Temporal Graph Representation Learning with Adaptive Augmentation Contrastive

Hongjiang Chen[1], Pengfei Jiao[1], Huijun Tang[1(✉)], and Huaming Wu[2]

[1] School of Cyberspace, Hangzhou Dianzi University, Hangzhou 310018, China
{hchen,pjiao,tanghuijune}@hdu.edu.cn
[2] Center for Applied Mathematics, Tianjin University, Tianjin 300072, China
whming@tju.edu.cn

Abstract. Temporal graph representation learning aims to generate low-dimensional dynamic node embeddings to capture temporal information as well as structural and property information. Current representation learning methods for temporal networks often focus on capturing fine-grained information, which may lead to the model capturing random noise instead of essential semantic information. While graph contrastive learning has shown promise in dealing with noise, it only applies to static graphs or snapshots and may not be suitable for handling time-dependent noise. To alleviate the above challenge, we propose a novel Temporal Graph representation learning with Adaptive augmentation Contrastive (TGAC) model. The adaptive augmentation on the temporal graph is made by combining prior knowledge with temporal information, and the contrastive objective function is constructed by defining the augmented inter-view contrast and intra-view contrast. To complement TGAC, we propose three adaptive augmentation strategies that modify topological features to reduce noise from the network. Our extensive experiments on various real networks demonstrate that the proposed model outperforms other temporal graph representation learning methods.

Keywords: Temporal graphs · Network embedding · Contrastive learning

1 Introduction

Temporal networks have become increasingly popular for modeling complex real-world scenarios, e.g., citation networks, recommendation systems, and engineering systems [3,7,9,16], where nodes represent interacting elements and temporal links denote their labeled interactions over time. These networks are inherently dynamic, with the topology and node properties evolving over time [32]. However, the real world is often affected by time-varying noise, which can have a significant impact on the network structure and its predictions. For instance, colleagues who work together on a project may interact frequently during the project's duration, but may rarely interact afterwards, leading to a decrease

D. Koutra et al. (Eds.): ECML PKDD 2023, LNAI 14170, pp. 683–699, 2023.
https://doi.org/10.1007/978-3-031-43415-0_40

in the amount of available information for future interactions. Therefore, it is imperative to investigate techniques for reducing the influence of time-varying noise on temporal graphs in order to improve the accuracy of predicting future interactions.

In recent years, there has been a surge in the development of temporal graph neural networks (TGNNs), which extend the capabilities of neural networks to structured inputs and have achieved state-of-the-art (SOTA) performance in various tasks, such as link prediction. However, one of the key challenges in temporal graph representation learning is the presence of time-varying noise, which can significantly affect the network's evolution. Existing methods [10,11,17,19,33] have primarily focused on capturing fine-grained information to obtain a more comprehensive node representation. This can lead to overfitting and the capture of random noise, which can obscure essential semantic information in the network as it evolves. Therefore, it is important to explore new approaches that balance the capture of both fine-grained and essential semantic information in order to improve the robustness and generalization ability of TGNNs.

Contrastive learning (CL) has emerged as a promising approach for addressing the aforementioned challenges in temporal graph representation learning by enabling the method to learn more generalized graph representations through the generation of multiple views for each instance using various data augmentations. This process helps reduce the impact of noise and improve method generalization and robustness [36]. However, current graph augmentation methods tend to focus primarily on capturing structural features at the node or graph level, while neglecting the temporal information of edge generation [34]. Incorporating temporal information related to edge generation into graph learning can help capture the dynamic evolution of the graph and improve the accuracy of node representations. Thus, there is a need to develop new approaches that effectively integrate temporal information into CL-based methods for temporal graph representation learning.

Consider the toy example of a temporal network shown in Fig. 1. When using the method of static graph augmentation (e.g., GCA [37]) to improve the temporal graph, the edge between nodes D and E may be inadvertently removed. As a result, TGNNs may not be able to accurately predict future interactions based on the enhanced graph because crucial temporal information has been lost. Specifically, the interaction between nodes D and E at the most recent time t_5 is crucial for accurately predicting future interactions, while the interaction between nodes B and C at time t_2 may be less important. Consequently, the static graph augmentation method fails to capture important temporal information that is essential for accurate predictions of future interactions in temporal graphs. To overcome this issue, incorporating temporal information into data augmentation and node representation can effectively capture the evolution of edge generation and improve the accuracy of future interaction predictions.

In this paper, we propose a novel contrastive model called Temporal Graph representation learning with Adaptive augmentation Contrastive (TGAC). Firstly, we utilize centrality measures to eliminate redundant topological

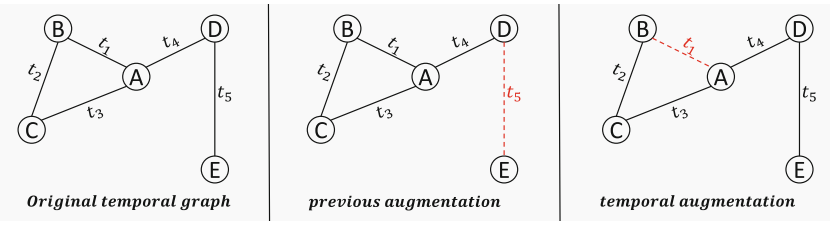


Fig. 1. The toy example illustrates the limitations of the static graph augmentation method when applied to a temporal graph. Specifically, the original temporal network (left) and the resulting loss of temporal information following the application of static graph augmentation (middle) are demonstrated. To address this issue, we propose a novel approach for augmenting temporal graphs by incorporating both topological and temporal information. This approach allows us to eliminate redundant information while preserving vital temporal information (right).

information from the input temporal graph by taking into account both structural and temporal influence. This process enhances the effectiveness of temporal graph augmentation. Subsequently, the pruned graph is subjected to perturbations to generate two distinct temporal views for augmentation. Finally, the model is trained using a contrastive loss function to maximize the agreement between node embeddings in the two views.

