with Feature-Oriented Optimization. *Proc. VLDB Endow.* 15, 11 (2022), 3240–3248.

[37] David C Liu, Stephanie Rogers, Raymond Shiau, Dmitry Kislyuk, Kevin C Ma, Zhigang Zhong, Jenny Liu, and Yushi Jing. 2017. Related pins at pinterest: The evolution of a real-world recommender system. In *Proceedings of the 26th international conference on world wide web companion*. 583–592.

[38] Haoyu Liu and Siqiang Luo. 2024. BIRD: Efficient Approximation of Bidirectional Hidden Personalized PageRank. *PVLDB* 17, 9 (2024), 2255–2268. https://doi.org/10.14778/3665844.3665855

[39] Yixin Liu, Ming Jin, Shirui Pan, Chuan Zhou, Yu Zheng, Feng Xia, and S Yu Philip. 2022. Graph self-supervised learning: A survey. *IEEE Transactions on Knowledge and Data Engineering* 35, 6 (2022), 5879–5900.

[40] Yuanfu Lu, Xiao Wang, Chuan Shi, Philip S. Yu, and Yanfang Ye. 2019. Temporal Network Embedding with Micro- and Macro-dynamics. In *CIKM*. ACM, 469–478.

[41] Siqiang Luo, Xiaokui Xiao, Wenqing Lin, and Ben Kao. 2019. Baton: Batch one-hop personalized pageranks with efficiency and accuracy. *IEEE Transactions on Knowledge and Data Engineering* 32, 10 (2019), 1897–1908.

[42] Dingheng Mo and Siqiang Luo. 2021. Agenda: Robust Personalized PageRanks in Evolving Graphs. In *CIKM*. ACM, 1315–1324.

[43] Dingheng Mo and Siqiang Luo. 2023. Single-Source Personalized PageRanks With Workload Robustness. *IEEE Trans. Knowl. Data Eng.* 35, 6 (2023), 6320–6334.

[44] Yujie Mo, Liang Peng, Jie Xu, Xiaoshuang Shi, and Xiaofeng Zhu. 2022. Simple unsupervised graph representation learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 7797–7805.

[45] Kenta Oono and Taiji Suzuki. 2020. Graph Neural Networks Exponentially Lose Expressive Power for Node Classification. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. https://openreview.net/forum?id=S1ldO2EFPr

[46] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748* (2018).

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

[48] Jiwoong Park, Minsik Lee, Hyung Jin Chang, Kyuewang Lee, and Jin Young Choi. 2019. Symmetric graph convolutional autoencoder for unsupervised graph representation learning. In *Proceedings of the IEEE/CVF international conference on computer vision*. 6519–6528.

[49] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. 2020. Gcc: Graph contrastive coding for graph neural network pre-training. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 1150–1160.

[50] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. 2020. Temporal Graph Networks for Deep Learning on Dynamic Graphs. In *ICML 2020 Workshop on Graph Representation Learning*.

[51] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. 2020. Temporal Graph Networks for Deep Learning on Dynamic Graphs. In *ICML 2020 Workshop on Graph Representation Learning*.

[52] Anshul Shah, Suvrit Sra, Rama Chellappa, and Anoop Cherian. 2022. Max-margin contrastive learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 8220–8230.

[53] Uriel Singer, Ido Guy, and Kira Radinsky. 2019. Node Embedding over Temporal Graphs. In *IJCAI*. ijcai.org, 4605–4612.

[54] Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Mehdi Azabou, Eva L Dyer, Remi Munos, Petar Veličković, and Michal Valko. 2021. Large-scale representation learning on graphs via bootstrapping. *arXiv preprint arXiv:2102.06514* (2021).

[55] Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Rémi Munos, Petar Veličković, and Michal Valko. 2021. Bootstrapped representation learning on graphs. In *ICLR 2021 Workshop on Geometrical and Topological Representation Learning*.

[56] Sheng Tian, Ruofan Wu, Leilei Shi, Liang Zhu, and Tao Xiong. 2021. Self-supervised representation learning on dynamic graphs. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 1814–1823.

[57] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. 2019. DyRep: Learning Representations over Dynamic Graphs. In *ICLR (Poster)*. OpenReview.net.

[58] Petar Velickovic, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. 2019. Deep Graph Infomax. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. https://openreview.net/forum?id=rklz9iAcKQ

[59] Sibo Wang, Renchi Yang, Xiaokui Xiao, Zhewei Wei, and Yin Yang. 2017. FORA: simple and effective approximate single-source personalized pagerank. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 505–514.

[60] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *International conference on machine learning*. PMLR, 6861–6871.

[61] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. 2020. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 753–763.

[62] Yaochen Xie, Zhao Xu, Jingtun Zhang, Zhengyang Wang, and Shuiwang Ji. 2022. Self-supervised learning of graph neural networks: A unified review. *IEEE transactions on pattern analysis and machine intelligence* 45, 2 (2022), 2412–2429.

