Self-supervised Representation Learning on Dynamic Graphs

Sheng Tian
Ant Group, China
tiansheng.ts@antgroup.com

Ruofan Wu
Ant Group, China
ruofan.wrf@antgroup.com

Leilei Shi
Ant Group, China
leilei.sll@antgroup.com

Liang Zhu
Ant Group, China
tailiang.zl@antgroup.com

Tao Xiong
Ant Group, China
weilue.xt@antgroup.com

Abstract

Graph representation learning has now become the de facto standard when dealing with graph-structured data. Using powerful tools from deep learning and graph neural networks, recent works have applied graph representation learning to time-evolving dynamic graphs and showed promising results. However, all the previous dynamic graph models require labeled samples to train, which might be costly to acquire in practice. Self-supervision offers a principled way of utilizing unlabeled data and has achieved great success in the computer vision community. In this paper, we propose debiased dynamic graph contrastive learning (DDGCL), the first self-supervised representation learning framework on dynamic graphs. The proposed model extends the contrastive learning idea to dynamic graphs via contrasting two nearby temporal views of the same node identity, with a time-dependent similarity critic. Inspired by recent theoretical developments in contrastive learning, we propose a novel debiased GAN-type contrastive loss as the learning objective to correct the sampling bias that occurred in the negative sample construction process. We conduct extensive experiments on benchmark datasets via testing the DDGCL framework under two different self-supervision schemes: pretraining and finetuning and multi-task learning. The results show that using a simple time-aware GNN encoder, the performance of the downstream tasks is significantly improved under either scheme to closely match, or even outperform state-of-the-art dynamic graph models with more elegant encoder architectures. Further empirical evaluations suggest that the proposed approach offers more performance improvement than previously established self-supervision mechanisms over static graphs.

CCS Concepts

• **Computing methodologies → Unsupervised learning**.

Keywords

Dynamic graphs; contrastive learning; self-supervision

ACM Reference Format:
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 and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, QLD, Australia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3459637.3482389

1 Introduction

The area of graph representation learning [5] has attracted much attention over the last few years. With the surging development of graph neural networks [9], graph representation learning has achieved great success over a wide range of tasks such as node classification [21], graph classification [8] and link prediction [13]. Most of the graph representation learning algorithms operate on static graphs, producing low-dimensional node embeddings that reflect the local structure of underlying nodes. While such approaches have shown to be both simple and effective, they do not apply to *dynamic*, or *temporal graphs* that evolve over time [24]. Naive application of static graph learning paradigms such as message-passing style GNN to dynamic graphs fails to capture the temporal information and was shown to be sub-optimal in dynamic graph scenarios [46].

Prior work on representation learning over dynamic graphs preprocesses the dynamic graphs into a sequence of snapshots, and use recurrent or attentive architectures to combine intermediate representations extracted via GNN type encoders [12, 30, 36], usually referred to as *discrete-time* methods [20]. The primal drawback of discrete-time approaches is that the right granularity for temporal discretization is often subtle, leading to coarsely captured temporal information. A recent line of work [35, 40, 43, 46] explored *continuous-time* approaches to integrate temporal information directly into the representation learning process and were shown to significantly improve time-independent and discrete-time approaches on standard datasets. Despite the success of these time-dependent proposals, all of them still require labeled data to train. However, it is often very costly to obtain annotated labels for graph-related tasks since they rely heavily on domain knowledge [38]. Hence to efficiently utilize the abundant unlabeled data, unsupervised or semi-supervised approaches are sometimes preferable.

Recently, self-supervision [25] has emerged as a principled way of obtaining useful representation without the need for human annotations. In self-supervised learning, representation vectors are learned via a predictive task (usually referred to as *pretext task*) with its prediction objective derived from unlabeled data. There are various ways to define pretext tasks, among which *instance discrimination* trained using a *contrastive objective* [14, 44] has received considerable success in the domain of computer vision [6]. The idea of contrastive learning is to learn representations that preserve *similarity*, via contrasting two or more semantically coherent views (obtained via data augmentations [6]) of the same underlying object. As notions of similarity exist in many domains, contrastive approaches have been shown to perform competitively in areas like natural language processing [26].

Early attempts on applying self-supervision to graph domain adopted reconstruction-based primitives [13, 22] and required carefully designed decoders. A more recent line of work [17, 32, 42, 48] explored the generalization of contrastive framework to graph representation learning. It was reported that node representations trained with contrastive learning [42] along with a simple linear classifier produced comparable performance with end-to-end trained graph neural networks. To the best of our knowledge, all the previous attempts on self-supervised graph representation learning focused on only static graphs. It is thus of interest to investigate how representation learning over dynamic graphs benefits from self-supervision.

Extending self-supervised learning frameworks on static graphs to dynamic setups is highly nontrivial. The only strategy we are aware of is to use the dynamic link prediction [20] as the pretext task for downstream tasks like dynamic node classification [35]. This could be regarded as an analog of a reconstruction-based pretraining strategy in dynamic graph learning. As argued in [42, 48], this strategy over-emphasizes *proximity*, and might not be beneficial even in static graph scenarios. Additionally, we identify two challenges of self-supervised dynamic graph representation learning: Firstly, previous empirical observations [48] suggest that the effectiveness of different graph data augmentation methods depends on the underlying graph data. Consequently, the learned representations may be less transferable compared with computer vision setups. Secondly, recent theoretical developments in contrastive learning [37] discovered the fact that negative samples which are "false negatives" hurt the generalization ability of contrastive frameworks. Hence it is worthwhile to design a better negative sampling process to improve the generalizability of contrastive approaches.

In this paper, we propose a novel *debiased dynamic graph contrastive learning (DDGCL)* framework that tackles the aforementioned challenges. The proposed framework utilizes an instance discrimination task that contrasts two temporal views of the same (evolving) node identity, which offers a natural and powerful method of graph data augmentation. Motivated by recent developments on contrastive learning [7, 34], We propose a novel debiased GAN-type contrastive objective that introduces a time-dependent extension of representation similarity, and corrects the sampling bias during training [7, 34]. Extensive empirical evaluations are conducted to verify the effectiveness of the DDGCL framework. In particular, for dynamic node classification tasks using a simple dynamic graph encoder [46], we can improve the supervised learning performance by a relative margin up to 10%, outperforming state-of-the-art dynamic graph models. We summarize our contribution as follows:

- We propose the first self-supervised representation learning framework on dynamic graphs called DDGCL which is based on contrastive learning. The proposed approach adopts a continuous-time formulation and introduces a time-dependent generalization of representation similarity metric that measures the agreement between two temporal views of the same node object. The framework is generic and applies to all kinds of graph encoders.
- We propose a debiased GAN-type contrastive loss that alleviates the sampling bias problem in contrastive learning scenarios. The loss is not confined to dynamic graph learning and is also applicable to static graphs.
- We conduct extensive experiments to verify the effectiveness of the proposed approach. For node classification and link prediction tasks, combining DDGCL with simple graph encoders like TGAT yield significant performance improvement over supervised training. On dynamic node classification tasks, DDGCL combined with TGAT outperforms state-of-the-art dynamic graph learning algorithms.
- We compare DDGCL with several previous self-supervision mechanisms over static graphs and show that DDGCL indeed produces the largest performance improvement over dynamic graph benchmarks, thereby verifying the effectiveness of our time-aware design.

