CeDFormer: Community Enhanced Transformer for Dynamic Network Embedding

Jiaqi Guo[1], Tianpeng Li[2], Minglai Shao[2], Wenjun Wang[2,3], Lin Pan[2], Xue Chen[2], and Yueheng Sun[2]

[1] School of future technology, Tianjin University, Tianjin, China
[2] College of Intelligence and Computing, Tianjin University, Tianjin, China
[3] Yazhou Bay Innovation Institute, Hainan Tropical Ocean University, Sanya Hainan, 572022, China

Abstract. Dynamic Graph Representation Learning methods have achieved significant success. However, real-world data often do not strictly follow the smoothness assumption in time-dependent relationships between dynamic graph snapshots. Existing methods sometimes fail to capture these relationships effectively. To address this issue, we propose the Community Enhanced Transformer for Dynamic Network Embedding (CeDFormer), which leverages transformers to capture temporal structures in dynamic graphs. CeDFormer is designed to adapt to highly variable graph structures and complex temporal dependencies. To mitigate the high computational cost and limited scalability of transformers on large-scale graph data, we introduce an optimization strategy involving parameter sharing within stable communities from a global perspective. This strategy enhances training speed by $30\% \sim 35\%$ without compromising model performance. Extensive experiments on real-world datasets demonstrate that CeDFormer outperforms most other methods on the majority of datasets. Code is available at https://github.com/gjqwanttogjq/CeDFormer

Keywords: Dynamic Network Embedding· Network Representation Learning · Dynamic Network Data.

1 Introduction

In recent years, graph representation learning, also known as network embedding or graph embedding, has experienced rapid development. Its primary objective is to map nodes within a network into low-dimensional, real-valued, dense vectors while preserving network structure, node features, labels, and other auxiliary information [21, 32, 17]. Most existing graph representation learning methods are based on the assumption that the graph is static. However, most real-world problems are better modeled as dynamic graphs, such as social networks in Facebook[30] and user-video interaction graphs in YouTube[5]. The structure and features of dynamic graphs change over time. While representation learning on static graphs has been studied extensively, dynamic graph representation learning is a relatively newer and more promising area of research. Introducing the temporal dimension adds complexity to the problem, making it more challenging, but also more applicable to real-world scenarios[33, 31].

Existing methods have made several attempts to address the temporal dimension in dynamic graph analysis. For instance, approaches like VGRNN[13] have introduced recurrent neural networks (RNNs) to capture latent representations of nodes, while DGCN [10] employs LSTM to update GCN weight parameters, capturing global structural information across all snapshots in dynamic graphs. Others like DySAT [26], have introduced attention mechanisms to encode dynamic information that changes over time into node embeddings. DyFormer[8] builds upon attention mechanisms, incorporating transformers. Nonetheless, both of these methods still follow traditional approaches to handle the temporal dimension.

However, we have observed that real-world dynamic graph data do not always exhibit a strict chain-like dependency relationship,i.e., smoothness assumption, among snapshots over time. In other words, for a given snapshot, it may not necessarily have stronger dependencies on its adjacent snapshots. In the context of the real world, the evolution of each node does not necessarily occur within the time window between two consecutive snapshots[14, 11]. If a set of nodes undergoes complex, non-smooth, non-uniform, and highly frequent changes over multiple snapshots, the application of recurrent neural networks (RNNs) would indeed pose challenges[16]. During the experimental phase, we have demonstrated that many datasets exhibit these characteristics. This observation aligns with reality; for instance, transportation networks evolve cyclically on a weekly basis[24], where the network on any given day is highly similar to the network from one week ago, rather than the network from the previous snapshot. Similarly, unexpected events in social networks can disrupt the assumption of network smoothness.

In response to the aforementioned challenges, we have developed a model based on the Transformer[28] architecture that is suitable for highly variable dynamic graph structures with intricate temporal dependencies between snapshots. To reduce overall complexity and training costs, we introduce an optimization strategy that involves parameter sharing within stable structural communities in dynamic graphs. Extensive experiments have confirmed that CeDFormer outperforms state-of-the-art methods on most real-world datasets, demonstrating the feasibility and effectiveness of our model and optimization strategy.

2 Related Work

2.1 Dynamic Graph Representation Learning

Previous methods for dynamic graph representation learning have typically extended static graph algorithms by introducing a temporal dimension to capture time-dependent relationships. Approaches for capturing temporal dependencies can be categorized into three main classes[8]:

Temporal Smoothness-Based Methods. These methods aim to ensure temporal smoothness in the embeddings of dynamic graph snapshots by incorporating temporal regularization. For example, DYNGEM[12] generates incremental embeddings of the current graph snapshot based on the embeddings of the previous snapshot. However, this approach heavily relies on the smoothness between snapshots and performs poorly when there are significant changes in node behavior between adjacent snapshots.
Recursive Methods. Recursive methods use recurrent neural networks (RNNs) to capture temporal dependencies. For instance, GCRN[27] combines convolutional neural networks (CNNs) and RNNs to simultaneously identify meaningful spatial structures and dynamic patterns in graph-structured data. Similarly, DGCN[10] uses Long Short-Term Memory (LSTM)[15] units to update GCN's weight parameters, capturing global structural information across all time steps. However, these methods have limitations when dealing with long-term dependencies and non-sequential dependencies, and they may lack scalability for large-scale dynamic graph data.
Attention-Based Methods. Attention mechanisms are used to aggregate spatial and temporal information. Traditional attention mechanisms, such as DySAT[26], aggregate information using self-attention mechanisms in both spatial and temporal dimensions. TGAT[31] encodes temporal information into node features and then applies attention mechanisms. However, traditional spatial attention mechanisms only consider attention calculations on existing edges, and temporal attention mechanisms focus solely on previous snapshots. In recent years, attention-based methods have seen significant development, especially with the introduction of transformers on graphs.