[63] Da Xu, Chuanwei Ruan, Evren Körpeoglu, Sushant Kumar, and Kannan Achan. 2020. Inductive representation learning on temporal graphs. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. https://openreview.net/forum?id=rJeW1yHYwH

[64] Yiming Xu, Bin Shi, Teng Ma, Bo Dong, Haoyi Zhou, and Qinghua Zheng. 2023. CLDG: Contrastive Learning on Dynamic Graphs. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 696–707.

[65] Menglin Yang, Min Zhou, Marcus Kalander, Zengfeng Huang, and Irwin King. 2021. Discrete-time temporal network embedding via implicit hierarchical learning in hyperbolic space. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 1975–1985.

[66] Jiaxuan You, Tianyu Du, and Jure Leskovec. 2022. ROLAND: Graph Learning Framework for Dynamic Graphs. In *KDD*. ACM, 2358–2366.

[67] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. *Advances in neural information processing systems* 33 (2020), 5812–5823.

[68] Wenchao Yu, Wei Cheng, Charu C. Aggarwal, Kai Zhang, Haifeng Chen, and Wei Wang. 2018. NetWalk: A Flexible Deep Embedding Approach for Anomaly Detection in Dynamic Networks. In *KDD*. ACM, 2672–2681.

[69] Hengrui Zhang, Qitian Wu, Junchi Yan, David Wipf, and Philip S Yu. 2021. From canonical correlation analysis to self-supervised graph neural networks. *Advances in Neural Information Processing Systems* 34 (2021), 76–89.

[70] Wentao Zhang, Mingyu Yang, Zeang Sheng, Yang Li, Wen Ouyang, Yangyu Tao, Zhi Yang, and Bin Cui. 2021. Node dependent local smoothing for scalable graph learning. *Advances in Neural Information Processing Systems* 34 (2021), 20321–20332.

[71] Tong Zhao, Wei Jin, Yozen Liu, Yingheng Wang, Gang Liu, Stephan Günnemann, Neil Shah, and Meng Jiang. 2022. Graph data augmentation for graph machine learning: A survey. *arXiv preprint arXiv:2202.08871* (2022).

[72] Yizhen Zheng, Shirui Pan, Vincent Lee, Yu Zheng, and Philip S Yu. 2022. Rethinking and scaling up graph contrastive learning: An extremely efficient approach with group discrimination. *Advances in Neural Information Processing Systems* 35 (2022), 10809–10820.

[73] Yanping Zheng, Hanzhi Wang, Zhewei Wei, Jiajun Liu, and Sibo Wang. 2022. Instant graph neural networks for dynamic graphs. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2605–2615.

[74] Changtao Zhong, Mostafa Salehi, Sunil Shah, Marius Cobzarenco, Nishanth Sastry, and Meeyoung Cha. 2014. Social bootstrapping: how pinterest and last. fm social communities benefit by borrowing links from facebook. In *Proceedings of the 23rd international conference on World wide web*. 305–314.

[75] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2020. Deep graph contrastive representation learning. *arXiv preprint arXiv:2006.04131* (2020).

[76] Zulun Zhu, Siqiang Luo, Wenqing Lin, Sibo Wang, Dingheng Mo, and Chunbo Li. 2024. Personalized PageRanks over Dynamic Graphs – The Case for Optimizing Quality of Service. In *ICDE*. IEEE.

[77] Zulun Zhu, Jiaying Peng, Jintang Li, Liang Chen, Qi Yu, and Siqiang Luo. 2022. Spiking Graph Convolutional Networks. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, Luc De Raedt (Ed.). ijcai.org, 2434–2440. https://doi.org/10.24963/ijcai.2022/338

[78] Zulun Zhu, Sibo Wang, Siqiang Luo, Dingheng Mo, Wenqing Lin, and Chunbo Li. [n. d.]. Personalized PageRanks over Dynamic Graphs–The Case for Optimizing Quality of Service.

[79] Yuan Zuo, Guannan Liu, Hao Lin, Jia Guo, Xiaoqian Hu, and Junjie Wu. 2018. Embedding Temporal Network via Neighborhood Formation. In *KDD*. ACM, 2857–2866.

A APPENDIX

A.1 Summary of Notations

The main notations used in this paper and their descriptions are summarized in Table 8.

Table 8: Frequently used notations in this paper.

