DYNAMIC NETWORK LINK PREDICTION BASED ON ENHANCED GRAPH CONVOLUTIONAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Dynamic graph representation learning (DGRL) methods based on Graph Convolutional Networks (GCNs) have been the main approaches for dynamic network tasks including link prediction and node classification applied in numerous fields. The current Graph Convolutional Network (GCN)-based methods mostly neglect the differences in importance of neighbor nodes which leads to an unsatisfied information aggregation and hold an excessively computation complexity by employing additional temporal encoders. To address these issues, we propose the Enhanced Graph Convolutional Networks (EGCN) framework for DGRL. First, we define Structurally Enhanced Weighted Resource Allocation (SE-WRA) similarity which introduces a logarithmic enhancement term and a structural interaction term simultaneously to distinguish the importance of neighboring nodes, and further utilizes this similarity to guide the aggregation. Second, to integrate the local and global structural information of dynamic graphs without temporal encoder, we propose a multi-snapshot gradient optimization (MSGO) method based on smoothed second-order moments to update the GCN weight parameters. Specifically, this method defines an exponential moving average (EMA) mechanism with second-order moments, which smooths the gradients at each time step to generate local smoothed gradients for capturing local structural information. Then, we propose an adaptive gradient aggregation strategy that incorporates a dynamic adjustment factor based on global graph structural information to generate a globally adaptive gradient for updating the GCN, thereby obtaining global information. Experiments on six public datasets show the advantage of our EGCN compared with existing baselines, where it has reached the optimum in fourteen out of fifteen performance metrics.

1 INTRODUCTION

Many complex systems in the real world, such as social networks, transportation networks, and financial transaction networks, can often be abstracted as graph-structured data. The structural relationships in these systems tend to evolve significantly over time Kazemi et al. (2020); Skarding et al. (2021). Graphs with such temporal evolution are commonly referred to as dynamic graphs. Dynamic graphs capture not only the temporal variations in entity relationships but also contain rich structural information and behavioral patterns. These characteristics are essential for uncovering the mechanisms underlying system evolution and for predicting future status, thus playing a critical role in dynamic network link prediction tasks. Recently, graph neural networks (GNNs) have achieved rapid progress in learning node embeddings based on local neighborhood information Hamilton et al. (2017); Kipf & Welling (2016); Wang & Zhang (2022). Under these circumstances, incorporating graph evolution information into GNNs (i.e., dynamic GNNs) shows a prominent prospect and has attracted considerable attention in very recent years Li & Chen (2021); You et al. (2022). Despite the great success, there are still two following limitations.

Node importance awareness issue. It indicates that the importance of a node reflects the extent to which it influences the representation of a target node in graph representation learning. However, most existing GCN-based graph representation learning methods compute node features by aggregating features of neighboring nodes on average, which may leave them especially less capable of identifying the importance of different neighboring nodes to a target node Long et al. (2020). Although existing attention mechanism-based methods (e.g.,GAT Veličković et al. (2017)) can distin-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

guish such importance between nodes by adaptively training the attention weights, the computational process is time consuming and involves the higher space complexity Bahdanau et al. (2016). To address these limitations, some studies have introduced similarity functions to assess the importance of neighboring nodes, such as Dice similarity Xie et al. (2018) and Weighted Resource Allocation similarity(WRA)Moradabadi & Meybodi (2018); Kumar et al. (2020); Zhao et al. (2015). Dice similarity measures structural similarity based on the number of shared neighbors, which is computationally simple but lacks consideration of edge weights and node-level structural roles. on the other hand, WRA incorporates edge weight information and node degree to enhance the expression of structural centrality. Nevertheless, it relies solely on common neighbors, lacks awareness of global structural information, and is sensitive to anomalous edge weights, which makes it particularly pronounced in dynamic graph scenarios with evolving structures over time. Therefore, designing an efficient and interpretable mechanism for node importance awareness is crucial for enhancing the structural expressiveness and generalization capability of dynamic graph neural networks.

The dependency on temporal encoders leads to complexity issues. In the study of dynamic GNNs, mainstream approaches typically employ GNNs to model the structural information of each temporal snapshot and leverage temporal modeling components to capture the dynamic evolution of the graph over time. Common temporal modeling strategies include recurrent neural networks (RNNs) Pareja et al. (2020), long short-term memory networks (LSTMs) Goyal et al. (2020), and more recently, Transformer-based architectures Sankar et al. (2020). These modules enhance the model's ability to represent graph evolution trends by learning sequential dependencies across snapshots. However, temporal encoders of this type often rely on deep stacked architectures and multi-step state propagation mechanisms to model cross-snapshot temporal dependencies, which significantly increase the parameter scale. As a result, memory consumption, computational overhead, and the difficulty of hyperparameter tuning increase noticeably during training, thereby significantly increasing the overall modeling complexity. Moreover, the large number of trainable parameters introduced by temporal encoders further elevates the risk of overfitting Zhu et al. (2023). Therefore, realizing a temporal encoder-free model to remove the use of these temporal encoders can improve the parameter efficiency of the model, thus resulting in performance increment and reduce complexity. However, it is challenging because the direct removal of the temporal encoder will cause the intractable modeling of graph dynamics Zhu et al. (2023).

Taking the above limitations together, we study the key research question: **How can we efficiently and robustly learn representations from dynamic graphs without relying on complex temporal encoders, while enhancing the model's ability to perceive the importance of neighboring nodes?** In this paper, as illustrated in Figure 1, we propose a novel dynamic GCN framework called EGCN (abbreviation of ehanced graph convolutional network) to establish efficient and robust dynamic graph representation learning. First, we introduce the Structurally Enhanced Weighted Resource Allocation (SE-WRA) similarity, which measures the similarity between a target node and its neighboring nodes by integrating the direct connection strength and structural information of shared neighbors. This similarity is then used to guide the feature aggregation process. Second, we develop a multi-snapshot gradient optimization (MSGO) method based on smoothed second-order moments to update the GCN parameters at each time step. Specifically, this method incorporates meta-learning strategies Yang et al. (2022) into a simple GCN structure, where the frame-wise loss is calculated on the current snapshot and applies an EMA mechanism with second-order moments to smooth the gradients, thereby generating stable local smoothed gradients that are propagated to the next snapshot. In this way, EGCN establishes a representation association between two adjacent snapshots and removes all temporal-specific encoders, thereby improving the generalization and parameter efficiency. In addition, we propose an adaptive gradient aggregation strategy that generates adaptive gradients by aggregating local smoothed gradients from multiple snapshots. In this process, to adaptively adjust the contribution weight of each snapshot, we design a dynamic adjustment factor based on global graph states, which compares the structural similarity between the current and historical snapshots to achieve a dynamic balance of information across different time steps.

We summarize the main contribution in our study as follows:

- A Structurally Enhanced WRA similarity is proposed to better specify the degree of similarity between two nodes, in order to evaluate the importance of different neighboring nodes, which can be further used to guide the aggregation.

- A multi-snapshot gradient optimization mechanism based on smoothed second-order moments is proposed to update GCN weights at each time step, effectively achieving a temporal encoder-free GNN design efficiently and robustly.
- Extensive experiments on six public datasets are performed, and the superiority of EGCN is demonstrated by the results. Meanwhile, EGCN can outperform the state-of-the-art baseline WinGNN even on a very large dynamic graph with more than 60 million edges.