2 Related work

2.1 Representation learning on dynamic graphs

Representation learning on graphs [16] produces node/graph level embeddings that preserves structural (topological) and compositional (attribute) information. Generalizations of graph representation learning to dynamic graphs [20] allow the node/graph representations to be time-dependent, with the model of temporal dependence formulated as either discrete-time or continuous-time [20]. While discrete-time paradigms may be of independent interest, we will focus on continuous-time models in this paper since much empirical evidence suggested the superiority of continuous-time approaches over dynamic graphs.

Compared to static graphs, the time-evolving nature of nodes in dynamic graphs provides additional information: historical interaction information would help identifying different structural roles of nodes that share very similar local neighborhoods [43, Figure 3]. As a consequence, most state-of-the-art architectures of dynamic graph modeling represent node identities as *updatable state vectors*, also called dynamic node embeddings[20]. Such kinds of design choice (which will be referred to as *stateful approaches hereafter*) often makes the resulting model align well with streaming update framework [27].

However, stateful approaches have several caveats in practice. Apart from the obvious high storage cost caused by maintaining a growing table of dynamic node embeddings, the fault tolerance of incremental computation systems is usually much more difficult to handle than stateless systems [4, 28]: problematic updates (which frequently happens in production environments) may have significant influences over later states, and the problem is even amplified when information cascading mechanisms are incorporated. Contrarily, stateless approaches [46] work by grabbing an appropriate temporal subgraph upon request and utilize standard GNN models to aggregate information from the temporal subgraph, alleviating the storage cost of the dynamic embedding table, and is more deployment-friendly because of better decoupling between algorithm execution and system architecture.

2.2 Contrastive learning on graphs

Contrastive learning [37] relies on a pre-established similarity knowledge to generate positive and negative sample pairs, along with a discrimination rule that classifies whether a given representation pair is similar. In graph-related scenarios, there are usually two sources of similarity [17]: structural (connectivity) similarity and compositional (attribute) similarity, which need not agree in general. As a consequence, different graph data augmentation schemes reflect different prior knowledge of the downstream task [48]. GCC [32] considers only the preservation of structural similarity and uses sampled local neighborhoods to obtain data augmentations. The GraphCL [48] framework construct semantic-preserving views of the same graph via four kinds of different tasks, the authors also conducted extensive empirical studies showing that no single augmentation scheme dominates.

Information maximization (infomax) frameworks on graphs [17, 42] is a closely related approach that could be viewed as a special form of contrastive learning, with the positive sample pairs constructed using a parameterized transformation, and the contrastive objective being recast as maximizing a variational approximation of some information preservation measure like mutual information or f-information [31].

2.3 Other self-supervised learning on graphs

Other types of pretext tasks on graph includes node-clustering [39], graph partition and graph completion [49]. It was reported in [49] that self-supervision techniques on graphs are often most beneficial when used in a multi-task learning fashion. In other words, a pretext task objective could be utilized as a regularization to the supervised task and appears often more effective than serving as the initial value of some supervised objective.

3 Preliminaries

Problem formulation We adopt the event stream representation of a temporal network [43]: let $\mathcal{E} = \{(u_1, v_1, t_1, \chi_1), (u_2, v_2, t_2, \chi_2), \ldots\}$ be an event stream that generates the temporal network, with an event (u, v, t, χ) indicating an interaction (link) happens from source node u to target node v at time t, with associating features χ. In this paper all events will be considered as homogeneous, i.e., we don't distinguish different types of events [40]. For any pair of nodes (u, v), a *causal path* from u to v with respect to the event stream $\mathcal{E}$ is a subset of events with the earliest occurring event having source u and the terminal event having target v. The (temporal) shortest path $d_t(u, v)$ is thus defined as the minimum size of the causal path from u to v whose terminal event occurs no later than t. Denote V_t as the set of nodes that has emerged up till time t. For each $v \in V_t$, define its (unrestricted) k-hop *temporal neighborhood*

$$\mathcal{N}_t^k(v) = \{u : d_t(u, v) \leq k\}$$

as the multiset of nodes that are at most k hops away (in a causal sense)to v up till time t. Note that the temporal neighborhood is different from the conventional neighborhood in static graphs as there might be multiple links between the same pair of node identities. [1]. For a temporal neighborhood multiset $\mathcal{N}_t^k(v)$, define its induced subgraph (with respect to the event stream $\mathcal{E}$) as $G_t^k(v)$, with its edge set generated by all the events whose source and target are in $\mathcal{N}_t^k(v)$ and occurred no later than t. In this paper, we focus on the two most representative tasks: (dynamic) node classification and (dynamic) link prediction [20].

Remark 1. *Technically, $G_t^k(v)$ is a multigraph in that there may exist multiple links between the same pair of nodes. In this paper, we follow the* stateless *approach in [46] that discards node identity information (with respect to the underlying temporal network) during subgraph construction and model computation. (see also the discussion in section 2.1).*

Time-aware GNN encoders graph neural networks [5] are a powerful tool that combines representation learning with graph connectivity information. The dominant formulation of GNNs is the message passing model [9, 47] that recursively aggregates neighborhood features. Applying GNN to the temporal subgraph $G_t^k(v)$ provides a principled way to incorporate historical interaction information into the node embedding of v at time t: let $h_t^l(v)$ be the hidden representation of node v in the l-th layer, a standard GNN aggregation rule is written as:

$$h_t^{l+1}(v) = \mathsf{AGG}\left(h_t^l(v), \{h_t^l(u), u \in \mathcal{N}_t(v)\}\right) \tag{1}$$

Where different types of aggregation rules AGG corresponds to different types of GNNs [47]. The rule (1) does not effectively exploit the temporal information. Recent works [35, 46] suggest that careful treatment of event times has significant impact on the performance of dynamic graph models, the treatment is based on an assumption that *time-dependent* aggregation schemes shall be translation-invariant in time, thereby obtaining a form of generic time-encoding [45, 46] that stems from the design of random Fourier features [33]. Time encoding defines a learnable map $\mathsf{TE}(t)$ that takes any $t \geq 0$ into a d_T dimensional embedding vector, and a time-aware GNN encoder is thus defined as:

$$\begin{aligned} h_t^{l+1}(v) = \mathsf{AGG}\Big(&\mathsf{MERGE}(h_t^l(v), \mathsf{TE}(0)), \\ &\{\mathsf{MERGE}(h_t^l(u), \mathsf{TE}(t_v - t_u)), u \in \mathcal{N}_t(v)\}\Big) \end{aligned} \tag{2}$$

where t_u and t_v are event times of u and v, and the MERGE operation fuses nodes' (hidden) features and their time encodings. According to [46], simple merging functions like summation or concatenation works reasonably well.

4 Debiased dynamic graph contrastive learning

4.1 A time-dependent setup

We follow the probabilistic setup in [7, 34, 37] and adapt it to a dynamic graph modeling context. We assume the input triple $(v, t, G_t^k(v))$ is drawn randomly according to a probability distribution p over the product set $\mathcal{V} \times \mathbb{R}_+ \times \mathcal{G}$, here $\mathcal{V}$ denotes the universe of all possible node identities, and $\mathcal{G}$ stands for the universe of (temporal) graphs defined on subsets of $\mathcal{V}$. For notational simplicity we will use x to denote the input triple. Contrastive learning assumes that for any x there exists a semantic class $c(x) \in C$, where c is the class assignment map that maps input tuple x to an element of a

[1] Working on unrestricted forms of the temporal neighborhood would require storing the entire event history which may become prohibitive for large-scale temporal graphs. A more practical modification of temporal neighborhood poses certain types of constraints on the neighborhood, for example, an upper limit on the cardinality of the multiset, or restricting the longest time span of the neighborhood.