3 CeDFormer Model

3.1 Problem Definition

Suppose we have a dynamic graph $\mathcal{G} = \{G^{(1)}, G^{(2)}, ..., G^{(T)}\}$, where $G^{(t)} = (\mathcal{V}^{(t)}, \mathcal{E}^{(t)})$ represents a snapshot of the dynamic graph at time t. Each snapshot corresponds to a graph with a node set $\mathcal{V}^{(t)}$ and an edge set $\mathcal{E}^{(t)}$. There are no constraints between $G^{(t)}$ and $G^{(t+1)}$, which means that over time, between snapshots, we can expect changes in the number of nodes, the appearance of new edges, and the disappearance of existing edges[29, 9]. CeDFormer takes variable-length sequences of adjacency matrices $\mathcal{A} = \{A^{(1)}, A^{(2)}, ..., A^{(T)}\}$ and node attribute matrices $\mathcal{X} = \{X^{(1)}, X^{(2)}, ..., X^{(T)}\}$ as input.

3.2 Method

In this section, we will introduce the architecture of the CeDFormer model (as illustrated in Figure 1). The model consists of two main parts: the encoder and the decoder. Furthermore, we introduce a strategy for optimizing parameter count and training speed within the encoder module through community discovery methods.

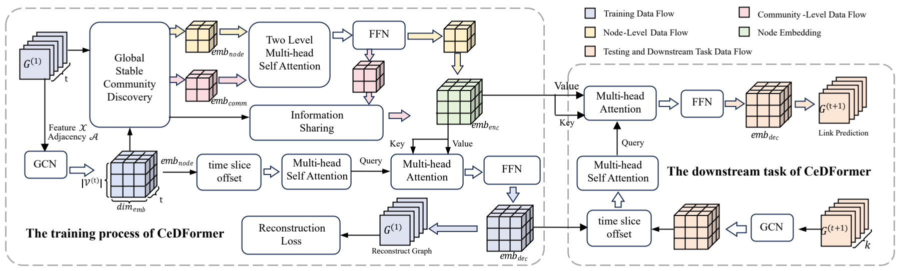


Fig. 1. The framework of the CeDFormer model, using link prediction as an example of downstream tasks, encompasses the learning process of the model from $G^{(1)}$ to $G^{(t)}$ and its application for link prediction from $G^{(t+1)}$ to $G^{(t+k)}$.

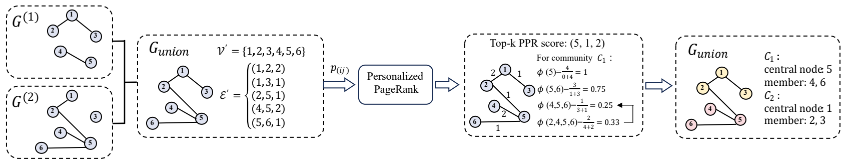

$G^{(1)}$
$G^{(2)}$
G_{union}
$\mathcal{V}' = \{1,2,3,4,5,6\}$
$\mathcal{E}' = \{(1,2,2), (1,3,1), (2,5,1), (4,5,2), (5,6,1)$
$p_{(ij)}$
Personalized PageRank
Top-k PPR score: (5, 1, 2)
For community C_1 :
$\phi(5) = \frac{4}{0+4} = 1$
$\phi(5,6) = \frac{3}{1+3} = 0.75$
$\phi(4,5,6) = \frac{1}{3+1} = 0.25$
$\phi(2,4,5,6) = \frac{2}{4+2} = 0.33$
C_1 :
central node: 5
member: 4, 6
C_2 :
central node: 1
member: 2, 3

Fig. 2. An example of global stable community discovery, demonstrating how to perform local community discovery based on global influence $top - k$ nodes on $G^{(1)}$ and $G^{(2)}$.

Global Stable Community Discovery Existing dynamic graph representation learning methods primarily focus on node-level embeddings, which pose scalability issues and computational inefficiencies, especially with large-scale graph data. To tackle this, we introduce a community perspective to detect stable and temporally invariant global community structures, illustrated in Figure 2. These communities comprise nodes sharing similar structures and features, facilitating parameter sharing and enhancing model efficiency. Expanding on this, we present a strategy for local community discovery, centered on the top-k nodes with significant global influence.

Firstly, we aggregate the dynamic graph $\mathcal{G} = \{G^{(1)}, G^{(2)}, ..., G^{(T)}\}$ into a joint temporal graph $\mathcal{G}_{union} = (\mathcal{V}^{'}, \mathcal{E}^{'})$. In this joint graph, $\mathcal{V}^{'}$ contains all nodes that appear in any time snapshot, i.e., $\mathcal{V}^{'} = \{ i \,|\, \exists i \in \mathcal{V}^{(t)}, t \in T\}$. Similarly, $\mathcal{E}^{'}$ consists of all edges present in any time snapshot, i.e., $\mathcal{E}^{'} = \{(i,j) | \exists (i,j) \in \mathcal{E}^{(t)}, t \in T\}$. The weight of any edge (i,j) in the joint graph is defined as: $\mathcal{W}_{ij}^{'} = \sum_{t \in T} A_{ij}^{(t)}$. From a global perspective, our goal is to identify highly influential nodes to serve as central points for local community discovery. These nodes are likely to be the most influential within their respective communities. Global influence reflects a node's impact on maintaining the global structure, while intra-community influence reflects its ability to represent the entire community. To find the top-k nodes with the highest global influence, we perform random walks[23] on the joint graph $\mathcal{G}_{union}$. The probability $p(i,j)$of transitioning from any node i to its neighboring node j is determined by the normalized weight of edge (i, j). This probability can be expressed as:

$$p(ij) = \frac{\mathcal{W}'_{ij}}{\sum_{k \in \mathcal{N}(i)} \mathcal{W}'_{ik}} \tag{1}$$

Here, $\mathcal{N}(i)$ represents the set of all neighboring nodes of node i.

