MemMap: An Adaptive and Latent Memory Structure for Dynamic Graph Learning

Shuo Ji
jishuo@buaa.edu.cn
CCSE Lab, Beihang University
Beijing, China

Mingzhe Liu
mzliu1997@buaa.edu.cn
CCSE Lab, Beihang University
Beijing, China

Leilei Sun*†
leileisun@buaa.edu.cn
CCSE Lab, Beihang University
Beijing, China

Chuanren Liu
cliu89@utk.edu
The University of Tennessee
Knoxville, USA

Tongyu Zhu†
zhutongyu@buaa.edu.cn
CCSE Lab, Beihang University
Beijing, China

ABSTRACT

Dynamic graph learning has attracted much attention in recent years due to the fact that most of the real-world graphs are dynamic and evolutionary. As a result, many dynamic learning methods have been proposed to cope with the changes of node states over time. Among these studies, a critical issue is how to update the representations of nodes when new temporal events are observed. In this paper, we provide a novel memory structure - Memory Map (MemMap) for this problem. MemMap is an adaptive and evolutionary latent memory space, where each cell corresponds to an evolving "topic" of the dynamic graph. Moreover, the representation of a node is generated from its semantically correlated memory cells, rather than linked neighbors of the node. We have conducted experiments on real-world datasets and compared our method with the SOTA ones. It can be concluded that: 1) By constructing an adaptive and evolving memory structure during the dynamic learning process, our method can capture the dynamic graph changes, and the learned MemMap is actually a compact evolving structure organized according to the latent "topics" of the graph nodes. 2) Our research suggests that it is a more effective and efficient way to generate node representations from a latent semantic space (like MemMap in our method) than from directly connected neighbors (like most of the previous graph learning methods). The reason is that the number of memory cells in latent space could be much smaller than the number of nodes in a real-world graph, and the representation learning process could well balance the global and local message passing by leveraging the semantic similarity of graph nodes via the correlated memory cells.

*Corresponding Author
†Also with Key Laboratory of Data Science and Intelligent Computing, International Innovation Institute, Beihang University, Hangzhou.

KDD '24, August 25–29, 2024, Barcelona, Spain

ACM ISBN 979-8-4007-0490-1/24/08
https://doi.org/10.1145/3637528.3672060

CCS CONCEPTS

• **Computing methodologies** → **Learning latent representations**; • **Information systems** → **Data mining**.

KEYWORDS

Dynamic Graph Learning, Graph Neural Network

ACM Reference Format:
Shuo Ji, Mingzhe Liu, Leilei Sun, Chuanren Liu, and Tongyu Zhu. 2024. MemMap: An Adaptive and Latent Memory Structure for Dynamic Graph Learning. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24), August 25–29, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3637528.3672060

1 INTRODUCTION

Dynamic graph learning has received increasing attention due to its growing prevalence in real-world applications, such as recommendation systems [1, 11, 32], social networks [14, 17], and community detection [6, 12]. Many real-world networks evolve over time, and extensive efforts have been made to model the evolving patterns of dynamic graphs. Specifically, there are mainly two distinct research settings for dynamic graph learning: Discrete-Time Dynamic Graphs (DTDG) [4, 17, 21, 30] and Continuous-Time Dynamic Graphs (CTDG) [2, 10, 15, 20, 31]. Since Discrete-Time Dynamic Graphs can be formulated as Continuous-Time Dynamic Graphs without information loss [23], we address the problem of dynamic graph learning through a focus on the Continuous-Time Dynamic Graphs.

However, learning the structural and temporal properties of continuous-time data is a challenging task due to the sophisticated and evolutionary graph topology. A fundamental and critical question is *how to update the representations of nodes at the arrival of a new link on the dynamic graph.* In recent years, two mainstream categories of approaches to this question have emerged: sequence-based methods [2, 27, 31] and memory-based methods[10, 20, 24]. The former methods sample a neighborhood for each queried link, such as extracting temporal walks from a huge link set, and then employ sequence models to learn embeddings for graph nodes[27]. However, the huge size of needed data and certain repetitive operations can lead to high computational complexity. In contrast, the latter methods maintain a memory for each node, and the memory

is continuously updated by Recurrent Neural Networks (RNN) to capture information in the link stream[20].

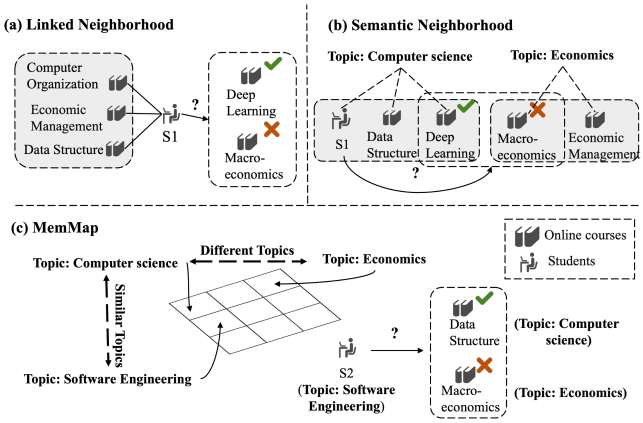

Figure 1: Illustration of the MemMap.

Despite the significant advances in existing works, there are several limitations. *First*, these methods capture structural features and learn dynamic patterns solely based on fixed topology, overlooking the hierarchical and latent correlations among graph nodes. However, such correlations are prevalent in real-world data. This is attributed to the fact that entities often exhibit specific distributions and evolutionary patterns, which we refer to as "topics". For example, when selecting online courses, students majoring in computer science tend to favor courses related to deep learning, with "computer science" serving as their central topic as shown in Figure 1(b). *Second*, these methods find it challenging to make accurate predictions for newly arriving graph nodes with a limited number of links, while the co-correlated topics could reveal rich information via a substantial semantic neighborhood. In Figure 1(c), hierarchical topic correlations are constructed and it is obvious that students majoring in software engineering are more likely to learn "data structure" from a similar topic. *Third*, for large-scale data, recursive sampling of more layers of neighbors or extracting longer temporal walks significantly increases modeling complexity and may introduce noise nodes, leading to counterproductive results. In Figure 1(a), the noise neighbor of "Economic Management" may lead to incorrect predictions. *Finally*, the existing methods tend to overlook collective evolution patterns of the co-correlated topics on dynamic graphs.

In this paper, we contend that modeling hierarchical node correlations and generating node representations from their semantic neighborhoods are crucial for dynamic graph learning. Therefore, we propose a novel structure - Memory Map (MemMap) that captures node correlations with a rectangular grid in the latent memory space. Each cell of the grid represents a specific evolving pattern, and the distances of cells symbolize their hierarchical correlations. Due to the correlations, we construct a semantic neighborhood for each node via its corresponding cell and neighboring cells. As the events stream gradually unfolds, the MemMap captures the historical events of corresponding cells and adaptively evolves with collective and individual trends through the local and global message passing. The efficient indexing of semantic neighborhoods expands the receptive field to the whole graph with linear time complexity. This expansion also benefits the personalized design of various aggregation strategies. Subsequently, we propose map-level aggregation mechanisms to capture temporal and high-order structural features. Our main contributions can be summarized as follows:

- **MemMap Structure:** We introduce MemMap, a novel structure that utilizes a rectangular grid in latent memory space to capture hierarchical correlations of nodes via cells. Our approach constructs an indexable semantic neighborhood and expands the receptive field scalably to the global graph.
- **MemMap Updating Scheme:** We propose a dual-layer MemMap updating scheme to facilitate continuous global and local message passing and to adaptively perceive both collective and individual evolution of graph representations.
- **Map-Level Aggregation Mechanisms:** Leveraging the efficient indexing of MemMap neighborhoods, we propose two personalized map-level aggregation mechanisms. These mechanisms are designed to capture high-order structural features and temporal evolving trends.