Specifically, the main contributions are summarized as follows.

- We present a novel approach for temporal graph contrast learning that incorporates temporal information during edge generation. This enables the model to better capture the structural evolution characteristics of graphs, resulting in improved representation learning.
- We propose a temporal graph augmentation method that leverages both the structural and temporal information of neighborhoods. By doing so, we are able to augment the original graph while preserving important temporal features.
- To further enhance important topology structures and improve node representations, we propose a graph pruning scheme that employs edge centrality measures to remove noisy or redundant connections prior to attention allocation.
- Experimental results demonstrate the superior performance of our proposed TGAC in tasks such as link prediction and node classification, when compared to other state-of-the-art temporal graph representation learning models.

2 Related Work

In this section, we will provide a concise overview of the existing literature on temporal graph representation learning. We will then delve into the topic of contrastive representation learning methods. Finally, we will compare and contrast our proposed method with related works in the field to better understand its unique contributions.

2.1 Temporal Graph Representation Learning

Graph representation learning methodologies are designed to generate embeddings that capture both structural and attribute information at either the node or graph level [4,24,26,31]. For temporal graphs, traditional representations can be expanded to incorporate time-dependency, where the model of temporal dependence is formulated either as snapshot-based or event-based methods [20]. These techniques aim to learn temporal node or graph embeddings that capture the evolution of the graph over time. While snapshot-based paradigms may have merit, our paper focuses primarily on event-based models, which have exhibited superior performance in empirical studies compared to models based on snapshot temporal graphs [27].

Temporal graphs exhibit the time-varying behavior of nodes, which provides distinct insights not present in static graphs. By incorporating historical interaction information, we can distinguish between nodes that have similar local neighborhoods but different structural roles. For instance, JODIE [14] learns the embeddings of evolving trajectories by leveraging past interactions. TGN [25] keeps track of a memory state for each node and updates it with new interactions. CAWs [32] capture the dynamic evolution of networks by using temporally anonymous random walks to extract temporal network motifs. Unfortunately, all of the aforementioned techniques do not take into account the impact of noise in the network, which can be detrimental to the ability to capture valuable temporal information.

2.2 Contrastive Representation Learning

Inspired by recent advancements of CL in computer vision [12] and natural language processing [18] domains, some research has been conducted to apply CL to graph data. For instance, DGI [30] combines Graph Neural Networks with infomax and concentrates on contrasting views at the node level by generating multiple augmented graphs through handcrafted augmentations. GRACE [36] generates two views by randomly masking node attributes and removing edges, while GCA [37] employs a similar framework to GRACE but emphasizes designing the adaptive augmentation strategy.

Although some studies have explored the potential of contrastive learning for temporal graphs, most of them focus on static graphs and snapshot-based temporal graphs [5,22,23]. In contrast, our proposed approach addresses the challenge of noise in temporal graphs by considering the importance of edges with respect to both temporal and topological features, and adaptively augmenting the graphs in an efficient manner. Our approach effectively enhances both the temporal and topological features of the graphs, distinguishing it from existing methods for temporal graph learning and graph contrastive learning.

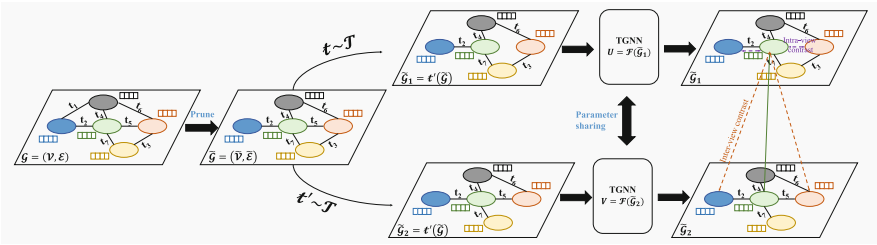


Fig. 2. Our proposed Temporal Graph representation learning with Adaptive augmentation Contrastive (TGAC) model. The input graph $\mathcal{G}$ is first pruned to be $\widetilde{\mathcal{G}}$, then use two augmentation t and t' are generate two temporal graphs $\widetilde{\mathcal{G}}_1$ and $\widetilde{\mathcal{G}}_2$. A shared TGNN $\mathcal{F}$ is employed to obtain two views' node representation. Finally, the model was trained by contrasting positive-negative pairs in both intra-view (in purple) and inter-view (in orange). (Color figure online)

3 The Proposed Method

In this section, we will introduce the notations and definitions used in this paper. Then, we will present the problem formulation and introduce the overall framework of TGAC. Finally, we will provide a detailed description of each component module (Fig. 2).

3.1 Preliminaries

First, we define the temporal graph based on the timestamps accompanying the node interactions.

Definition 1 (Temporal Graph). *A temporal graph is represented as* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$*, where* $\mathcal{V}$ *is the set of nodes and* $\mathcal{E}$ *is the set of sequences of node interactions with timestamps labels. For any edge* $(u, v, t) \in \mathcal{E}$*, there exists a set of timestamps* $\mathcal{E}_{u,v} = (u, v, t_1), (u, v, t_2), \cdots, (u, v, t_n)$*, indicating that nodes* u *and* v *have interacted at least once at each of the corresponding timestamps. Two interacting nodes are referred to as neighbors. It is important to note that in temporal graphs, the concept of interaction replaces the concept of edges, and multiple interactions can occur between two nodes.*

A good representation learning method for temporal networks should be able to accurately and efficiently predict how these networks will evolve over time. In this context, the problem can be formulated as follows.