Once we have calculated the transition probabilities between any two nodes in the joint graph, we compute the Personalized PageRank (PPR) [22] scores for all nodes on $\mathcal{G}_{union}$. The top-k nodes with the highest PPR scores are selected as central points for local community discovery[2]. These influential nodes serve as central points, forming the initial community structure. We then include all neighboring nodes within the community. Conductance[3] is used to measure the community structure, aiming to minimize inter-community edges while maximizing intra-community edges. For a community C, its conductance is defined as:

$$\phi(C) = \frac{\sum_{i \in C, k \in \overline{C}, (i,k) \in \mathcal{E}'} \mathcal{W}'_{ik}}{\sum_{i,j \in C, (i,j) \in \mathcal{E}'} \mathcal{W}'_{ij} + \sum_{i \in C, k \in \overline{C}, (i,k) \in \mathcal{E}'} \mathcal{W}'_{ik}} \tag{2}$$

If attempting to pull in a certain node can reduce the conductance of a local community, it is considered to strengthen that local community, and we formally include the node in the local community. The community structures will be used in the encoder section of the model. In our design, we can gradually aggregate node information through multiple levels to achieve the transformation from the node level to the community level. The number of levels can be selected to strike a balance between performance and efficiency.

Embedding In the encoder section, we start by aggregating the features of each snapshot of the dynamic graph using GCN (Graph Convolutional Network)[18]. For the snapshot $G^{(t)}$ at time t, we extract features, resulting in a feature matrix $\mathcal{F}_t = GCN(X^{(t)}, A^{(t)})$ of size $n \times dim_{emb}$, where n represents the number of nodes and dim_{emb} represents the node embedding dimensions. After aggregating features for all snapshots, we obtain a time-level node feature matrix of size $t \times n \times dim_{emb}$. This matrix is then used as input to the encoder layers.

To differentiate the temporal information between snapshots, we use position encoding in the transformer model to encode time information for each snapshot. The time dimension of the graph feature matrix enters the encoder layer and undergoes a dimension transformation, changing from a $t \times n \times dim_{emb}$ matrix to an $n \times t \times dim_{emb}$ matrix. This transformed matrix corresponds to the embeddings of all nodes at different time steps. Next, we optimize the model using the information from the global stable structural communities. For a community C, the central node i aggregates information from surrounding nodes using GCN. This process also aggregates the embeddings of other nodes within the community, propagating their information to the central node i. As a result, the embeddings of other nodes in community C can be replaced by the embedding of the central node $i : emb^{i}_{node}$ across all time steps. Based on this idea, we treat the embedding of the central node i as the embedding of community $C : emb^{C}_{comm}$, introducing a community-level encoder alongside the node-level encoder.

Encoder Due to the fact that global community detection does not assign community membership to all nodes, there are still a few isolated nodes outside of all communities from a global perspective. These isolated nodes need to be embedded at the node level using an encoder-decoder structure. Both the embeddings of isolated nodes at each time step and community embeddings are used as inputs in the two hierarchical levels. Since the dimensions of embeddings are the same in both levels, the encoder structure does not need any additional modifications, meaning that the encoder structure is reusable between the two levels. In the encoder, we take the node embeddings emb_{node} and community embeddings emb_{comm}, both generated by the embedding module, as inputs for the node-level and community-level respectively. Due to the reusability of the encoder structure, we denote the inputs for both levels as emb_{enc}. emb_{enc} serves as the input for Key, Query, and Value in the multi-head attention layer, allowing multiple heads to create distinct subspaces for the model to focus on different aspects of information.

In the encoder structure, introducing a community-level perspective can indeed lead to significant savings in training time. Assuming that during the phase of discovering stable structural communities at a global level, you find that there are c communities, each with m members, and the number of isolated nodes is $|\mathcal{V}'| - c * m$. Then, compared to not using the optimization strategy, the overall optimization rate for the entire encoder part can be calculated as $\frac{c*(m-1)}{|\mathcal{V}'|}$.

Decoder In the decoder part, we start again from the node perspective to compute embeddings for each node. Additionally, the process in the decoder layer differs between the training phase and the downstream task phase.

In the decoder part, we approach the problem from a node-centric perspective, calculating embeddings for each node. Furthermore, the decoder layer has two multi-head attention layers.

The first one is similar to the multi-head attention layer in the encoder, but we have made two improvements. Firstly, we introduce a concept similar to the "Beginning of Sentence" (BOS) token used in natural language processing (NLP). In CeFormer, the concept of the BOS token is specifically manifested in the initial graph embedding. During the training process, the initial graph embedding is placed before the first time snapshot. It is obtained through random initialization during the training phase and, in the downstream task phase, it is derived from the encoder's output in the training phase. Additionally, we remove the last time snapshot to maintain a consistent input size. This approach is designed to ensure that the training data at the current moment does not affect the calculation of node embeddings at the current time snapshot. Secondly, we have also introduced an additional sequence mask mechanism in the decoder, similar to the sequence mask used in transformers. The mask mechanism ensures that the model relies only on information from previous snapshots during node embedding calculations. Through these two methods, when reconstructing any snapshot at time t, we rely only on information from snapshots $0, ..., t-1$ (for t

> 1). The graph at time 0 is reconstructed from learnable, randomly initialized initial graph embeddings.

The second multi-head attention layer takes the output of the first one as a query and the output of the encoder to recalculate the node embedding matrix $emb^{'}_{enc}$, with dimensions of $n \times t \times dim_{emb}$, which is used as key and value. The embedding of any node i within $emb^{'}_{enc}$, denoted as $emb^{'}_{enc}i$, can be represented as follows:

$$emb^{'}_{enc}i = \begin{cases} emb^{'}_{node}i & \nexists m,\ s.t.\ i \in m \\ emb^{'}_{comm}m & \exists m,\ s.t.\ i \in m \end{cases} \tag{3}$$

Similar to the encoder, the output of the second multi-head attention layer in the decoder undergoes normalization and a feedforward layer to obtain embeddings for each node across all time steps. These embeddings are then combined, transformed to the matrix emb_{dec}. After passing through a linear layer, emb_{dec} is further transformed into reconstruction snapshots of dynamic graphs with dimensions $n \times n$.