2 RELATED WORK

Dynamic graph representation learning is a prominent direction in contemporary research, finding extensive applications in diverse domains such as social networks [14, 17], intelligent transportation [5, 22, 33], and recommendation systems [1, 11, 32]. Based on the formalization of dynamic graph modeling, existing methods for dynamic graph learning can be broadly categorized into two classes: methods for Discrete-Time Dynamic Graphs (DTDG) and methods for Continuous-Time Dynamic Graphs (CTDG)[8]. In the realm of DTDG methods, researchers typically conceive dynamic graphs as a series of snapshots, applying static graph learning techniques to each snapshot [4, 21, 30]. While the intuitiveness of DTDG methods positioned it as the start for exploration, its reliance on manually determined time intervals and disregard for temporal information led to a significant loss of crucial continuous-time dynamic information. This temporal information loss has steered recent research endeavors toward CTDG methods.

Within the domain of CTDG methods, two pivotal branches have emerged: sequence-based methods and memory-based methods. GraphMixer [2] stands as a pioneering sequence-based method, employing the aggregation of first-order neighbor information to construct node representations, thereby supporting various downstream tasks. Building upon this foundation, novel approaches [31] leverage techniques from long sequence learning [3, 16], endowing sequence-based methods with the ability to capture long-term dependencies. However, inherent limitations in modeling node associations arise in sequence-based methods due to their sampling of only first-order neighbors or a large group of temporal walks. In contrast, JODIE [10] and TGN [20], as early examples of memory-based methods, maintain a memory unit for each node and update representations by considering the interaction sequence. Subsequently, numerous methods have proposed improvements in optimizing the efficiency of memory module updates [13, 26] or refining sampling and aggregation techniques [7, 27]. Nevertheless, memory-based methods often depend on existing topologies

when aggregating neighbors and updating memory, which may potentially limit their effectiveness in modeling correlations with long-range dependencies.

While the existing methods are limited to generating representations with directly linked neighbors, We propose constructing an adaptive and evolutionary latent memory space with continuous global and local message passing, and generating node representations by retrieving information from the space.

3 PRELIMINARIES

Dynamic Graph. We represent a dynamic graph as a sequence of timestamped interaction events $\mathcal{G}_T = \{(v_i, v_j, t)|t < T\}$, where v_i, v_j denote the nodes and t denotes the timestamp. The temporal link may have an edge feature $e_{(v_i,v_j,t)}$.

Dynamic Representation Learning. Given a dynamic graph $\mathcal{G}_t$, the aim of dynamic representation learning is to learn a function $f : \mathcal{G}_t \rightarrow h_i(t), h_j(t)$, where $h_i(t), h_j(t) \in \mathcal{R}^d$ represent temporal representations of v_i and v_j, respectively.

Link Prediction. The temporal representation could be utilized for various downstream tasks. In the paper, we evaluate the effectiveness of representations via the dynamic link prediction task.

4 METHODOLOGY

4.1 Definitions of MemMap

The intuition behind MemMap is to capture the inherent correlations among nodes and generate representations from a latent semantic space. A good choice to model these correlations is evaluating the semantic similarity of their representations, which incorporate individual evolutionary patterns. Compared to a directly linked neighborhood, the temporal neighborhood based on representation similarity captures more comprehensive and meaningful information from nodes with similar patterns. However, constructing the temporal neighborhood necessitates separate considerations for each link, introducing a time-consuming problem. To achieve that, we construct an adaptive and evolutionary latent memory space, named Memory Map (MemMap), offering a scalable solution for neighborhood updating and aggregation. The MemMap acts as a map, recording all historical events in global order and facilitating efficient inquiries for related information. We present the formal definition for MemMap in Definition 1.

Definition 1. *(MemMap).* Let $\mathcal{P}_t$ be the MemMap of a dynamic graph $\mathcal{G}_t$, defined as a memory space with rectangular grids. Each cell of $\mathcal{P}_t$, denoted as $p_{(a,b)}$, is associated with a $d-$dimensional representation $s^p_{(a,b),t}$ reflecting a specific pattern. The indexs (a, b) of cells in the grid represent their global correlations.

The MemMap representations $S^p_t \in \mathcal{R}^{n_p \times n_p \times d}$ are initially set as zero vectors. With the arrival of the event stream, we continuously update these representations as detailed in Section 4.3. Through this process, the representation of each cell gradually captures a specific evolutionary pattern, encompassing both temporal and structural information derived from nodes with the similar pattern. We define a projection relation for nodes to cells based on representation similarity in Definition 2. Additionally, with continuous global and local message passing, cells form hierarchical correlations related to their index distances. Specifically, cells with closer index distance exhibit more similar patterns, enabling us to construct the k-layer MemMap neighborhood $\mathcal{N}^k_{(a,b),t}$ of $p_{(a,b)}$ by indexing neighboring cells shown in Definition 3.

Definition 2. Let $f_{p,t}$ be a projection function designed to establish the relation between nodes and map cells in the graph $\mathcal{G}_t$. This function maps each node to a specific *MemMap* cell as:

$$f_{p,t} : v_{i,t} \rightarrow p_{(a,b),t}, \tag{1}$$

Furthermore, with the application of $f_{p,t}$, we obtain the corresponding nodes of the cell $p_{(a,b)}$, denoted as $\mathcal{N}_{(a,b),t}$.

Definition 3. *(K-layer MemMap Neighborhood)* Let $\mathcal{N}^k_{(a,b),t}$ represent the k-layer *MemMap* neighborhood of $p_{(a,b)}$ at t, formally defined as:

$$\mathcal{N}^k_{(a,b),t} = \{\mathcal{N}_{(a',b'),t}|a - k \leq a' \leq a + k, b - k \leq b' \leq b + k\}, \tag{2}$$

With the corresponding map cell of each node, the MemMap neighborhood could be defined as a union of MemMap cells within an index distance from the central cell. The querying of multi-layer neighborhoods incurs only linear complexity growth, eliminating the need for recursive searches of neighbors as required by previous methods[20, 29].

4.2 Overall Framework

The key novelty of our method is the construction of MemMap structure. Figure 2 displays an overview of our method which consists of three modules: MemMap updating scheme, map-based aggregation mechanism and prediction module. To record historical information and update the model with new events, our method consistently maintains a memory vector $s_i(t)$ for each node and a map representation matrix $S^p(t) \in \mathcal{R}^{n_p \times n_p \times d}$ for the entire graph. Cells with closer index distances of the matrix exhibit more similar patterns and the map grid reflects the relation of nodes to cells, allowing for the construction of an indexable MemMap neighborhood for each node.

In the MemMap updating scheme, four fundamental steps encode events and propagate information to local and global cells. Through global and local message passing, our method gradually constructs an adaptive and evolutionary memory space. Subsequently, the map-based aggregation mechanisms, including MemMap neighborhood aggregation and trajectory aggregation, capture high-order structural features and temporal evolving trends, respectively. At last, a prediction module integrates obtained features to generate temporal node representations.

4.3 Memory Map Updating Scheme

In the section, we present an efficient scheme for constructing and updating the MemMap of $\mathcal{G}_t$. The primary objective is to implicitly capture hierarchical correlations of nodes via cells and construct temporal semantic neighborhoods. Through the updating process, temporal and structural features, as well as collective and individual evolving trends, are learned by the corresponding map cells. To achieve this, the scheme primarily consists of four key steps.