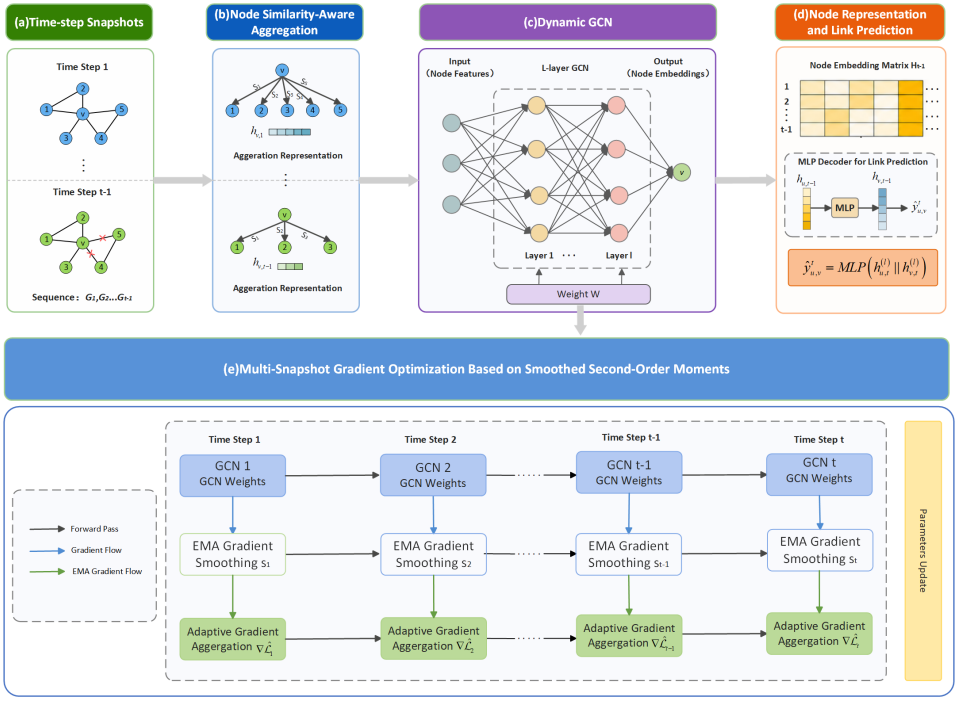

Figure 1: Overall framework of EGCN, which consists of (a) Time-step Snapshots, (b) Node Similarity-Aware Aggregation, (c) Dynamic GCN, (d) Node Representation, and (e) Multi-Snapshot Gradient Optimization Based on Smoothed Second-Order Moments. In (a), "x" denotes disconnected links.

2 Related Work

2.1 Graph Convolutional Networks

GCNs have demonstrated remarkable capability in analyzing graph-structured data and have been widely applied to various tasks, including graph representation learning, node classification, link prediction, and recommendation systems Gao et al. (2019). Existing GCN models can generally be categorized into two groups,that is, the spectral-based methods and spatial-based methods Wu et al. (2020). Spectral-based approaches perform convolution operations in the frequency domain by mapping graph signals through Fourier transforms and utilizing the eigendecomposition of the Laplacian matrix Defferrard et al. (2016b), as exemplified by classical models such as ChebNet Defferrard et al. (2016a) and CayleyNet Levie et al. (2018). In contrast, spatial-based methods directly aggregate features from local neighborhood structures. For example, GraphSAGE Hamilton et al. (2017) improves feature integration by sampling neighboring nodes and designing various aggregation functions, while GAT Veličković et al. (2017) introduces an attention mechanism to adaptively assign weights based on the importance of neighboring nodes, thereby enhancing the model's ability to capture local structural information. In addition to these feature aggregation-based models, several approaches focus on exploring the connectivity strength between nodes to enhance the expressive power of graph structural information. Such methods include deep learning-based models such as SDNE Wang et al. (2016) and TLC-GNN Yan et al. (2021), non-negative matrix

factorization-based methods such as M-NMF Wang et al. (2017), and random walk-based methods such as Node2vec Grover & Leskovec (2016). To further capture the potential complex relationships within graphs, FactorGCN proposes a factorizable GCN that improves the representation of node relationships through disentangled modeling Yang et al. (2020). Similarly, TWP introduces a topology-aware weight-preserving strategy, which effectively enhances model robustness by retaining local structural information Liu et al. (2021). Moreover, with the continuous development of graph neural networks, researchers have proposed various model variants by designing more sophisticated aggregation strategies You et al. (2019); Ye et al. (2019); Si et al. (2020). For instance, P-GNN introduces a nonlinear distance-weighted aggregation method that guides feature fusion by computing the spatial distances between nodes and their neighbors You et al. (2019). On the other hand, CurvGN designs a novel aggregation function based on discrete graph curvature, enabling better characterization of complex geometric structures within graphs Ye et al. (2019).

However, directly applying GCN to representation learning on dynamic graphs still faces a challenge. A common limitation is that aggregation functions typically update node representations by averaging the features of neighboring nodes, making it difficult to distinguish the importance of different neighbors. This problem is particularly pronounced in dynamic graph scenarios where node relationships continuously evolve and structural changes occur frequently, severely affecting the model's ability to capture temporal characteristics and long-term dependencies.

2.2 Dynamic Graph Representation Learning

Dynamic GNN models are designed for either discrete or continuous time settings, depending on temporal information to be encoded Zuo et al. (2018); Kumar et al. (2019); Lu et al. (2019); Gao et al. (2022). Discrete-Time Dynamic Graphs (DTDGs) represent dynamic networks as a series of static snapshots sampled at fixed time intervals, while Continuous-Time Dynamic Graphs (CTDGs) focus on modeling interactions between entities at arbitrary timestamps with fine granularity. Both types of dynamic GNNs fundamentally encode graph temporal dynamics into continuously updated node representations, a process often referred to as dynamic node embeddings Tian et al. (2021); Qu et al. (2024); Zhang et al. (2023); Ma et al. (2024), which are directly applicable to downstream tasks such as recommendation Wang et al. (2022) and link prediction Qu et al. (2020).

Discrete-Time Dynamic Graphs. Previous studies on DTDGs have primarily adopted hybrid architectures that combine GNNs with sequence modeling modules such as LSTM, self-attention mechanisms, and Transformers Pareja et al. (2020); Goyal et al. (2020); Sankar et al. (2020). For example, DySAT Sankar et al. (2020) simultaneously employs graph attention networks and self-attention mechanisms to model local structural features and temporal dependencies, respectively. EvolveGCN Pareja et al. (2020) leverages LSTM and GRU to dynamically update the weights of the internal GNN structure. ROLAND You et al. (2022) proposes a general framework that maintains node states across layers and incorporates incremental training and meta-learning strategies to enable efficient extension of static GNNs to dynamic graph tasks. Targeting large-scale discrete-time dynamic graphs, SEIGN adopts parameter-free message passing and a dual-evolution mechanism to model dynamic changes in node representations. DTFormer Chen et al. (2024) introduces a Transformer-based architecture that combines multi-block partitioning and attention mechanisms to capture the graph topology and its dynamic evolution across different timestamps.

Continuous-Time Dynamic Graphs. Research on CTDGs primarily focuses on modeling interaction events that occur at specific timestamps, with the goal of precisely characterizing the fine-grained temporal dependencies exhibited during the structural evolution of graphs over time. The study of CTDGs began with Jodie Kumar et al. (2019); Tian et al. (2024), which dynamically updates node representations using dual RNNs and a projection operator. DyRep Trivedi et al. (2019) further proposes a deep temporal point process model that introduces two distinct time scales to separately model node evolution and communication events, thereby capturing more fine-grained temporal dynamics in continuous time. HTNE Zuo et al. (2018), on the other hand, adopts a Hawkes process to model historical interactions between nodes, capturing the influence of historical neighbors on current ones. Its embedding mechanism demonstrates strong performance across various dynamic network tasks.