Training The model's objective function is derived from the reconstruction error for each snapshot. The reconstruction error $Loss_t$ is computed using the following formula:

$$\begin{aligned} Loss_t = -\frac{1}{|\mathcal{V}^{(t)}|^2} \sum_{i=1}^{|\mathcal{V}^{(t)}|} \sum_{j=1}^{|\mathcal{V}^{(t)}|} & adj(t)_{ij} \cdot \log(adj_{rec}(t)_{ij}) \\ & + (1 - adj(t)_{ij}) \cdot \log(1 - adj_{rec}(t)_{ij}) \end{aligned} \tag{4}$$

Here, $adj(t)_{i,j}$ represents the value of the edge i, j in the original adjacency matrix at time t, and $adj_{rec}(t)_{i,j}$ represents the corresponding value of the edge i, j in the reconstructed snapshot at time t. The $adj_{rec}(t)$ values are obtained by passing the embeddings at time t, $emb_{dec}(t)$, through a linear layer.

The overall loss for the entire model is obtained by summing up the losses for each time step, represented as:

$$Loss = \sum_{t=1}^{T} Loss_t \tag{5}$$

During the model training phase, the encoder learns to obtain better node representations, while the decoder learns how to effectively utilize node representations and historical information to reconstruct the dynamic graph.

3.3 Model complexity analysis

For a dynamic graph with a time slice length of t, maximum nodes v, maximum edges e, embedding dimension d, transformer layers n, and attention heads h, neglecting global community optimization, CeDFormer's time complexity is $O(etd + nvt^2d)$, and space complexity is $O(hnvd^2t^2)$. CeDFormer's design involves applying transformers at each node, thus scaling with increasing nodes without causing an unacceptable decline in model efficiency.

4 Experiments

4.1 Experimental Setup

Dataset and Baseline Models. To demonstrate CeDFormer's performance, we conducted an in-depth study of standard datasets from related literature. We selected five widely-used datasets: enron10[6], DBLP[13], highSchool[10], Facebook[25], Email[10] and bitcoinOTC[10]. These datasets were used for tasks such as link prediction, clustering, and time efficiency comparisons. We also analyzed the temporal dependencies in these datasets.Our baseline models included: VGRNN[13], DySAT[26],DyFormer[8], DGCN[10], HGWaveNet[4]. We followed the settings described in the original papers of these baseline models. The specific number of nodes, edges, the number of snapshots, and the number of test snapshots for each dataset are listed in Table 1.

Table 1. Dataset description

Dataset	Nodes	Edges	Snapshots	Test Snapshots
Enron	184	4784	11	3
DBLP	315	5104	10	3
highSchool	327	26870	9	3
Facebook	663	23394	9	3
Email	2029	28090	29	3
bitcoinOTC	5053	11651	12	3

Evaluation Metrics. In the time snapshot dependency analysis section, for a dynamic graph $\mathcal{G} = \{G^{(1)}, G^{(2)}, ..., G^{(T)}\}$, where $t > 1$, we calculate the cosine similarity between snapshot $G^{(t)}$ and all the previous time snapshots $G^{(1)}, G^{(2)}, ...G^{(t-1)}$. This calculation is used to measure the dependency relationship between any two time snapshots. The calculation of cosine similarity involves reducing the dimensionality of the two snapshots and then computing the cosine similarity on two one-dimensional vectors.

In the context of dynamic graphs with partially observed snapshots $\mathcal{G} = \{G^{(1)}, G^{(2)}, ..., G^{(T)}\}$, link prediction tasks can be described as follows:

1) Link Prediction: This involves predicting the edges in $G^{(t+1)}$, which means forecasting which edges will exist in the next snapshot.
2) New Link Prediction: This task specifically focuses on predicting edges in $G^{(t+1)}$ that are not present in $G^{(t)}$.

Hyper-parameters. The link prediction experiments follow the same settings as in VGRNN. We randomly select 5%and 10% of the edges from each snapshot as the validation and test sets for the link prediction task. The embedding dimension for all tasks was set to 32. In the experiments, CeDFormer defaults to using all nodes for the global stable community optimization strategy. All experiments were conducted using a single RTX3090 GPU with 24GB of VRAM.

4.2 Experiment Results

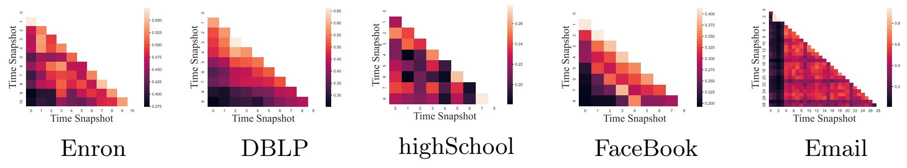


Fig. 3. Time Snapshot Dependency Analysis

Time Snapshot Dependency Analysis. Based on Figure 3, we conducted dependency analysis on the mentioned datasets. It can be observed that the diagonal blocks in the dependency matrices for the Enron and DBLP data have lighter colors, corresponding to cosine similarities exceeding 0.5. This indicates stronger dependencies between each snapshot and the previous one, aligning these datasets more closely with traditional time series data and adhering more to the Markov assumption: that the conditional probability distribution of future states depends only on the current state. In particular, DBLP data highlights the transitivity of time-dependent relationships, meaning that the similarity between any two snapshots is influenced by the time gap between them.

Similarly, the Facebook data also exhibits the transitivity of dependencies, but the snapshot dependencies within the Facebook dataset itself are not very pronounced, with diagonal elements in the matrix representing cosine similarities not exceeding 0.4.

For the highSchool data, it does not exhibit clear dependencies, especially in snapshots 4 and 6, where it shows highly inconsistent characteristics compared to the previous snapshot. Such data is more variable, displays weaker time dependencies, and tends to exhibit dependencies across snapshots.