Step 1. Event embedding. The event embedding serves as the intermediary for propagating event information to MemMap

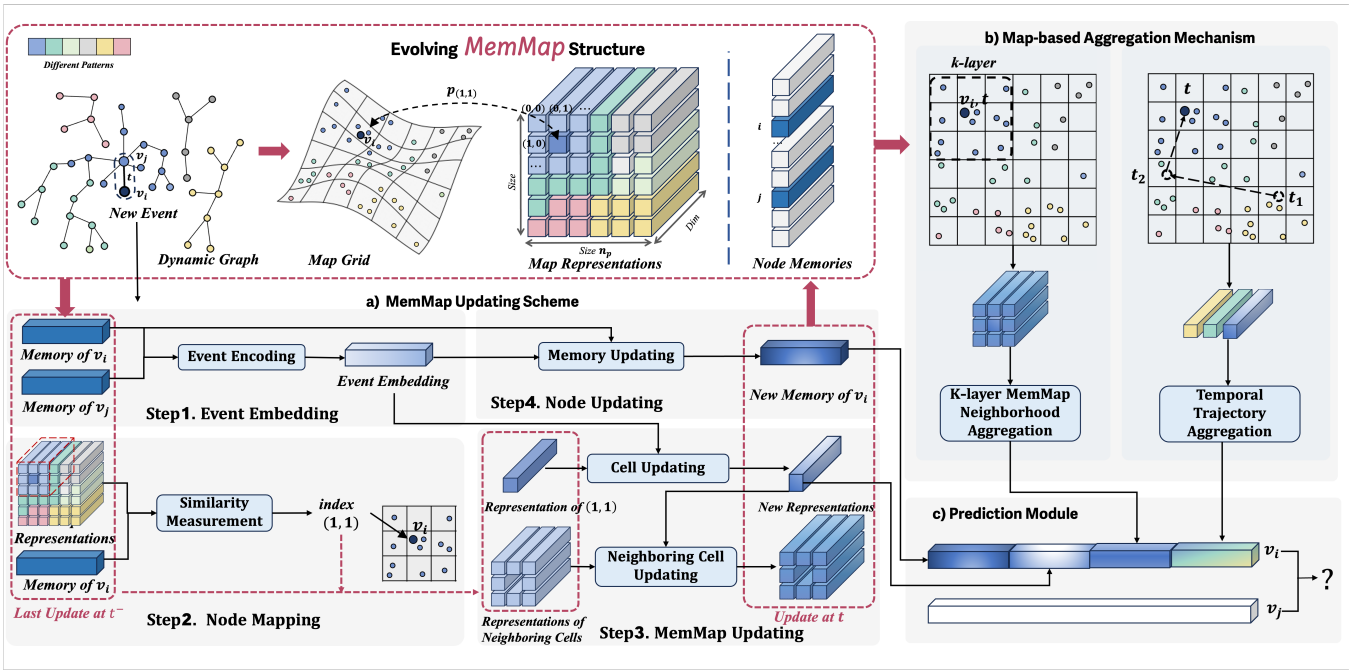

Figure 2: Framework of the MemMap structure. MemMap is a latent memory space where each cell represents an evolving pattern. The framework is composed of a MemMap updating scheme, two map-based aggregation mechanisms, and a prediction module. We generate representations for nodes from the MemMap space.

representations. Thus, each interaction event is compressed into a vectorized representation in this step.

Upon the occurrence of a node-level event (v_i, v_j, t), we encode the event information and generate embeddings $m_i(t), m_j(t)$ for v_i, v_j, respectively. Here, we focus on v_i, and compute the event embedding $m_i(t)$ as follows:

$$m_i(t) = f_m\left(s_i(t^-), s_j(t^-), f_t(t), e_{ij,t}\right), \tag{3}$$

where f_m is an event encoding function, such as MLPs or concatenation. $s_i(t^-), s_j(t^-)$ represent the memories of involved nodes before time t, and $e_{ij,t}$ is the edge feature of the event. Inspired by [29], timestamps are encoded by $f_t(\cdot)$ to introduce rich temporal information, formulated as follows:

$$f_t(t) = [cos(w_1\Delta t_i + b_1), ..., cos(w_n\Delta t_i + b_n)], \tag{4}$$

$$\Delta t_i = t - t_i^-, \tag{5}$$

where $w = [w_1, ..., w_n]$ and $b = [b_1, ..., b_n]$ are trainable parameters.

Step 2. Node mapping. The node mapping process constructs the dynamic relations between nodes and cells. The node representations are employed to identify cells with similar patterns and the projection function defined in Definition 2 establishes an ordered temporal mapping from high-dimensional node representations to two-dimensional grids in the continuous memory space. In this step, the projection function consists of a distance computation function between nodes and cells and a minimum function, which can be formulated as follows:

$$(a, b) = \text{argmin}\left(f_{dis}(s_i(t^-), S^p(t^-))\right), \tag{6}$$

where $s_i(t^-)$ is the memory of v_i, and $S^p(t^-)$ is the representation of MemMap. The function $f_{dis}(\cdot)$ can be various similarity measurement functions, and in the paper we choose the Euclidean distance function. (a, b) represents the index of the corresponding map cell of v_i. With these considerations, we capture the inherent relation between nodes and cells.

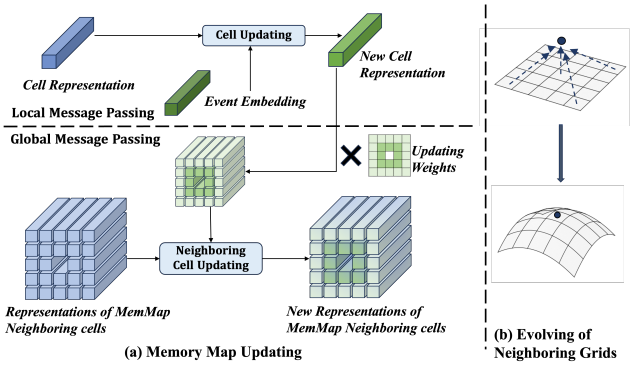

Figure 3: Illustration of MemMap Updating.

Step 3. Memory Map Updating. The updating process is a core submodule that implements a dual-layer updating mechanism to propagate local and global information and capture hierarchical correlations of nodes and cells. Firstly, we update the representation of the corresponding map cell $p_{(a,b)}$ which records historical events of the same pattern, with the embedding of newly arrived event.

Subsequently, we propagate information from $p_{(a,b)}$ to neighboring grids within an index distance k^p to facilitate the gathering of cells with similar patterns and capture hierarchical correlations among cells. Through this process, an indexable semantic MemMap neighborhood is constructed.

For the arrival of the event (v_i, v_j, t), we have obtained the corresponding cell index (a, b) and event embedding $m_i(t)$ for v_i in the previous steps. To incorporate the event information into the local map cell and learn the evolving pattern, we employ a recurrent neural network f_s to fuse the event embedding $m_i(t)$ and map representation $s^p_{(a,b)}(t^-)$ in the following mechanism:

$$s^p_{(a,b)}(t) = f_s\left(s^p_{(a,b)}(t^-), m_i(t)\right). \tag{7}$$

This local message passing process allows the cell representation to capture the evolving patterns and summarize local temporal and structural features of related nodes. In addition, the hierarchical relatedness of evolving patterns is also widely present. To adaptively model the global distribution and capture collective patterns, we update representations $S^p_{\mathcal{N}^k_{(a,b)}}(t)$ of neighboring cells with the representation of $p_{(a,b)}$ to varying extents, formulated as follows:

$$S^p_{\mathcal{N}^k_{(a,b)}}(t) = w^k \cdot s^p_{(a,b)}(t) + (1 - w^k) \cdot S^p_{\mathcal{N}^k_{(a,b)}}(t^-), \tag{8}$$

$$w^k = f_w(w_0, k, t), 0 < k \leq k^p, \tag{9}$$

where k^p is the max layer of global message passing, and w^k represents the weight of neighboring cell updating, monotonically shrinking with time and distance to the central cell. From the perspective of grids in latent space, during the updating process, cells tend to move towards regions with higher node density to model node distribution and learn evolving patterns[9]. Figure 3 provides an illustrative example. To achieve this, the weights need to be larger to quickly expand the memory space and adapt to node distribution in the early stage. After a basic global order emerges, the evolution of the map is primarily driven by dynamic patterns. Contributing to this order, the correlation between cells could be reflected in the index distance of the map.

Step 4. Node memory updating. Given that nodes exhibit dynamic characteristics in reality, it is crucial to update node memories with a recurrent network to ensure accurate node mapping and map updating at the end of each updating process as follows:

$$s_i(t) = f_s\left(s_i(t^-), m_i(t)\right). \tag{10}$$

Node memories reflect the individual preferences and evolving patterns of nodes.

4.4 Map-based Aggregation Mechanism

In the updating process, the MemMap captures hierarchical correlations and the representation of each cell preserves local historical information. We can easily retrieve information from a union of cells by indexing the representation matrix in Definition 1. With this capability, the receptive field can be expanded to the entire graph. MemMap provides a scalable solution for designing various personalized aggregations for specific information. In this module, we introduce two categories of map aggregation mechanisms for high-order structural features and temporal evolving trends.