However, existing dynamic graph methods still face certain limitations. Most models that rely on sequence modeling architectures exhibit high time complexity and are difficult to parallelize, which hinders the scalability of current dynamic graph models to large-scale graphs.

3 PRELIMINARY

Let $\mathcal{G} = \{G_t\}_{t=1}^{T}$ denote a discrete-time dynamic graph sequence. $\mathcal{V} = \{v_1, \ldots, v_n\}$ is a finite set of nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges. Each snapshot is a static graph $G_t = (V_t, E_t)$, where V_t and E_t are its node set and edge set. They satisfy $\mathcal{V} = \bigcup_{t=1}^{T} V_t$ and $\mathcal{E} = \bigcup_{t=1}^{T} E_t$. The adjacency matrix of a snapshot at time t is denoted by $A_t \in \mathbb{R}^{n \times n}$. If $v_i, v_j \in V_t$ and $(v_i, v_j) \in E_t$, then $(A_t)_{ij} = 1$; otherwise, it is 0. For a weighted snapshot, $w_{ij}^t \geq 0$ denotes the weight of the direct edge from v_i to v_j at time t, with $w_{ij}^t = 0$ when the edge is absent. For an unweighted graph, $w_{ij}^t = (A_t)_{ij}$. When the time step is clear from context, we omit the superscript t on edge weights. A node feature matrix $X_t \in \mathbb{R}^{n \times k}$ is provided at each time step, where k is the feature dimension; for time-invariant features, $X_t = X = \{x_v \mid v \in \mathcal{V}\}$. Similar to many studies, link prediction is an ideal task for evaluating dynamic graph learning. It requires the model to learn a function f that takes the historical graph sequence $G_{1:t-1}$ and node features X as input, and predicts whether an edge exists between a given node pair at time t:

$$f : (G_{1:t-1}, X) \longrightarrow E_t \tag{1}$$

4 METHODOLOGY

We introduce the proposed EGCN model in this section, as the overall framework is presented in Figure 1. The EGCN can be summarized into two key parts. One part is for local structural representation learning at a specific time step, and the other part is for dynamic graph evolution modeling across time steps. For local structural modeling, we first compute a structurally enhanced WRA similarity matrix based on the topology of the current snapshot to capture fine-grained structural correlations between nodes. Subsequently a single-layer GCN takes the constructed similarity matrix as guidance to perform importance-aware feature aggregation and generate the node representation matrix. For dynamic graph evolution modeling, EGCN introduces a MSGO mechanism based on smoothed second-order moments, where an Exponential Moving Average (EMA) of the second-order moments is defined to smooth the gradients at each time step, resulting in local smoothed gradients. Furthermore, a dynamic adjustment factor based on global graph states is incorporated to adaptively adjust the contribution weights of historical snapshots during parameter updates. Through these designs, EGCN effectively captures the long-term structural evolution of dynamic graphs without relying on explicit temporal encoders.

4.1 A STRUCTURALLY ENHANCED WRA SIMILARITY

Measuring the degree of similarity between a target node and its neighboring nodes is critically important for graph-based analysis, which provides information that can determine the impact of different neighboring nodes on the target node. The WRA similarity is a classic function used to

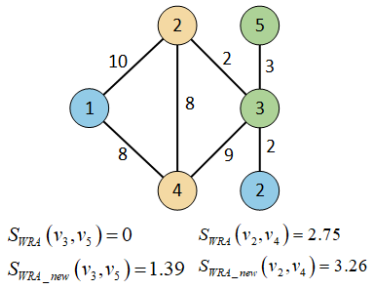

Figure 2: Comparison of the original WRA similarity and the proposed SE-WRA similarity.

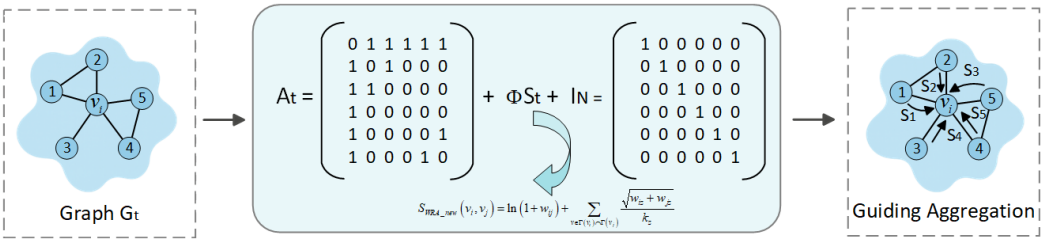

Figure 3: Example to illustrate the proposed aggregation strategy. A graph with six nodes at time step t first generates the corresponding adjacency matrix A_t and the SE-WRA similarity matrix $S^t_{\text{WRA_new}}$. Then, each element in the sum of these two matrices is expressed as the importance between the two nodes, which is used to guide the aggregation according to the numerical value of such importance.

measure the similarity between two nodes because it has advantages in comprehensively considering both edge weight strength and the influence of common neighbors Kumar et al. (2020). Additionally, since the WRA similarity is calculated directly based on the network's topology and edge weight information, it offers relative interpretability while also saving computational costs. Given two nodes v_i and v_j, the WRA similarity is defined as:

$$S_{\text{WRA}}(v_i, v_j) = \sum_{v_z \in \Gamma(v_i) \cap \Gamma(v_j)} \frac{w_{iz} + w_{jz}}{k_z} \tag{2}$$

Where $\Gamma(v_i)$ denotes the set of neighbors of node v_i, w_{iz} and w_{jz} represent the weights of the edges from v_i and v_j to their common neighbor v_z, respectively, and k_z denotes the degree of node v_z. In the following equation, w_{ij} denotes the weight of the direct edge between v_i and v_j; it is 0 when no such edge exists.

According to the above formula, when calculating the similarity between two nodes, the presence or absence of a direct edge between them does not affect the result. However, in real-world networks, the existence of an edge should reflect a higher level of similarity. For instance, two users may both follow the same blogger, but only if they interact with each other can a real connection be established. Moreover, the WRA metric is highly sensitive to edge weights. Extreme edge weights can excessively amplify the influence of certain common neighbors, leading to unstable similarity values. For example, v_3 and v_5 are directly connected but share no common neighbors in Figure 2, so $S_{\text{WRA}}(v_3, v_5) = 0$. In contrast, nodes v_2 and v_4 have no direct edge but share multiple strong common neighbors, resulting in $S_{\text{WRA}}(v_2, v_4) = 2.75$. This fails to reflect the absence of a true connection and falsely implies high similarity, which is undesirable in realistic scenarios.