In the case of the Email dataset, we observed a high degree of change and periodic variations. This dataset undergoes significant changes from snapshot 3 to 4, resulting in minimal dependencies between later snapshots and snapshots 0 to 3.

As designed, CeDFormer performs better on data resembling highSchool data, leading to more significant improvements compared to baseline methods.

Link Prediction. The experimental results for link prediction on various datasets are presented in Table 2. From the table, we can observe that CeDFormer performs comparably to HGWaveNet on the Enron dataset and is approximately 1.3% behind HGWaveNet on the DBLP dataset. However, on the highSchool, bitcoinOTC, Facebook, and Email datasets, CeDFormer outperforms the latest

Table 2. AUC and AP scores of link prediction. The best are bolded and the second best are underlined.

Metrics	Model	Enron	DBLP	highSchool	FaceBook	Email	bitcoinOTC
AUC	VGRNN	93.42±0.70	85.80±0.78	89.66±0.32	89.79±0.34	91.92±1.18	81.34±3.25
	DySAT	88.81±1.10	86.74±1.51	91.51±0.48	88.88±0.89	90.42±0.91	83.06±2.71
	Dyformer	90.35±0.45	77.74±0.63	85.01±0.29	83.74±0.76	91.08±0.44	83.62±1.01
	DGCN	85.27±0.85	72.03±0.54	66.18±0.53	68.65±0.29	95.12±0.30	84.35±4.88
	HGWaveNet	94.45±0.26	**88.95±0.47**	91.64±0.22	86.98±0.66	92.49±0.36	80.06±1.27
	CeDformer	**94.67±0.51**	87.64±0.92	**94.33±0.80**	**90.05±0.50**	**95.83±1.01**	**89.23±1.97**
AP	VGRNN	**94.50±0.66**	88.64±0.58	88.71±0.65	89.14±0.41	93.34±0.70	88.62±2.37
	DySAT	87.30±1.54	89.64±1.00	89.17±1.12	88.61±0.96	89.19±1.17	81.18±2.59
	Dyformer	90.78±0.44	78.94±0.51	84.25±0.34	81.16±0.72	92.39±0.49	84.54±1.74
	DGCN	84.51±0.79	73.14±0.69	66.09±0.56	68.78±0.34	95.32±0.31	81.42±3.74
	HGWaveNet	94.37±0.32	**91.72±0.38**	90.86±0.25	85.96±0.71	94.02±0.32	81.15±1.24
	CeDformer	93.87±0.39	90.52±0.48	**94.10±0.83**	**89.84±0.56**	**96.65±0.42**	**88.63±2.70**

Table 3. The average training time for one epoch on different datasets, in seconds/epoch.

Model	**Enron**	**DBLP**	**highSchool**	**FaceBook**	**Email**
DySAT	2.240	2.268	2.740	4.622	8.620
Dyformer	1.123	2.474	4.728	6.822	17.774
HGWaveNet	67.740	95.316	127.335	262.109	683.212
CeDFormer	2.267	3.319	2.945	5.972	25.719

baselines, indicating that CeDFormer performs better on highly variable and periodic data, as supported by the analysis of time snapshot dependencies.

The Table 3 illustrates the average runtime per epoch on the dataset for each method (measured in seconds) for link prediction. We compare the efficiency of CeDFormer with other deep baseline methods. Consistent with our hypothesis, despite utilizing a transformer-based architecture, through optimization techniques, the efficiency of our model is comparable to existing methods and significantly outperforms the HGWaveNet model on hyperbolic space.

Table 4. AUC and AP scores of new link prediction. The best are bolded and the second best are underlined.

Metrics	Model	Enron	DBLP	highSchool	FaceBook	Email	bitcoinOTC
AUC	VGRNN	87.41±0.96	75.87±1.65	88.09±0.28	86.76±0.54	90.37±1.28	80.22±2.78
	DySAT	82.58±3.00	76.57±2.30	90.42±0.78	85.97±1.21	81.71±3.97	83.09±2.61
	Dyformer	87.35±0.52	74.76±0.66	84.23±0.33	81.58±0.69	90.62±0.68	81.31±1.69
	DGCN	81.87±0.70	62.48±1.54	62.08±0.28	64.98±0.43	93.21±0.40	83.50±1.83
	HGWaveNet	89.38±0.36	**83.73±0.55**	89.63±0.24	83.97±0.61	92.05±0.38	79.97±0.99
	CeDformer	**90.46±0.59**	80.41±0.88	**91.35±0.78**	**88.37±0.46**	**94.36±0.49**	**88.72±1.55**
AP	VGRNN	88.76±0.70	78.38±1.21	86.81±0.54	85.30±0.66	92.22±0.77	84.85±3.45
	DySAT	83.07±2.89	78.81±2.13	88.76±1.56	84.78±1.79	74.96±3.13	82.24±2.94
	Dyformer	88.00±0.46	73.96±0.73	84.25±0.31	80.66±0.55	91.24±0.76	80.02±2.06
	DGCN	82.03±0.57	64.72±1.52	62.15±0.37	65.65±0.53	93.36±0.43	81.38±1.15
	HGWaveNet	87.71±0.33	**86.89±0.47**	88.48±0.27	82.30±0.67	93.29±0.38	79.86±1.25
	CeDformer	**89.74±0.71**	83.39±0.64	**90.09±0.72**	**87.91±0.45**	**95.75±0.53**	**87.11±2.01**

New Link Prediction. New link prediction is used to predict new edges that will appear in the next snapshot, evaluating the model's inductive ability. This task is more challenging, and both link prediction and new link prediction tasks are performed in the same model training process. Compared to link prediction tasks, the performance of new link prediction tasks may experience a certain degree of decline. However, as shown in Table 4, CeDFormer demonstrates better generalization on the Enron dataset, surpassing HGWaveNet and achieving better results. Furthermore, it maintains a leading position on the highSchool, bitcoinOTC, Facebook, and Email datasets. Compared to other self-attention-based methods or transformer-based methods, we harness the attention mechanism's performance more effectively on dynamic graph data.

