Self-Supervised Dynamic Graph Representation Learning via Temporal Subgraph Contrast

KE-JIA CHEN, LINSONG LIU, LINPU JIANG, and JINGQIANG CHEN, Nanjing University of Posts and Telecommunications, China

Self-supervised learning on graphs has recently drawn a lot of attention due to its independence from labels and its robustness in representation. Current studies on this topic mainly use static information such as graph structures but cannot well capture dynamic information such as timestamps of edges. Realistic graphs are often dynamic, which means the interaction between nodes occurs at a specific time. This article proposes a self-supervised dynamic graph representation learning framework DySubC, which defines a temporal subgraph contrastive learning task to simultaneously learn the structural and evolutional features of a dynamic graph. Specifically, a novel temporal subgraph sampling strategy is firstly proposed, which takes each node of the dynamic graph as the central node and uses both neighborhood structures and edge timestamps to sample the corresponding temporal subgraph. The subgraph representation function is then designed according to the influence of neighborhood nodes on the central node after encoding the nodes in each subgraph. Finally, the structural and temporal contrastive loss are defined to maximize the mutual information between node representation and temporal subgraph representation. Experiments on five real-world datasets demonstrate that (1) DySubC performs better than the related baselines including two graph contrastive learning models and five dynamic graph representation learning models, especially in the link prediction task, and (2) the use of temporal information cannot only sample more effective subgraphs, but also learn better representation by temporal contrastive loss.

CCS Concepts: • Theory of computation → Dynamic graph algorithms; • Computing methodologies → Unsupervised learning: Learning latent representations;

Additional Key Words and Phrases: Self-supervised learning, temporal subgraph contrast, dynamic graph representation learning

ACM Reference format:
Ke-Jia Chen, Linsong Liu, Linpu Jiang, and Jingqiang Chen. 2023. Self-Supervised Dynamic Graph Representation Learning via Temporal Subgraph Contrast. ACM Trans. Knowl. Discov. Data. 18, 1, Article 18 (September 2023), 20 pages.
https://doi.org/10.1145/3612931

1 INTRODUCTION

Graph-structured data, such as social networks [1], collaboration networks [17], and chemical molecular graphs [13], are ubiquitous in the real world. They naturally represent entities and their relationships. Graph representation learning aims to transform nodes into low-dimensional dense
embeddings that preserve attributive and structural features of the graph. The method using graph neural networks (GNNs) [9, 26, 33, 36] has recently drawn considerable attention and achieved excellent performance. However, the supervised or semi-supervised GNNs heavily rely on labels with high acquisition cost, which could lead to poor generalization or weak robustness under label-related adversarial attacks [15].

Self-supervised learning can effectively alleviate the above problems. By designing appropriate training tasks on the graph, the self-supervised model can learn a more generalized graph representation in the absence of labels. Most of the existing methods focus on learning representations from the structural perspective by contrasting graph elements of different scales, such as node and local subgraph contrasting [7], node and global graph contrasting [5, 34], and so on.

However, neither graph sampling nor graph contrastive learning in these methods take the graph dynamics (especially the temporal information of edge generation) into account. As a result, the learned node representation fails to reflect the graph evolution. Take the social network in Figure 1 as an example, where the edge represents the friend relationship of two person nodes. Each edge is marked with a timestamp \( t_1 < t_2 < \cdots < t_9 \), indicating the moment when they became friends. The central node Mary used to be an engineer, and most of her neighbor nodes were engineers before \( t_6 \). Recently, Mary changed her career to become a teacher and started to have teacher neighbor nodes (e.g., she will befriend a teacher at time \( t_9 \)). The static graph representation learning model samples subgraphs and encodes the nodes only based on structures, so it could predict that Mary will befriend an engineer instead of a teacher at time \( t_9 \). By using temporal information, both subgraph sampling and node representation can capture Mary’s relationship evolution, which may give a more accurate prediction.

Nodes are usually more closely related to their regional neighbors [7]. So intuitively, frequent changes of neighbors will reflect potential representation of the central node and the latest neighbors interacting with the central node have higher influence on its representation. This article proposes a novel self-supervised dynamic graph representation learning method DySubC, which learns node representation by sampling and contrasting with temporal subgraphs. To be specific, a temporal subgraph sampling strategy is firstly proposed, which takes each node as the central node to sample the corresponding temporal subgraph. The subgraphs are sampled by considering not only the local structure but also the temporal information of neighborhood interactions. Then, a certain GNN encoder is used to learn node representation for each temporal subgraph, thus outputs the representation of the central node and the summary of the temporal subgraph calculated by a time-aware readout function defined in this article. Finally, the central node is paired with (a) its corresponding time-weighted subgraph as a positive sample, (b) any other temporal subgraph
in a shuffled sampling set as a structural negative sample, and (c) its corresponding unweighted subgraph as a temporal negative sample to train the whole model by maximizing the mutual information of the central node and its temporal subgraph.

In summary, the main contributions of the DySubC framework include:

- It poses the task of dynamic graph contrast learning for the first time and aims to capture the structural evolution characteristics of graphs by introducing temporal information.
- A temporal subgraph sampling method is proposed, which simultaneously uses the structural and temporal information of neighborhoods.
- A new readout function is defined to get the summary of a temporal subgraph, which takes the influence of the neighborhood nodes on the central node into consideration.
- A temporal subgraph contrast loss is defined including structure contrast loss and time contrast loss.
- Extensive experiments verify the superiority of DySubC in link prediction compared to the related continuous time graph representation learning models and graph contrastive learning models. The ablation study further proves the effectiveness of each time-enhanced module.

The rest of this article includes the following sections. In Section 2, the related work on dynamic graph representation learning and self-supervised graph learning is briefly reviewed. Section 3 introduces the proposed DySubC framework in detail. The experimental results are shown and analyzed in Section 4. Section 5 concludes the article.

2 RELATED WORK

2.1 Dynamic Graph Representation Learning

GNNs [9, 26, 33, 36] have achieved the competitive performance in static graph representation learning [3, 24, 30, 35]. Recently, the dynamic graph representation learning has drawn more and more attentions due to its ability to incorporate temporal information into representations. There are two different types of dynamic graphs, i.e., discrete time graphs and continuous time graphs. Accordingly, dynamic graph representation learning models are roughly divided into two categories.

Discrete time graph refers to a dynamic graph formalized as a series of graph snapshots with the same time interval between them. This type of methods generally conduct representation learning on each snapshot first and then learns the evolution features of the graph structure over time through a sequence learning model (e.g., RNN [38], self-attention [32], etc.). Representative methods include DynGEM [2], DynamicTriad [40], EvolveGCN [22], DySAT [29], and so on, which focus more on capturing the global evolution features rather than the features of local continuous changes.