Therefore, we propose a new WRA similarity measure to address the issue where the presence of an edge between two nodes has little positive impact on similarity calculation and the method is overly sensitive to extreme edge weights. Specifically, as shown in equation (3), we introduce a logarithmic enhancement term for the direct edge weights and integrate the structural interaction term of edge weights on both sides of the common neighbors. As shown in Figure 2, the similarity between nodes v_3 and v_5, as well as between nodes v_2 and v_4, can be recalculated and corrected as $S_{\text{WRA_new}}(v_3, v_5) = 1.39$ and $S_{\text{WRA_new}}(v_2, v_4) = 2.36$, respectively.

$$S_{\text{WRA_new}}(v_i, v_j) = \ln(1 + w_{ij}) + \sum_{v_z \in \Gamma(v_i) \cap \Gamma(v_j)} \frac{\sqrt{w_{iz} + w_{jz}}}{k_z} \tag{3}$$

4.2 Node Similarity-Aware Aggregation

To enhance the structural awareness of graph neural networks during the neighborhood aggregation process, we integrate the proposed structural similarity $S_{\text{WRA_new}}$ into the propagation mechanism of GCN. This similarity matrix considers both the direct connection strength between nodes and the structural interactions of edge weights on both sides of their common neighbors. By normalizing through the degree of intermediary nodes, it effectively models the strength of associations in the local structure.

At a specific time step t, we propose a structural-aware graph representation encoder to generate the node representation matrix $H_t \in \mathbb{R}^{n\times d}$. Considering the computational efficiency of the model, we adopt a design based on a single-layer GCN. The propagation function of this encoder is defined as follows:

$$H_t = \sigma_{\text{gcn}}\left(\hat{D}_t^{-1/2}\hat{S}_t\hat{D}_t^{-1/2}X_tW_t\right) \tag{4}$$

Where $\hat{S}_t = A_t + \phi S_t + I_n$, $S_t \in \mathbb{R}^{n\times n}$ is the similarity matrix constructed by calculating the SE-WRA similarity on G_t, $I_n \in \mathbb{R}^{n\times n}$ is the identity matrix, and $\phi \in [0, 1]$ controls the contribution of S_t. The diagonal degree matrix $\hat{D}_t \in \mathbb{R}^{n\times n}$ is defined by $(\hat{D}_t)_{ii} = \sum_{j=1}^{n}(\hat{S}_t)_{ij}$. Here, $X_t \in \mathbb{R}^{n\times k}$ is the node feature matrix, $W_t \in \mathbb{R}^{k\times d}$ is the learnable linear transformation weight matrix updated by MSGO, and $\sigma_{\text{gcn}}(\cdot)$ is the ReLU activation function.

As shown in Figure 3, different from traditional GCN-based methods that use $A_t + I_n$ to guide the aggregation of node features, we add the new WRA similarity as an additional item to further guide the aggregation. Under the new aggregation strategy, a target node can identify the importance of different neighboring nodes toward it and further utilize such importance to guide the update of its features. Also, we introduce a parameter ϕ to control the contribution of the new WRA similarity.

4.3 Multi-Snapshot Gradient Optimization Based on Smoothed Second-Order Moments

To eliminate the reliance on heavy temporal encoders while efficiently capturing long-term structural evolution features in dynamic networks, we propose a MSGO method based on smoothed second-order moments. This method is built upon single-snapshot feature learning, and adaptively integrates information from the current and historical snapshots to achieve parameter updates across time steps. Specifically, as shown in Figure 4, the framework clearly illustrates the loss gradient propagation process in EGCN. By incorporating EMA gradient smoothing and adaptive gradient aggregation, this framework effectively integrates information from different time steps, aiding in model optimization and capturing the long-term structural evolution in dynamic graphs. Formally, for each time step G_t, EGCN first performs GCN feature aggregation based on the Structurally Enhanced WRA similarity to obtain the node representation matrix H_t.

Let $\mathcal{P}_t = \mathcal{E}_t^+ \cup \mathcal{E}_t^-$ be the sampled set of candidate node pairs at time t, containing positive and negative pairs. Let $Y^t = \{y_{u,v}^t \mid (u, v) \in \mathcal{P}_t\}$ be their binary labels, where $y_{u,v}^t = 1$ if $(u, v) \in E_t$ and 0 otherwise. Since we use a single-layer GCN, $\boldsymbol{h}_{u,t} = H_t[u, :] \in \mathbb{R}^d$ denotes the representation of node u at time t. The MLP takes the concatenation of two node representations as input and predicts the probability that an edge from node u to node v exists:

$$\hat{y}_{u,v}^t = \text{MLP}\left(\boldsymbol{h}_{u,t} \| \boldsymbol{h}_{v,t}\right) \in (0, 1) \tag{5}$$

where $\|$ denotes the concatenation operation. Based on the label data Y^t, the training loss is the binary cross-entropy loss:

$$\mathcal{L}_t = -\sum_{(u,v)\in\mathcal{P}_t}\left[y_{u,v}^t \log \hat{y}_{u,v}^t + \left(1 - y_{u,v}^t\right)\log\left(1 - \hat{y}_{u,v}^t\right)\right] \tag{6}$$

where $\mathcal{L}_t$ denotes the training loss at snapshot G_t. Note that such loss can also be replaced by other loss functions depending on the downstream tasks. At each training step, θ_t denotes all trainable parameters at time t, including W_t and the MLP parameters, and the frame-wise loss gradient is formulated as follows:

$$\nabla\mathcal{L}_t = \frac{\partial\mathcal{L}_t}{\partial\theta_t} \tag{7}$$

EMA Gradient Smoothing. In the process of dynamic graph learning, significant fluctuations often occur in the standard gradient $\nabla\mathcal{L}_t$ due to rapid structural changes between snapshots. Traditional first-order EMA methods can provide a certain degree of gradient smoothing, but they struggle to effectively capture dynamic variations. Introducing third-order or higher-order moments can improve sensitivity to short-term changes; however, it often results in numerical instability and increased computational overhead. To achieve a balance between stability and responsiveness, we formally define a novel EMA mechanism that incorporates second-order moment estimation. This

mechanism not only effectively smooths gradient oscillations but also avoids the issues caused by higher-order moment estimation.

For time step t, the update formula of the smoothed second-order EMA is defined as:

$$s_t = \alpha s_{t-1} + (1-\alpha)(\nabla \mathcal{L}_t)^2 \tag{8}$$

where $\alpha \in [0,1]$ is the smoothing coefficient used to balance the influence between historical and current information, s_t is the accumulated second-order moment estimate, and $s_0 = \mathbf{0}$. The square is taken element-wise. Based on the resulting smoothed tensor s_t, the formula for computing the local smoothed gradient is given by:

$$\nabla \mathcal{L}_t^{ls} = \frac{\nabla \mathcal{L}_t}{\sqrt{s_t} + \epsilon} \tag{9}$$

Where $\epsilon > 0$ is a small positive constant (typically set to 10^{-8}) used to prevent division by zero. The square root and division are element-wise. This gradient is then utilized to update the model parameters in the next snapshot.

Adaptive gradient aggregation. Relying solely on the current snapshot for gradient optimization may cause an underestimation of long term evolution trends. To address this issue, we propose an adaptive gradient aggregation strategy based on the global graph representation. This mechanism dynamically adjusts the weights of gradients at different time steps according to the structural similarity between historical and current snapshots, thereby effectively enhancing the integration of temporal information.