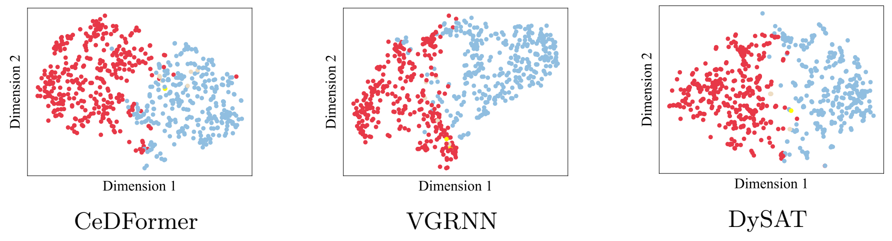


Fig. 4. The results of node representation analysis, t-SNE Visualization of Graph Embeddings

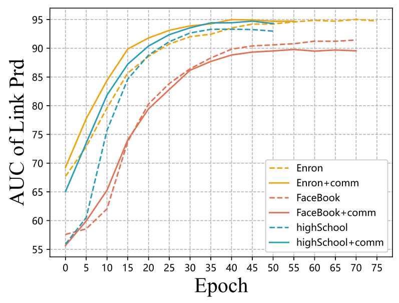
4.3 Node Representation Analysis

In this experiment, we used t-SNE (t-Distributed Stochastic Neighbor Embedding)[20] to reduce the node embeddings of the Facebook dataset snapshots to two dimensions for visualization. We employed the same t-SNE parameters for each model, setting the cluster density (*perplexity*) to 10 and the maximum number of iterations(n_iter) to 300.

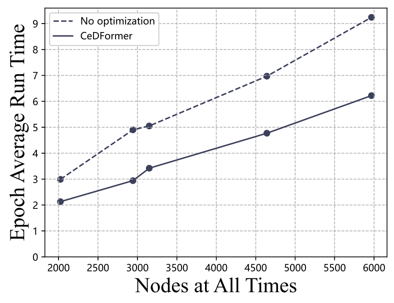
From Figure 4, it can be observed that although CeDFormer requires more complex methods like calculating attention and using linear layers in the decoder module to reconstruct snapshots for node embeddings, the quality of node embeddings doesn't deteriorate. Compared to VGRNN, CeDFormer's t-SNE visualization shows smaller areas of intersection between two classes of labels (red and blue), and it is on par with DySAT based on self-attention mechanisms.

4.4 Ablation Study

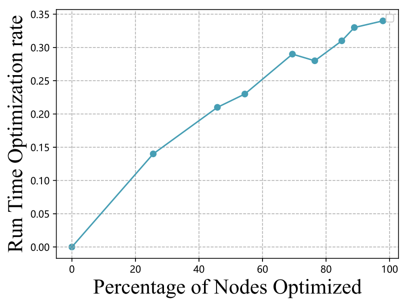
To further validate the reliability of our model's global community optimization strategy, we conducted ablation experiments on the relevant datasets. We also explored the impact of the proportion of nodes in community levels and different community discovery methods on optimization performance and efficiency.



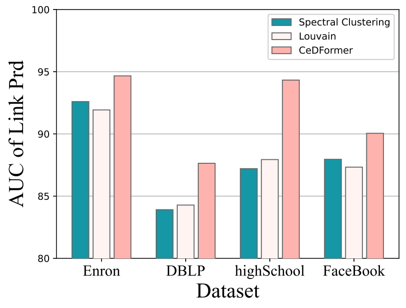
(a) Performance Comparison



(b) Time Complexity



(c) Community Enhanced Percentage



(d) Community Detection Method

Fig. 5. The results of ablation experiment

Performance Comparison. As shown in Figure 5(a), we conducted experiments on the Enron, Facebook, and highSchool datasets. Different colored lines represent different datasets. The solid lines represent CeDFormer, while the dashed lines represent the model without community-level optimization. There is no significant difference in performance before and after removing the optimization strategy. CeDFormer converges faster on the Enron and highSchool datasets, with around a 1.8% performance improvement on the highSchool dataset. However, after removing the global stable community structure optimization strategy, CeDFormer's performance decreases by around 1.6% on the Facebook dataset.

Time Efficiency Analysis. As shown in Figure 5(b), we compared the running time of the model per epoch before and after removing the optimization strategy at different node counts (where the total node count is obtained by adding the number of nodes from all snapshots). It can be observed that as the node count increases, using the optimization strategy still reduces training time by 30% $\sim$ 35% . The optimization effect is more significant on datasets with a more pronounced community structure.

Analysis of the Proportion of Community-Level Nodes. The global stable structure optimization strategy allows us to choose the proportion of nodes participating in the community level, relative to all nodes. As shown in Figure 5(c), when this proportion is set to 0, the CeDFormer reverts to having only the node-level. The proportion of community-level nodes starts from 25.6%, given that the initial selection of nodes generates larger local communities. It can be observed that the optimization rate does not increase linearly with this proportion. This is because the process of selecting central nodes is sorted by influence, and nodes selected later have less influence, resulting in smaller local community structures and less effective optimization through parameter sharing.

Community Detection Methods Analysis. As shown in Figure 5(d), we conducted a performance analysis by replacing our global influence-based local community detection method on four datasets. It can be observed that the Louvain algorithm[7] and spectral clustering methods[1, 19] did not perform well in the context of joint graph community detection for global stability. In particular, the Louvain algorithm had excessive time overhead on large-scale graph data and is not a viable choice. This experiment further validates the effectiveness of our optimization strategy.

Table 5. AUC and AP scores of link prediction on the BitcoinOTC Dataset.