The continuous time graph is another formalization of dynamic graph where edges are marked with continuous timestamps. CTDNE [19] proposes a temporal random walk method and then uses skip-gram to obtain node representation. HTNE [43] introduces the Hawkes process theory into the dynamic graph model and learns node representation based on the fact that the influence of neighbors on the central node will change over time. DyRep [31] captures the interleaved dynamics of communication processes and correlation processes, thereby updating node representation. TGN [27] combines the memory module and graph-based operators to update node representations and improves the computational efficiency. MetaDyGNN [37] proposes a meta-learner with hierarchical time interval-wise and node-wise adaptions for few-shot scenarios. The method proposed in this article also falls into the category of continuous time graph representation learning, which mainly uses the changes of regional neighbors to learn node representations.
2.2 Self-Supervised Graph Learning

Self-supervised learning uses pretext tasks to train the model with constructed supervised information from large-scale unsupervised data. It not only alleviates the problem of high cost of acquiring data labels, but also learns effective features. It has been successfully applied in computer vision [8] and natural language processing [14]. For graph representation learning, DGI [34] is the first graph self-supervised learning method that uses the pretext task of maximizing mutual information between node representations and global graph representations. After that, a series of graph contrastive learning models emerged. Sankararaman et al. [5] propose a multi-view graph contrastive learning model MVGRL, which treats the original graph structure and graph diffusion as two different views to maximize mutual information between nodes and cross-view representations of large-scale graphs. The graph contrastive learning in GMI [23] is achieved by maximizing the mutual information between the representation of each node and the original features of its one-hop neighbors. Sub-Con [7] maximizes the mutual information between node representations and subgraph representations, which can be used for large-scale graphs.

In addition to the above methods, [25, 41, 42] introduce the contrastive learning method in machine vision into graph representation learning, which adopts different strategies for positive and negative sample construction and different loss functions.

However, the existing graph self-supervised methods do not take into consideration the fact that graphs in real world are often dynamic. The temporal information is not well used not only in the pretext task but also in the definition of the objective function. The work of this article attempts to bridge this gap.

3 THE PROPOSED METHOD

3.1 Preliminaries

Before detailing the model, a formal definition of dynamic graph is given. Considering that the interaction between nodes occurs at a specific time and the graph is constantly changing over time, this article models the dynamic graph as a continuous time graph.

**Definition 1 (Dynamic Graphs).** Given a graph \( G = (V, E_t, X) \), \( V \) is a set of vertices, \( E_t \subseteq V \times V \times \mathbb{R}^+ \) is a set of edges with timestamp \( t \subseteq \mathbb{R}^+ \) and \( X \) denotes the matrix of node attributes. \( e = (u, v, t) \subseteq E_t \) represents the interaction between node \( u \) and \( v \) at time \( t \). Note that when the nodes are not attributed, one-hot encoding is often used to initialize \( X \).

**Problem 1 (Dynamic Graph Representation Learning).** For a dynamic graph \( G = (V, E_t, X) \), the task is to learn the mapping function \( f : V \rightarrow \mathbb{R}^D \) to embed the node in a \( d \)-dimensional vector space. The node representation is supposed to contain both structural and temporal information and suitable for downstream machine learning tasks such as link prediction.

3.2 Overview

The proposed DySubC (Dynamic graph representation learning via temporal Subgraph Contrast) framework uses graph contrastive learning that captures the structural and temporal features from continuous-time graphs during the training phase without additional supervision. An overview of the DySubC framework is shown in Figure 2, which mainly includes three time-enhanced modules.

- **Temporal subgraph sampling.** Firstly, a temporal subgraph for each node \( i \) in the original graph is sampled using both structural and temporal information, generating a time-weighted subgraph \( G_i = (X_i, A_i) \) and an unweighted counterpart \( G'_i = (X'_i, A'_i) \).
- **Node and subgraph representation.** Secondly, \( G_i \) and \( G'_i \) are encoded through GNN and then the summary of each subgraph is represented by a readout function, respectively. For
Fig. 2. The overall framework of DySubC. DySubC first samples the temporal subgraph $G_i$ for each node. Then, taking the yellow central node as an example, DySubC encodes all the nodes in $G_i$ including the central node $i$ (represented as $h_i$). The time-weighted subgraph representation $s_i$ and the unweighted subgraph representation $s'_i$ are calculated by two readout functions, respectively. Meanwhile, the other temporal subgraphs are shuffled to get a subgraph $\tilde{G}_i$ with its representation $\tilde{s}_i$. Finally, one positive sample and two negative samples are generated to calculate contrastive loss and train the model.

each node $i$, we have its representation $h_i$, a time-weighted subgraph representation $s_i$ and an unweighted subgraph representation $s'_i$.

- **Temporal contrastive learning.** Finally, a positive sample $s_i$, a temporal negative sample $s'_i$ and a structural negative sample $\tilde{s}_i$ are constructed for each central node $i$ represented by $h_i$. The model is trained by maximizing the mutual information of the central node representation and the time-weighted subgraph representation.

Note that the temporal subgraph sampling can be completed independently before the start of training. Thus, it does not take up the running time of the model.

### 3.3 Temporal Subgraph Sampling

The temporal subgraph sampling module is first proposed to generate training samples for self-supervised learning.

By considering both the structure of neighbors and the timestamp of edge interactions, the temporal subgraph sampler (see Algorithm 1) can sample a temporal subgraph with a fixed number of nodes $k$ for each central node $i$. The specific steps are as follows.

Firstly, all first-order neighbors of $i$ are sampled. Since the number of first-order neighbors is usually less than $k$, the second-order or even the higher-order neighbors of $i$ may be sampled until the number of candidate nodes is greater than or equal to $k$.

Then, if the number of candidate nodes exceeds $k$, a selection strategy is adopted to select more important candidate nodes into the sampling pool according to their importance score, which is defined as the combination of structural importance score $S^j_{structure}$ (Equation (1)) and temporal importance score $S^j_{time}$ (described below), where $j$ is the node id in the sampling path.

The structural importance score $S^j_{structure}$ is defined as the degree of the node:

$$S^j_{structure} = Degree(j).$$

Actually, it can also be defined using other metrics such as the eigenvector centrality [18], influence in PageRank [18], and so on. This article finally adopts the node degree metric, as it can better reflect the importance of local structure. The comparative analysis of different structure importance metrics can be seen in the experimental section.
ALGORITHM 1: Temporal Subgraph Sampler

Input: Dynamic graph $G = (V, E_t, X)$; Subgraph size $k$.
Output: A time-weighted subgraph $G_i = (X_i, A_i)$ and an unweighted subgraph $G_i' = (X_i', A_i')$ for each node $i$.