MemMap Neighborhood Aggregation. Learning high-order structural information is a crucial but challenging task due to the high time complexity associated with querying a sufficiently large neighborhood. In our approach, the updating process incorporates global and local message passing and hierarchical correlation establishment based on index distance, enabling us to directly access the map representations of a k-layer MemMap neighborhood. We utilize a temporal attention mechanism to aggregate high-order structural features as follows:

$$h^n_i(t) = f_{agg}\left(s^p_{(a,b)}(t), S^p_{\mathcal{N}^k_{(a,b)}}(t), t\right), \tag{11}$$

where $S^p_{\mathcal{N}^k_{(a,b)}}(t)$ denotes the map representations of neighboring cells. High-order structural information is particularly beneficial for understanding graph topology, especially when dealing with large-scale graphs and new node prediction. For large-scale datasets, the MemMap neighborhood aggregation benefits in obtaining a more comprehensive structure. In the case of newly arrived nodes, we obtain enriched insights from a broad receptive field of similar patterns owing to the extracting of high-order information.

Trajectory Aggregation. In real-world scenarios, both individual and collective evolution are common phenomena. As a result, the patterns of nodes may change, leading to the formation of trajectories on the MemMap grid. These trajectories encapsulate rich information about the evolving trends of nodes and semantic neighborhoods at different stages. Nodes with similar evolving patterns tend to exhibit similar trajectories. To incorporate trajectory representations with temporal features, we employ a temporal attention mechanism as follows:

$$h^t_i(t) = f_{agg}\left(s^p_{(a,b)}(t), S^p_{\mathcal{T}_i}(t), t\right), \tag{12}$$

where $S^p_{\mathcal{T}_i}(t)$ denotes the map representations of cells in the trajectory. While individual node evolution may exhibit some randomness, collective evolution follows certain patterns and is more reliable for prediction.

4.5 Prediction Module

In the aforementioned sections, we construct MemMap, an evolutionary latent space, from which we obtain the representation $s^p_{(a,b)}(t)$ of the corresponding cell, preserving local temporal and structural information. Through personalized map-based aggregation mechanisms, the representation $h^n_i(t)$ of MemMap neighborhood, captures high-order structural information and the representation $h^t_i(t)$ of trajectory cells learns temporal evolutionary patterns. To generate node representation, we concat these representations and employ an MLP function as follows:

$$h_i(t) = MLP\left(s_i(t)||s^p_{(a,b)}(t)||h^n_i(t)||h^t_i(t)\right), \tag{13}$$

where $h_i(t)$ is the representation of v_i and $h_j(t)$ can be computed in the same manner. With temporal representations, we perform tasks detailed in Section 5.1.3. Since all events are split into separate batches at the training stage, the most recent interactions may be neglected. Thus, we incorporate information of occurred events within the same batch as additional context for some datasets.

5 EXPERIMENTS

In this section, we conduct experiments to evaluate the effectiveness of MemMap on six real-world datasets guided by the following research questions (RQs):

- **RQ1**: How does MemMap perform compared with the state-of-the-art baselines?
- **RQ2**: Is the semantically correlated neighborhood a better choice compared with the directly connected neighborhood in terms of both effectiveness and efficiency?
- **RQ3**: How do the different components of MemMap contribute to its performance?
- **RQ4**: How does MemMap evolve over time and improve its performance?
- **RQ5**: What is the inherent correlation between nodes or cells when they are adjacent or on the same trajectory?
- **RQ6**: How do the choices of key hyperparameters influence the learning of MemMap?

5.1 Experiments Settings.

5.1.1 Datasets and Baselines. The experiments have been conducted on six real-world datasets and compared with seven SOTA baselines. Detailed descriptions of datasets and baselines can be found in Appendix A.1 and A.2.

5.1.2 Evaluation Tasks. We evaluate models for dynamic link prediction in two settings: transductive setting and inductive setting. For each dataset, we split the event stream into three intervals $[0, T_{train}]$, $[T_{train}, T_{val}]$ and $[T_{val}, T_{test}]$ with $T_{train}/T_{test} = 0.7$ and $T_{val}/T_{test} = 0.85$. Moreover, we remove duplicate links with the same timestamps. In the transductive task, all events in the training dataset can be observed, and we predict the occurrence probability of each link based on historical timestamped links. In the inductive task, we randomly select 10% of the nodes, delete their related links in the dataset and predict the occurrence probability of each link based on the remaining historical timestamped links. Average Precision (AP) and Area Under the Receiver Operating Characteristic Curve (AUC-ROC) are utilized as evaluation metrics for all models.

5.1.3 Model Configurations. For training the models, we utilize the Adam optimizer with an initial learning rate of 0.0001, the binary cross-entropy loss as loss function and an early stopping strategy with a patience of 10. The batch size is set to 50 for UCI and Enron datasets, and 200 for the other datasets. A detailed analysis of model parameters can be found in Appendix C. All specific best model parameters, along with our code and data can be found at: https://github.com/AhaSokach/MemMap.

5.1.4 Hardware. Experiments are conducted on an Ubuntu machine equipped with one Intel(R) Xeon(R) Gold 6226R CPU @ 2.90GHz with 16 physical cores and one GPU(NVIDIA GeForce RTX 3090).

5.2 Performance Comparison (RQ1)

Table 1 and Table 2 report the results of transductive and inductive dynamic link prediction using the AUC-ROC metric. In summary, MemMap achieves state-of-the-art performance in both tasks. The best and second best results are highlighted by **bold** and underlined fonts and the results are multiplied by 100 to better display. The results for the AP metric in the two tasks can be found in Appendix B.1 and B.2.

Firstly, MemMap outperforms the state-of-the-art methods in most cases of the transductive task which may be because: (1) we employ the MemMap structure to construct hierarchical correlation and build a semantic neighborhood that expands the receptive field and offers richer and more complex information. (2) Two specific aggregation mechanisms are utilized aiming at high-order structural information and incorporation of local and collective temporal evolving trends.

Secondly, in the inductive task, MemMap demonstrates a significant performance advantage over all baselines. This improvement can be attributed to our approach of generating node representations from the MemMap space, which constructs a temporal semantic neighborhood and expands the receptive field to the entire graph. New nodes are able to obtain sufficient information from cells with similar patterns, while the information from linked neighbors in other methods remains comparatively limited.

We also conduct experiments on the key parameters and analyze the results in Appendix C.

5.3 Linked v.s. Semantic Neighborhood (RQ2)

5.3.1 Effective Evaluation. To verify the superiority of semantically correlated neighborhoods as a strong support of our motivation, we conduct experiments over two neighborhood selection strategies named as follows:

- **LinkNbr (Linked Neighborhood)**: samples directly interacted historical neighbors upon the arrival of each event.
- **SemanNbr (Semantic Neighborhood)**: calculates representation similarity of involved nodes with all other nodes and samples nodes with the most similar representations upon the arrival of each event.

The numbers of sampled neighbors are set to 20, 50, and 100. To isolate and accurately assess the impact of neighborhood selection strategies, we conduct experiments within dynamic node embedding framework (Step 4 in Section 4.3), utilizing the simplest graph summing operator as follows:

$$h_t(t) = W_2(h_{\mathcal{N}}(t)||s_i(t)), \tag{14}$$

$$h_{\mathcal{N}}(t) = \mathrm{ReLU}\left(\sum_{v_z \in \mathcal{N}} W_1 s_z(t^-)\right), \tag{15}$$

where $\mathcal{N}$ is the sampled neighborhood and $s_z(t^-)$ is the memories of nodes in $\mathcal{N}$. The results on transductive link prediction are displayed in Table 3. The results of MemMap neighborhood strategies (MemNbr) are also presented as a special type of semantic neighborhood, without incorporating additional information. Due to the independence of MemMap neighborhood from the number of neighbors, the table shows a repetition of identical results. Observations from the table include: (1) Regardless of the chosen number of the neighborhood, the semantic neighborhood outperforms the directly linked neighborhood, possibly due to the semantic neighborhood carrying more related and meaningful information. (2) Aggregating more neighbors is not always beneficial, as some noisy information

Table 1: AUC-ROC for transductive dynamic link prediction.

