DyG-Mamba: Continuous State Space Modeling on Dynamic Graphs

Dongyuan Li
The University of Tokyo
lidy@csis.u-tokyo.ac.jp

Shiyin Tan
Institute of Science Tokyo
tanshiyin@lr.pi.titech.ac.jp

Ying Zhang
RIKEN AIP
ying.zhang@riken.jp

Ming Jin
Griffith University
mingjinedu@gmail.com

Shirui Pan
Griffith University
s.pan@griffith.edu.au

Okumura Mamabu
Institute of Science Tokyo
oku@pi.titech.ac.jp

Renhe Jiang*
The University of Tokyo
jiangrh@csis.u-tokyo.ac.jp

Abstract

Dynamic graph learning aims to uncover evolutionary laws in real-world systems, enabling accurate social recommendation (link prediction) or early detection of cancer cells (classification). Inspired by the success of state space models, *e.g.*, Mamba, for efficiently capturing long-term dependencies in language modeling, we propose DyG-Mamba, a new continuous state space model (SSM) for dynamic graph learning. Specifically, we first found that using inputs as control signals for SSM is not suitable for continuous-time dynamic network data with irregular sampling intervals, resulting in models being insensitive to time information and lacking generalization properties. Drawing inspiration from the Ebbinghaus forgetting curve, which suggests that memory of past events is strongly correlated with time intervals rather than specific details of the events themselves, we directly utilize irregular time spans as control signals for SSM to achieve significant robustness and generalization. Through exhaustive experiments on 12 datasets for dynamic link prediction and dynamic node classification tasks, we found that DyG-Mamba achieves state-of-the-art performance on most of the datasets, while also demonstrating significantly improved computation and memory efficiency.

Keywords

Continuous-Time Dynamic Graphs, Temporal Link Prediction, State Space Models.

ACM Reference Format:
Dongyuan Li, Shiyin Tan, Ying Zhang, Ming Jin, Shirui Pan, Okumura Mamabu, and Renhe Jiang. 2018. DyG-Mamba: Continuous State Space Modeling on Dynamic Graphs. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 9 pages. https://doi.org/XXXXXXX.XXXXXXX

*Corresponding author.

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Dynamic graph learning aims to understand the node and link evolution laws behind many complex systems, *e.g.*, recommender systems [52, 54], traffic systems [2, 14], and social networks [1, 26, 43]. Despite the great success of current methods, there is still a core limitation: *existing methods lack the ability to efficiently and effectively track long-term temporal dependencies.* Specifically, RNN-based methods, *e.g.*, JODIE [26] and TGN [39], recurrently update the representation of current nodes by incorporating historical information. Although they efficiently capture long-term temporal dependencies, they suffer from vanishing/exploding gradients, leading to unsatisfactory performance. In contrast, Transformer-based models, *e.g.*, SimpleDyG [50] and DyGFormer [53], can effectively capture long-term dependencies but struggle with expensive computation costs, *i.e.*, the complexity of the self-attention mechanism grows quadratically as the size of input increases. Although many efforts have been dedicated to enhancing the efficiency of self-attention, *e.g.*, patching [21] or convolution [53], these methods inevitably make trade-offs between effectiveness and efficiency, thereby limiting their capability to establish long-term dependencies. Other methods using multi-layer perceptrons (MLPs) or graph neural networks (GNNs), *e.g.*, GraphMixer [6], FreeDyG [45], and TGAT [49], focus on short-term dependencies, and their performance often decreases as the sequence length increases [6].