Model	**bitcoinotc**		**bitcoinotc-4k**		**bitcoinotc-3k**		**bitcoinotc-2k**		**bitcoinotc-1k**	
	AUC	**AP**	**AUC**	**AP**	**AUC**	**AP**	**AUC**	**AP**	**AUC**	**AP**
VGRNN	81.34	88.62	82.02	89.31	81.19	89.05	76.95	85.99	63.07	71.40
DySAT	83.06	81.18	86.32	85.20	84.19	81.29	87.37	85.90	68.54	74.31
CeDFormer	89.23	88.63	93.48	94.05	93.22	95.03	88.42	85.21	73.01	75.20

4.5 Model robustness experiment

Node Quantity Robustness Experiment We partitioned the bitcoinOTC data into subsets containing 1000, 2000, 3000, and 4000 nodes. We conducted experiments to assess CeDFormer's robustness with increasing node counts across these datasets, using VGRNN and DySAT as baselines. The specific experimental results are presented in Table 5. Observing the results, CeDFormer outperformed both VGRNN and DySAT across all versions of the bitcoinotc dataset. It is worth noting that CeDFormer demonstrates excellent performance on datasets with 2k and 3k nodes, indicating that the model can effectively reconstruct missing information when some information is absent. However, on the 1k-node dataset, due to its sparse nature and significant information loss, the model's performance may decline, yet it still outperforms VGRNN and DySAT.

As illustrated in Figure 6, We plotted the average training time per epoch related to the increase in the node count on the bitcoinOTC dataset. It is evident that the overall runtime of CeDFormer is almost linearly correlated with the number of nodes, consistent with the results of our previous time complexity analysis.

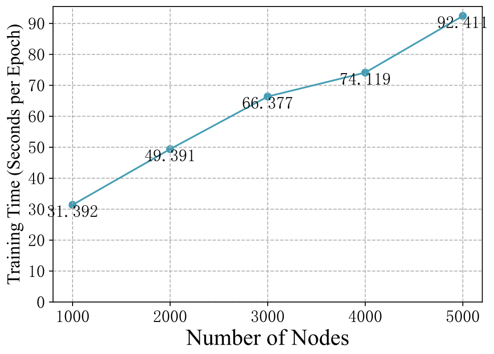


Fig. 6. The training time per epoch (seconds/epoch) as a function of the number of nodes

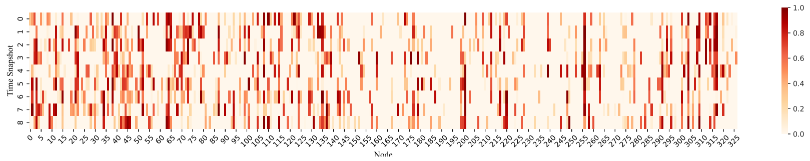
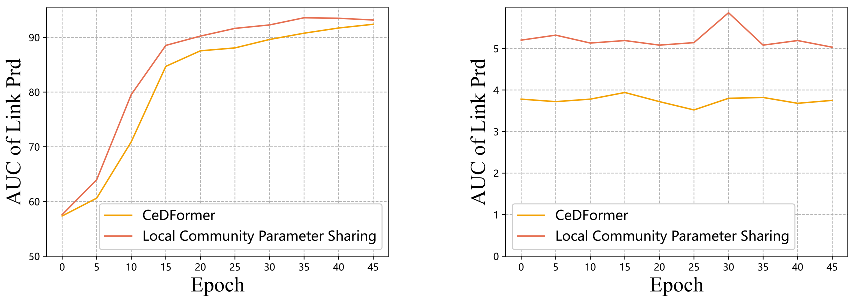


Fig. 7. Analysis of influential nodes in the highSchool Dataset

Robustness experiment of influential nodes We replicated the method used in CeDFormer to select influential nodes. This process was executed in each snapshot, extracting the top 100 influential nodes and comparing their distributions across all snapshots. The visualization of node influence for each snapshot is depicted in Figure 7, where the heatmap rows represent individual snapshots, columns represent nodes, and color intensity indicates the node's influence within a snapshot. Darker colors denote higher influence. The analysis revealed that the distribution of the top 100 influential nodes across each snapshot in the highSchool dataset appeared disordered. There was significant variation in influential nodes across snapshots.

Then we replaced CeDFormer's global community optimization strategy with a strategy involving parameter sharing within communities in each snapshot.



(a) Performance Comparison (b) Time Complexity

Fig. 8. The results of ablation experiment

This approach strictly identified each influential node, even in cases of variability. It's important to note that this strategy did not identify stable global communities, eliminating the efficiency optimization introduced by CeDFormer's community hierarchy, and only reduced the parameter count due to parameter sharing. Figure 8 illustrates the effectiveness and efficiency of this approach. Even in datasets with highly variable influential nodes, CeDFormer performs slightly less than the local parameter sharing strategy for influential nodes in time slices, but it achieves optimization in efficiency, remaining consistent with the original design motivation of the model.

5 Conclusion

In this paper, we introduced CeDFormer, a model designed to adapt better to highly variable graph structures and complex dependencies between snapshots. We proposed an optimization strategy that involves sharing parameters within stable communities from a global perspective on the graph structure. Extensive experiments demonstrate the effectiveness and versatility of our proposed framework compared to state-of-the-art methods. In future experiments, we will progressively aggregate node information through a bottom-up, multi-level community detection approach, aiming to transform node-level information into community-level representation. We will also explore methods and metrics to select the optimal number of community levels to balance performance and efficiency.

6 Acknowledge

This work was supported in part by Key R&D Project of Hainan (Grant No.ZDYF 2024SHFZ051), National Key R&D Plan of China (Grant No.2022YFC2602305) and ENNGroup under Grant 2023GKF-1220.