Datasets	JODIE	DyRep	TGAT	TGN	CAWN	TCL	GraphMixer	Ours
Wikipedia	96.16±0.09	95.40±0.29	97.10±0.09	98.64±0.04	98.60±0.02	97.47±0.03	97.12±0.08	**98.85±0.04**
MOOC	82.80±1.98	85.23±0.25	88.39±0.10	92.00± 0.80	82.11±0.07	85.22±0.71	84.82±0.06	**94.00±0.16**
LastFM	69.90±0.81	71.17±0.11	71.26±0.19	78.27±2.42	82.35±0.23	68.88± 4.22	73.93±0.15	**84.11±0.41**
Reddit	98.43±0.03	98.33±0.07	97.12±0.07	98.73±0.05	**99.08±0.01**	98.18±0.01	97.42±0.02	98.80±0.01
Enron	87.52±2.93	82.90 ±1.77	76.54±0.99	84.09±1.46	92.56±0.86	82.10±1.45	87.45±0.08	**93.64±0.24**
UCI	90.32± 0.19	64.01±7.49	79.91±0.57	90.49±2.52	93.45±0.04	84.42±1.07	91.72±0.46	**94.48±0.25**

Table 2: AUC-ROC for inductive dynamic link prediction.

Datasets	JODIE	DyRep	TGAT	TGN	CAWN	TCL	GraphMixer	Ours
Wikipedia	95.32±0.12	92.77±0.38	96.76±0.05	98.28±0.09	98.21±0.03	97.62±0.06	97.45±0.06	**98.84±0.01**
MOOC	82.72±0.71	81.51±1.19	84.91±0.22	89.03±0.97	80.70±0.57	80.07±0.22	80.35±0.34	**90.49±0.60**
LastFM	83.57±1.62	82.30±2.48	77.47±0.24	88.89±1.46	85.54±0.13	77.81±0.45	83.89±0.08	**91.87±0.47**
Reddit	96.32±00.28	96.34±0.15	97.16±0.07	97.34±0.33	98.12±0.03	94.95±0.56	96.84±0.05	**98.21±0.09**
Enron	82.30±0.16	60.81±0.49	61.11±0.48	80.64±0.68	87.28±0.42	64.21±0.24	70.22±0.38	**94.57±0.78**
UCI	89.66±0.20	54.34±4.72	78.71±0.22	93.37±0.20	92.55±0.10	83.61±0.20	91.64±0.56	**93.59±0.31**

Table 3: Comparison of different neighborhood sampling strategies by AP.

Datasets	Num.	LinkNbr	SemanNbr	MemNbr
MOOC	100	84.02±1.37	85.73±0.48	86.05±0.58
	50	83.80±2.45	85.47±0.17	86.05±0.58
	20	85.11±0.45	85.37±0.84	86.05±0.58
LastFM	100	72.87±2.32	75.44±1.94	76.33±0.83
	50	71.25±0.93	74.74±0.90	76.33±0.83
	20	72.47±1.96	76.02±0.62	76.33±0.83

Table 4: Average running time(s) of training epoch with respect to different layer settings.

Datasets	layer	TGN	MemMap	CAWN
MOOC	1	206.51	242.64	1083.97
	2	291.7	279.05	1083.97
	3	1314.71	285.76	1083.97
LastFM	1	167.14	932.14	2354.45
	2	505.93	951.04	2354.45
	3	3708.58	970.23	2354.45

Table 5: Maximum memory usage(MB) of training epoch with respect to different layer settings.

Datasets	layer	TGN	MemMap	CAWN
MOOC	1	1469.30	647.56	4971.47
	2	2049.37	705.73	4971.47
	3	8693.22	793.06	4971.47
LastFM	1	1243.70	1159.12	5547.42
	2	1914.13	1214.58	5547.42
	3	8559.61	1302.32	5547.42

may lead to a decline in performance, especially for linked neighborhoods. (3) Compared with SemanNbr which necessitates the calculation of node similarity between all nodes, MemMap serves as a scalable solution for semantic neighborhood selection strategy. It is efficient and potentially superior, possibly because we build hierarchical correlations among cells and capture collective evolving trends.

5.3.2 Efficiency Evaluation. In addition to evaluating the prediction performance, we conduct scalability evaluations on both directly linked neighborhoods and MemMap neighborhoods across layers 1, 2, and 3. We use TGN and CAWN as representative memory-based and sequence-based methods, respectively, with both relying on directly linked neighborhoods. And TGN is known for its effectiveness and efficiency. In the CAWN model, the concept of layers is not explicitly present. Instead, the model is configured with a sequence length of 64.

The results in Table 4 and Table 5 present the average training time and maximum memory usage per epoch respectively. Several observations can be made: (1) CAWN exhibits significantly higher computational time and memory usage compared to other methods, likely due to its online sampling of neighborhoods with temporal walks. (2) As the number of layers increases, computational time and memory usage of linked neighborhood aggregation quickly rise, attributed to recursive inquiry for neighborhoods. In contrast, MemMap shows relatively smaller increases in computational time, possibly because it efficiently retrieves information via an indexed map matrix. Additionally, MemMap demonstrates notably lower memory usage, likely owing to its aggregation and updating mechanisms based on a smaller number of cells.

5.4 Ablation Study (RQ3)

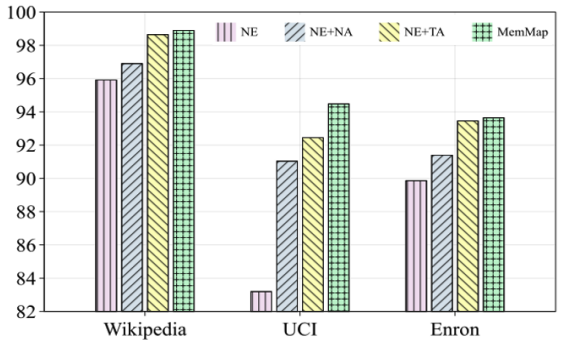

Figure 4: AUC-ROC for Ablation Study.

To investigate the effectiveness and characteristics of MemMap, we conduct an ablation study on three datasets with varying node scales, including Wikipedia (9227 nodes), UCI(1899 nodes) and Enron(184 nodes). We name variations of MemMap as follows:

- **NE (Node Embedding Only)**: removes the MemMap structure, utilizing the dynamic node memory only.
- **NE+NA (Node Embedding + Neighboring Aggregation)**: integrates MemMap neighborhood aggregation into NE.
- **NE+TA (Node Embedding + Trajectory Aggregation)**: integrates map-based trajectory aggregation into NE.

Figure 4 summarizes the ablation results. As expected, we could make a conclusion that all components contribute to the final results to a certain extent. Firstly, removing the MemMap(NE) leads to a significant decline, which indicates the effectiveness of the MemMap structure. Secondly, the introduction of neighborhood aggregation leads to varied improvements compared with NE across the three datasets, with the most notable enhancement observed on the UCI dataset. This suggests a significant impact of high-order structural information on this particular dataset. Conversely, the improvements on the Wikipedia and Enron datasets were relatively modest, which could be attributed to the low proportion of new edges on the Wikipedia dataset[19] and the limited number of nodes on the Enron dataset. These factors constrain the efficacy of high-order structural information for these two datasets. Subsequently, the trajectory aggregation module yields a significant improvement. This outcome underscores the importance of temporal information and dynamic characteristics. Additionally, the map cells constituting the trajectory have inherently aggregated local structural information, which may be another reason for the better performance of NE+TA.

5.5 Visualization of Memory Map (RQ4)

5.5.1 Memory Map projection. In this section, to gain deeper insights into how the memory map evolves and contributes to improved prediction performance, we employ PCA [28] to project the dynamic representations of MemMap at intervals of five batches, which are divided by time on the Wikipedia dataset. The color scheme denotes the index distance from the center of the map: darker regions represent the edges of the map, while lighter regions represent the center.

The results in Figure 5, present several intriguing phenomena. In the initial stage (mainly including figures of the first row), the MemMap gradually expands from a single node. As an interaction event arrives, the MemMap cells with the similar pattern give an active response to the event and adaptively move to a particular domain in the memory space. In the subplots of the second row, there is a noticeable stretching and distortion of MemMap, indicating that map cells develop into specific detectors of different patterns. Throughout the process, cells in the memory space gradually learn specific patterns and evolving trends. The pictures in the third row display their different evolving directions driven by dynamic patterns with a consistent global order. The results illustrate that MemMap succeeds in learning individual and collective evolving trends of patterns, effectively improving its performance.