1: Preprocess $G$ to get the time-weighted adjacency matrix $A$ and the unweighted adjacency matrix $A'$.
2: for each node $i$ do
3: Initialize the queue $q = \emptyset$, the sampling pool $Pl = \emptyset$ and the number of sampled nodes $count = 0$.
4: Add $i$ to $q$ and $Pl$ respectively, $count = 1$.
5: repeat
6: Add $\text{Neighbor}(v)$ ($\forall v \in q$) into the candidate set $Cand$.
7: if $|Cand| < (k - count)$ then
8: Add all the nodes in $Cand$ into $q$ and $Pl$, respectively.
9: $count \leftarrow count + |Cand|$.
10: else
11: Calculate the importance score $S$ (Equations (1)–(3)) of each node in the $Cand$.
12: Take the $k - count$ nodes with the largest $S$ value from $Cand$ and add them into $Pl$.
13: end if
14: until $count == k$
15: Calculate $G_i$ and $G_i'$ with $Pl$ using Equation (4).
16: end for

The temporal importance score $S_{time}^j$ is a normalized latest timestamp of the edge connected to node $j$:

$$S_{time}^j = (t^j - t_{min})/(t_{max} - t_{min}) \ast 10,$$

where $t^j$, $t_{min}$ and $t_{max}$ are the timestamp of node $j$, the minimum timestamp and the maximum timestamp, respectively. We believe that the neighbor node with a more recent connection has a greater impact on the central node.

As a result, the importance score of node $j$ is denoted as

$$S^j = S_{time}^j + \alpha S_{structure}^j,$$

where, $\alpha$ is the hyperparameter used to balance two influence scores.

An example is shown in Figure 3, where $t$ represents the timestamp and $S$ represents the importance score. Assuming that $k = 10$, for the yellow central node, its 4 first-order neighbors are first sampled and then its 11 second-order neighbors are also sampled, increasing to 16 nodes in the candidate set. According to Equation (3), 5 blue nodes with the highest importance score will be saved as the sampled nodes.

Finally, after the $k$ nodes are sampled, the time-weighted subgraph $G_i$ and unweighted subgraph $G_i'$ of node $i$ are obtained referring to the original graph [7], represented by the adjacency matrix $A_i$ and $A_i'$, respectively. The edge weight in $A_i$ is the normalization of its latest timestamp. The feature matrix $X_i$ and the adjacency matrix $A_i$ of $G_i$ are:

$$X_i = X_{idx,i}, A_i = A_{idx,idx},$$

where, $idx$ represents the index of the sampled node. $X_{idx,:}$ is the row-wise (i.e., node-wise) indexed feature matrix. $A_{idx,idx}$ is the row-wise and col-wise indexed adjacency matrix. Similarly,
the feature matrix $X'_i$ and adjacency matrix $A'_i$ of $G'_i$ are also obtained. $A'_i$ shares the same set of edges as $A_i$, but has different weights on edges.

As a result, the output of the temporal subgraph sampling module for each node $i$ includes a time-weighted subgraph $G_i = (X_i, A_i)$ and an unweighted subgraph $G'_i = (X'_i, A'_i)$. Both of them will be used in the subsequent contrastive learning task.

### 3.4 Node and Subgraph Representation

After obtaining $G_i = (X_i, A_i)$ and $G'_i = (X'_i, A'_i)$, the encoder $E_1$ and $E_2$ are used to output their representation $H_i$ and $H'_i$, respectively:

$$H_i = E_1(X_i, A_i), H'_i = E_2(X'_i, A'_i).$$  \hfill (5)

For simplicity, both $E_1$ and $E_2$ adopt a one-layer graph convolutional network (GCN) [9] that can efficiently aggregate neighbor information. Take $G_i$ as an example, the propagation rule is as follows:

$$E(X_i, A_i) = \sigma \left( \hat{D}_i^{-\frac{1}{2}} \hat{A}_i \hat{D}_i^{-\frac{1}{2}} X_i W \right),$$  \hfill (6)

where $\hat{A}_i = A_i + I$ is the adjacency matrix inserted with a self-loop, and $\hat{D}_i$ is the corresponding degree matrix. The non-linear function $\sigma$ is the parametric ReLU (PReLU) function [6] and $W$ is a learnable linear transformation.

The representation $h_i$ of the central node $i$ is then picked out from the representation matrix $H_i$:

$$h_i = \mathcal{P}(H_i),$$  \hfill (7)

where $\mathcal{P}$ denotes the pick-out operation to select the representation of node $i$ in $H(i)$.

To facilitate the subsequent contrast learning task, the time-weighted subgraph representation $s_i$ and the unweighted subgraph representation $s'_i$ are obtained by using two readout functions $\mathcal{R}_1$ and $\mathcal{R}_2$, respectively.

$\mathcal{R}_1$ is a time-aware readout function designed in this article, which calculates the influence score of any node $i$ on the central node $j$ (Equation (8)) and then performs a weighted average of node representations to get $s_i$ (Equation (9)).

$$\text{Inf}_{ij} = \tau(i, j) + \beta \frac{1}{\text{Dist}(i, j)},$$  \hfill (8)
where, $\tau(i, j)$ represents the latest interaction timestamp between $i$ and $j$, $\text{Dist}(i, j)$ represents the shortest distance between $i$ and $j$ and $\beta$ is a hyperparameter.

\[ s_i = \frac{1}{\sum_{j=1}^{k} \text{Inf}_j} \sum_{j=1}^{k} \text{Inf}_j h_j, \quad (9) \]

where, $k$ is the number of nodes in $G_i$.

$\mathcal{R}_2$ is used to average all node representations in the subgraph $G'_i$:

\[ s'_i = \frac{1}{k} \sum_{j=1}^{k} h'_j, \quad (10) \]

**Algorithm 2:** The DySubC Algorithm

**Input:** Dynamic graph $G = (V, E_t, X)$.

**Output:** node representations $\{h_1, h_2, \ldots, h_{|V|}\}$

1: Sample $\{G_1, G_2, \ldots, G_{|V|}\}$ and $\{G'_1, G'_2, \ldots, G'_{|V|}\}$ for each node by the temporal subgraph sampler.

2: while not converge do
   3: for each $G_i$ and $G'_i$ do
      4: Encode $G_i$ and $G'_i$ into $H_i$ and $H'_i$ through $\mathcal{E}_1$ and $\mathcal{E}_2$ (Equation (5)).
      5: Pick out $h_i = \mathcal{P}(H_i)$ (Equation (7)).
      6: Obtain $s_i$ and $s'_i$ through the readout function $\mathcal{R}_2$ and $\mathcal{R}_1$ respectively (Equation (9)–(10)).
   7: end for
   8: Shuffle the set of $\{s_1, s_2, \ldots, s_{|V|}\}$ to generate $\{	ilde{s}_1, \tilde{s}_2, \ldots, \tilde{s}_{|V|}\}$.
   9: Construct a positive sample $s_i$, a temporal negative sample $s'_i$ and a structural negative sample $\tilde{s}_i$.
   10: Update parameters of $\mathcal{E}_1$ and $\mathcal{E}_2$ by Equation (14).
11: end while
12: for each node $i$ do
   13: Calculate the representation $h_i = \mathcal{P}(H_i)$ by Equation (7).
14: end for

### 3.5 Temporal Contrastive Learning

Pretext tasks and the generation of positive and negative samples are crucial for self-supervised learning. As mentioned above, the dynamics of neighborhood nodes have greater influence on the central node than that of distant nodes. The pretext task is designed to make the central node strongly correlated to its regional neighbors. In our method, the mutual information of the central node and its corresponding temporal subgraph is maximized.