References

1. Amini, A.A., Chen, A., Bickel, P.J., Levina, E.: Pseudo-likelihood methods for community detection in large sparse networks (2013)
2. Andersen, R., Chung, F.: Detecting sharp drops in pagerank and a simplified local partitioning algorithm. In: International Conference on Theory and Applications of Models of Computation. pp. 1–12. Springer (2007)
3. Andersen, R., Chung, F., Lang, K.: Local graph partitioning using pagerank vectors. In: 2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06). pp. 475–486. IEEE (2006)
4. Bai, Q., Nie, C., Zhang, H., Zhao, D., Yuan, X.: Hgwavenet: A hyperbolic graph neural network for temporal link prediction. In: Proceedings of the ACM Web Conference 2023. pp. 523–532 (2023)
5. Benevenuto, F., Duarte, F., Rodrigues, T., Almeida, V.A., Almeida, J.M., Ross, K.W.: Understanding video interactions in youtube. In: Proceedings of the 16th ACM international conference on Multimedia. pp. 761–764 (2008)
6. Benson, A.R., Abebe, R., Schaub, M.T., Jadbabaie, A., Kleinberg, J.: Simplicial closure and higher-order link prediction. Proceedings of the National Academy of Sciences **115**(48), E11221–E11230 (2018)
7. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. Journal of statistical mechanics: theory and experiment **2008**(10), P10008 (2008)
8. Cong, W., Wu, Y., Tian, Y., Gu, M., Xia, Y., Chen, C.c.J., Mahdavi, M.: Dyformer: A scalable dynamic graph transformer with provable benefits on generalization ability. In: Proceedings of the 2023 SIAM International Conference on Data Mining (SDM). pp. 442–450. SIAM (2023)
9. Gao, C., Fan, Y., Jiang, S., Deng, Y., Liu, J., Li, X.: Dynamic robustness analysis of a two-layer rail transit network model. IEEE Transactions on Intelligent Transportation Systems **23**(7), 6509–6524 (2021)
10. Gao, C., Zhu, J., Zhang, F., Wang, Z., Li, X.: A novel representation learning for dynamic graphs based on graph convolutional networks. IEEE Transactions on Cybernetics (2022)
11. Goyal, P., Chhetri, S.R., Canedo, A.: dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. Knowledge-Based Systems **187**, 104816 (2020)
12. Goyal, P., Kamra, N., He, X., Liu, Y.: Dyngem: Deep embedding method for dynamic graphs. arXiv preprint arXiv:1805.11273 (2018)
13. Hajiramezanali, E., Hasanzadeh, A., Narayanan, K., Duffield, N., Zhou, M., Qian, X.: Variational graph recurrent neural networks. Advances in neural information processing systems **32** (2019)
14. Harary, F., Gupta, G.: Dynamic graph models. Mathematical and Computer Modelling **25**(7), 79–87 (1997)
15. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
16. Karita, S., Chen, N., Hayashi, T., Hori, T., Inaguma, H., Jiang, Z., Someki, M., Soplin, N.E.Y., Yamamoto, R., Wang, X., et al.: A comparative study on transformer vs rnn in speech applications. In: 2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU). pp. 449–456. IEEE (2019)
17. Kazemi, S.M., Goel, R., Jain, K., Kobyzev, I., Sethi, A., Forsyth, P., Poupart, P.: Representation learning for dynamic graphs: A survey. The Journal of Machine Learning Research **21**(1), 2648–2720 (2020)

18. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
19. de Lange, S.C., de Reus, M.A., van den Heuvel, M.P.: The laplacian spectrum of neural networks. Frontiers in computational neuroscience **7**, 189 (2014)
20. Linderman, G.C., Rachh, M., Hoskins, J.G., Steinerberger, S., Kluger, Y.: Efficient algorithms for t-distributed stochastic neighborhood embedding. arXiv preprint arXiv:1712.09005 (2017)
21. Liu, X., Tang, J.: Network representation learning: A macro and micro view. AI Open **2**, 43–64 (2021). https://doi.org/https://doi.org/10.1016/j.aiopen.2021.02.001, https://www.sciencedirect.com/science/article/pii/S2666651021000024
22. Page, L.: The pagerank citation ranking: Bringing order to the web. technical report. Stanford Digital Library Technologies Project, 1998 (1998)
23. 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)
24. Rahimi, A., Cohn, T., Baldwin, T.: Semi-supervised user geolocation via graph convolutional networks. arXiv preprint arXiv:1804.08049 (2018)
25. Rossi, R., Ahmed, N.: The network data repository with interactive graph analytics and visualization. In: Proceedings of the AAAI conference on artificial intelligence. vol. 29 (2015)
26. Sankar, A., Wu, Y., Gou, L., Zhang, W., Yang, H.: Dysat: Deep neural representation learning on dynamic graphs via self-attention networks. In: Proceedings of the 13th international conference on web search and data mining. pp. 519–527 (2020)
27. Seo, Y., Defferrard, M., Vandergheynst, P., Bresson, X.: Structured sequence modeling with graph convolutional recurrent networks. In: Neural Information Processing: 25th International Conference, ICONIP 2018, Siem Reap, Cambodia, December 13-16, 2018, Proceedings, Part I 25. pp. 362–373. Springer (2018)
28. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
29. Wang, Z., Wang, C., Gao, C., Li, X., Li, X.: An evolutionary autoencoder for dynamic community detection. Science China Information Sciences **63**, 1–16 (2020)
30. Weaver, J., Tarjan, P.: Facebook linked data via the graph api. Semantic Web **4**(3), 245–250 (2013)
31. Xu, D., Ruan, C., Korpeoglu, E., Kumar, S., Achan, K.: Inductive representation learning on temporal graphs. arXiv preprint arXiv:2002.07962 (2020)
32. Zhu, Y., Xu, W., Zhang, J., Du, Y., Zhang, J., Liu, Q., Yang, C., Wu, S.: A survey on graph structure learning: Progress and opportunities. arXiv preprint arXiv:2103.03036 (2021)
33. Zhu, Y., Lyu, F., Hu, C., Chen, X., Liu, X.: Encoder-decoder architecture for supervised dynamic graph learning: A survey. arXiv preprint arXiv:2203.10480 (2022)