Definition 2 (Problem formulation). *For any temporal graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$*, the task is to learn the mapping function* $f : \mathcal{V} \rightarrow \mathbb{R}^d$ *to embed the node in a* d*-dimensional vector space, where* $d \ll |\mathcal{V}|$*. The node representation is supposed to contain both structural and temporal information and is suitable for downstream machine-learning tasks such as link prediction, and node classification.*

3.2 Overview

The proposed model utilizes graph contrastive learning to capture the structural and temporal features from temporal graphs during the training phase. The model prunes the input temporal graph, generates contrasting views, and uses a loss function that includes both link prediction and contrastive loss to learn effective node representations.

3.3 Temporal Graph Pruning

To ensure effective node representation learning for downstream tasks, it is necessary to remove noisy edges from the original time graph topology. TGAC achieves this by computing the importance of each edge, which takes into account both the node centrality and temporal information. As a result, the pruned time graph provides richer information for TGNN to learn node representations more effectively. The centrality of each edge is assessed based on a combination of node properties, graph topology, and temporal characteristics. By removing noisy links based on their centrality attributes, the pruned temporal graph facilitates improved information use for node representation learning through TGNN.

Node centrality is a common method for measuring the importance of nodes in large-scale complex networks. Various techniques have been proposed to measure node centrality, some of which are outlined below:

- Degree centrality (DE) is considered one of the elementary measures of centrality, which quantifies the number of edges incident to a particular node in a network. It is a widely used and effective approach for evaluating the significance of a node in a network. Specifically, in social networks such as Twitter, nodes represent people, while edges represent the following connections among them. Nodes with a high degree of centrality tend to correspond to more important people.
- Eigenvector centrality (EV) is another important centrality measure that considers not only the number of connections of a particular node but also the centrality of its neighboring nodes. The idea is that if a node is connected to other nodes with high centrality, its own centrality is subsequently augmented. Consequently, a node's eigenvector centrality may not necessarily be high even if it has a substantial degree, in cases where all its connections have low centrality. Subsequent paragraphs, however, are indented.
- PageRank centrality (PR) is a measure of centrality determined by utilizing the PageRank algorithm. This algorithm involves developing a random walk model on a directed graph and calculating the likelihood of visiting each node under specific conditions. The resulting stable probability value of each node is its PageRank value, which serves as an indicator of the node's importance or centrality within the network.

These three methods of calculating node centrality have distinct advantages and limitations. DE is a straightforward and efficient method, making it suitable

for datasets that are not very sensitive to node characteristics. EV takes into account both node characteristics and topology and performs well across a wide range of datasets. PR is especially effective for analyzing complex topological networks. Consequently, we can use the notation $\varphi(\cdot)$ to indicate the specific node centrality method used for a given dataset.

Additionally, since temporal graphs contain temporal information that static graphs lack, we need to consider the temporal dimension when measuring the impact of each edge. To achieve this, we define the centrality of each edge as ϕ_{uv}^t, which is determined by the centralities of the two nodes it connects and the time of its occurrence. In undirected graphs, $\phi^t uv$ is computed as the product of the average centralities of its two nodes and the time at which the edge is formed. This can be expressed mathematically as follows:

$$\phi_{uv}^t = (\varphi(u) + \varphi(v))/2 + \alpha t_{uv}. \tag{1}$$

This definition enables us to capture the evolving nature of the temporal graph and obtain more precise node representations that can be used for downstream tasks. In the case of a directed graph, we define the centrality of an edge as the product of the centrality of the node it is pointing to and the time at which the connection is established. This reflects the impact of the edge in directing the flow of information or influences toward the target node, while also taking into account the time factor. Hence, the edge centrality for a directed graph is defined as:

$$\phi_{uv}^t = \varphi(v) + \alpha t_{uv}. \tag{2}$$

After obtaining the centrality score for each edge, we sort all the edges in descending order based on their centrality scores and then select the top k edges to retain while pruning the rest. The value of k is determined by the formula $k = E \times (1 - c)$, where E represents the total number of edges in the temporal graph, and c is the pruning ratio. The temporal graph after pruning is illustrated below:

$$\widetilde{\mathcal{E}} = \left\{ u_i, v_i, t_i | \phi_{u_i v_i}^{t_i} \in \mathrm{TopK}(\phi(\mathcal{E}), k) \right\}. \tag{3}$$

This method helps to remove redundant and noisy edges from the temporal graph and obtain a pruned temporal graph that can be used for subsequent training, which facilitates the acquisition of improved representation results.