For the central node $i$, the pretext task is to contrast its real temporal subgraph to its fake temporal subgraph. Specifically, a positive sample, a structural negative sample and a temporal negative sample are constructed for $h_i$. The positive sample is the time-weighted subgraph representation $s_i$. Consistent with the negative sample generation strategy in Sub-Con [7], the structural negative sample is generated by shuffling the representation set of time-weighted subgraphs, denoted as

\[ \{	ilde{s}_1, \tilde{s}_2, \ldots, \tilde{s}_{|V|}\} = \text{Shuffle}(\{s_1, s_2, \ldots, s_{|V|}\}). \quad (11) \]
The subgraph representation of other central nodes is regarded as negative sample so that nodes are closely related to their context subgraphs and weakly associated with other subgraphs. The temporal negative sample is the unweighted subgraph representation $s'_i$, with the purpose to make the node representation $h_i$ closer to the time-weighted subgraph representation $s_i$ and farther away from the unweighted subgraph representation $s'_i$. The temporal contrast information is therefore emphasized.

Finally, the margin triplet loss is used to train the model as it is more favorable for subgraph contrastive learning [7]. The margin loss of the structural negative sample is defined as

$$L_1 = \frac{1}{|V|} \sum_{i=1}^{|V|} \mathbb{E}_{(X,A)}(-\max(\sigma(h_is_i) - \sigma(h_i\tilde{s}_i) + \phi, 0)),$$

where $\sigma(x) = 1/(1 + \exp(-x))$ is the sigmoid function and $\phi$ is the margin value. Similarly, the margin loss of the temporal negative sample is defined as

$$L_2 = \frac{1}{|V|} \sum_{i=1}^{|V|} \mathbb{E}_{(X,A)}(-\max(\sigma(h_is_i) - \sigma(h_is'_i) + \phi, 0)),$$

where $\phi$ is the margin value. As a result, the total loss function of the model is

$$\mathcal{L} = \lambda L_1 + \lambda L_2,$$

where $\lambda$ is a hyperparameter to balance two loss.

The process of DySubC is summarized in Algorithm 2.

4 EXPERIMENTS

The following experiments are conducted to evaluate the DySubC model from various aspects.

- The performance of DySubC and the related baseline methods on link prediction and node classification are compared, thereby reflecting their representation learning ability.
- The effectiveness of each proposed time-enhanced module and how it affects the overall model is evaluated.
- In the temporal subgraph sampling module, different structural importance metrics are compared, the impact of the subgraph size on the model is explored, and the balance between memory and performance is analyzed.
- The sensitivity of the model to hyperparameters is analyzed.
- The visualization of node representations obtained by DySubC and Sub-Con is compared.
- The complexity of DySubC and other baseline models is analyzed.

Before detailing the experimental results and analysis, we start with a brief introduction of the datasets, the experimental settings, and the baselines.

4.1 Datasets

For a comprehensive comparison, we use seven widely used datasets collected from different types of real dynamic networks, the first five for link prediction and the last two for node classification. The detailed statistics of the datasets are summarized in Table 1.

- Forum [20]. The dataset is a forum network similar to Facebook, obtained from online social networks. The directed edge $<u, v, t>$ means that user $u$ and user $v$ interacted at time $t$.
- Bitcoin Alpha [10, 11]. This is a who-trusts-whom network among people who trade using Bitcoin on a platform called Bitcoin Alpha. The directed edge $<u, v, t>$ indicates that user $u$ trusted user $v$ at time $t$. 

ACM Transactions on Knowledge Discovery from Data, Vol. 18, No. 1, Article 18. Publication date: September 2023.
Table 1. Statistics of Dynamic Graph Datasets

| Dynamic graph   | |V|| |E| | Timespan (days) | Nodes with dynamic labels | Labels |
|-----------------|---|---|---|---|---|---|---|---|
| Forum           | 899 | 33,720 | 164.49 | – | – |
| Bitcoin Alpha   | 3,783 | 24,186 | 1,901.00 | – | – |
| Wikipedia       | 7,118 | 107,071 | 1,378.34 | – | – |
| Movielens       | 16,528 | 95,580 | 1,108.97 | – | – |
| Math Overflow   | 16,836 | 203,639 | 2,349.00 | – | – |
| Mooc            | 1,735 | 26,169 | 37.01 | 234 | 2 |
| Reddit          | 11,000 | 672,447 | 31.00 | 366 | 2 |

– Wikipedia [28]. The dataset contains the administrator election and voting data based on the latest complete dump of Wikipedia page edit history (from January 3, 2008). The directed edge \(<u, v, t>\) indicates that \(u\) voted for \(v\) at time \(t\).

– Movielens [28]. This bipartite network represents the tagging behaviors of MovieLens users. The nodes represent users and movies. The directed edge \(<u, v, t>\) means that the user \(u\) tagged the movie \(v\) at time \(t\).

– Math Overflow [21]. It is a temporal interaction network on the stack exchange website Math Overflow. The directed edge \(<u, v, t>\) means that user \(u\) commented on user \(v\)’s question at time \(t\).

– Mooc [12]. This dataset is a social interaction network from online course content units. Each node is a student, and the directed edge \(<u, v, t>\) means that student \(u\) interacts with student \(v\) at time \(t\). The label of a node indicates whether a student drops out of a course or not, which may change over time. The number of nodes with dynamic labels and the number of labels are listed in Table 1.

– Reddit [12]. It is a social network. The nodes represent users and posts. The directed edge \(<u, v, t>\) means that user \(u\) requests to publish post \(v\) at time \(t\). The label of a node means whether a user is banned or not.

4.2 Experimental Settings

Since the nodes in the above datasets have no features, we use one-hot encoding as the initial features of the node. For link prediction, we first sort the edges in the graph in ascending order of time, using the recent 25% randomly divided as validation set (10%) and test set (15%) and the remaining 75% as the training set. For the recent 25% edges (i.e., positive samples), we randomly sample the same number of negative samples (unconnected node pairs). Node embeddings at the last moment are used for prediction. For node classification, we use 75% nodes as the training set, and the remaining 25% nodes as the test set.