State space models (SSMs) seem to be a suitable framework to solve this problem, as they have shown great potential for long-term sequential modeling with linear time complexity [13, 15]. In particular, compared to previous SSMs like Hippo [11] and S4 [12], Mamba [10] sets parameters as input data-dependent variables to selectively copy previous hidden states, achieving better performance than Transformers in various tasks [37, 44, 55]. However, when vanilla Mamba is applied for dynamic graph learning, Figure 1(a) shows that Mamba significantly underperforms the Transformer-based DyGFormer. We explore three main reasons that hindered the application of Mamba to dynamic graphs. *Firstly, Mamba lacks sufficient utilization of time information with irregular time intervals.* The occurrence of events in dynamic graphs often follows different periodic patterns, resulting in irregular interaction timestamps. Effectively uncovering the underlying regularities in irregular temporal data can enhance dynamic graph learning capabilities [22]. However, as shown in Figure 1(b-d), when the time information is removed, only the Transformer-based DyGFormer shows a performance drop in most cases. It indicates that most previous studies including Mamba lack the ability to leverage irregular temporal

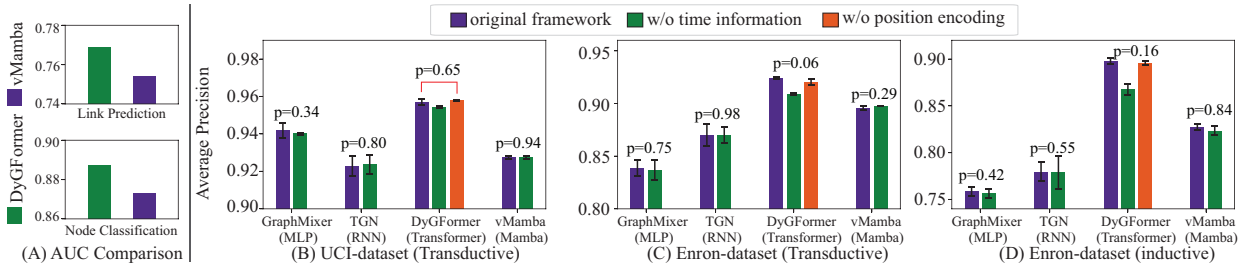

Figure 1: Experimental results for models with and w/o time information, and also for DyGFormer with positional encoding. $p = *$ indicates the p-value in the Student's t-test [34]. vMamba represents the vanilla Mamba.

information [22]. To further explore what DyGFormer learns from irregular time intervals, we simply replace time encoding with positional encoding [47], *i.e.*, replaces irregular time-spans as regular. We found that DyGFormer shows a slight performance drop. This indicates that it uses limited irregular time information to indicate sequence order. *Secondly, Mamba's data-dependent selection leads to poor generalization for unseen nodes in inductive settings.* As shown in Figure 1(c-d), compared to DyGFormer, we observed that Mamba exhibits a more obvious performance degradation in inductive settings than in transductive settings. Because Mamba set core parameters related to input data for selective copy, which limits its ability to effectively predict interactions between nodes that are unseen during the training process. *Thirdly, Mamba cannot support interactions between two input sequences.* For dynamic graph learning, many tasks require considering the semantic relatedness between two temporal neighborhoods (*i.e.*, history behaviors), which may also be a causal factor for the target interaction [48]. For example, in a temporal friendship network, if A and B are both friends with C, this can prompt a potential new friendship between A and B. Therefore, modeling mutual influences between two temporal neighborhoods can help in creating informative dynamic representations.

To address these issues, we propose DyG-Mamba, a new architecture for dynamic graph learning. Firstly, drawing inspiration from the Ebbinghaus Forgetting Curve theory [7], which suggests that "*human beings adhere to the same forgetting pattern for most things, which is strongly correlated with time rather than the content*", we directly change the data-dependent parameter to a learnable time-spans-dependent parameter that automatically learns the periodicity of historical events with irregular time intervals and balances the aggregation between historical state and current input. Secondly, we theoretically and empirically demonstrate that among Mamba's three data-dependent input parameters, the step size parameter Δ is the primary cause of its poor generalization. This parameter needs to be replaced with one that is time-span dependent. The other two parameters, $\boldsymbol{B}$ and $\boldsymbol{C}$, are crucial for assessing the significance of historical states and should remain data-dependent. Finally, to enhance dynamic link prediction without adding extra computational overhead, we designed a linear cross-attention layer on top of the DyG-Mamba layer. This addition improves the model's performance by better supporting the interaction of nodes in historical event sequences. The contributions are summarized as follows:

- To the best of our knowledge, we are the first to introduce SSMs for dynamic graph learning, achieving high efficiency and effectiveness in handling long-term temporal dependencies.
- We theoretically and empirically analyze the role of three core parameters of Mamba and replace one data-dependent parameter Δ with a learnable time-span-dependent parameter to better utilize irregular time information and enhances model generalization.
- We design a linear cross-attention architecture for DyG-Mamba to facilitate interaction between two historical event sequences, improving effectiveness without sacrificing efficiency.
- Extensive experimental results on 12 open datasets, including dynamic link prediction and node classification tasks, show that DyG-Mamba achieves SOTA performance with superior generalization and robustness. DyG-Mamba achieves higher precision with linear time and memory complexity than DyGFormer.

2 Related Work

Representation learning on dynamic graphs has recently attracted great attention [8, 19, 29, 30]. *Discrete-time methods* manually divide the dynamic graph into a sequence of snapshots with different resolutions (one day/an hour) and then combine GNNs (snapshot encoder) with recurrent models (dynamic tracker) to learn the representation of nodes [5, 36, 40, 41, 48, 51]. Their main common drawback is the necessity to predetermine the time granularity to create snapshots, ignoring the fine-grained temporal order of nodes/edges within each snapshot [49, 53]. In contrast, *continuous-time methods* directly use timestamps for representation learning. Based on the neural architectures, they can be classified into four classes, including *RNN-based methods*, *e.g.*, JODIE and TGN, *GNN-based methods*, *e.g.*, TGAT and DySAT [40], *MLP-based methods*, *e.g.*, GraphMixer and FreeDyG [45], and *Transformer-based methods*, *e.g.*, SimpleDyG [50] and DyGFormer. Additional techniques, such as ordinary differential equations [31, 32], random walks [22, 49], and temporal point process [18, 20], are also incorporated to learn continuous temporal information. Table 1 provides a detailed comparison between our method and the SOTAs including JODIE [26], DyRep [46], TGN [39], TGAT [49], CAWN [49], EdgeBank [38], TCL [48], GraphMixer [6], and DyGFormer [53], from the following angles: if the method could effectively handle unseen nodes during training (*i.e.*, inductive), capture long-term dependencies with both time and memory efficiency, exhibit robustness against noise, and effectively leverage irregular time-span information.

Originating from control systems [23, 27], SSMs inspire great attention for long-sequence modeling with HiPPO initialization [11, 13]. To enhance computational efficiency, the structured state space models (S4) normalize the parameters into diagonal structure [12]. Since then, many flavors of S4 sprang up, *e.g.*, parallel scan (S5) [42],

Table 1: Comparison of Dynamic Graph Baselines. With a batch size of 200 and a sequence length of 512, a model is considered time and memory efficient if the running time and memory usage are less than GraphMixer, *i.e.*, running time 250 seconds and memory usage 30,000 MB. Adding 50% noisy temporal edges, a performance drop of less than 10% indicates robustness.

	JODIE	DyRep	TGN	TGAT	CAWN	EdgeBank	TCL	GraphMixer	DyGFormer	DyG-Mamba
Inductive	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓
Long-Term Dependency Capability	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓
Time Efficient	✓	✗	✗	✗	✗	✓	✓	✓	✗	✓
Memory Efficient	✓	✗	✗	✗	✗	✓	✗	✓	✗	✓
Noise Robust	✗	✗	✓	✗	✗	✓	✗	✗	✓	✓
Irregular Time-Span Supportive	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓

shift SSM for linear attention (H3) [9] and diagonal structure design for parameters [16]. To remove the linear time-invariance constraint for better generalization, Mamba introduces a data-dependent selection mechanism into S4 to capture long-range context with increasing sequence length and a hardware-aware algorithm for efficient implementation [10], outperforming Transformers on various benchmarks [37, 44, 55]. Then, Graph Mamba [3] present a framework for a new class of GNNs based on selective SSMs. STG-Mamba [28] proposes Kalman Filtering GNNs for the adaptive enhancement of the graph structure of snapshots and adopts Mamba layers to capture the dynamic evolution of spatial-temporal graphs.