3.4 Temporal Graph Encoder

The temporal graph encoder is based on TGN [25] and consists of interchangeable and independent modules. Each node in the model has a memory vector that represents its past interactions in a compressed form. When a new event occurs, the mailbox module calculates the message for each related node, which is then used to update the node's memory vector. To address the issue of stale information, the embedding module calculates node embeddings at each time step by using their neighborhood and memory state. In other words, the encoder updates the memory state of each node with new interactions and employs a

node memory update mechanism. In the node memory storage module, at time t, the model stores the memory of each node u it has encountered so far in a vector denoted by $s_u(t)$. Whenever a new interaction occurs with a node, its compressed historical information is used to update its memory state. During the message passing and updating phase, the model calculates the memory vectors for the source and target nodes u and v affected by each event. This is done using the msg method, which computes the message sent from the source node to the target node. The message is then used to update the memory vectors of both nodes. We formulate message passing function as

$$m_u(t) = msg\left(s_u(t^-), s_v(t^-), t\right), \tag{4}$$

$$\bar{m}_u(t) = agg\left(m_u(t_1), \ldots, m_u(t)\right). \tag{5}$$

To clarify, the message passing and aggregator part involves calculating the message using the msg method for the nodes u and v affected by each event, where $s_u(t^-)$ represents the information at node u before time t. The message is then aggregated with the information obtained before the node, and the resulting information is then updated to yield the $s_u(t)$ value for node u. This process involves the utilization of a learnable information method, such as MLP, followed by information aggregation techniques, such as RNNs or attention mechanisms, and concluded with information update operations. In scenarios where nodes u and v are affected by an interaction event, their information is updated using a memory cell such as GRU [2] or LSTM [8]. The process can be mathematically formulated as follows:

$$s_u(t) = mem\left(\bar{m}_u(t), s_u(t^-)\right). \tag{6}$$

Finally, after obtaining $s_u(t)$, the node representation is obtained by concatenating it with the current input features of node u at time t, followed by a non-linear transformation to obtain the final embedding $h_u(t)$. Specifically, the concatenation operation is defined as follows:

$$z_u(t) = emb(u,t) = \sum_{v \in \mathcal{N}_u^k([0,t])} h\left(s_u(t), s_v(t)\right), \tag{7}$$

where h is a learnable function. The resulting $z_u(t)$ can be used for downstream tasks such as node classification or link prediction.

3.5 Temporal Contrastive Learning

Contrastive learning aims to learn node or graph representations by bringing positive samples closer and pushing negative samples farther apart. We use a general contrastive learning framework to maximize representation consistency across different views. Two views of the pruned graph are generated using random augmentation operations. Existing methods struggle with topological random disturbances, as selecting positive and negative samples is crucial. After

a disturbance, ineffective neighborhood information can make optimizing contrastive targets difficult. We must perturb the graph to preserve its internal mode as much as possible. Our method removes edges randomly with a probability but assigns a weight to each edge to decrease the probability of removing important edges and increase that of removing redundant ones.

To achieve this, we introduce a removal probability for each edge and improve the perturbation process of the temporal graph by considering edge importance. Similar to temporal graph pruning, we compute edge importance based on topology and time information and use it to calculate the removal probability for each edge. However, since the importance values may be relatively large, we first normalize them by setting them to $w_{uv}^t = \lg \phi_{uv}^t$. After normalization, we obtain the removal probability for each edge as follows:

$$p_{uv}^t = \min\left(\frac{w_{max}^t - w_{uv}^t}{w_{max}^t - \mu_w^t} \cdot p_e, p_r\right), \tag{8}$$

where p_e is a hyperparameter that controls the overall probability of edge removal, w_{max}^t and μ_w^t are the maximum and average values of w_{uv}^t, respectively. We set a cut-off probability $p_r < 1$ to prevent the removal probability from becoming too high and corrupting the graph topology. The resulting temporal graph is pruned and looks like this:

$$P\left\{(u, v, t) \in \widetilde{\mathcal{E}}\right\} = 1 - p_{uv}^t. \tag{9}$$

To enhance the quality of node representations, we propose topological perturbations that generate distinct views during each iteration of training, denoted as $\widetilde{\mathcal{E}}_1$ and $\widetilde{\mathcal{E}}_2$. The probabilities of generating these two views are represented by p_e^1 and p_e^2, respectively. To prevent excessive perturbation that may lead to the degradation of the graph topology, we set p_e to 0.7, ensuring that p_r does not surpass 0.7.

3.6 Loss Function

Task Loss: To learn the parameters of TGNN for each view node, we utilize a link prediction binary cross-entropy loss function, define as follows:

$$\mathcal{L}(u, v, t) = -\log \sigma(-{z_u^t}^{\mathrm{T}} z_v^t) - Q\mathbb{E}_{v' \sim P(v)} \log \sigma({z_u^t}^{\mathrm{T}} z_{v'}^t). \tag{10}$$

The loss function aims to maximize the likelihood of the observed edges while minimizing the likelihood of negative edges. Since two views both have this task, the loss for the two views is defined similarly. The overall objective to be maximized is defined as the average over two views, formally given by:

$$\mathcal{L}_{task} = \sum_{(u_1, v_1, t_1) \in \widetilde{\mathcal{E}}_1} \mathcal{L}(u_1, v_1, t_1) + \sum_{(u_2, v_2, t_2) \in \widetilde{\mathcal{E}}_2} \mathcal{L}(u_2, v_2, t_2). \tag{11}$$

Contrastive Loss: We use a comparison objective for the two generated views to differentiate nodes with the same identifier in different views from other embeddings. For any node v_i in one view, its corresponding node u_i in the other view is considered as an anchor, and v_i and u_i form positive sample pairs. All other nodes from both views form negative samples, guiding the model to maximize the consistency of node representations across the two views. The representations of each node in the two views should be similar and distinct from those of other nodes.