Specifically, we define the global graph representation $g_t \in \mathbb{R}^d$ for each snapshot based on the node representation matrix $H_t \in \mathbb{R}^{n \times d}$. The global graph representation is computed using average pooling followed by a nonlinear transformation:

$$g_t = \mathcal{R}(H_t) = \sigma_{\text{pool}}\left(\frac{1}{n}\sum_{i=1}^{n} h_i^t\right) \tag{10}$$

where $\mathcal{R}(\cdot)$ denotes the graph-level pooling function, h_i^t denotes the representation of node i at time step t, and $\sigma_{\text{pool}}(\cdot)$ is a nonlinear activation function, such as sigmoid or tanh. For $t > 1$, we define the discrepancy between the current and a historical global graph representation as the cosine distance $D(g_t, g_j) = 1 - \frac{g_t^\top g_j}{\|g_t\|_2 \|g_j\|_2}$. We then design a dynamic adjustment factor γ_j to measure the contribution of the local gradient at historical time step j to the current time step t. The computation is as follows:

$$\gamma_j = \frac{\exp\left(-D(g_t, g_j)\right)}{\sum_{k=1}^{t-1} \exp\left(-D(g_t, g_k)\right)} \tag{11}$$

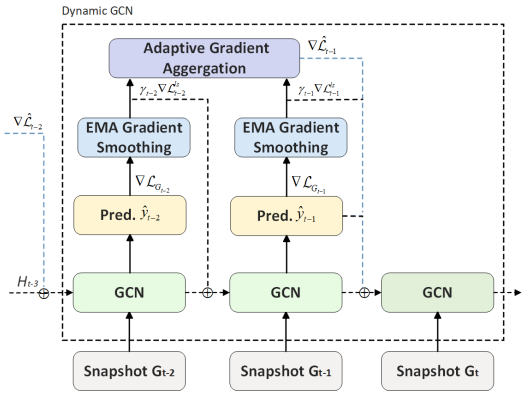

Figure 4: The loss gradient propagation process in EGCN. The black dashed arrows and blue dashed arrows represent frame-wise gradient propagation and adaptive loss gradient propagation, respectively.

Table 1: Overall performance comparison on six datasets. The best results are highlighted in bold. Note that $*$ OOM denotes the out-of-memory error when we tried to run the model on our environment.

Dataset	Metric	EvolveGCN-H	EvolveGCN-O	DGNN	dyngraph2vec	ROLAND	DTFormer	WinGNN	EGCN
Bitcoin-Alpha	Accuracy	50.29±0.25	56.46±0.41	OOM	OOM	67.13±2.16	81.12±0.63	82.16±0.47	**90.32±0.25**
	AUC	63.71±1.03	68.93±0.91	OOM	OOM	90.21±1.18	87.23±1.23	**90.63±0.33**	89.61±0.49
	MRR	3.28±0.34	2.52±0.18	OOM	OOM	13.88±0.46	32.64±2.44	36.74±3.94	**39.62±2.56**
Bitcoin-OTC	Accuracy	49.28±0.03	51.62±2.31	53.63±0.68	58.29±1.55	86.60±0.52	86.42±1.26	87.14±2.39	**91.39±1.17**
	AUC	55.38±1.62	59.82±2.57	59.13±6.49	62.12±9.75	90.77±1.30	86.65±0.7618	88.64±0.62	**92.23±0.74**
	MRR	11.27±0.58	11.31±0.50	16.03±0.26	36.35±1.48	16.54±1.22	32.62±1.6214	37.88±2.16	**39.66±1.58**
DBLP	Accuracy	64.12±0.41	65.24±0.53	66.12±0.33	64.36±0.42	64.31±0.49	62.13±0.65	68.43±0.41	**86.71±0.51**
	AUC	70.19±0.38	72.64±0.47	73.26±0.44	75.84±0.33	77.79±0.17	72.96±0.26	77.87±0.31	**83.43±0.54**
	MRR	2.65±0.01	2.48±0.01	2.62±0.01	2.71±0.01	6.60±0.01	7.16±0.01	8.63±0.03	**12.31±0.02**
Reddit-title	Accuracy	85.85±0.02	77.46±1.27	OOM	OOM	93.42±0.07	91.25±0.69	97.25±0.01	**97.56±0.01**
	AUC	93.87±0.01	97.17±0.52	OOM	OOM	97.90±0.01	89.55±0.24	**97.97±0.01**	94.73±0.02
	MRR	3.28±0.02	1.31±0.02	OOM	OOM	35.11±0.09	26.18±0.17	**31.96±0.19**	29.62±0.11
Stack Overflow	Accuracy	OOM	OOM	OOM	OOM	64.61±1.91	88.42±0.16	94.82±0.13	**95.31±0.16**
	AUC	OOM	OOM	OOM	OOM	74.39±1.16	81.56±0.27	92.63±0.02	**96.12±0.02**
	MRR	OOM	OOM	OOM	OOM	27.55±1.88	36.65±3.43	42.11±1.68	**50.22±1.44**
UCI	Accuracy	59.85±2.54	49.91±1.45	50.91±0.14	50.88±3.16	81.83±0.64	80.32±1.69	89.27±1.16	**93.27±1.24**
	AUC	71.56±1.83	65.03±3.81	52.11±0.56	52.10±0.15	94.05±1.07	82.14±2.86	89.17±0.52	**92.14±0.60**
	MRR	8.17±0.43	10.81±0.53	1.52±0.01	17.84±1.12	11.84±0.26	18.68±0.32	**38.64±0.34**	32.98±0.29

Table 2: The basic statistics of six datasets in the experiment.

Dataset	# Nodes	# Edges	# Snapshots	Avg. Density
Bitcoin-Alpha	3,783	24,186	226	2.5890×10^{-3}
Bitcoin-OTC	5,881	35,592	262	1.7396×10^{-4}
DBLP	28,086	162,451	27	9.5423×10^{-5}
Reddit-title	54,075	571,927	178	1.9592×10^{-3}
Stack Overflow	2,601,977	63,497,050	92	2.5503×10^{-6}
UCI	1,899	59,835	28	1.1191×10^{-3}

where $D(g_t, g_j)$ is the cosine distance defined above. The globally adaptive gradient is given by:

$$\nabla\hat{\mathcal{L}}_t = \omega \cdot \nabla\mathcal{L}_t^{ls} + (1-\omega) \cdot \sum_{j=1}^{t-1} \gamma_j \cdot \nabla\mathcal{L}_j^{ls} \tag{12}$$

where $\omega \in [0, 1]$ balances the contribution of the current snapshot's gradient and the aggregated gradients from historical snapshots, and γ_j is the dynamic adjustment factor that adaptively adjusts the influence of historical time step j based on its global structural similarity to the current snapshot. At $t = 1$, we use only the local smoothed gradient, i.e., $\nabla\hat{\mathcal{L}}_1 = \nabla\mathcal{L}_1^{ls}$.

Parameter Update. Finally, based on the aggregated gradient, the model parameters are updated according to the following rule:

$$\theta_t = \theta_{t-1} - \beta\nabla\hat{\mathcal{L}}_t \tag{13}$$

where β is the learning rate that controls the magnitude of the parameter update.

Then the parameters of the EGCN are iteratively updated with time progression. Finally, the learned node representations are used to perform dynamic link prediction through an additional MLP layer during the inference phase.

4.4 Complexity Analysis

The time complexity of EGCN is significantly lower than that of existing dynamic graph representation learning methods that rely on explicit temporal sequence modeling. Overall, it achieves a more favorable computational scale while maintaining strong modeling capacity, thus offering higher training efficiency and scalability, especially for large-scale dynamic graph construction tasks.