5.5.2 Cell Correlation Analysis. To visually illustrate the correlations of cells in MemMap, we utilize K-Means to cluster the MemMap representations and assign distinct colors to the clusters. The left subplot of Figure 6 showcases the results, where each grid unit corresponds to a map cell. It is evident that MemMap forms distinct clusters, affirming its structure in capturing the hierarchical correlation and similar pattern.

5.6 Analysis of Aggregation Strategies (RQ5)

In this section, to further conceptualize the personalized map-based aggregation strategies, we utilize PCA to project the representations of corresponding map cells and nodes on the Wikipedia dataset. The results illustrate their characteristics and superiority.

5.6.1 Neighborhood Aggregation Projection. In this subsection, we zoom in on a 3×3 grid and visualize the nodes within the grid in a two-dimensional space. Nodes of different cells are represented by various colors, and different shapes indicate nodes in two different directly linked neighborhoods. We focus on the larger v_a and v_b nodes, both of which are associated with the center cell. It can be observed that the directly linked nodes are positioned around these larger nodes, i.e. in their close cells, especially those with frequent or recent links in the original dataset, respectively. Moreover, the MemMap neighborhood includes another category of correlation, also represented by the larger v_a and v_b nodes. Although they were not directly linked before, they exhibit similar patterns, as evidenced by their linked neighbors in the common cells. This similarity suggests that they have a similar pattern, and it is likely that the larger v_a and v_b nodes will interact with each other in the future.

5.6.2 Trajectory Aggregation Projection. In this part, we provide a detailed analysis of the trajectory of the v_{137} node. We project the representations of v_{137} at three different times when it undergoes the evolving pattern. It can be observed that the node changes its corresponding cell two times, as marked in the left figure, and nodes in the three different cells are labeled in different colors. The dots represent the projected representations of nodes that interact with the v_{137} node at different stages. An obvious observation is that the change in interacted neighbors corresponds to the pattern evolving, verifying the accurate capture of the collective trend. And we can

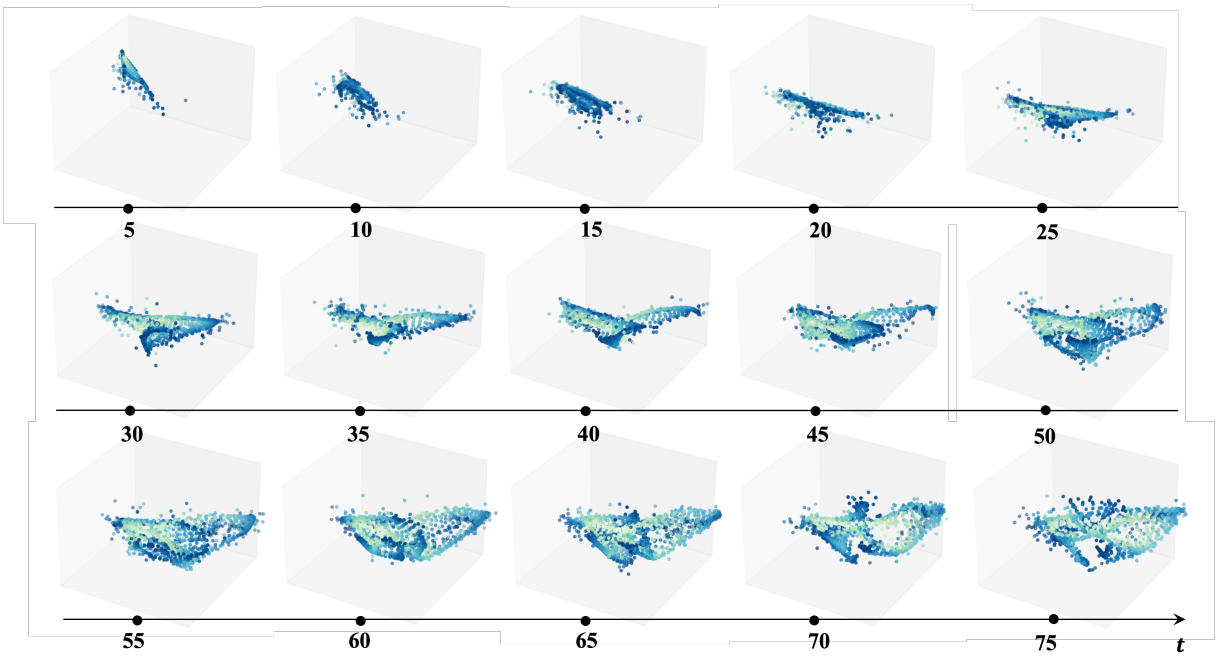

Figure 5: Illustration of how MemMap evolves on Wikipedia dataset, where each point denotes a map cell representation after dimension reduction and deepening colors indicate the expansion of the map grid from the center to the edges.

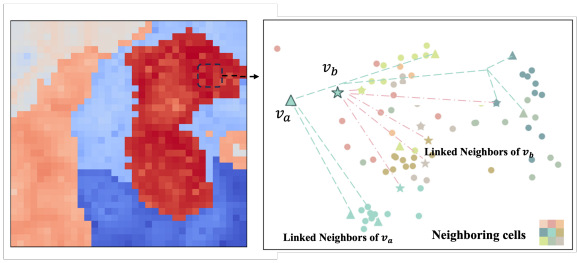

Figure 6: The MemMap neighborhood of node v_a and v_b.

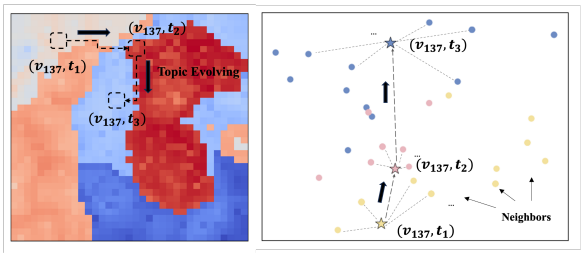

Figure 7: A trajectory composed of different cells of v_{137}.

conclude that if the pattern of a node changes, it is more likely to interact with nodes under the new pattern.

6 CONCLUSION

In the paper, we introduce a novel memory structure - Memory Map for dynamic graph learning, a latent memory space with a rectangular grid where each cell represents an evolving pattern. The MemMap serves as a framework to model hierarchical correlations between nodes and cells and generate node representations from semantically correlated memory cells, breaking free from the dependency on fixed topology. Moreover, the updating scheme of MemMap facilitates both global and local message passing, capturing collective and individual evolving trends. The updating process transforms irregular, complex graph data into a structured, regular matrix. Leveraging the convenience of matrix inquiries, we utilize two aggregation mechanisms to learn high-order structural information and pattern transition trends. Experiments on real-world datasets validate the effectiveness of MemMap and provide insights into inherent correlations and evolution processes. We posit that MemMap is a significant and intriguing exploration, and various aggregation approaches can be explored based on MemMap to adapt to different tasks, including graph-level tasks.

ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers for their constructive comments on this research. This work was supported by the National Natural Science Foundation of China (62272023, 51991391) and the Fundamental Research Funds for the Central Universities (YWF-23-L-1203).

REFERENCES

[1] Jianxin Chang, Chen Gao, Yu Zheng, Yiqun Hui, Yanan Niu, Yang Song, Depeng Jin, and Yong Li. 2021. Sequential recommendation with graph neural networks. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*. 378–387.

[2] Weilin Cong, Si Zhang, Jian Kang, Baichuan Yuan, Hao Wu, Xin Zhou, Hanghang Tong, and Mehrdad Mahdavi. 2023. Do We Really Need Complicated Model Architectures For Temporal Networks? *arXiv preprint arXiv:2302.11636* (2023).

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).

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

[5] Liangzhe Han, Xiaojian Ma, Leilei Sun, Bowen Du, Yanjie Fu, Weifeng Lv, and Hui Xiong. 2022. Continuous-Time and Multi-Level Graph Representation Learning for Origin-Destination Demand Prediction. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 516–524.