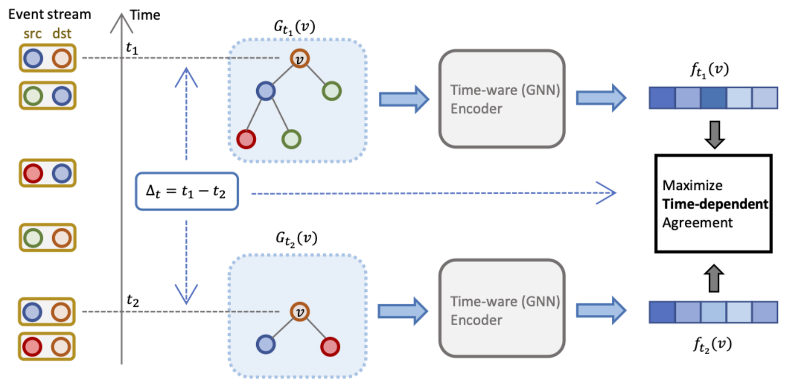

Figure 1: The proposed contrastive learning framework for dynamic graphs. For a given node identity, two temporal views are generated via grabbing its temporal subgraph at two different time points. A time-aware GNN encoder (which may be shared across two views depending on the learning procedure) is learned to maximize a time-dependent agreement using a debiased contrastive loss.

finite set C, and we say that two observations x and x' are similar if they share the same semantic class, i.e., $c(x) = c(x')$. Under this assumption, we formalize a positive sample x^+ of a given observation x (also called an *anchor point*) as an element drawn from the conditional distribution $p^+(x'|x) = p(x'|c(x') = c(x))$, similarly, a negative sample x^- is drawn from the conditional distribution $p^-(x'|x) = p(x'|c(x') \neq c(x))$. In practice, we do not have direct access to the sampling distributions p^+ and p^-. The main ingredients of a contrastive learning framework are therefore: (i) Proxies of p^+ and p^- given anchors x. (ii) A loss function that discriminates positive and negative samples.

4.2 Positive sample construction via contrasting temporal views

To obtain a proxy for p^+, a common approach is to design two different views of the same underlying object [6]. In computer vision problems, the two views are easily constructed via semantic preserving transformations. However, the analog is not obvious in graph related problems [48]. In [48], it was reported that different graph perturbation strategies for static graph representation learning (i.e., node dropping, edge perturbation) behave differently according to the distributions of the underlying graph data. It is thus of interest to explore more generic approaches for obtaining positive samples. Fortunately, in dynamic graph setups, a natural approach exists for constructing positive samples: note that in dynamic scenarios, individual identities (nodes) evolve over time. It is thus reasonable to assume that the evolution process is in general "smooth", which means that for a small amount of elapsed time, the semantic of each individual object remains the same. This observation motivates the positive sample construction process which we term *temporal views*, formally defined as follows: given an input tuple $(v, t, G_t^k(v))$ and an upper limit

$$\delta_{\max} = \max\{s : s \in (0, t), c(v, t, G_t^k(v)) = c(v, t-s, G_{t-s}^k(v))\} \quad (3)$$

that represents the longest retrospecting duration such that the semantic of $(v, t, G_t^k(v))$ is not altered. Then we sample a past time t^+ uniformly from $[t - \delta_{\max}, t)$, and constructs the positive sample as $(v, t^+, G_{t^+}^k(v))$. In practice we don't have access to semantic classes and $\delta_{\max}$ is chosen either as a fixed duration, or a proportion of the absolute time γt, with $\gamma \in (0, 1)$ a hyperparameter. Note that the sampling procedure is not guaranteed to succeed since for stale nodes [20], i.e., nodes that abstained from getting update for a very long period of time, there might be no meaningful temporal views that reflects the evolution pattern of the node. We summarize the positive sampling procedure in algorithm 1

Algorithm 1 Temporal view construction with a given anchor

```
Require: Anchor $x = (v, t, G_t^k(v))$, desired number of hops $k$, sampling duration proportion $\gamma$, event stream $\mathcal{E}$.
1: if No events in $\mathcal{E}$ with target $v$ happens during $((1-\gamma)t, t)$ then return
2: Sample $t^+ \sim \mathrm{Unif}((1-\gamma)t, t)$
3: Construct $k$-hop temporal graph $G_{t^+}^k(v)$ of $v$ at $t^+$.
4: return $x^+ = (v, t^+, G_{t^+}^k(v))$
```

4.3 The contrastive loss function

In this paper we use a GAN-type objective with a time-dependent critic. The GAN-type objective was previously shown to exhibit superior performance on unsupervised representation learning tasks over static graphs [17, 42] with information-theoretic motivations. Denote the GNN encoding map as f that maps an input triple x into an element of the d-dimensional euclidean space. Under the true positive and negative sampling distributions, we adopt the following form of contrastive loss:

$$-\mathbb{E}_x \left[\mathbb{E}_{x^+ \sim p^+} \log \frac{1}{1 + e^{-\mathrm{sim}(f(x^+), f(x))}} + \mathbb{E}_{x^- \sim p^-} \log \frac{1}{1 + e^{\mathrm{sim}(f(x^-), f(x))}} \right] \quad (4)$$

where the similarity function sim is a learnable, time-dependent map that measures the similarity of two representations, previous time-independent constrastive learning approaches [29, 42] frequently adopted the following bilinear critic:

$$\mathsf{sim}\,(x, y) = x^T \Omega y \tag{5}$$

In dynamic setups, using a constant weighting matrix Ω maybe problematic. Consider two nodes with very similar k-hop temporal subgraphs but are far apart on the timeline. A time-independent critic would output a high similarity score. However the large time gap between the two nodes suggest that there might be potentially a large number of excluded events that are more than k-hops away from the later node but occurs later than the earlier node. We give a pictorial illustration in figure 2. Therefore, it is reasonable to add an extra penalty with respect to time gap in the weighting matrix $\Omega(t_x, t_y) = \Omega(|t_x - t_y|)$. The functional form $\Omega(t)$ is assumed arbitrary, and we utilize the rich expressive power of generic time encoding [45] to obtain the following *nonparametric* approximation of the time-dependent similarity metric:

$$\widehat{\mathsf{sim}}\,(x, y) = \sum_{1 \le i,j \le d} \mathsf{RELU}\left(\langle \omega_{ij}, \mathsf{TE}(|t_x - t_y|)\rangle\right) x_i y_j \tag{6}$$

Hereafter with slight abuse of notation we will use the symbol sim for $\widehat{\mathsf{sim}}$.

4.4 Debiasing the contrastive objective

Under the setup in section 4.1, the standard approach to obtain negative samples is to draw from the marginal distribution p, in practice p is often approximated using the empirical distribution of the data. There are two issues with this sampling process. The first issue is *sampling bias*. As was pointed out in [7], sampling from $p(x)$ instead of the (ideal) $p^-(x'|x) = p(x'|c(x') \neq c(x))$ produces a biased sampling scheme for the negative samples, since chances are that a sample with the same semantic class is drawn, resulting in a larger loss value and might produce very different learned representations [7, Figure 2]. The second one is *hardness-unawareness*: even we may get rid of the sampling bias problem and select negative samples over the "true negatives", the benefits of different negative sample instances to the learning procedure may vary. This phenomenon was formally quantified in [34], where the authors showed both theoretically and empirically that placing higher weights on "good" negative samples leads to improved contrastive representation learning. By "good" negative sample we mean an element x^- with different semantic class from the anchor point x, but the representation $f(x^-)$ is close to the representation $f(x)$ with respect to the similarity critic. Now we generalize the idea in [34] to general learnable similarity criterion.