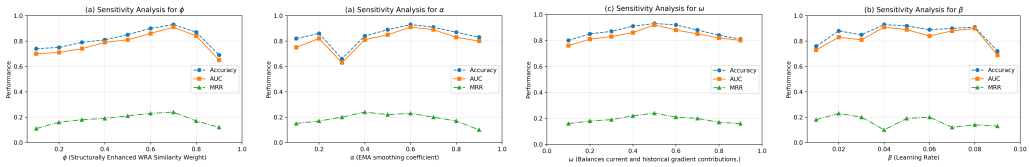

Figure 5: Performance of EGCN on UCI dataset with different hyper-parameter settings, including the SE-WRA similarity weight, EMA smoothing coefficient, balances current and historical gradient contributions, learning Rate.

For analytical purposes, let the number of nodes be n, the number of edges at each time step be $|E_t|$, the node feature dimension be d, and the average degree be d_{avg}. Let t denote the current training time step and T the maximum length of the historical time window. In addition, we define the following notation for subsequent analysis: L is the number of GCN or Transformer layers, h denotes the number of Transformer attention heads, λ_m is the number of input segments per node in DTFormer, and l is the length of each segment. Next, we examine the most recent setting and analyze the primary computational costs during single-step training. First, in the similarity construction stage, computing the similarity matrix based on structural sensitivity and the SWRA_new similarity requires a complexity of $\mathcal{O}(n \cdot d_{\text{avg}}^2)$. Then, GCN encoding is performed over a sparsely connected graph structure, with complexity $\mathcal{O}(|E_t| \cdot d)$; the projection of nonlinear state representations into global dynamic latent space requires $\mathcal{O}(n \cdot d)$. In the optimization phase, we introduce a two-level EMA gradient smoothing mechanism, with computational cost $\mathcal{O}(d)$, and use global variance-based dynamic adjustment for historical EMA gradient aggregation, also with cost $\mathcal{O}(d)$. Therefore, the overall computational complexity of EGCN at time step t is $\mathcal{O}(|E_t| \cdot d + n \cdot d_{\text{avg}}^2 + t \cdot d)$. The space complexity is mainly determined by the storage of historical representations and gradient states, given by:$\mathcal{O}(t \cdot n \cdot d)$. In comparison, the per-step complexity of ROLAND is $\mathcal{O}(L \cdot |E_t| \cdot d \cdot n \cdot d^2)$, where L is the number of neural network layers. This method relies on GRU or MLP to explicitly update embedded states, while DTFormer uses multi-path attention to encode features and multiple Transformer layers. Its complexity is $\mathcal{O}(L \cdot h \cdot \lambda_m^2 \cdot d)$, where h is the number of attention heads and λ_m is the segment length of each node's local temporal sequence, which results in significantly higher computational cost when dealing with long sequences or high-dimensional temporal modeling. In contrast, EGCN achieves a more lightweight design with lower training overhead and better scalability.

5 Experiments

5.1 Experimental Settings

Datasets. We perform the experiment on six typical public datasets which are widely evaluated by existing dynamic graph representation learning studies. All of these datasets can be obtained from the SNAP website (https://snap.stanford.edu/). The datasets are selected with different edge densities and distributions, including Bitcoin-Alpha, Bitcoin-OTC, DBLP, Reddit-title, Stack Overflow, UCI and Elliptic. The Stack Overflow dataset includes a very large dynamic graph with more than 60 million edges, so the efficiency and performance on it can better reflect the application potential of the model. To make a fair comparison of reproducible results, we adopt the standard partition of snapshots generated by GraphGym You et al. (2020). We show the basic statistics of these six datasets in Table 2.

Evaluation Metrics. We apply three commonly-used metrics, namely accuracy, macro-AUC(AUC), MRR to testify the superiority of EGCN against other baselines.

- **Accuracy** indicates the basic classification ability of the model by predicting whether there is a link (i.e., edge) between the two given nodes. Let $E'_{\pm}$ denote the set of positive and negative candidate edges in the test set, $T(E'_{\pm})$ the ground-truth labels for these candidates, and $R(E'_{\pm})$ their predicted labels. The accuracy is defined as:

$$\text{Accuracy} = \frac{|T(E'_{\pm}) \cap R(E'_{\pm})|}{|T(E'_{\pm})|}, \tag{14}$$

Table 3: Ablation performance of EGCN on UCI dataset.

Model	Accuracy (%)	AUC (%)	MRR (%)
EGCN	93.27±1.24	92.14±0.60	32.98±0.29
-SE-WRA	74.14±1.37	72.62±1.19	18.74±0.43
-S.G	82.42±1.58	79.89±1.02	17.67±0.37
-A.G	72.33±1.49	71.11±1.28	13.41±0.41

where $T(E'_{\pm})$ denotes all ground-truth labels and $R(E'_{\pm})$ the corresponding predicted labels.

- **Area Under Curve (AUC)** uses the area of the receiver operating characteristic curve (ROC) to evaluate the classification quality.
- **Mean Reciprocal Rank (MRR)** is a measure of predicting list which counts reciprocal ranks of the correctly predicted node in the whole list. For N test queries, let $\mathrm{Rank}(q)$ be the rank of the correct node for query q, and let K be the evaluation cutoff. The MRR is defined as:

$$\mathrm{MRR}\,@K = \frac{1}{N}\sum_{q=1}^{N}\frac{\mathbb{I}[\mathrm{Rank}(q) \leq K]}{\mathrm{Rank}(q)}, \tag{15}$$

where $\mathbb{I}[\cdot]$ is the indicator function. The rank is determined by ordering candidate nodes according to predicted logit scores.

Baselines. We evaluate our proposed EGCN by comparing the performance with various dynamic GNN baselines, including EvolveGCN Pareja et al. (2020), DGNN Manessi et al. (2020), dyngraph2vec Goyal et al. (2020), ROLAND You et al. (2022), DTFormer Chen et al. (2024) and WinGNN Zhu et al. (2023).

- **EvolveGCN** employs RNNs to evolve graph convolutional network parameters.
- **DGNN** is a dynamic graph neural network that learns node embeddings by combining temporal information and graph structures.
- **dyngraph2vec** leverages autoencoders to capture the changes in node embeddings in dynamic graphs.
- **ROLAND** updates node embeddings in dynamic networks using a meta-learning framework for real-time updates.
- **DTFormer** employs the Transformer model, incorporating the modeling of intersection relationships between nodes and a multi-patching module to enhance the expressive power of the model.
- **WinGNN** learns dynamic graphs efficiently through a random sliding window mechanism and multi-snapshot gradient optimization.

Implementation Details. We perform our comparsions on a Ubuntu 20.04.2 LTS server with AMD EPYC 7642 48-Core Processor, 1008GB RAM, and an NVIDIA A100 Tensor Core GPU. EGCN by using the PyTorch 1.9.1 and DGL 0.9.1 framework. We also utilize the modules for data processing as well as evaluation from GraphGym. There are four major hyper-parameters in the EGCN framework, which are: weight ϕ of the SE-WRA similarity, EMA smoothing factor α, the gradient aggregation weight ω and Learning Rate β. We derive the optimal performance by searching different parameter setting as follows: $\phi \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$; $\alpha \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$; $\omega \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$; $\beta \in \{0.02, 0.04, 0.06, 0.08, 0.1\}$. Additionally, the Adam optimizer Kingma & Ba (2014) is used as the optimizer for gradient descent and the Xavier Glorot & Bengio (2010) is utilized as the basic initializer for each parameter matrix.