Notations	Descriptions
$\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$	Directed graph after i edge updates
n, m	Numbers of nodes and edges in the current graph
F	Dimension of the node attribute
K	The layer of graph propagation
$\boldsymbol{X}$	The node attribute matrix and $\boldsymbol{X} \in \mathbb{R}^{n\times F}$
$\boldsymbol{x}_i$	The attribute vector of the i-th dimension and $\boldsymbol{x}_i \in \mathbb{R}^{n\times 1}$
$\mathcal{N}_{out}(v)$, $\mathcal{N}_{in}(v)$	The out-neighbors and in-neighbors of v
Y	The label set, where the element $y_i \in \mathbb{R}$
$e_i = \{u_i, v_i\}$	The i-th update event related to node u_i and v_i
$\pi(G_i, s, t)$	PPR of node t from the source node s in graph G_i
$\boldsymbol{r}_i$	Residue vector of the i-th dimension and $\boldsymbol{r}_i \in \mathbb{R}^{n\times 1}$
$\boldsymbol{h}_i$	Reserve vector of the i-th dimension and $\boldsymbol{h}_i \in \mathbb{R}^{n\times 1}$
$\boldsymbol{z}_s$	The node embedding vector of node s and $\boldsymbol{z}_s \in \mathbb{R}^{1\times F}$
$^{-}\boldsymbol{z}_s, ^{+}\boldsymbol{z}_s$	The negative and positive sample vectors of node s and $^{-}\boldsymbol{z}_s, ^{+}\boldsymbol{z}_s \in \mathbb{R}^{1\times F}$
α	Teleport probability of random walks
r^b_{max}	Threshold in *Reverse Push*

A.2 More Related Work

A.2.1 Dynamic Graph Representation Learning. There exist two major types of approaches, including the methods focusing on DTDG and CTDG.

Discrete-Time Dynamic Graph. Representation learning over dynamic graphs primarily focused on discrete-time dynamic graphs (DTDGs), which are represented by a series of graph snapshots captured at discrete time points. Literature on DTDGs has been extensively studied due to ease of implementation. Many architectures for DTDGs are based on combining static GNNs with sequence models, such as recurrent neural networks (RNNs), which can be applied recurrently to the network parameters [47], hierarchical node states [66], or to a set of embeddings encoded from each graph snapshot [35]. Another line of research involves performing temporal random walks on graph snapshots to obtain dynamic node representations [53, 68].

Continuous-Time Dynamic Graph. Although DTDG methods have proven a powerful tool in learning the low-dimensional node representations for dynamic graphs, one limitation is their difficulty in capturing fine-grained temporal dynamics and continuous changes in the evolving structure. Recently, continuous-time dynamic graphs (CTDGs) have been widely studied, which consider graphs where the temporal evolution is modeled as a continuous process. Early works employ self-attention mechanisms to learn a dynamic node representation by attending over its neighbors and historical representations [56, 63]. Sequence models are also promising methods for CTDGs. For example, JODIE [33] and TGN [51] use RNNs to propagate messages across interactions to update node representations. In addition, temporal point processes (TPP) are applied for modeling continuous-time event sequences [40, 57, 79]. Unlike prior methods that recompute node embeddings from scratch for each prediction in dynamic scenarios, our approach utilizes incremental computation, optimizing both computation efficiency and memory usage.

A.2.2 Graph Contrastive Learning. Graph contrastive learning leverages the concept of positive and negative samples combined with a discrimination rule to learn the representation of nodes, capturing statistical dependencies and enhancing the meaningful patterns from graphs. DGI [58] is the first approach to maximize the mutual information between the node embeddings and graph embeddings, which enables the graph encoder to learn both local and global information. BYOL [15] and BGRL[55] attempt to get rid of the high overhead for sampling negative pairs and achieve promising performance when only utilizing positive pairs. GGD[72] simplifies the loss computation of DGI and conducts the pretraining only with a few epochs. Lastly, DDGCL[56] aims to capture the dynamics in the graphs and accommodate the temporal information during the process of positive sampling.

Due to such label scarcity, graph contrastive learning can also be used to improve the model quality by enhancing the consistency of different augmentation views. GRAND [13] generates different augmentation views of the graph by dropping nodes and masking features, and then the mutual information of the scarce labeled nodes based on different views is minimized to enhance the prediction performance. NASA [7] randomly replaces the 1-hop neighbors with 2-hop neighbors for the labels nodes and use a neighbor-constrained regularization to improve the consistency of adjacent nodes.

Table 9: Parameter settings. Here "lr" means the learning rate, "K" means the number of convolution layers, "hidden" means the hidden size of the network, and "batch number" means the number of neighbor sampling.

Datasets	lr	K	hidden	batch number
Arxiv	1e-3	4	1024	12
Mag	1e-3	4	512	12
Products	1e-3	4	512	12
Patent	1e-3	4	512	12

A.3 Experiment Settings and Extral results

We summarize the experimental setting of all baseline in Table 9. To provide enough comparison with the baselines, we employ the batched processing with neighbour sampling on GCN, SGC, DGI and GGD when processing on *Mag*, *Products* and *Patent* datasets. We keep the same setting with the common parameters like learning rate, K, hidden size. Following the work in [72], we set the pretraining epoch of IDOL and GGD as 1, and the epoch number of others is set as 200.