Hardness-promoting negative sampling distribution We suggest the following improved negative sampling distribution:

$$p^-(x'|x; \beta) = q_\beta(x'|c(x') \neq c(x)) \tag{7}$$

The distribution q_β is defined as

$$q_{\beta|x}(x') \propto \underbrace{p(x')}_{\text{prior}} \times \underbrace{e^{\beta \mathsf{sim}(f(x), f(x'))}}_{\text{likelihood}} \tag{8}$$

where $\beta > 0$ is a hyperparameter that controls the impact of hard negative samples on the contrastive learning process: points that lies closer to the anchor in representation space have a larger probability of getting sampled. The definition (8) has a natural Bayesian interpretation: without any information about the anchor x, the best we've got is the marginal distribution. Upon observing the anchor x, we are then able to place more probability mass on sample points with similar representations. Next, we derive an unbiased negative sampling procedure with the target sampling distribution being (7) under the loss (4).

Debiasing the contrastive loss The debiasing procedure utilizes the following relation [7, 34]: for any sampling distribution q_β defined in (8):

$$q_\beta(x') = (1 - \tau^+) p^-(x'|x) + \tau^+ q_\beta(x'|c(x') = c(x)) \tag{9}$$

where $\tau^+ = p(c(x') = c(x))$ is the class prior of x's semantic class. In practice, this term could be estimated from data [19], or treated as a hyperparameter [7]. Rearranging (9) yields:

$$p^-(x'|x) = \frac{1}{1 - \tau^+}\left(q_\beta(x') - \tau^+ q_\beta(x'|c(x') = c(x))\right) \tag{10}$$

Since sampling from q_β is tractable, it remains to address how to sample from $q_\beta(x'|c(x') = c(x))$. From the definition (8), it follows that:

$$\begin{aligned} q_\beta(x'|c(x') = c(x)) &\propto 1/\tau^+ q(x') \\ &= p(x')/\tau \times e^{\beta \mathsf{sim}(f(x), f(x'))} \\ &= e^{\beta \mathsf{sim}(f(x), f(x'))} \times p^+(x'|x) \end{aligned} \tag{11}$$

Combining (10) and (11), we can use samples from q_β and p^+ to construct an unbiased estimate of the second term of (4) via importance sampling technique [34]. Plug (10) into the definition and condition on x, we get:

$$\begin{aligned} &\mathbb{E}_{x^- \sim p^-} \log\left(\frac{1}{1 + e^{\mathrm{sim}(f(x^-), f(x))}}\right) \\ &= \frac{1}{1 - \tau^+}\mathbb{E}_{x^- \sim p}\left[\log\left(\frac{1}{1 + e^{\mathrm{sim}(f(x^-), f(x))}}\right) \times \frac{q_\beta(x^-)}{p(x^-)}\right] \\ &\quad - \frac{\tau^+}{1 - \tau^+}\mathbb{E}_{v \sim p^+}\left[\log\left(\frac{1}{1 + e^{\mathrm{sim}(f(v), f(x))}}\right) \times \frac{q^+_\beta(v)}{p^+(v)}\right] \\ &= \frac{1}{1 - \tau^+}\mathbb{E}_{x^- \sim p}\left[\log\left(\frac{1}{1 + e^{\mathrm{sim}(f(x^-), f(x))}}\right) \times \frac{e^{\beta \mathrm{sim}(f(x^-), f(x))}}{Z_\beta}\right] \\ &\quad - \frac{\tau^+}{1 - \tau^+}\mathbb{E}_{v \sim p^+}\left[\log\left(\frac{1}{1 + e^{\mathrm{sim}(f(v), f(x))}}\right) \times \frac{e^{\beta \mathrm{sim}(f(v), f(x))}}{Z^+_\beta}\right] \\ &:= J_1 - J_2 \end{aligned} \tag{12}$$

where Z_β and Z^+_β are partition functions of p and p^+ defined as

$$\begin{aligned} Z_\beta &= \mathbb{E}_{y \sim p}\left(e^{\mathrm{sim}(f(y), f(x))}\right) \\ Z^+_\beta &= \mathbb{E}_{z \sim p^+}\left(e^{\mathrm{sim}(f(z), f(x))}\right) \end{aligned} \tag{13}$$

Which could be estimated using negative and positive samples. In particular, using a set of N_{pos} positive samples and N_{neg} negative samples, the two components of the previous display are separately

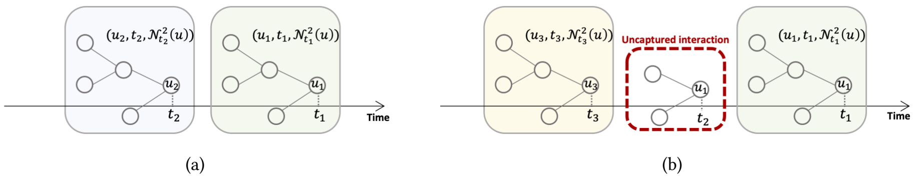
$(u_2, t_2, \mathcal{N}^2_{t_2}(u))$
u_2
t_2
$(u_1, t_1, \mathcal{N}^2_{t_1}(u))$
u_1
t_1
Time
(a)
$(u_3, t_3, \mathcal{N}^2_{t_3}(u))$
u_3
t_3
Uncaptured interaction
u_1
t_2
$(u_1, t_1, \mathcal{N}^2_{t_1}(u))$
u_1
t_1
Time
(b)

Figure 2: An illustration about the necessity of time-dependent similarity formulation (6)**. In figure (a) above, the structure of two temporal subgraphs corresponding to nodes u_1 and u_2 are identical, and between the scope of these two subgraphs there are no relevant interactions targeting u_1. The situation becomes more complicated in figure (b), where two identical temporal subgraphs (with target nodes u_1 and u_3 respectively) exhibit a fairly large time gap. There might be interactions targeting node u_1 that are not included in either subgraph, under time-independent setting, case (a) and (b) would share the same similarity score, which is problematic because of the uncaptured interaction that happens at t_2 in figure (b)**

estimated as follows:

$$\widehat{J}_1 = \frac{1}{1-\tau^+}\sum_{i=1}^{N_{\text{neg}}}\left(\frac{e^{\beta \text{sim}(f(x_i^-),f(x))}}{\sum_j e^{\beta \text{sim}(f(x_j^-),f(x))}} \times \log \frac{1}{1+e^{\text{sim}(f(x_i^-),f(x))}}\right)$$

$$\widehat{J}_2 = \frac{\tau^+}{1-\tau^+}\sum_{i=1}^{N_{\text{pos}}}\left(\frac{e^{\beta \text{sim}(f(x_i^+),f(x))}}{\sum_j e^{\beta \text{sim}(f(x_j^+),f(x))}} \times \log \frac{1}{1+e^{\text{sim}(f(x_i^+),f(x))}}\right) \quad (14)$$