Furthermore, we use a two-layer MLP to transform node representations into a feature space for comparison. A similarity function $\theta(u, v) = s(g(u), g(v))$ is used to measure different node representations, where s can be either cosine or Euclidean distance and $g(\cdot)$ denotes the non-linear projection of the MLP. To achieve contrastive learning in multi-view, we use a loss function similar to InfoNCE. For each positive sample pair u_i and v_i, the objective function is defined as follows:

$$\mathcal{L}_{cl} = \sum_{u_i, v_i \in \mathcal{V}} \log \frac{P_i}{P_i + N_i^{inter} + N_i^{intra}}, \tag{12}$$

where $P_i = e^{\theta(u_i, v_i)/\tau}$ is positive pair, N_i^{inter} and N_i^{intra} are inter-view and intra-view negative pairs, respectively, which are given by the following:

$$N_i^{inter} = \sum_{k \neq i} e^{\theta(u_i, v_k)/\tau}, \tag{13}$$

$$N_i^{intra} = \sum_{k \neq i} e^{\theta(u_i, u_k)/\tau}, \tag{14}$$

where τ denotes the temperature coefficient.

Total Loss: The total loss function is a combination of the task loss $\mathcal{L}_{task}$ and contrastive loss $\mathcal{L}_{cl}$. The definition of the total loss function $\mathcal{L}$ is established formally by utilizing Eqs. 11 and 12. Specifically, the total loss function $\mathcal{L}$ is expressed as follows:

$$\mathcal{L} = \lambda \mathcal{L}_{task} + \mathcal{L}_{cl}, \tag{15}$$

where λ is a hyperparameter that balances the weights of the two loss functions. The task loss function $\mathcal{L}_{task}$ evaluates the predictive capability of the model in identifying observed edges in the temporal graph, whereas the contrastive loss function $\mathcal{L}_{cl}$ encourages the consistency of representations of the same node across the two augmented views.

4 Experiments

In this section, we evaluate the performance of TGAC against a variety of baselines on different datasets. We further conduct an ablation study on relevant modules and hyperparameter analysis.

4.1 Experimental Setup

Datasets. We evaluate the performance of TGAC on the tasks of temporal link prediction and dynamic node classification using four public temporal graph datasets, namely, Wikipedia [14], Reddit [1], MOOC [14], and CollegeMsg [15]. A detailed description of the statistical characteristics of these datasets is presented in Table 1.

Table 1. Statistics of the datasets.

Datasets	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	Feature	Label
Wikipedia	9,227	157,474	172	2
Reddit	10,984	672,447	172	2
Mooc	7,144	411,749	0	2
CollegeMsg	1,899	59,835	0	0

Baselines. To evaluate the performance of TGAC, we compare ten state-of-the-art graph embedding methods on both static and temporal graphs. For static graph embedding methods, including GAE, VGAE [13], GraphSAGE [6] and GAT [29]. For temporal graph embedding methods, including CTDNE [21], JODIE [14], DyRep [28], TGAT [35], TGN [25] and CAWs [32].

Parameter Settings. In the parameter settings, we select the optimizer with the Adam algorithm, the learning rate is 0.0001, and the dropout probability is 0.1. The dimension of both node embedding and time embedding is set to 100, memory dimension is set to 172. The temporal information weight α and contrastive loss weights λ are set at 10 and 0.1. For the baseline methods, we keep their default parameter settings.

4.2 Temporal Link Prediction

For temporal link prediction, we follow the evaluation protocols of TGN [35]. The goal of this task is to predict whether a temporal link will exist between given two nodes at a certain future point in time. We consider two different downstream tasks for evaluation: transductive and inductive link prediction. In the transductive link prediction task, we aim to predict the presence or absence of a link between two nodes that were observed during the training phase. In the inductive link prediction task, we aim to predict the presence or absence of a link between two new nodes that were not observed during the training phase. We divide the ratios of training, validation, and testing are 70%, 15%, and 15%, respectively.

Table 2. ROC AUC(%) and Average Precision(%) for the transductive temporal link prediction on Wikipedia, Reddit, Mooc and CollegeMsg. The means and standard deviations are computed for ten runs.