For each experiment, the dimension of node representations is set to 128. A simple logistic regression classifier is trained and tested for link prediction using the embedding results. We train the model for 10 times on different data splits and report the average performance for fair evaluation. Consistent with previous work [19, 27, 31, 43], we use AUC and accuracy indicator to evaluate the performance of link prediction. We also use mean average rank (MAR) and Hits@10 [31] to rank positive edges against negative edges in the prediction results. Accuracy and precision are used to evaluate the performance of node classification.

In training, the Adam optimizer is used with an initial learning rate of 0.001. For all datasets, the size of subgraphs is set to 20. Both the margin value \(\varphi\) and \(\phi\) for the loss function are set to 0.75. The parameters \(\alpha\), \(\beta\) and \(\lambda\) are set to the optimal values in each datasets.
| Forum | Bitcoin Alpha | Wikipedia | Movielens | Math Overflow |
|-------|---------------|-----------|-----------|---------------|
| node2vec | 157.69 ± 2.7 | 86.04 ± 0.8 | 75.19 ± 0.3 | 92.48 ± 0.5 |
| graphSAGE | 141.91 ± 2.3 | 79.32 ± 0.5 | 70.71 ± 0.3 | 90.25 ± 0.5 |
| SEAL | 195.63 ± 2.3 | 93.64 ± 0.7 | 80.93 ± 0.3 | 89.27 ± 0.3 |
| DGI | 251.62 ± 2.7 | 85.90 ± 0.8 | 72.78 ± 0.3 | 90.87 ± 0.5 |
| Sub-Con | 193.91 ± 2.3 | 79.32 ± 0.5 | 70.71 ± 0.3 | 90.25 ± 0.5 |
| GCA | 219.63 ± 2.3 | 93.64 ± 0.7 | 80.93 ± 0.3 | 89.27 ± 0.3 |
| CTDNE | 180.11 ± 2.7 | 85.90 ± 0.8 | 72.78 ± 0.3 | 90.87 ± 0.5 |
| HTNE | 252.34 ± 2.7 | 85.90 ± 0.8 | 72.78 ± 0.3 | 90.87 ± 0.5 |
| DyRep | 188.56 ± 2.7 | 85.90 ± 0.8 | 72.78 ± 0.3 | 90.87 ± 0.5 |
| TGN | 173.49 ± 2.7 | 85.90 ± 0.8 | 72.78 ± 0.3 | 90.87 ± 0.5 |
| MetaDyGNN | 179.62 ± 2.7 | 85.90 ± 0.8 | 72.78 ± 0.3 | 90.87 ± 0.5 |

The best result is bolded.

| Forum | Bitcoin Alpha | Wikipedia | Movielens | Math Overflow |
|-------|---------------|-----------|-----------|---------------|
| node2vec | 215.69 ± 2.7 | 86.04 ± 0.8 | 75.19 ± 0.3 | 92.48 ± 0.5 |
| graphSAGE | 141.91 ± 2.3 | 79.32 ± 0.5 | 70.71 ± 0.3 | 90.25 ± 0.5 |
| SEAL | 195.63 ± 2.3 | 93.64 ± 0.7 | 80.93 ± 0.3 | 89.27 ± 0.3 |
| DGI | 251.62 ± 2.7 | 85.90 ± 0.8 | 72.78 ± 0.3 | 90.87 ± 0.5 |
| Sub-Con | 193.91 ± 2.3 | 79.32 ± 0.5 | 70.71 ± 0.3 | 90.25 ± 0.5 |
| GCA | 219.63 ± 2.3 | 93.64 ± 0.7 | 80.93 ± 0.3 | 89.27 ± 0.3 |
| CTDNE | 180.11 ± 2.7 | 85.90 ± 0.8 | 72.78 ± 0.3 | 90.87 ± 0.5 |
| HTNE | 252.34 ± 2.7 | 85.90 ± 0.8 | 72.78 ± 0.3 | 90.87 ± 0.5 |
| DyRep | 188.56 ± 2.7 | 85.90 ± 0.8 | 72.78 ± 0.3 | 90.87 ± 0.5 |
| TGN | 173.49 ± 2.7 | 85.90 ± 0.8 | 72.78 ± 0.3 | 90.87 ± 0.5 |
| MetaDyGNN | 179.62 ± 2.7 | 85.90 ± 0.8 | 72.78 ± 0.3 | 90.87 ± 0.5 |

The best result is bolded.

### 4.3 Baselines
We use three types of representative baseline methods for comparison: (1) Continuous-time dynamic graph representation learning methods: TGN [27], DyRep [31], CTDNE [19], HTNE [43], and MetaDyGNN [37]; (2) State-of-the-art static graph self-supervised methods: DGI [34], Sub-Con [7] and GCA [42]; (3) Static graph representation learning benchmarks: Node2vec [3], GraphSAGE [4] and SEAL [39]. Note that when reproducing the codes of different models, we carefully select the reported optimal hyperparameters to ensure a fair comparison.

### 4.4 Performance on Link Prediction
The comparative results of all methods in link prediction performance are summarized in Table 2 and Table 3. Overall, our proposed model shows a competitive performance on both global prediction and ranking of nodes. The performance of DySubC is slightly lower than that of TGN on the Math Overflow dataset. The possible reason for the good performance of TGN on the Math Overflow dataset is that its memory module will be more advantageous for datasets with a large time span. node2vec achieves highest Hits@10 score on Forum but the scores on other datasets are low. A possible reason is that Forum is denser than other datasets, so it is more beneficial for node2vec to learn information in local field. The methods that use self-supervised learning (DGI, Sub-Con, GCA, and DySubC) perform better than methods that do not use self-supervised learning (node2vec and graphSAGE).
More detailed observations and analysis are as follows. Firstly, the performance of our model is significantly higher than static self-supervised graph representation learning methods (i.e., DGI and Sub-Con). Especially on the Math Overflow dataset, the AUC score of DySubC is nearly 0.9 higher than DGI and 0.1 higher than Sub-Con. It verifies the importance of temporal information for graph representation learning models and the temporal subgraph contrastive method in this article can effectively capture temporal information. Secondly, on the bipartite graph dataset (i.e., Movielens), the performance of most continuous-time dynamic graph representation learning methods is poor. In contrast, the DySubC model achieves the best performance, which indicates that our method is more robust and can be applied to different types of graphs. Thirdly, DySubC performs significantly better than other models on Bitcoin Alpha and Movielens datasets. These two graphs are relatively sparse, which could cause the deterioration in model performance. However, DySubC may alleviate the above problem caused by the graph sparsity since it learns at the sub-graph level. Finally, the performance of Sub-Con on the Movielens dataset is poor, probably because its subgraph sampling strategy is a personalized PageRank algorithm \[18\]. It may not sample sufficient first-order neighbors, which are particularly important for the central node in the bipartite graph.