3 Preliminary

Dynamic Graph Learning. *Dynamic graphs* can be modeled as a sequence of non-decreasing chronological interactions $\mathcal{G} = \{(u_1, v_1, t_1), \ldots, (u_\tau, v_\tau, \tau)\}$ with $0 \le t_1 \le \cdots \le \tau$, where $u_i, v_i \in \mathcal{V}$ denote the source and destination nodes of the i-th link and $\mathcal{V}$ denote all nodes. Each node is associated with a node feature $\boldsymbol{x} \in \mathbb{R}^{d_V}$ and each interaction has a link feature $\boldsymbol{e}^t \in \mathbb{R}^{d_E}$, where d_V and d_E denote the dimensions of the node and the link features. Given the source node u, destination node v, timestamp t, and historical interactions before t, *i.e.*, $\{(u', v', t') \,|t' < t\}$, *dynamic graph learning* aims to learn time-aware representations $\boldsymbol{h}_u^t$ and $\boldsymbol{h}_v^t$ for nodes u and v. We validate the learned representations via two common tasks: (i) *dynamic link prediction*, which predicts whether u and v are connected at t; and (ii) *dynamic node classification*, which infers the class of u and v at t.

State Space Models. SSMs [10, 33, 37] define a linear mapping from input $\boldsymbol{u}(t) \in \mathbb{R}^d$ to output $\boldsymbol{y}(t) \in \mathbb{R}^d$ through a state-variable $\boldsymbol{h}(t) \in \mathbb{R}^{m \times d}$, formulated by:

$$\boldsymbol{h}'(t) = \boldsymbol{A}\boldsymbol{h}(t) + \boldsymbol{B}\boldsymbol{u}(t), \quad \boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{h}(t) + D\boldsymbol{u}(t), \tag{1}$$

where $\boldsymbol{A} \in \mathbb{R}^{m \times m}$, $\boldsymbol{B}, \boldsymbol{C} \in \mathbb{R}^m$ are the weighting trainable parameters, and D always equals to 0. For application to a discrete input sequence $(\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_\tau)$ instead of a continuous function, Eq.(1) can be discredited with a step size Δ, indicating the input's resolution. Following [10], we consider to discretize SSMs using the zero-order hold (ZOH) discretization rule, which is formulated by:

$$\begin{aligned} &\boldsymbol{h}_t = \overline{\boldsymbol{A}}\boldsymbol{h}_{t-1} + \overline{\boldsymbol{B}}\boldsymbol{u}_t, \quad \boldsymbol{y}_t = \boldsymbol{C}\boldsymbol{h}_t, \\ &\text{where } \overline{\boldsymbol{A}} = \exp(\Delta\boldsymbol{A}), \quad \overline{\boldsymbol{B}} = (\Delta\boldsymbol{A})^{-1}(\exp(\Delta\boldsymbol{A}) - \boldsymbol{I})(\Delta\boldsymbol{B}). \end{aligned} \tag{2}$$

By transforming the parameters from $(\Delta, \boldsymbol{A}, \boldsymbol{B})$ to $(\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}})$, the SSM model becomes a sequence-to-sequence mapping framework from discrete input $\{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_\tau\}$ to output $\{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_\tau\}$.