Task	Methods	Wikipedia		Reddit		Mooc		CollegeMsg	
		AUC	AP	AUC	AP	AUC	AP	AUC	AP
Transductive	GAE	91.47 ± 0.3	91.12 ± 0.1	95.87 ± 1.2	96.57 ± 1.0	87.89 ± 0.6	90.70 ± 0.3	73.15 ± 1.5	70.00 ± 1.17
	VGAE	82.43 ± 1.6	82.50 ± 4.0	92.70 ± 0.4	91.53 ± 0.7	88.21 ± 0.6	**91.00** ± 0.3	74.07 ± 0.9	70.66 ± 1.0
	GraphSAGE	92.00 ± 0.3	92.34 ± 0.3	97.75 ± 0.1	97.85 ± 0.1	56.17 ± 0.3	60.63 ± 0.2	62.38 ± 1.3	62.48 ± 0.9
	GAT	92.76 ± 0.5	93.17 ± 0.5	97.90 ± 0.1	97.07 ± 0.1	67.24 ± 0.1	66.66 ± 0.8	78.09 ± 0.5	75.97 ± 0.7
	CTDNE	82.36 ± 0.7	80.86 ± 0.7	85.32 ± 2.0	87.31 ± 1.4	88.37 ± 2.6	89.27 ± 2.0	81.88 ± 0.7	80.25 ± 0.8
	JODIE	94.94 ± 0.3	94.65 ± 0.6	97.62 ± 0.2	97.07 ± 0.4	79.75 ± 2.8	74.85 ± 3.1	59.85 ± 6.0	54.50 ± 4.4
	DyRep	94.22 ± 0.2	94.63 ± 0.2	98.01 ± 0.1	98.05 ± 0.1	80.57 ± 2.1	77.30 ± 2.2	54.75 ± 6.8	51.89 ± 4.8
	TGAT	94.99 ± 0.3	95.29 ± 0.2	98.07 ± 0.1	98.17 ± 0.1	66.02 ± 1.0	63.82 ± 0.9	81.05 ± 0.6	79.16 ± 0.6
	TGN	98.42 ± 0.1	98.50 ± 0.1	98.69 ± 0.1	98.73 ± 0.1	**89.07** ± 1.6	86.96 ± 2.1	85.06 ± 5.9	85.38 ± 6.4
	CAWs	98.39 ± 0.1	98.62 ± 0.1	98.05 ± 0.1	98.66 ± 0.1	69.48 ± 5.3	70.11 ± 6.2	90.02 ± 0.2	92.55 ± 0.1
	TGAC-DE	98.85 ± 0.0	98.89 ± 0.0	98.70 ± 0.0	98.73 ± 0.0	85.39 ± 1.0	82.20 ± 1.0	91.39 ± 0.6	92.91 ± 0.5
	TGAC-EV	**98.86** ± 0.0	**98.91** ± 0.0	98.71 ± 0.1	98.74 ± 0.0	88.54 ± 0.8	86.02 ± 0.8	**91.55** ± 0.7	**93.03** ± 0.5
	TGAC-PR	98.85 ± 0.0	98.90 ± 0.0	**98.76** ± 0.1	**98.76** ± 0.1	88.14 ± 1.4	85.47 ± 1.3	91.49 ± 0.7	92.98 ± 0.5
Inductive	GraphSAGE	88.60 ± 0.3	88.94 ± 0.5	94.28 ± 0.4	94.51 ± 0.1	53.68 ± 0.4	55.35 ± 0.4	49.64 ± 1.5	51.83 ± 0.8
	GAT	89.11 ± 0.5	89.82 ± 0.4	94.30 ± 0.4	94.58 ± 0.3	53.43 ± 2.1	54.80 ± 0.9	68.98 ± 1.2	66.22 ± 1.2
	JODIE	92.75 ± 0.3	93.11 ± 0.4	95.42 ± 0.2	94.50 ± 0.6	81.43 ± 0.8	76.82 ± 1.4	51.59 ± 3.2	50.02 ± 2.2
	DyRep	91.03 ± 0.3	91.96 ± 0.2	95.79 ± 0.5	95.75 ± 0.5	82.06 ± 1.7	79.17 ± 1.6	49.05 ± 4.1	49.30 ± 2.6
	TGAT	93.37 ± 0.3	93.86 ± 0.3	96.46 ± 0.1	96.61 ± 0.2	69.09 ± 0.8	67.65 ± 0.7	72.27 ± 0.5	72.53 ± 0.6
	TGN	97.72 ± 0.1	97.83 ± 0.1	97.54 ± 0.1	97.63 ± 0.1	**89.03** ± 1.6	**86.70** ± 2.0	78.54 ± 3.9	80.77 ± 3.7
	CAWs	98.16 ± 0.2	**98.52** ± 0.1	97.56 ± 0.1	97.06 ± 0.1	74.79 ± 2.3	76.02 ± 2.2	**89.11** ± 1.5	**91.79** ± 1.4
	TGAC-DE	**98.29** ± 0.0	98.35 ± 0.1	98.95 ± 0.0	98.98 ± 0.0	84.00 ± 1.3	80.02 ± 1.5	88.42 ± 0.5	90.70 ± 0.4
	TGAC-EV	98.28 ± 0.1	98.35 ± 0.1	98.94 ± 0.1	98.97 ± 0.1	88.23 ± 0.6	85.30 ± 0.7	88.49 ± 0.5	90.75 ± 0.4
	TGAC-PR	98.28 ± 0.1	98.34 ± 0.0	**98.96** ± 0.1	**98.98** ± 0.1	88.16 ± 1.5	85.16 ± 1.7	88.49 ± 0.5	90.73 ± 0.4

The results of our method and the baseline method on the temporal link prediction task are compared in Table 2. We leverage the Area Under the ROC Curve (AUC) and Average Precision (AP) as performance metrics. On both transductive and inductive tasks, we make the following observations.

- Baseline temporal graph embedding methods outperform static graph embedding methods such as GAE, VGAE, GraphSAGE, and GAT in link prediction tasks on four real-world datasets that include temporal information.
- For the temporal graph embedding methods, compare with the methods which combine time embedding, node features, and graph topology (i.e., CTDNE, TGAT) are worse than the use of a special module to update node embeddings based on temporal interactions (i.e., TGN, TGAC).
- Our method outperforms several existing methods on multiple datasets, although it is not as effective as CAWs on some of them. However, CAWs uses online time random walk sampling to obtain time node representations, which cannot be parallelized on the GPU and therefore require significant processing time. By incorporating prior knowledge into our time map and utilizing message passing, our method improves efficiency compared to TGN and achieves faster processing speeds than CAWs.

4.3 Dynamic Node Classification

For dynamic node classification, we also follow the evaluation protocols of TGN. The goal of this task is to predict the state label of the source node while giving the node link and future timestamps. Specifically, we use the model obtained from the previous transductive link prediction as the pre-training model for node classification. The node classification task trains a classifier decoder separately, such as a three-layer MLP. We evaluate the task on three datasets with dynamic node labels (i.e., Wikipedia, Reddit, and Mooc), excluding the CollegeMsg dataset because there are no node labels.

The results of our method and the baseline method on the Dynamic Node Classification task are compared in Table 3. We leverage the Area Under the ROC Curve (AUC) as performance metrics. Our results demonstrate superior performance on all three datasets, underscoring the effectiveness of our model's use of contrastive learning. By bringing the distance between nodes in one view closer while pushing away nodes in the other view, our model learns more optimized node representations for downstream classification tasks. This approach has proven to be more effective than alternative methods, as evidenced by the superior performance of our model.