(14) suggests that the bias correction procedure is implemented as a *reweighting* procedure. We summarize the above derivations into a final empirical objective: with a sample $\mathbf{X} = \{x_1, \ldots, x_N\}$ of size N, for each anchor $x \in \mathbf{X}$ we sample N_{pos} positive sample $\left\{x_1^+, \ldots, x_{N_{\text{pos}}}^+\right\}$ using the temporal view construction method, and sample N_{neg} negative samples $\left\{x_1^-, \ldots, x_{N_{\text{neg}}}^-\right\}$ uniformly from X without replacement, the debiased contrastive objective is:

$$\begin{aligned}
\mathcal{L}_{\text{DDGCL}} &= \\
&- \frac{1}{N}\sum_{x\in\mathbf{X}}\sum_{l=1}^{N_{\text{pos}}}\frac{1}{N_{\text{pos}}}\log\frac{1}{1+e^{-\text{sim}(f(x_l^+),f(x))}} \\
&- \frac{1}{N(1-\tau^+)}\sum_{x\in\mathbf{X}}\sum_{i=1}^{N_{\text{neg}}}\left(\frac{e^{\beta \text{sim}(f(x_i^-),f(x))}}{\sum_j e^{\beta \text{sim}(f(x_j^-),f(x))}}\right)\log\frac{1}{1+e^{\text{sim}(f(x_i^-),f(x))}} \\
&+ \frac{\tau^+}{N(1-\tau^+)}\sum_{x\in\mathbf{X}}\sum_{i=1}^{N_{\text{pos}}}\left(\frac{e^{\beta \text{sim}(f(x_i^+),f(x))}}{\sum_j e^{\beta \text{sim}(f(x_j^+),f(x))}}\right)\log\frac{1}{1+e^{\text{sim}(f(x_i^+),f(x))}}
\end{aligned}$$

For the rest of the paper, we will focus on the case where $N_{\text{pos}} = 1$, i.e., only a single positive sample is used.

4.5 Learning procedure

Different mechanisms for using contrastive losses There are multiple strategies for parameter learning under contrastive losses. We present here two representative learning mechanisms: end-to-end (E2E) update and momentum contrast [18] (MoCo) update. In E2E update rule, the set of negative samples is chosen as the points in the current mini-batch, and all the parameters involved in (15) are learned via back-propagation. According to the discussion in [18], the key drawback of the E2E mechanism is the coupling of negative sample size with training batch size, and obstacles in large batch training may prevent efficient usage of a large number of negative samples. The MoCo update rule overcomes this problem via decoupling the parameter updates of the anchor representation (also called *query* representation) $f_{\theta_q}(x) := f(x)$ with positive and negative representations (also called *key* representations) $f_{\theta_k}(x^+) := f(x^+), f_{\theta_k}(x^-) := f(x^-)$. The query parameter θ_q are updated via back-propagation. The set of negatives samples is maintained as a FIFO queue with size Q, which is often set to be larger than the batch size, and the key parameter θ_k is updated via a momentum update rule $\theta_k \leftarrow m\theta_k + (1-m)\theta_q$ with $m \in [0, 1)$ being a momentum coefficient. MoCo was previously shown to benefit contrastive learning in computer vision [18] and was also adopted in graph contrastive learning frameworks [32].

Two schemes of self-supervision There are several ways to use a pretext task to assist the downstream model. In this paper we will consider two most representative self-supervision schemes [49]: **Pretraining and finetuning (PF)** and **Multi task learning (MTL)**. Specifically, let θ_{enc} denote all the parameters of the time-aware GNN encoder, and θ_{cl}) denote the learnable parameters in the contrastive learning module. For a downstream supervised task with loss function $\mathcal{L}_{\text{sup}}(\theta_{\text{enc}})$, the pretraining and finetuning scheme learns θ_{enc} via a two-stage process: first we obtain the minimizer of the pretext task:

$$\theta^*_{\text{enc}}, \theta^*_{\text{cl}} \in \arg\min_{\theta_{\text{enc}},\theta_{\text{cl}}} \mathcal{L}_{\text{DDGCL}}\ (\theta_{\text{enc}}, \theta_{\text{cl}}) \quad (15)$$

Then we use θ^*_{enc} as a warm start for the supervised loss minimization problem. Alternatively, multi task learning scheme combines the above two stage into a single optimization problem:

$$\theta^*_{\text{enc}}, \theta^*_{\text{cl}} \in \arg\min_{\theta_{\text{enc}},\theta_{\text{cl}}} \mathcal{L}_{\text{sup}}(\theta_{\text{enc}}) + \alpha\mathcal{L}_{\text{DDGCL}}\ (\theta_{\text{enc}}, \theta_{\text{cl}}) \quad (16)$$

Here $\alpha > 0$ is a hyperparameter for balancing the contributions of two losses.

5 Experiments

In this section, we report empirical evaluation results on benchmark datasets to validate the effectiveness of the proposed DDGCL framework. Across all tasks we adopt TGAT [46] as the time-aware GNN encoder, and we will use TGAT-CL to represent TGAT models trained under the assistance of self-supervision. Two types of tasks are for evaluation: dynamic node classification (user state change prediction) and dynamic link prediction (future interaction prediction). A detailed comparison between DDGCL and other self-supervised graph learning approaches is also presented.

5.1 Experimental setup

Datasets We use three benchmark datasets, all of which are bipartite interaction graphs[23]: Wikipedia [3] is a dynamic network between wiki pages and human editors, where an interaction event represents a user editing a page. Reddit [2] is a dynamic network between posts and users on subreddits, where an interaction represents a user writes a post to the subreddit. MOOC [1] is a dynamic network of students and online course content units, where an interaction event represents user actions on the course activity. For all tasks and datasets we adopt the same chronological split with 70% for training, and 15% for validation and testing [35, 46].

Evaluation tasks We evaluate DDGCL for node classification task over all the three datasets. For Wikipedia and Reddit datasets, there's a binary label on each edge indicating whether the user got banned right after this interaction. For the MOOC dataset, there's a binary label on each edge indicating whether the user dropped out of the course right after this interaction. The edge label is translated into node label as a state change. A notable fact about these datasets is that positive labels are very rare [23]. We follow the transductive setup in the node classification task so that in inference time, node identities are allowed to be observed during training time.

We evaluate DDGCL for link prediction tasks over Wikipedia and Reddit datasets. We consider both transductive setups, where we predict future interactions between nodes observed during training and inductive setups where links of completely fresh nodes are predicted. For a fair comparison, we follow the negative sampling approach in previous works [35, 43, 46] that sample an equal amount of negative links to cast the link prediction problem into a binary classification problem.

5.2 Baselines

We compare the proposed framework with the following two types of baselines:

GNNs over static graphs This type of baselines perform time-independent aggregations over the k-hop temporal subgraph. We present results corresponding to two representative GNN aggregation rules GraphSAGE [15] and GAT [41] that are capable of handling both transductive and inductive settings.

Temporal Graph Embedding algorithms These algorithms are specifically designed for handling temporal networks. JODIE [23] uses time projection embedding to capture temporal information. TGAT [46] is an adaptation of GNN type aggregation rule into temporal setups with temporal information captured via generic time encodings. TGN [34] enhances TGAT with time-dependent node state vectors. CAW [43] utilizes causal anonymous walks to produces relative identity embeddings that have a tailor-made inductive bias over link prediction tasks.