5.2 Future Link Prediction

In the link prediction task, we comprehensively evaluate the performance of various dynamic graph neural network models across different types of datasets using three metrics: Accuracy, AUC, and

MRR. For the baseline models where applicable, some datasets have been previously addressed, we follow the settings and results from these prior studies. We use historical snapshot data as training data and predict the existence of edges in future snapshots. During training, data from later snapshots could serve as a self-supervision signal for earlier snapshots. Ultimately, we select the model that performed best on the validation set for testing and predict future links on the test set.

As summarized in Table 1. The experimental results indicate that several existing baseline models, such as EvolveGCN, DGNN, and dyngraph2vec, exhibit significant performance variability across dynamic graph datasets with differing density distributions. This phenomenon is particularly pronounced in the Bitcoin-Alpha and DBLP datasets, suggesting that these models struggle to effectively capture spatiotemporal patterns in highly dynamic or structurally sparse graphs. In contrast, the proposed EGCN model demonstrates remarkably consistent and superior performance across all six datasets, achieving the best results in 14 out of 18 evaluations across the Accuracy, AUC, and MRR metrics. Specifically, on the Bitcoin-Alpha dataset, EGCN surpasses the previous best-performing model by 8.16% in Accuracy, and achieves improvements of 2.88% in MRR, respectively, highlighting its strong capability in modeling the evolution of sparse social networks. Furthermore, on more structurally complex datasets such as DBLP, EGCN outperforms the second-best model by over 18% in Accuracy, indicating its robust ability to discern link existence between node pairs. Additionally, on large-scale graphs like Reddit-title and Stack Overflow, EGCN maintains efficient training under resource-constrained environments, achieving an Accuracy of 95.31%, significantly outperforming other models; whereas methods like EvolveGCN and DGNN fail to complete training due to excessive resource consumption.

It is worth mentioning that when evaluating the performance on large dynamic graphs (i.e., Stack Overflow), we found that EGCN's training process exhibits no significant fluctuations and the results remain stable across multiple runs. This further validates the parameter efficiency and generalization capability of its node similarity-aware aggregation mechanism and encoder-free framework. Thus, taking the above aspects together, our observations conclude that EGCN demonstrates superior performance in link prediction tasks compared to existing baseline models.

5.3 Exploration of EGCN

After verifying the effectiveness of the proposed EGCN, we perform the exploration study to figure out the necessity of its various components and parameter settings on the performance. Due to space limitation, we selected UCI as the standard dataset, and conduct analysis from two aspects: ablation experiment and hyper-parameter setting.

Model ablation study. As presented in Table 3, we explored the contribution of three key components in EGCN to the final performance. In particular, "-SE-WRA" denotes the model without Structurally Enhanced WRA Similarity, where feature aggregation relies solely on basic adjacency information. "-S.G." refers to the model variant that removes the EMA gradient smoothing mechanism based on second-order moments, using raw gradients for parameter updates instead. The "-A.G." variant completely discards the adaptive gradient aggregation across historical snapshots, relying only on the current snapshot's gradient and the previous model parameters for updating.

The results demonstrate that the adaptive gradient aggregation mechanism contributes significantly to overall EGCN's performance. When this module is removed, the model's accuracy and AUC drop by nearly 20%, and the MRR metric decreases by over 10%. This indicates that without explicitly modeling global structural correlations across historical snapshots, EGCN struggles to capture the evolving patterns of dynamic graphs, leading to instability and reduced generalization in node representation. The removal of the SE-WRA similarity module also leads to significant performance degradation, with the model's accuracy dropping by approximately 19% and the AUC declining by nearly 20%, clearly highlighting the critical role of structural similarity modeling in guiding the selection of important neighbors. Furthermore, when the second-moment-based EMA gradient smoothing mechanism is removed, there is a slight performance decline in both accuracy and AUC, suggesting the robustness introduced by this component. It plays a vital role in alleviating gradient oscillations during cross-snapshot training and in stabilizing the model's convergence process. Overall, these components do not function in isolation but rather work collaboratively under dynamic conditions. The SE-WRA module enhances local structural awareness, the second-moment-based EMA gradient smoothing improves the adaptability of the learning process, and the

adaptive gradient aggregation introduces critical global evolution modeling capabilities. Together, these components form an efficient, stable, and generalizable framework for dynamic graph learning.

Hyper-parameter sensitivity analysis. In the EGCN model, there are four major hyper-parameters which are the weight ϕ of the SE-WRA similarity, the EMA smoothing coefficient α, the gradient aggregation weight ω that balances the contributions of current and historical gradients, and the learning rate β. To verify the degree to which the model is affected by these four hyper-parameters, different settings are used to observe changes in performance, and the result is shown in Figure 5.

As illustrated in Figure 5-(a), the parameter ϕ controls the degree to which the SE-WRA similarity participates in the feature aggregation process. We observe that as ϕ increases from 0.1 to 0.7, the model's Accuracy, AUC, and MRR improve steadily, indicating that the similarity matrix plays a valuable role in capturing local topological patterns. However, when ϕ exceeds 0.7, the performance begins to degrade, suggesting that over-reliance on the similarity term may introduce noise. This finding is consistent with the results from the ablation study, which showed that removing the SE-WRA similarity module leads to a significant drop in model performance.

The EMA smoothing coefficient α governs the stability of gradient updates across snapshots. As shown in Figure 5-(b), the model achieves optimal performance when α is set to 0.6. As α increases further, critical gradient signals may be overly smoothed out, resulting in performance degradation.

Figure 5-(c) presents the effect of the gradient aggregation weight ω, which balances the influence of gradients from the current snapshot and aggregated historical snapshots. The model performs best when ω reaches 0.5, which means that maintaining a balance between short-term reactivity and long-term consistency is crucial. A small ω makes the model less responsive to current structural changes, while a large ω may impair generalization due to insufficient use of historical information.

Figure 5-(d) analyzes the impact of the learning rate β on training. The results show that the model reaches a good trade-off between convergence speed and final performance when β is set to 0.04. A larger learning rate may lead to unstable training or oscillations, whereas a smaller learning rate slows down convergence and negatively affects performance.

Although EGCN introduces four hyper-parameters, the model demonstrates strong stability across different parameter configurations. Satisfactory performance can be achieved through moderate adjustments without requiring complex hyper-parameter tuning.

6 Conclusion

This work establishes a bridge between dynamic graph representation learning and temporal data management, addressing the critical need for efficient, scalable, and query-aware learning over evolving graph data—a problem that lies at the intersection of graph neural networks and database systems. We propose a novel representation learning framework for DTDGs, aiming to achieve efficient and accurate dynamic link prediction without relying on complex temporal encoders. EGCN innovatively enhances the capability of perceiving the importance of neighboring nodes by introducing a SE-WRA similarity, thereby improving the expressiveness of local feature aggregation. Meanwhile, we design a multi-snapshot gradient optimization based on smoothed second-order moments to model the dynamic properties of graph structures over time. By smoothing historical gradients and incorporating dynamic adjustment mechanism based on global graph structuraly information, EGCN adaptively updates the weight parameters at each time step, effectively capturing long-term temporal dependencies and improving model stability and efficiency. Extensive experiments conducted on six public datasets demonstrate that the proposed model achieves superior performance and scalability compared to existing methods. Further ablation studies validate the critical contributions of each core module to performance improvement and training stability, highlighting the broad application potential of EGCN in the field of dynamic graph learning.

AI use statement

Generative AI was used to assist with formatting this manuscript for ICLR 2027. The authors reviewed the resulting changes and take responsibility for the final content of this work.