[6] Shuo Ji, Xiaodong Lu, Mingzhe Liu, Leilei Sun, Chuanren Liu, Bowen Du, and Hui Xiong. 2023. Community-based Dynamic Graph Learning for Popularity Prediction. ACM, 930–940.

[7] Ming Jin, Yuan-Fang Li, and Shirui Pan. 2022. Neural temporal walks: Motif-aware representation learning on continuous-time dynamic graphs. *Advances in Neural Information Processing Systems* 35 (2022), 19874–19886.

[8] Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. 2020. Representation learning for dynamic graphs: A survey. *The Journal of Machine Learning Research* 21, 1 (2020), 2648–2720.

[9] Teuvo Kohonen. 1998. The self-organizing map. *Neurocomputing* 21, 1-3 (1998), 1–6. https://doi.org/10.1016/S0925-2312(98)00030-7

[10] Srijan Kumar, Xikun Zhang, and Jure Leskovec. 2019. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 1269–1278.

[11] Xiaohan Li, Mengqi Zhang, Shu Wu, Zheng Liu, Liang Wang, and S Yu Philip. 2020. Dynamic graph collaborative filtering. In *2020 IEEE international conference on data mining (ICDM)*. IEEE, 322–331.

[12] Yu-Ru Lin, Yun Chi, Shenghuo Zhu, Hari Sundaram, and Belle L. Tseng. 2008. Facetnet: a framework for analyzing communities and their evolutions in dynamic networks. ACM, 685–694.

[13] Yuhong Luo and Pan Li. 2022. Neighborhood-aware scalable temporal network representation learning. In *Learning on Graphs Conference*. PMLR, 1–1.

[14] Yao Ma, Ziyi Guo, Zhaocun Ren, Jiliang Tang, and Dawei Yin. 2020. Streaming graph neural networks. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*. 719–728.

[15] Giang Hoang Nguyen, John Boaz Lee, Ryan A. Rossi, Nesreen K. Ahmed, Eunyee Koh, and Sungchul Kim. 2018. Continuous-Time Dynamic Network Embeddings. In *Companion of the The Web Conference 2018 on The Web Conference 2018, WWW 2018, Lyon , France, April 23-27, 2018*, Pierre-Antoine Champin, Fabien Gandon, Mounia Lalmas, and Panagiotis G. Ipeirotis (Eds.). ACM, 969–976. https://doi.org/10.1145/3184558.3191526

[16] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2022. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730* (2022).

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

[18] James W Pennebaker, Martha E Francis, and Roger J Booth. 2001. Linguistic inquiry and word count: LIWC 2001. *Mahway: Lawrence Erlbaum Associates* 71, 2001 (2001), 2001.

[19] Farimah Poursafaei, Shenyang Huang, and Kellin Pelrine and. 2022. Towards Better Evaluation for Dynamic Link Prediction. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

[20] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. 2020. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637* (2020).

[21] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang. 2020. Dysat: Deep neural representation learning on dynamic graphs via self-attention networks. In *Proceedings of the 13th international conference on web search and data mining*. 519–527.

[22] Hongzhi Shi, Quanming Yao, Qi Guo, Yaguang Li, Lingyu Zhang, Jieping Ye, Yong Li, and Yan Liu. 2020. Predicting origin-destination flow via multi-perspective graph convolutional network. In *2020 IEEE 36th International conference on data engineering (ICDE)*. IEEE, 1818–1821.

[23] Amauri H. Souza, Diego Mesquita, Samuel Kaski, and Vikas Garg. 2022. Provably expressive temporal graph networks.

[24] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. 2019. DyRep: Learning Representations over Dynamic Graphs. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

[25] Lu Wang, Xiaofu Chang, Shuang Li, Yunfei Chu, Hui Li, Wei Zhang, Xiaofeng He, Le Song, Jingren Zhou, and Hongxia Yang. 2021. TCL: Transformer-based Dynamic Graph Modelling via Contrastive Learning. *CoRR* abs/2105.07944 (2021). arXiv:2105.07944 https://arxiv.org/abs/2105.07944

[26] Xuhong Wang, Ding Lyu, Mengjian Li, Yang Xia, Qi Yang, Xinwen Wang, Xinguang Wang, Ping Cui, Yupu Yang, Bowen Sun, et al. 2021. Apan: Asynchronous propagation attention network for real-time temporal graph embedding. In *Proceedings of the 2021 international conference on management of data*. 2628–2638.

[27] Yanbang Wang, Yen-Yu Chang, Yunyu Liu, Jure Leskovec, and Pan Li. 2021. Inductive representation learning in temporal networks via causal anonymous walks. *arXiv preprint arXiv:2101.05974* (2021).

[28] Svante Wold, Kim Esbensen, and Paul Geladi. 1987. Principal component analysis. *Chemometrics and intelligent laboratory systems* 2, 1-3 (1987), 37–52.

[29] Da Xu, Chuanwei Ruan, Evren Körpeoglu, Sushant Kumar, and Kannan Achan. 2020. Inductive representation learning on temporal graphs. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

[30] Jiaxuan You, Tianyu Du, and Jure Leskovec. 2022. ROLAND: graph learning framework for dynamic graphs. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2358–2366.

[31] Le Yu, Leilei Sun, Bowen Du, and Weifeng Lv. 2023. Towards Better Dynamic Graph Learning: New Architecture and Unified Library. *arXiv preprint arXiv:2303.13047* (2023).

[32] Mengqi Zhang, Shu Wu, Xueli Yu, Qiang Liu, and Liang Wang. 2022. Dynamic graph neural networks for sequential recommendation. *IEEE Transactions on Knowledge and Data Engineering* 35, 5 (2022), 4741–4753.

[33] Ruixing Zhang, Liangzhe Han, Boyi Liu, Jiayuan Zeng, and Leilei Sun. 2022. Dynamic graph learning based on hierarchical memory for origin-destination demand prediction. *arXiv preprint arXiv:2205.14593* (2022).

A DETAILED EXPERIMENTAL SETTINGS

A.1 Datasets

We conduct experiments on six real-world datasets collected by [19], and more details of datasets are listed as follows.

- **Wikipedia**: is a bipartite interaction network between editors and Wiki pages over the span of a month. Links in the network represent timestamped edit events, each characterized by a 172-dimensional LIWC feature vector[18] extracted from the content of the Wikipedia pages.
- **MOOC**: consists of interactions between students and various online course content units. Each link signifies a student's access to specific content, marked with a timestamp, and is associated with a 4-dimensional feature vector.
- **LastFM**: is a bipartite interaction graph detailing the events of users listening to songs over a month. Each link corresponds to a timestamped user-listens-to-song interaction.
- **Reddit**: records interactions between users and subreddits over a one-month period. Users and subreddits are represented as nodes, while the links are timestamped posting requests accompanied by 172-dimensional LIWC features.
- **Enron**: encompasses email correspondence events among employees of the ENRON energy company, spanning three years. Each link represents a timestamped message.
- **UCI**: is an online communication network that records communication events among students of the University of California at Irvine, along with timestamps.

We present the statistics of the datasets in Table 6, where the number of nodes for bipartite graphs is displayed as sources/destinations.

Table 6: Statistics of datasets.

Datasets	Bipartite	#Nodes	#Links	Duration
Wikipedia	True	8,227/1,000	157,474	1 month
MOOC	True	7,047/97	411,749	17 months
LastFM	True	980/1,000	1,293,103	1 month
Reddit	True	10,000/984	672,447	1 month
Enron	False	184	125,235	3 years
UCI	False	1899	59,835	196 days

A.2 Baselines

We compare our method with seven state-of-the-art baselines that could be mainly grouped into two categories:(1) memory-based methods: JODIE[10], DyRep[24], TGAT[29], TGN[20]; (2) sequence-based methods: CAWN[27], TCL[25], GraphMixer[2]. More details are listed as follows.