5.3 Training configurations

We adopt binary cross entropy as the loss function for the supervised tasks. For model hyperparameters, we fix the following configuration across all experiments without further tuning: We use a 2 layer, 8 head TGAT with a hidden representation of 128 dimensions. We use a Bochner-type functional time encoding [45, 46]

$$\mathrm{TE}(t) = \sqrt{\frac{1}{d}}\,[\cos(\phi_1 t), \sin(\phi_1 t), \ldots, \cos(\phi_d t), \sin(\phi_d t)] \tag{17}$$

of $d = 128$ dimensions and learnable $\{\phi_i\}$s. For the contrastive auxiliary task, we use $\gamma = 0.4$ to generate temporal views, and device another Bochner-type time encoding for (6). The two additional hyperparameters are the class prior τ and hardness-controlling parameter β. We tune τ over the range $[0, 0.05]$ and β over the range $[1, 10]$, and we found that picking $\tau = 0.01$ and $\beta = 2$ performs reasonably well across all tasks, and we report the results based on this setting. For multi task self-supervision scheme, we fix $\alpha = 1$. For MoCo type learning procedure, we use a queue size $Q = 256$ and momentum coefficient $m = 0.999$. We use Adam as the optimizer with a learning rate of 0.0002, during training we use a batch size of 100 and train for 10 epochs. All parameter tunings are based on the validation set performance.

We found during evaluation that using the MoCo learning rule consistently outperforms the E2E learning rule, hence without further clarification, the results are produced using MoCo. We will make an additional assessment on the performance difference between MoCo and E2E learning in section 5.7.

Table 1: Dynamic node classification performance in average AUC (mean in percentage ± standard deviation over 10 trial runs). The baselines TGN(S) and TGAT(S) mean that they're trained directly using the labeled data. We use different colors to highlight the first, second and third best performing algorithms.

	Wikipedia	Reddit	MOOC
GAT [41]	82.34 ± 0.8	64.52 ± 0.5	66.21 ± 0.4
GraphSAGE [15]	82.42 ± 0.7	61.24 ± 0.6	65.17 ± 0.5
JODIE [23]	84.84 ± 1.2	61.83 ± 2.7	66.69 ± 0.9
CAW-N [43]	86.77 ± 0.3	67.46 ± 0.8	68.77 ± 0.4
TGN (S) [35]	86.80 ± 1.1	64.03 ± 1.0	69.89 ± 0.6
TGN [35]	**88.56** ± 0.3	**68.63** ± 0.7	**71.64** ± 0.3
TGAT(S) [46]	81.28 ± 0.7	64.80 ± 0.8	68.28 ± 0.4
TGAT [46]	83.69 ± 0.7	65.56 ± 0.7	69.46 ± 0.4
TGAT-CL(PF)	**87.48** ± 0.5	**68.71** ± 0.7	**73.41** ± 0.3
TGAT-CL(MTL)	**89.32** ± 0.5	**71.13** ± 0.8	**74.54** ± 0.2

5.4 Performance

Evaluation strategy For the node classification task, we compute average AUC over 10 trial runs on the test sets. For the link prediction task, we use average precision over 10 trial runs as the evaluation metric.

Results We present node classification evaluations in table 1 and link prediction evaluations in table 2. We have the following observations:

- On node classification, DDGCL significantly improves its base model's pure supervision performance. For the multi-task learning scheme, the relative average improvement on three datasets is 9.9%, 8.5%, 7.3% respectively, clearly dominant the improvement achieved via the link prediction pretext task. Besides, both self-supervision schemes enhance the base model's performance to state-of-the-art level: with multi-task learning as an auxiliary

Table 2: Dynamic link prediction performance in average precision (mean in percentage ± standard deviation over 10 trial runs).

	Wikipeida		Reddit	
	Transductive	Inductive	Transductive	Inductive
GAT [41]	94.73 ± 0.2	91.27 ± 0.4	97.33 ± 0.2	95.37 ± 0.3
GraphSAGE [15]	93.56 ± 0.3	91.09 ± 0.3	97.65 ± 0.2	96.27 ± 0.2
JODIE [23]	94.62 ± 0.4	93.11 ± 0.4	97.11 ± 0.3	94.36 ± 1.1
CAW-N [43]	**99.51** ± 0.1	**99.74** ± 0.1	**99.72** ± 0.1	**99.37** ± 0.1
TGN [35]	98.46 ± 0.3	97.81 ± 0.1	**98.70** ± 0.1	97.55 ± 0.1
TGAT [46]	95.34 ± 0.1	93.99 ± 0.3	98.12 ± 0.2	96.62 ± 0.3
TGAT-CL(PF)	**98.54** ± 0.1	**98.22** ± 0.1	98.63 ± 0.1	**98.19** ± 0.1
TGAT-CL(MTL)	**98.79** ± 0.1	**99.06** ± 0.1	**99.13** ± 0.1	**98.45** ± 0.1

task, a TGAT encoder outperforms more elegant state-based models like TGN over all three datasets by a statistically significant margin.

- On link prediction, DDGCL exhibits large performance improvement over its base model. The gap is particularly evident on the Wikipedia dataset, and the trained TGAT encoder outperforms all baselines except for CAW-N. The inferiority to CAW-N is reasonable since the CAW model entails a carefully calibrated inductive bias for interaction prediction [43] and was previously shown to be strictly more expressive than the TGAT architecture on link prediction. However, the inductive bias brought by CAW may not be preferable in node classification settings, as illustrated in table 1.
- Across all tasks, the multi-task learning scheme consistently achieves better improvement over the base model than the pretraining and finetuning scheme, which agrees with former observations in [49].

5.5 Comparison with other graph contrastive learning frameworks

In this section we present a detailed comparison between DDGCL and other previous approaches on graph contrastive learning, namely GraphCL [48] and GCC [32]. Our model differs from previous approaches both in the construction process of positive sample pairs and in the adoption of a different contrastive loss.

Evaluation scheme We conduct the comparison under the node classification task on all three datasets. The GraphCL [48] framework proposes four different graph augmentation techniques: node dropping (n), edge perturbation (l), attribute masking (m) and subgraph (s). In our comparisons, we choose the five best performing 2-combinations of the GraphCL augmentation techniques and applied them to the temporal subgraph according to the results in [48]. The GCC model [32] uses a three-step sampling process to construct positive sample pairs. For a fair comparison, we use MoCo learning rule with $Q = 256$ and $m = 0.999$ in our GCC experiments.

Results We list the comparison results in table 3. Our model is shown to dominate other baselines in five out of six tasks. We attribute the superiority to the efficient integration of temporal information into graph contrastive learning, and a bias correction scheme that was not included in all the baseline algorithms. In comparison to graph augmentation techniques over static graphs, DDGCL exploits the *temporal dynamic* of the underlying graph to guide the construction of semantic-preserving views, providing a natural and efficient way for self-supervision on dynamic graphs.