4.5 Performance on Node Classification

The results of node classification are listed in Table 4. Overall, dynamic graph methods achieve better results than static graph methods, indicating that the leverage of time information is also helpful for learning the node distribution. Our method can achieve better node classification performance than most methods, but slightly lower than Sub-Con and TGN on Mooc and slightly lower than TGN in Reddit. Sub-Con performs best on Mooc. The sampling strategy of Sub-Con samples nearby nodes in the structure, while that of DySubC may pick nodes with high temporal importance but far distance, so the former can better learn the local distribution of node features. The corruption strategy of Sub-Con determines the differentiation of nodes with different contexts, which is also beneficial for node classification task. For Mooc, its scale is relatively small and the node labels change less, so the learning ability of the static Sub-Con method to local structures is further highlighted. The overall performance of TGN is higher than DySubC. TGN can be trained for a variety of tasks such as link prediction or node classification, so its loss function is favorable for node classification. DySubC is a self-supervised learning framework and may not perform as well as supervised or semi-supervised models on a specific task.

4.6 Ablation Studies

In order to further observe the impact of three time-enhanced modules (i.e., temporal subgraph sampling, time-weighted subgraph representation and temporal contrastive learning) and different structural importance score strategies (i.e., degree, eigenvector centrality, PageRank) of DySubC, we conduct a series of ablation experiments by replacing each of the three modules with the counterpart that does not utilize the time information and replacing structural importance score function with different measures. The subscript \(-S\) represents that the model replaces the temporal subgraph sampling with a subgraph sampling that only uses structural information. The subscript \(-N\) means that the model does not use the negative sample \(s'_j\) for contrastive learning. The subscript \(-R\) stands for replacing our designed readout function with a simple average function in time-weighted subgraph representation. It is worth noting that DySubC\(-S\)--\(-N\)--\(-R\) and Sub-Con \[7\] are not equivalent, as their subgraph sampling strategies are different. The subscripts \(-DE\), \(-EC\) and \(-PR\) represent using degree, eigenvector centrality, or PageRank as structural importance metrics, respectively.
Table 4. Performance of Node Classification in Terms of Accuracy and Precision (Mean in Percentage ± Standard Deviation over 10 Trial Runs)

|          | Mooc            | Reddit          |
|----------|-----------------|-----------------|
| Accuracy | Precision       | Accuracy        | Precision       |
| node2vec | 84.10 ± 0.3     | 70.73 ± 0.4     | 90.20 ± 0.6     | 86.54 ± 0.4     |
| graphSAGE| 86.76 ± 0.3     | 75.62 ± 0.2     | 91.83 ± 0.5     | 87.59 ± 0.3     |
| SEAL     | 85.32 ± 0.4     | 75.39 ± 0.6     | 90.95 ± 0.3     | 87.42 ± 0.3     |
| DGI      | 85.48 ± 0.4     | 73.07 ± 0.1     | 89.88 ± 0.2     | 86.31 ± 0.5     |
| Sub-Con  | **89.86 ± 0.5** | **80.75 ± 0.1** | 90.92 ± 0.4     | 87.03 ± 0.3     |
| GCA      | 87.93 ± 0.4     | 78.08 ± 0.3     | 91.79 ± 0.5     | 88.47 ± 0.4     |
| CTDNE    | 84.56 ± 0.7     | 73.64 ± 0.5     | 89.60 ± 0.4     | 86.11 ± 0.2     |
| HTNE     | 85.94 ± 0.3     | 74.86 ± 0.3     | 90.52 ± 0.2     | 86.16 ± 0.1     |
| DyRep    | 86.07 ± 0.6     | 75.36 ± 0.4     | 91.10 ± 0.5     | 88.46 ± 0.3     |
| TGN      | 89.71 ± 0.3     | 80.66 ± 0.2     | **94.28 ± 0.6** | **90.05 ± 0.4** |
| MetaDyGNN| 87.43 ± 0.5     | 77.81 ± 0.4     | 92.37 ± 0.2     | 89.58 ± 0.3     |
| **DySubC**| **88.94 ± 0.5** | **79.51 ± 0.4** | **93.57 ± 0.3** | **89.74 ± 0.3** |

The best result is bolded.

Table 5. Results of the Ablation Study on Link Prediction Task

|          | Forum | Bitcoin Alpha | Wikipedia | Movielens | Math Overflow |
|----------|-------|---------------|-----------|-----------|---------------|
|          | AUC   | Accuracy      | AUC       | Accuracy  | AUC           |
|          |       |               |           |           |               |
| DySubC_−\text{N−R} | 84.42 ± 0.2 | 76.61 ± 0.1 | 89.67 ± 0.5 | 81.86 ± 0.4 | 88.74 ± 0.3 |
| DySubC_−\text{R}   | 85.72 ± 0.5 | 77.82 ± 0.3 | 91.49 ± 0.2 | 83.88 ± 0.1 | 91.24 ± 0.3 |
| DySubC_−\text{E}   | 88.21 ± 0.4 | 80.51 ± 0.3 | 92.03 ± 0.3 | 84.15 ± 0.1 | 92.18 ± 0.2 |
| **DySubC**        | **88.61 ± 0.3** | **80.82 ± 0.2** | **92.21 ± 0.4** | **84.57 ± 0.6** | **92.29 ± 0.3** |

Performances are measured in average AUC score and accuracy (mean in percentage ± standard deviation over 10 trial runs). The best result performance is bolded.

Table 6. Results of Different Structural Importance Score Strategies on Link Prediction Task

|          | Forum | Bitcoin Alpha | Wikipedia | Movielens | Math Overflow |
|----------|-------|---------------|-----------|-----------|---------------|
|          | AUC   | Accuracy      | AUC       | Accuracy  | AUC           |
|          |       |               |           |           |               |
| DySubC_−\text{D-E} | 88.61 ± 0.3 | 80.82 ± 0.2 | 92.21 ± 0.4 | 84.57 ± 0.6 | 92.29 ± 0.5 |
| DySubC_−\text{E-C} | 88.26 ± 0.4 | 80.32 ± 0.6 | 92.19 ± 0.4 | 84.41 ± 0.9 | 92.13 ± 0.3 |
| DySubC_−\text{E-P} | 88.52 ± 0.6 | 80.69 ± 0.5 | 91.91 ± 0.6 | 84.26 ± 0.7 | 92.04 ± 0.4 |

Performances are measured in average AUC score and accuracy (mean in percentage ± standard deviation over 10 trial runs). The best result performance is bolded.