Table 3. ROC AUC(%) for the transductive dynamic node classification on Wikipedia, Reddit and Mooc. The means and standard deviations are computed for ten runs. We use bold and underline to highlight the best and second best performers.

	Wikipedia	Reddit	Mooc
CTDNE	84.86 ± 1.5	54.38 ± 7.5	71.84 ± 1.0
JODIE	84.40 ± 0.9	61.51 ± 1.2	70.03 ± 0.5
DyRep	83.25 ± 0.5	60.86 ± 1.7	64.64 ± 1.4
TGAT	84.41 ± 1.5	65.98 ± 1.6	65.79 ± 0.5
TGN	87.56 ± 0.7	65.51 ± 0.8	63.93 ± 0.3
CAWs	84.88 ± 1.3	66.52 ± 2.2	68.77 ± 0.4
TGAC-DE	87.69 ± 0.2	68.54 ± 0.4	$\underline{70.13 \pm 0.2}$
TGAC-EV	$\mathbf{90.13} \pm 0.2$	$\mathbf{71.70} \pm 0.4$	61.83 ± 0.7
TGAC-PR	$\underline{88.85 \pm 0.2}$	$\underline{71.06} \pm 0.8$	$\mathbf{71.10} \pm 0.3$

4.4 Ablation Experiment

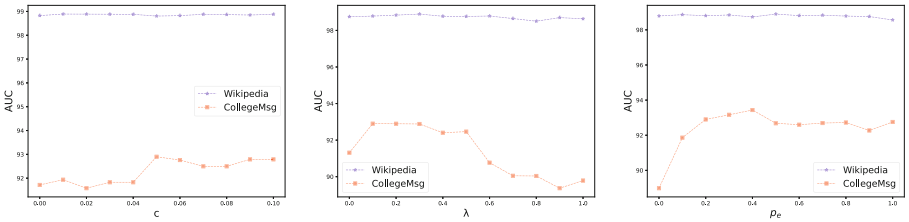
We conducted a series of experiments on the CollegeMsg dataset to evaluate the effectiveness of pruning on temporal graphs, using different centrality measures. Our findings, presented in Table 4, indicate a notable enhancement in the model's performance upon the removal of extraneous links through the application of diverse node centrality principles. Herein, "T" refers to the TGNN function,

Table 4. Ablation study result on CollegeMsg for Pruning schemes

	T	T+DE	T+EV	T+PR	T+DE+P	T+EV+P	T+PR+P
AUC	85.06	90.38	90.57	90.58	**92.39**	**92.55**	**92.49**

Table 5. ROC AUC(%) for both the transductive and inductive temporal link prediction on Wikipedia, Reddit, and CollegeMsg.

	Wikipedia		Reddit		CollegeMsg	
	Transductive	Inductive	Transductive	Inductive	Transductive	Inductive
TGAC w/o CL	98.29	98.37	98.54	98.57	85.06	87.41
TGAC w/o Prune	98.32	98.40	98.62	98.65	90.57	87.93
TGAC	**98.53**	**98.64**	**98.82**	**98.86**	**92.71**	**88.79**



(a) Pruning ratio c (b) Balance weight λ (c) Drop ratio p_e

Fig. 3. Parameter Sensitivity.

while "P" denotes the Prune function. Furthermore, we conducted an ablation study to assess the impact of contrastive learning, and the results are depicted in Table 5. Upon removing both the pruning and contrastive learning aspects, the model became a conventional TGN model. Our findings demonstrate that the absence of pruning and contrastive learning resulted in a significant decline in the performance of the TGN model.

4.5 Parameter Sensitivity

Our proposed method requires a thorough analysis of hyperparameters' impact on temporal link prediction performance on the datasets. These hyperparameters are the temporal graph pruning ratio c, the balance parameter λ, and the temporal graph enhancement factor p_e. We use a range of evaluation metrics to gauge the efficacy of various parameter values. We evaluate them on Wikipedia and CollegeMsg datasets using link prediction as the downstream task. We investigate the impact of the temporal graph pruning ratio on the model's ability to learn effective information. Additionally, we explore the balance between link prediction and contrastive learning. Figure 3 illustrates the sensitivity of our model's performance to various hyperparameters, including c,

λ, and p_e. Our experiments show that the proposed method achieves the best results when $c = 0.05$, $\lambda = 0.1$, and $p_e = 0.4$.

5 Conclusion

This paper introduces a novel temporal graph contrastive learning model named TGAC. The proposed model employs a pruning and adaptive augmentation technique that incorporates topological and temporal information with prior knowledge. This approach leads to the generation of enhanced temporal graph information, which in turn improves the performance of TGNN. The experimental results demonstrate that the TGAC model outperforms state-of-the-art methods on most of the datasets.

Acknowledgments. This work was supported by the Fundamental Research Funds for the Provincial Universities of Zhejiang Grant GK229909299001-008 and GK239909299001-028, Zhejiang Laboratory Open Research Project under Grant K2022QA0AB01, National Natural Science Foundation of China under Grant 62071327.

Ethical Statement. 1. The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

2. To the best of our knowledge, this work does not have potential negative social impacts.

3. All authors have already known that they intend to submit to the ecml-pkdd conference, and there is no multiple submission of one manuscript.