Table 3: Comparison of DDGCL with other previous approaches on graph contrastive learning on the dynamic node classification task. Classification performances are measured in average AUC (mean in percentage ± standard deviation over 10 trial runs)

Pretraining and finetuning			
	Wikipedia	Reddit	MOOC
Link prediction	83.69 ± 0.7	65.56 ± 0.7	69.46 ± 0.4
GraphCL(on)	87.36 ± 0.8	65.64 ± 0.7	70.50 ± 0.3
GraphCL(nn)	**87.68** ± 0.8	66.10 ± 0.7	69.44 ± 0.3
GraphCL(nl)	87.44 ± 0.8	66.26 ± 0.7	69.92 ± 0.3
GraphCL(nm)	86.82 ± 0.7	**66.71** ± 0.7	**70.17** ± 0.4
GraphCL(ns)	**87.60** ± 0.7	65.98 ± 0.7	70.08 ± 0.4
GCC	**87.91** ± 0.6	**67.35** ± 0.7	**72.34** ± 0.3
DDGCL	87.48 ± 0.5	**68.71** ± 0.7	**73.41** ± 0.3
Multi task learning			
	Wikipedia	Reddit	MOOC
Link prediction	84.45 ± 0.8	65.41 ± 0.7	73.38 ± 0.4
GraphCL(on)	87.67 ± 0.9	64.72 ± 0.6	73.48 ± 0.5
GraphCL(nn)	87.86 ± 0.6	65.69 ± 0.7	73.81 ± 0.4
GraphCL(nl)	87.52 ± 0.9	66.13 ± 0.7	73.73 ± 0.4
GraphCL(nm)	**87.89** ± 0.8	**66.16** ± 0.7	73.64 ± 0.6
GraphCL(ns)	87.10 ± 0.6	64.40 ± 0.7	**73.89** ± 0.4
GCC	**88.46** ± 0.6	**69.83** ± 0.8	**73.94** ± 0.3
DDGCL	**89.32** ± 0.5	**71.13** ± 0.8	**74.54** ± 0.2

5.6 On representation similarity versus time

In this section, we study the learned similarity critic (6) under the node classification task. The evaluation procedure is detailed as follows: For each point (called raw sample) $(v, t, G_t^k(v))$ in the test set, we extract 20 past temporal subgraphs (past samples) with

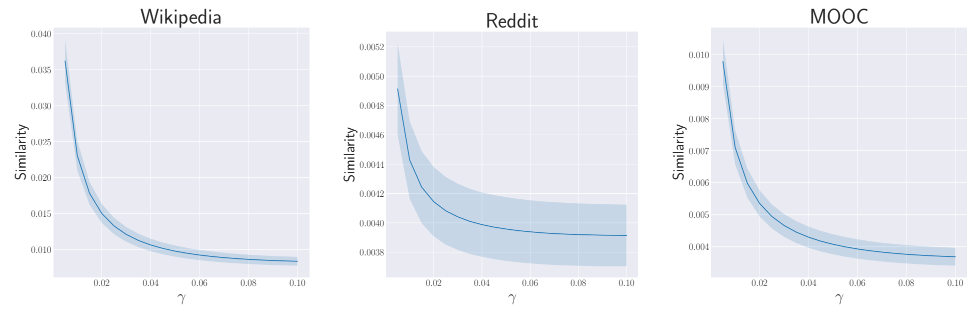

Figure 3: Evaluations of learned similarity with respect to node classification tasks over three benchmark datasets. The model parameters are chosen to be the best performing configuration according to table 1. The horizontal axis γ corresponds to twenty equidistant relative time gap configurations. Vertical axis denotes average similarity score over the whole test set at different relative time gaps. With a 95% Gaussian confidence band plotted as the shaded area.

equal time-gaps over the time span $[0.9t, t)$, and computes the representation similarity between the raw sample and every past sample, which could be viewed as similarity with "past selves". The results are shown in figure 3. The results imply that the learned representation similarity decays smoothly as time increases, with a flattening slope that resembles some sort of *exponential decay*, a model assumption that is frequently adopted in event history analysis [10, 11].

5.7 Ablation study

Finally, we study the contribution of different components of the DDGCL framework, namely the GAN-type loss with time-dependent similarity module, the debiasing technique, and the learning rule. Specifically, apart from the supervised results under the original TGAT setup, we conduct three experiments under both the End-2-End (E2E) and the MoCo learning rule with the multi-task learning self-supervision scheme, with the following configurations:

w/ CPC We use the standard CPC framework [29], with constrastive loss chosen as InfoNCE. This setup is the most ad-hoc choice when considering contrastive learning as an auxiliary task.

w/ Ours (biased) We use the GAN-type loss with time-dependent critic as described in (4).

w/ Ours (debiased) We use the debiased GAN-type loss with time-dependent critic , which corresponds to our full model.

Results We report the results of the ablation study in table 4. We summarize our observations as follows: Firstly, the contrastive auxiliary task improves the base model’s performance even in the vanilla setup (CPC framework). Secondly, our proposed GAN-type loss with time-dependent critic function was shown to achieve solid improvements over InfoNCE loss across all datasets. Thirdly, the debiasing technique consistently makes further improvement. Finally, the MoCo rule is shown to dominate E2E rule. We make an extra remark here that even E2E rule was shown to be inferior to MoCo rule, the best result under E2E learning still outperforms all the other baselines in terms of average precision, on two of three datasets the margin is statistically significant.

Table 4: Results of the ablation study on the dynamic node classification task. Classification performances are measured in average AUC (mean in percentage ± standard deviation over 10 trial runs)

	Wikipedia	Reddit	MOOC
TGAT	81.28 ± 0.7	64.80 ± 0.8	68.28 ± 0.4
End-to-end			
w/ CPC	86.59 ± 0.9	66.94 ± 0.8	72.20 ± 0.4
w/ Ours (biased)	88.13 ± 0.6	69.26 ± 0.8	73.79 ± 0.3
w/ Ours (debiased)	88.70 ± 0.5	70.57 ± 0.7	74.22 ± 0.3
MoCo			
w/ CPC	87.58 ± 0.6	69.63 ± 0.8	74.13 ± 0.3
w/ Ours (biased)	88.59 ± 0.6	70.51 ± 0.7	74.40 ± 0.3
w/ Ours (debiased)	89.32 ± 0.5	71.13 ± 0.8	74.54 ± 0.2

6 Discussion

The DDGCL framework proposed in this paper accepts stateless GNN type encoders. Despite its convincing performance, stateful approaches like [43] were previously shown to contain a richer inductive bias than stateless approaches. The idea of temporal view construction seems "redundant" under the presence of individual-specific embeddings. Instead one may explore semantic similarity *between* identities, which is a highly nontrivial task and we leave it to future explorations.

7 Conclusion

We proposed debiased dynamic graph contrastive learning, a self-supervised framework to assist GNN-based encoders over dynamic graphs. The proposed framework is based on a time-dependent formulation of contrastive learning, which uses temporal dynamic of nodes to create natural and effective positive samples, and utilized a debiased GAN-type contrastive loss to overcome the sampling bias problem in negative samples construction. Our framework does not bring extra computational overhead as compared to previous self-supervised approaches over static graphs, with improved performance on benchmark datasets.

References

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

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

[3] Wikipedia edit history dump. https://meta.wikimedia.org/wiki/Data_dumps.

[4] Akidau, T., Balikov, A., Bekiroğlu, K., Chernyak, S., Haberman, J., Lax, R., McVeety, S., Mills, D., Nordstrom, P., and Whittle, S. Millwheel: Fault-tolerant stream processing at internet scale. *Proc. VLDB Endow. 6*, 11 (Aug. 2013), 1033–1044.

[5] Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261* (2018).

[6] Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning* (13–18 Jul 2020), H. D. III and A. Singh, Eds., vol. 119 of *Proceedings of Machine Learning Research*, PMLR, pp. 1597–1607.

[7] Chuang, C.-Y., Robinson, J., Yen-Chen, L., Torralba, A., and Jegelka, S. Debiased contrastive learning. *arXiv preprint arXiv:2007.00224* (2020).