The ablation results on time-enhanced modules are listed in Table 5. It verifies the effectiveness of each time-enhanced module, which consistently improves the model performance on all datasets. Especially when considering the negative sample $s_j^i$, the performance of the model is significantly improved (see the comparative results of DySubC_−\text{R} and DySubC_−\text{N−R}). The combination of three modules achieves the best performance. On the Math Overflow dataset, the final DySubC model gains an improvement of 0.05 AUC score compared to the base model with no time-enhanced module enabled. The ablation results on structural importance metrics are listed in Table 6. It shows that the performance of eigenvector centrality and PageRank as structural score metrics is not as good as that of degree. The possible reason is that the first two metrics prefer to select nodes that are important in the global structure, while the subgraph sampling strategy in this article tends to find the locally important nodes. Therefore, a local structural metric like degree can work well in our method and achieve the best results in downstream task.
4.7 Subgraph Size Analysis

This section studies the impact of the subgraph size in DynSubC on the five datasets. We adjust the subgraph size from 10 to 100 (including the central node), and the evaluation results are shown in Figure 4. Note that in the experiment on MovieLens and MathOverflow datasets, the maximum subgraph size is set as 50 due to limited computational memory. The corresponding result with size 100 in the figure is approximated by the result with size 50, since the model performance tends to be stable.

As shown in the figure, the model achieves better performance when the size of the subgraph is larger. It is probably because neighborhood nodes contain more structural and temporal information, which helps to obtain a higher quality representation. However, in the Forum dataset, the model with the subgraph size 100 performs worse than the model with the subgraph size 50. It may be because the dataset is small in size, a large size subgraph will contain nodes far away from the central node, which could be not beneficial to representation learning. When the size of subgraph is too small (e.g., 10), the performance of the model in all datasets is poor, which indicates that necessary information for learning is lost. When the subgraph size is 20, the model performs quite well on all five datasets and consumes less system memory than the model with a larger subgraph size.

4.8 Parameter Sensitivity Study

4.8.1 Temporal and Structural Contrast. The hyperparameter $\lambda$ balances the relationship between the structure and time negative samples, which plays a vital role in DynSubC. In this section, the sensitivity analysis is conducted on hyperparameters $\lambda$. Specifically, the value of $\lambda$ is increased from 0 to 2 with the step size 0.5. The parameters $\alpha$ and $\beta$ are set to the optimal values in each datasets. The results are shown in Figure 5. With the increase of the hyperparameter $\lambda$, the AUC score increases first and then decreases slowly. The model performs best when $\lambda$ is 0.5. It demonstrates that the structural properties of graphs are still dominant in dynamic graph representation learning. When the value of $\lambda$ is 0, the model performance is poor, which also indicates that temporal information is also essential for dynamic graph representation learning. At the same time, when $\lambda$ is 0.5 in all three datasets, the model has the best effect, which indicates that different datasets have little influence on the choice of $\lambda$.

4.8.2 Hyperparameters $\alpha$ and $\beta$. Hyperparameters $\alpha$ and $\beta$ determine the quality of temporal subgraph sampling and time-weighted subgraph representation. In this section, the value of $\alpha$ is increased from 2 to 20 with the step size 2 and the value of $\beta$ is increased from 0.4 to 3.6 with the step size 0.4. $\lambda$ is set to 0.5. The stability of the model under the perturbation of the two

Fig. 4. The impact of the subgraph size on DySubC.
The impact of hyperparameter $\lambda$ on DySubC.

Fig. 6. The performance of DySubC on the Forum and Bitcoin Alpha datasets with the change of two hyperparameters $\alpha$ and $\beta$.

hyperparameters is observed. The results on the Forum and Bitcoin Alpha datasets are shown in Figure 6. On the Forum dataset, the model performs best when $\alpha$ is 10 and $\beta$ is 1.6. On the Bitcoin Alpha dataset, the model performs best when $\alpha$ is 10 and $\beta$ is 1.2. Both of that are around the median of their respective value ranges on different datasets. It demonstrates that DynSubC is less sensitive to datasets for hyperparameter $\alpha$ and $\beta$. However, from the uneven plan view, it can be seen that DynSubC is sensitive to these two hyperparameters. As we mentioned before, $\alpha$ is used to balance the structural and temporal information in the subgraph sampling. The result shows that the temporal information is at least as important as structural information when sampling. $\beta$ is used to balance the influence of the latest interaction and its distance to the central node. Both are verified to be indispensable in subgraph representation.

4.9 Visualization Analysis

In this section, we use the tSNE algorithm [16] to visualize the node representation and the latest interactions obtained by DySubC and Sub-Con. Figure 7 shows the result on the Bitcoin Alpha dataset. The red lines denote the latest 10 interactions. The visualization result shows that DySubC can better embed the evolitional features of nodes into the representation, so that the latest interacted nodes are also closer in the embedding space compared to Sub-Con, which is a static subgraph contrast method. Moreover, it is observed that the nodes that are recently active are more clustered in the visualization of DySubC than that of Sub-Con. It is consistent with the characteristics of the dataset, that is, users who trust each other are often in a community structure, so they are supposed to be closer in the embedding space.
Fig. 7. Visualization comparison of node embedding and 10 latest interactions obtained by Sub-Con and DySubC on the Bitcoin Alpha dataset.

Fig. 8. Comparison of training time on link prediction on five datasets.

4.10 Complexity Analysis

Finally, we analyzed the complexity of our model and other baseline models on a Linux server with RTX 3090 and i9-10920x. The results of training time of all models are summarized in Figure 8. The training time of each model increases gradually with the increase of the graph size. Different models have different sensitivities to the number of nodes and the sparseness of graphs. Although our model has the longest training time on some datasets (e.g., Movielens and Math Overflow), it is acceptable considering its advantages in performance improvement. For each central node, we sample $k$ neighbors to construct subgraph and need $O(kF)$ and $O(k^2)$ space overheads to store feature matrix and adjacency matrix, respectively, for each subgraph, where $F$ is the dimension of node features. The total space overheads are $O(|V|(kF + k^2))$. Since the time complexity of GCN
is $O(|E|Fd)$ [9], where $d$ is the dimension of node embeddings. The time complexity of DySubC is $O(|V||\bar{E}|Fd)$, where $|\bar{E}|$ is the average number of edges in each subgraph. Note that $|\bar{E}|$, $F$, and $d$ are much smaller than $|V|$.

5 CONCLUSION

In this article, a novel self-supervised dynamic graph representation learning framework (DySubC) based on temporal subgraph contrast is proposed. The model learns the representation with both structural and temporal information by maximizing the mutual information of the node representation and its temporal subgraph representation. DySubC proposes three time-enhanced modules, which cannot only sample more effective subgraphs, but also learn better representation by temporal contrast loss. The effectiveness of DySubC compared with related graph contrast learning methods and dynamic graph representation learning methods is demonstrated by empirical evaluation on multiple datasets. The success of DySubC provides an insight for the future study on continuous-time graph representation learning.