REFERENCES

Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. End-to-end attention-based large vocabulary speech recognition. In *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 4945–4949. IEEE, 2016.

Xi Chen, Yun Xiong, Siwei Zhang, Jiawei Zhang, Yao Zhang, Shiyang Zhou, Xixi Wu, Mingyang Zhang, Tengfei Liu, and Weiqiang Wang. Dtformer: A transformer-based method for discrete-time dynamic graph representation learning. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pp. 301–311, 2024.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016a.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016b.

Chao Gao, Junyou Zhu, Fan Zhang, Zhen Wang, and Xuelong Li. A novel representation learning for dynamic graphs based on graph convolutional networks. *IEEE Transactions on Cybernetics*, 53(6):3599–3612, 2022.

Hongchang Gao, Jian Pei, and Heng Huang. Conditional random field enhanced graph convolutional neural networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 276–284, 2019.

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

Palash Goyal, Sujit Rokka Chhetri, and Arquimedes Canedo. dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. *Knowledge-Based Systems*, 187:104816, 2020.

Aditya Grover and Jure Leskovec. 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.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. Representation learning for dynamic graphs: A survey. *Journal of Machine Learning Research*, 21(70):1–73, 2020.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Ajay Kumar, Shashank Sheshar Singh, Kuldeep Singh, and Bhaskar Biswas. Link prediction techniques, applications, and performance: A survey. *Physica A: Statistical Mechanics and its Applications*, 553:124289, 2020.

Srijan Kumar, Xikun Zhang, and Jure Leskovec. 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.

Ron Levie, Federico Monti, Xavier Bresson, and Michael M Bronstein. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing*, 67(1):97–109, 2018.

Haoyang Li and Lei Chen. Cache-based gnn system for dynamic graphs. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pp. 937–946, 2021.

Huihui Liu, Yiding Yang, and Xinchao Wang. Overcoming catastrophic forgetting in graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 8653–8661, 2021.

Qingqing Long, Yilun Jin, Guojie Song, Yi Li, and Wei Lin. Graph structural-topic neural network. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 1065–1073, 2020.

Yuanfu Lu, Xiao Wang, Chuan Shi, Philip S Yu, and Yanfang Ye. Temporal network embedding with micro-and macro-dynamics. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pp. 469–478, 2019.

Ying Ma, Haijie Lou, Ming Yan, Fanghui Sun, and Guoqi Li. Spatio-temporal fusion graph convolutional network for traffic flow forecasting. *Information Fusion*, 104:102196, 2024.

Franco Manessi, Alessandro Rozza, and Mario Manzo. Dynamic graph convolutional networks. *Pattern Recognition*, 97:107000, 2020.

Behnaz Moradabadi and Mohammad Reza Meybodi. Link prediction in weighted social networks using learning automata. *Engineering Applications of Artificial Intelligence*, 70:16–24, 2018.

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

Liang Qu, Huaisheng Zhu, Qiqi Duan, and Yuhui Shi. Continuous-time link prediction via temporal dependent graph neural network. In *Proceedings of the web conference 2020*, pp. 3026–3032, 2020.

Yunke Qu, Liang Qu, Tong Chen, Xiangyu Zhao, Quoc Viet Hung Nguyen, and Hongzhi Yin. Scalable dynamic embedding size search for streaming recommendation. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pp. 1941–1950, 2024.

Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang. 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.

Chenyang Si, Ya Jing, Wei Wang, Liang Wang, and Tieniu Tan. Skeleton-based action recognition with hierarchical spatial reasoning and temporal stack learning network. *Pattern Recognition*, 107:107511, 2020.

Joakim Skarding, Bogdan Gabrys, and Katarzyna Musial. Foundations and modeling of dynamic networks using dynamic graph neural networks: A survey. *iEEE Access*, 9:79143–79168, 2021.

Sheng Tian, Ruofan Wu, Leilei Shi, Liang Zhu, and Tao Xiong. Self-supervised representation learning on dynamic graphs. In *Proceedings of the 30th ACM international conference on information & knowledge management*, pp. 1814–1823, 2021.

Yuxing Tian, Yiyan Qi, and Fan Guo. Freedyg: Frequency enhanced continuous-time dynamic graph model for link prediction. In *The Twelfth International Conference on Learning Representations*, 2024.

Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. Dyrep: Learning representations over dynamic graphs. In *International conference on learning representations*, 2019.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1225–1234, 2016.

Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. Community preserving network embedding. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

Xiyuan Wang and Muhan Zhang. How powerful are spectral graph neural networks. In *International conference on machine learning*, pp. 23341–23362. PMLR, 2022.

Yifan Wang, Yifang Qin, Fang Sun, Bo Zhang, Xuyang Hou, Ke Hu, Jia Cheng, Jun Lei, and Ming Zhang. Disenctr: Dynamic graph-based disentangled representation for click-through rate prediction. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*, pp. 2314–2318, 2022.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

Yu Xie, Maoguo Gong, Shanfeng Wang, and Bin Yu. Community discovery in networks with deep sparse filtering. *Pattern Recognition*, 81:50–59, 2018.

Zuoyu Yan, Tengfei Ma, Liangcai Gao, Zhi Tang, and Chao Chen. Link prediction with persistent homology: An interactive view. In *International conference on machine learning*, pp. 11659–11669. PMLR, 2021.

Cheng Yang, Chunchen Wang, Yuanfu Lu, Xumeng Gong, Chuan Shi, Wei Wang, and Xu Zhang. Few-shot link prediction in dynamic networks. In *Proceedings of the fifteenth ACM international conference on web search and data mining*, pp. 1245–1255, 2022.

Yiding Yang, Zunlei Feng, Mingli Song, and Xinchao Wang. Factorizable graph convolutional networks. *Advances in Neural Information Processing Systems*, 33:20286–20296, 2020.

Ze Ye, Kin Sum Liu, Tengfei Ma, Jie Gao, and Chao Chen. Curvature graph network. In *International conference on learning representations*, 2019.

Jiaxuan You, Rex Ying, and Jure Leskovec. Position-aware graph neural networks. In *International conference on machine learning*, pp. 7134–7143. PMLR, 2019.

Jiaxuan You, Zhitao Ying, and Jure Leskovec. Design space for graph neural networks. *Advances in Neural Information Processing Systems*, 33:17009–17021, 2020.

Jiaxuan You, Tianyu Du, and Jure Leskovec. Roland: graph learning framework for dynamic graphs. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, pp. 2358–2366, 2022.

Kaike Zhang, Qi Cao, Gaolin Fang, Bingbing Xu, Hongjian Zou, Huawei Shen, and Xueqi Cheng. Dyted: Disentangled representation learning for discrete-time dynamic graph. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 3309–3320, 2023.

Jing Zhao, Lili Miao, Jian Yang, Haiyang Fang, Qian-Ming Zhang, Min Nie, Petter Holme, and Tao Zhou. Prediction of links and weights in networks by reliable routes. *Scientific reports*, 5(1): 12261, 2015.

Yifan Zhu, Fangpeng Cong, Dan Zhang, Wenwen Gong, Qika Lin, Wenzheng Feng, Yuxiao Dong, and Jie Tang. Wingnn: dynamic graph neural networks with random gradient aggregation window. In *Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining*, pp. 3650–3662, 2023.

Yuan Zuo, Guannan Liu, Hao Lin, Jia Guo, Xiaoqian Hu, and Junjie Wu. Embedding temporal network via neighborhood formation. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2857–2866, 2018.