[8] Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems* (2016), pp. 3844–3852.

[9] Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning* (International Convention Centre, Sydney, Australia, 06–11 Aug 2017), D. Precup and Y. W. Teh, Eds., vol. 70 of *Proceedings of Machine Learning Research*, PMLR, pp. 1263–1272.

[10] Gomez-Rodriguez, M., Leskovec, J., and Krause, A. Inferring networks of diffusion and influence. *ACM Transactions on Knowledge Discovery from Data (TKDD) 5*, 4 (2012), 1–37.

[11] Gomez-Rodriguez, M., Leskovec, J., and Schölkopf, B. Modeling information propagation with survival theory. In *Proceedings of the 30th International Conference on Machine Learning* (Atlanta, Georgia, USA, 17–19 Jun 2013), S. Dasgupta and D. McAllester, Eds., vol. 28 of *Proceedings of Machine Learning Research*, PMLR, pp. 666–674.

[12] Goyal, P., Chhetri, S. R., Mehrabi, N., Ferrara, E., and Canedo, A. Dynamicgem: A library for dynamic graph embedding methods. *arXiv preprint arXiv:1811.10734* (2018).

[13] Grover, A., and Leskovec, J. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining* (2016), pp. 855–864.

[14] Hadsell, R., Chopra, S., and LeCun, Y. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)* (2006), vol. 2, pp. 1735–1742.

[15] Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *Advances in neural information processing systems* (2017), pp. 1024–1034.

[16] Hamilton, W. L., Ying, R., and Leskovec, J. Representation learning on graphs: Methods and applications. *IEEE Data Eng. Bull. 40*, 3 (2017), 52–74.

[17] Hassani, K., and Khasahmadi, A. H. Contrastive multi-view representation learning on graphs. In *Proceedings of the 37th International Conference on Machine Learning* (13–18 Jul 2020), H. D. III and A. Singh, Eds., vol. 119 of *Proceedings of Machine Learning Research*, PMLR, pp. 4116–4126.

[18] He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 9729–9738.

[19] Jain, S., White, M., and Radivojac, P. Estimating the class prior and posterior from noisy positives and unlabeled data. In *Proceedings of the 30th International Conference on Neural Information Processing Systems* (2016), pp. 2693–2701.

[20] Kazemi, S. M., and Goel, R. Representation learning for dynamic graphs: A survey. *Journal of Machine Learning Research 21*, 70 (2020), 1–73.

[21] Kipf, T. N., and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[22] Kipf, T. N., and Welling, M. Variational graph auto-encoders, 2016.

[23] Kumar, S., Zhang, X., and Leskovec, J. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2019), pp. 1269–1278.

[24] Leskovec, J., Kleinberg, J., and Faloutsos, C. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining* (2005), pp. 177–187.

[25] Liu, X., Zhang, F., Hou, Z., Wang, Z., Mian, L., Zhang, J., and Tang, J. Self-supervised learning: Generative or contrastive, 2021.

[26] Logeswaran, L., and Lee, H. An efficient framework for learning sentence representations. In *International Conference on Learning Representations* (2018).

[27] Ma, Y., Guo, Z., Ren, Z., Tang, J., and Yin, D. Streaming graph neural networks. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (2020), pp. 719–728.

[28] Murray, D. G., McSherry, F., Isard, M., Isaacs, R., Barham, P., and Abadi, M. Incremental, iterative data processing with timely dataflow. *Commun. ACM 59*, 10 (Sept. 2016), 75–83.

[29] Oord, A. v. d., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748* (2018).

[30] Pareja, A., Domeniconi, G., Chen, J., Ma, T., Suzumura, T., Kanezashi, H., Kaler, T., Schardl, T. B., and Leiserson, C. E. EvolveGCN: Evolving graph convolutional networks for dynamic graphs. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence* (2020).

[31] Polyanskiy, Y., and Wu, Y. Lecture notes on information theory. *Lecture Notes for ECE563 (UIUC) and 6*, 2012-2016 (2014), 7.

[32] Qiu, J., Chen, Q., Dong, Y., Zhang, J., Yang, H., Ding, M., Wang, K., and Tang, J. Gcc: Graph contrastive coding for graph neural network pre-training. KDD '20, Association for Computing Machinery, p. 1150–1160.

[33] Rahimi, A., and Recht, B. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems* (2008), J. Platt, D. Koller, Y. Singer, and S. Roweis, Eds., vol. 20, Curran Associates, Inc.

[34] Robinson, J., Chuang, C.-Y., Sra, S., and Jegelka, S. Contrastive learning with hard negative samples. *arXiv preprint arXiv:2010.04592* (2020).

[35] Rossi, E., Chamberlain, B., Frasca, F., Eynard, D., Monti, F., and Bronstein, M. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637* (2020).

[36] Sankar, A., Wu, Y., Gou, L., Zhang, W., and Yang, H. Dysat: Deep neural representation learning on dynamic graphs via self-attention networks. In *Proceedings of the 13th International Conference on Web Search and Data Mining* (2020), pp. 519–527.

[37] Saunshi, N., Plevrakis, O., Arora, S., Khodak, M., and Khandeparkar, H. A theoretical analysis of contrastive unsupervised representation learning. In *Proceedings of the 36th International Conference on Machine Learning* (Long Beach, California, USA, 09–15 Jun 2019), K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97 of *Proceedings of Machine Learning Research*, PMLR, pp. 5628–5637.

[38] Sun, F.-Y., Hoffman, J., Verma, V., and Tang, J. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *International Conference on Learning Representations* (2019).

[39] Sun, K., Lin, Z., and Zhu, Z. Multi-stage self-supervised learning for graph convolutional networks on graphs with few labeled nodes. *Proceedings of the AAAI Conference on Artificial Intelligence 34*, 04 (Apr. 2020), 5892–5899.

[40] Trivedi, R., Farajtabar, M., Biswal, P., and Zha, H. Dyrep: Learning representations over dynamic graphs. In *International Conference on Learning Representations* (2019).

[41] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *International Conference on Learning Representations* (2018).

[42] Veličković, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y., and Hjelm, R. D. Deep graph infomax. In *International Conference on Learning Representations* (2019).

[43] Wang, Y., Chang, Y.-Y., Liu, Y., Leskovec, J., and Li, P. Inductive representation learning in temporal networks via causal anonymous walks. In *International Conference on Learning Representations* (2021).

[44] Wu, Z., Xiong, Y., Stella, X. Y., and Lin, D. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018).

[45] Xu, D., Ruan, C., Korpeoglu, E., Kumar, S., and Achan, K. Self-attention with functional time representation learning. In *Advances in Neural Information Processing Systems* (2019), pp. 15889–15899.

[46] Xu, D., Ruan, C., Korpeoglu, E., Kumar, S., and Achan, K. Inductive representation learning on temporal graphs. *arXiv preprint arXiv:2002.07962* (2020).

[47] Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *International Conference on Learning Representations* (2019).

[48] You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., and Shen, Y. Graph contrastive learning with augmentations. In *Advances in Neural Information Processing Systems* (2020), H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., pp. 5812–5823.

[49] You, Y., Chen, T., Wang, Z., and Shen, Y. When does self-supervision help graph convolutional networks? In *Proceedings of the 37th International Conference on Machine Learning* (13–18 Jul 2020), H. D. III and A. Singh, Eds., vol. 119 of *Proceedings of Machine Learning Research*, PMLR, pp. 10871–10880.