REFERENCES

[1] Jie Chen, Tengfei Ma, and Cao Xiao. 2018. FastGCN: Fast learning with graph convolutional networks via importance sampling. In Proceedings of the 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada. April 30–May 3, 2018, Conference Track Proceedings. OpenReview.net. Retrieved from https://openreview.net/forum?id=rytstxWAW

[2] Palash Goyal, Nitin Kamra, Xinran He, and Yan Liu. 2018. DynGEM: Deep embedding method for dynamic graphs. arXiv:1805.11273. Retrieved from https://arxiv.org/abs/1805.11273

[3] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, August 13-17, 2016. Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi (Eds.). ACM, 855–864. DOI: https://doi.org/10.1145/2939672.2939754

[4] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA. Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 1024–1034. Retrieved from https://proceedings.neurips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html

[5] Kaveh Hassani and Amir Hosein Khas Ahmadi. 2020. Contrastive multi-view representation learning on graphs. In Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event (Proceedings of Machine Learning Research), Vol. 119. PMLR, 4116–4126. Retrieved from http://proceedings.mlr.press/v119/hassani20a.html

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In Proceedings of the 2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015. IEEE Computer Society, 1026–1034. DOI: https://doi.org/10.1109/ICCV.2015.123

[7] Yizhu Jiao, Yun Xiong, Jiawei Zhang, Yao Zhang, Tianqi Zhang, and Yangyong Zhu. 2020. Sub-graph contrast for scalable self-supervised graph representation learning. In 20th IEEE International Conference on Data Mining, ICDM 2020, Sorrento, Italy, November 17-20, 2020. Claudia Plant, Haixun Wang, Alfredo Cuzzocrea, Carlo Zaniolo, and Xindong Wu (Eds.). IEEE, 222–231. DOI: https://doi.org/10.1109/ICDM50108.2020.00031

[8] Longlong Jing and Yingli Tian. 2021. Self-supervised visual feature learning with deep neural networks: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence 43, 11 (2021), 4037–4058. DOI: https://doi.org/10.1109/TPAMI.2020.2992393

[9] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In Proceedings of the 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net. Retrieved from https://openreview.net/forum?id=SJU4ayYgl

[10] Srijan Kumar, Bryan Hooi, Disha Makhiija, Mohit Kumar, Christos Faloutsos, and V. S. Subrahmanian. 2018. REV2: Fraudulent user prediction in rating platforms. In Proceedings of the 11th ACM International Conference on Web Search and Data Mining, WSDM 2018, Marina Del Rey, CA, USA, February 5-9, 2018. Yi Chang, Chengxiang Zhai, Yan Liu, and Yoelle Maarek (Eds.). ACM, 333–341. DOI: https://doi.org/10.1145/3159652.3159729
[28] Ryan A. Rossi and Nesreen K. Ahmed. 2015. The network data repository with interactive graph analytics and visualization. In Proceedings of the 29th AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas. Blai Bonet and Sven Koenig (Eds.), AAAI Press, 4292–4293. Retrieved from http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9553

[29] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang. 2020. DySAT: Deep neural representation learning on dynamic graphs via self-attention networks. In WSDM ’20: The Thirteenth ACM International Conference on Web Search and Data Mining, Houston, TX, February 3-7, 2020. James Caverlee, Xia (Ben) Hu, Mounia Lalmas, and Wei Wang (Eds.). ACM, 519–527. DOI: https://doi.org/10.1145/3336191.3371845

[30] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale information network embedding. In Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015. Aldo Gangemi, Stefano Leonardi, and Alessandro Panconesi (Eds.). ACM, 1067–1077. DOI: https://doi.org/10.1145/2736277.2741093

[31] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. 2019. DyRep: Learning representations over dynamic graphs. In Proceedings of the 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, May 6-9, 2019. OpenReview.net. Retrieved from https://openreview.net/forum?id=HyePzhR5KX

[32] Ashish Vaswani, Noami Shazeer, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 5998–6008. Retrieved from https://papers.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[33] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In Proceedings of the 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30–May 3, 2018, Conference Track Proceedings. OpenReview.net. Retrieved from https://openreview.net/forum?id=rJXMpikCZ

[34] Petar Velickovic, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. 2019. Deep graph infomax. In Proceedings of the 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, May 6-9, 2019. OpenReview.net. Retrieved from https://openreview.net/forum?id=rzklk9lAcKQ

[35] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, August 13-17, 2016. Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi (Eds.). ACM, 1225–1234. DOI: https://doi.org/10.1145/2939672.2939753

[36] Felix Wu, Amauri H. Souza Jr., Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. 2019. Simplifying graph convolutional networks. In Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California (Proceedings of Machine Learning Research). Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.), Vol. 97. PMLR, 6861–6871. Retrieved from http://proceedings.mlr.press/v97/wu19e.html

[37] Cheng Yang, Chunchen Wang, Yuanfu Lu, Xumeng Gong, Chuan Shi, Wei Wang, and Xu Zhang. 2022. Few-shot link prediction in dynamic networks. In WSDM ’22: The Fifteenth ACM International Conference on Web Search and Data Mining, Virtual Event / Tempe, AZ, USA, February 21 - 25, 2022. K. Selcuk Candan, Huan Liu, Leman Akoglu, Xin Luna Dong, and Jiliang Tang (Eds.). ACM, 1245–1255. DOI: https://doi.org/10.1145/3485650.3498417

[38] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. arXiv:1409.2329. Retrieved from https://arxiv.org/abs/1409.2329

[39] Muhan Zhang and Yixin Chen. 2018. Link prediction based on graph neural networks. In Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada. Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (Eds.). 5171–5181. Retrieved from https://proceedings.neurips.cc/paper/2018/hash/53f0d7c537d99b3824f0f99d62ea2428-Abstract.html

[40] Le-kui Zhou, Yang Yang, Xiang Ren, Fei Wu, and Yueting Zhuang. 2018. Dynamic network embedding by modeling triadic closure process. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, February 2-7, 2018. Sheila A. McIlraith and Kilian Q. Weinberger (Eds.). AAAI Press, 571–578. Retrieved from https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16572

[41] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2020. Deep graph contrastive representation learning. arXiv:2006.04131. Retrieved from https://arxiv.org/abs/2006.04131

[42] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2021. Graph contrastive learning with adaptive augmentation. In WWW ’21: The Web Conference 2021, Virtual Event/Ljubljana, Slovenia, April 19-23, 2021.
Jure Leskovec, Marko Grobelnik, Marc Najork, Jie Tang, and Leila Zia (Eds.). ACM/IW3C2, 2069–2080. DOI: https://doi.org/10.1145/3442381.3449802

[43] Yuan Zuo, Guannan Liu, Hao Lin, Jia Guo, Xiaoqian Hu, and Junjie Wu. 2018. Embedding temporal network via neighborhood formation. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018. Yike Guo and Faisal Farooq (Eds.). ACM, 2857–2866. DOI: https://doi.org/10.1145/3219819.3220054

Received 3 July 2022; revised 1 December 2022; accepted 26 July 2023