- **JODIE**: employs a coupled recurrent neural network model to update embeddings of users and items and a projection operator to learn the future embeddings trajectory.
- **DyRep**: builds a deep recurrent architecture to update and compute node representations. The temporally attentive aggregation utilizes temporal attentiveness to aggregate structural information.
- **TGAT**: proposes the temporal graph attention layer to aggregate temporal neighborhood and a time encoding function to learn temporal information for nodes.
- **TGN**: presents a general memory-based framework that maintains a memory vector for each node and updates the memory upon the arrival of new events.
- **CAWN**: relies on the huge link set to extract diverse temporal walks and employs sequence models to learn walk embeddings that are further aggregated for each interaction.
- **TCL**: designs a two-stream encoder based on co-attentional transformer to capture temporal-topological information of sampled temporal subgraph.
- **GraphMixer**: proposes a link-encoder based on MLP-mixer with a fixed time-encoding function, and a node-encoder based on mean pooling for one-hop neighbors.

B PERFORMANCE COMPARISON IN AP

B.1 Performance in AP for Transductive Dynamic Link Prediction

We report the results in AP metric for transductive dynamic link prediction in Table 7.

B.2 Performance in AP for Inductive Dynamic Link Prediction

We report the results in AP metric for inductive dynamic link prediction in Table 8.

C INFLUENCE OF HYPERPARAMETERS (RQ6)

To assess the influence of crucial hyperparameters, we evaluate the performance for MemMap updating and aggregation under different settings. This involves examining variations in map size, neighborhood aggregation layer and updating weight with adjustments made at different levels to comprehensively analyze their effects.

C.1 Influence of map size.

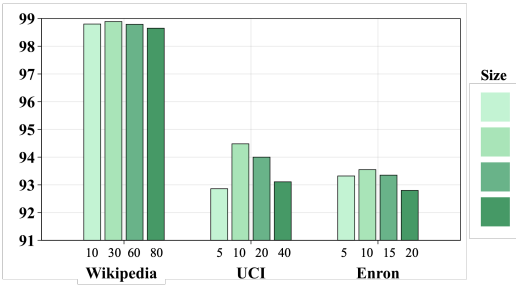

Figure 8: Effect of map size in AUC-ROC.

As MemMap takes the form of a regular two-dimensional grid, the map size n_p refers to the side length of the grid, defined in Section 4.1. A larger size corresponds to a sparser node density. We vary the map size from 5, 10, 20, 40 for UCI, for Enron, and 10, 30, 60, 80 for Wikipedia according to the number of nodes.

The results are presented in Figure 8, where we observe an initial improvement followed by a decline in performance with an increase in size. This trend is attributed to the adverse impact of excessively sparse or dense point densities, affecting the effective aggregation of information. Generally, maintaining a ratio of node number to $n_p \times n_p$ around 10 appears to be a suitable proportion. However, the observed differences are relatively minor, likely stemming from the grid's proactive adaptation to the data space. Throughout the continuous updating process, the grid dynamically adjusts its shape based on the density of nodes. For instance, when the number of grids is excessive, some grids may remain unoccupied.

C.2 Influence of neighborhood aggregation layer.

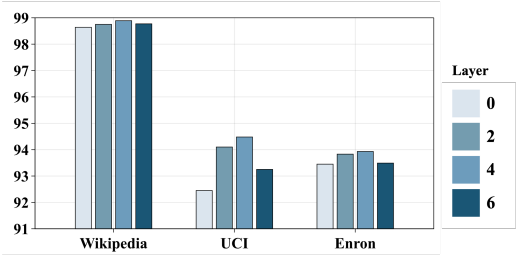

Figure 9: Effect of aggregation layer in AUC-ROC.

Table 7: AP for transductive dynamic link prediction.

Datasets	JODIE	DyRep	TGAT	TGN	CAWN	TCL	GraphMixer	Ours
Wikipedia	95.94±0.21	95.75±0.28	97.33±0.08	98.69±0.04	98.80±0.01	97.90±0.03	97.41±0.08	**98.89±0.03**
MOOC	78.57±2.31	81.90±0.38	87.09±0.04	91.89±1.28	81.61±0.06	84.69±0.88	83.50±0.08	**92.65±0.15**
LastFM	69.74±1.54	71.57±1.13	72.78±0.29	77.32±2.20	84.41±018	73.29±5.99	76.02±0.19	**84.83±0.62**
Reddit	98.47±0.02	98.35±0.07	97.41±0.08	98.76±0.04	**99.16±0.01**	98.18±0.01	97.53±0.01	98.78±0.01
Enron	84.12 ± 3.62	78.56±2.76	76.82±1.08	81.84±2.41	**92.52±0.61**	85.02±1.32	85.85±0.27	92.18±0.37
UCI	88.40 ± 0.89	58.54±6.71	80.14 ±0.42	91.00±3.90	94.77±0.01	85.16±1.63	93.26±0.46	**95.14±0.18**

Table 8: AP for inductive dynamic link prediction.

Datasets	JODIE	DyRep	TGAT	TGN	CAWN	TCL	GraphMixer	Ours
Wikipedia	95.67±0.15	93.44±0.26	97.03±0.05	98.36±0.10	98.44±0.03	97.95±0.08	97.65±0.06	**98.91±0.01**
MOOC	78.28±1.35	77.25±1.76	84.39±0.13	87.66 ±0.76	81.31±0.47	80.29±0.21	80.20±0.26	**89.41±0.31**
LastFM	83.73±0.99	82.64±2.96	78.73±0.27	88.29±1.72	88.02±1.02	82.41±0.45	85.65±0.25	**91.99±0.82**
Reddit	96.17±0.35	96.43±0.09	97.28±0.21	97.46±0.41	**98.37±0.02**	95.27±0.59	97.03±0.06	98.28±0.09
Enron	76.20±0.41	58.65±0.63	61.39±0.46	79.08±1.29	88.03±0.08	66.77±0.36	66.95±0.53	**94.74±0.81**
UCI	87.10±0.19	50.10±3.27	80.25±0.23	93.78±0.11	94.21±0.09	84.59±0.27	93.45±0.38	**94.37±0.24**

The neighborhood aggregation module is employed to capture structural features and aggregate high-order information. Neighborhood aggregation layer k represents the layer of MemMap neighborhood that undergoes aggregation defined in Section 4.4, with k-layer corresponding to a $(2k-1) \times (2k-1)$ matrix. We vary the layer from 0 (indicating no neighborhood aggregation) to 2, 4, 6.

The results can be observed in Figure 9. The improvements from 0 to 4 layers indicate the effectiveness of high-order information. However, with an increase to 6 layers, a marginal decline in performance may occur, possibly due to the introduction of redundant information. This observation aligns with the findings from the ablation experiments, highlighting a heightened sensitivity to high-order structural information, particularly on the UCI dataset where its influence is most prominent. Moreover, it is worth noting that despite significant differences in node scales across the three datasets, the layer parameter yielding optimal results remains consistent. This consistency substantiates the adaptive nature of the MemMap structure to the feature space, successfully maintaining the information content of the grid within an appropriate range.

C.3 The influence of update weight.

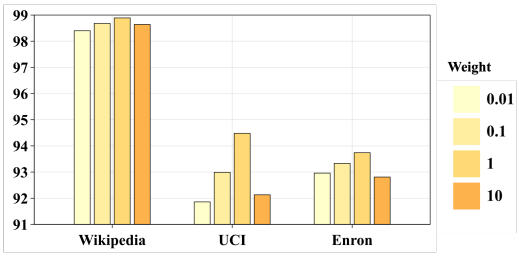

Figure 10: Effect of update weight in AUC-ROC.

The update weight w_0 plays a crucial role in memory map updating as it dictates the scale and extent of propagation in Equation (9). We systematically vary the update weight across four orders of magnitude: 0.01, 0.1, 1, and 10.

The results, displayed in Figure 10, notably reveal the model's heightened sensitivity to this particular parameter compared to others. Achieving the optimal value significantly influences the overall performance, with deviations in either direction leading to a decline in results. This sensitivity can be attributed to the fact that an excessively large parameter restricts the propagation range and diminishes the update intensity, making it challenging for the map to establish a global topological order. In such cases, each cell operates independently. Conversely, an excessively small parameter broadens the propagation range excessively, and the global propagation may overshadow inherent information, hindering the acquisition of effective information. This underscores the critical importance of establishing a global topological order on the map.