4. There is no conflict of interest in this study. Any questions or problems, please feel free to contact us.

References

1. Baumgartner, J., Zannettou, S., Keegan, B., Squire, M., Blackburn, J.: The pushshift reddit dataset. In: Proceedings of the International AAAI Conference on Web and Social Media, vol. 14, pp. 830–839 (2020)
2. Cho, K., et al.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
3. Gorochowski, T.E., Grierson, C.S., Di Bernardo, M.: Organization of feed-forward loop motifs reveals architectural principles in natural and engineered networks. Sci. Adv. **4**(3), eaap9751 (2018)
4. Grover, A., Leskovec, J.: node2vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 855–864 (2016)
5. Gutmann, M.U., Hyvärinen, A.: Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. J. Mach. Learn. Res. **13**(2) (2012)
6. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems, vol. 30 (2017)

7. Hamilton, W.L., Ying, R., Leskovec, J.: Representation learning on graphs: methods and applications. arXiv preprint arXiv:1709.05584 (2017)
8. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
9. Holme, P., Saramäki, J.: Temporal networks. Phys. Rep. **519**(3), 97–125 (2012)
10. Huang, C., Wang, L., Cao, X., Ma, W., Vosoughi, S.: Learning dynamic graph embeddings using random walk with temporal backtracking. In: NeurIPS 2022 Temporal Graph Learning Workshop (2022)
11. Jin, M., Li, Y.F., Pan, S.: Neural temporal walks: motif-aware representation learning on continuous-time dynamic graphs. In: Advances in Neural Information Processing Systems (2022)
12. Jing, L., Tian, Y.: Self-supervised visual feature learning with deep neural networks: a survey. IEEE Trans. Pattern Anal. Mach. Intell. **43**(11), 4037–4058 (2020)
13. Kipf, T.N., Welling, M.: Variational graph auto-encoders. arXiv preprint arXiv:1611.07308 (2016)
14. Kumar, S., Zhang, X., Leskovec, J.: Predicting dynamic embedding trajectory in temporal interaction networks. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1269–1278 (2019)
15. Leskovec, J., Krevl, A.: Snap datasets: Stanford large network dataset collection (2014)
16. Liben-Nowell, D., Kleinberg, J.: The link prediction problem for social networks. In: Proceedings of the Twelfth International Conference on Information and Knowledge Management, pp. 556–559 (2003)
17. Liu, M., Liu, Y.: Inductive representation learning in temporal networks via mining neighborhood and community influences. In: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 2202–2206 (2021)
18. Liu, X., et al.: Self-supervised learning: generative or contrastive. IEEE Trans. Knowl. Data Eng. **35**(1), 857–876 (2021)
19. Liu, Y., Ma, J., Li, P.: Neural predicting higher-order patterns in temporal networks. In: Proceedings of the ACM Web Conference 2022, pp. 1340–1351 (2022)
20. Longa, A., et al.: Graph neural networks for temporal graphs: state of the art, open challenges, and opportunities. arXiv preprint arXiv:2302.01018 (2023)
21. Nguyen, G.H., Lee, J.B., Rossi, R.A., Ahmed, N.K., Koh, E., Kim, S.: Continuous-time dynamic network embeddings. In: Companion Proceedings of the Web Conference 2018, pp. 969–976 (2018)
22. Park, N., et al.: CGC: contrastive graph clustering for community detection and tracking. In: Proceedings of the ACM Web Conference 2022, pp. 1115–1126 (2022)
23. Peng, Z., et al.: Graph representation learning via graphical mutual information maximization. In: Proceedings of the Web Conference 2020, pp. 259–270 (2020)
24. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 701–710 (2014)
25. Rossi, E., Chamberlain, B., Frasca, F., Eynard, D., Monti, F., Bronstein, M.: Temporal graph networks for deep learning on dynamic graphs. arXiv preprint arXiv:2006.10637 (2020)
26. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: Line: large-scale information network embedding. In: Proceedings of the 24th International Conference on World Wide Web, pp. 1067–1077 (2015)

27. Tian, S., Wu, R., Shi, L., Zhu, L., Xiong, T.: Self-supervised representation learning on dynamic graphs. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, pp. 1814–1823 (2021)
28. Trivedi, R., Farajtabar, M., Biswal, P., Zha, H.: Dyrep: learning representations over dynamic graphs. In: International Conference on Learning Representations (2019)
29. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y., et al.: Graph attention networks. Stat **1050**(20), 10–48550 (2017)
30. Velickovic, P., Fedus, W., Hamilton, W.L., Liò, P., Bengio, Y., Hjelm, R.D.: Deep graph infomax. In: ICLR (Poster), vol. 2, no. 3, p. 4 (2019)
31. Wang, D., Cui, P., Zhu, W.: Structural deep network embedding. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1225–1234 (2016)
32. Wang, Y., Chang, Y.Y., Liu, Y., Leskovec, J., Li, P.: Inductive representation learning in temporal networks via causal anonymous walks. arXiv preprint arXiv:2101.05974 (2021)
33. Wen, Z., Fang, Y.: Trend: temporal event and node dynamics for graph representation learning. In: Proceedings of the ACM Web Conference 2022, pp. 1159–1169 (2022)
34. Wu, L., Lin, H., Tan, C., Gao, Z., Li, S.Z.: Self-supervised learning on graphs: contrastive, generative, or predictive. IEEE Trans. Knowl. Data Eng. (2021)
35. Xu, D., Ruan, C., Korpeoglu, E., Kumar, S., Achan, K.: Inductive representation learning on temporal graphs. arXiv preprint arXiv:2002.07962 (2020)
36. Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., Wang, L.: Deep graph contrastive representation learning. arXiv preprint arXiv:2006.04131 (2020)
37. Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., Wang, L.: Graph contrastive learning with adaptive augmentation. In: Proceedings of the Web Conference 2021, pp. 2069–2080 (2021)