4 Methodology

The overview of DyG-Mamba is shown in Figure 2(a), with two DyG-Mamba frameworks for dynamic link prediction in Figure 2(b) and node classification in Figure 2(c). To help readers better understand, we take the dynamic link prediction task as an instantiation. Specifically, to predict the interaction between nodes u and v at timestamp τ, we first extract first-hop interaction sequences of nodes u and v before τ. Next, in addition to computing the encoding of nodes, links, time, and co-neighbor frequencies, we also compute the encoding of normalized time-spans between any two continuous timestamps as control signals for the continuous SSM. Finally, the outputs are averaged to derive representations of u and v at timestamp τ (*i.e.*, $\boldsymbol{h}_u^\tau$ and $\boldsymbol{h}_v^\tau$) for the dynamic link prediction.

4.1 Continuous-Time Dynamic Graph Encoding

Node, Edge, and Time Encodings. Most previous methods require nodes' historical interactions from multiple hops for dynamic graph learning [39, 46]. Unlike them, we learn only from the nodes' historical first-hop interactions, turning dynamic graph learning into simpler sequence learning problems. Specifically, the first-hop interaction sequence of length n_u for node u before time τ is defined as $S_u^\tau = \{(u, k_1, t_1), \ldots, (u, k_{n_u}, t_{n_u}) | t_{n_u} < \tau\}$. Then, the encoding of S_u^τ includes node encodings $\boldsymbol{X}_{u,V}^\tau = \{\boldsymbol{v}_{k_1}, \ldots, \boldsymbol{v}_{k_{n_u}}\} \in \mathbb{R}^{n_u \times d_V}$, edge encodings $\boldsymbol{X}_{u,E}^\tau = \{\boldsymbol{e}_{k_1} \ldots, \boldsymbol{e}_{k_{n_u}}\} \in \mathbb{R}^{n_u \times d_E}$, and time encodings $\boldsymbol{X}_{u,T}^\tau = \{\boldsymbol{t}_1, \ldots, \boldsymbol{t}_{n_u}\} \in \mathbb{R}^{n_u \times d_T}$, where d_* ($* \in \{E, V, T\}$) are the dimensions of the encodings. If the graph is non-attributed, we simply set the node feature and link feature to zero vectors. For time encodings, we use angular frequency features $\boldsymbol{\omega} = \{\alpha^{-(i-1)/\beta}\}_{i=1}^{d_T}$ to encode the relative time intervals $\Delta t_j = \tau - t_j$ by the encoding function $\cos(\boldsymbol{\omega}\Delta t_j)$ into a d_T-dimensional vector. α and β are parameters that make $\Delta t_{\max} \times \alpha^{-(i-1)/\beta} \to 0$ when $i \to d_T$, and the cosine function helps project $\boldsymbol{\omega}\Delta t$ into $[-1, +1]$. $\boldsymbol{\omega}$ remains constant during training to facilitate easier model optimization.

Co-occurrence Frequency Encodings. To consider the potential semantic relatedness between nodes' historical interactions, we employ co-occurrence frequency encodings, following [53]. Formally, let the historical interactions of u and v be $\{a, b, v\}$ and $\{b, b, c, a\}$. The frequency of a, b, c, and v in two sequences is $[1, 1]$, $[1, 2]$, $[0, 1]$, and $[1, 0]$, respectively. The co-occurrence features of u and v could be denoted by $\boldsymbol{C}_u^\tau = [[1, 1], [1, 2], [1, 0]]^\top$ and $\boldsymbol{C}_v^\tau = [[1, 2], [1, 2], [0, 1], [1, 1]]^\top$. Then, we apply a function $f(\cdot) : \mathbb{R}^1 \to \mathbb{R}^{d_C}$ to encode the co-occurrence features by:

$$\boldsymbol{X}_{*,C}^\tau = (f(\boldsymbol{C}_*^\tau[:, 0]) + f(\boldsymbol{C}_*^\tau[:, 1]))\boldsymbol{W}_C + \boldsymbol{b}_C, \tag{3}$$

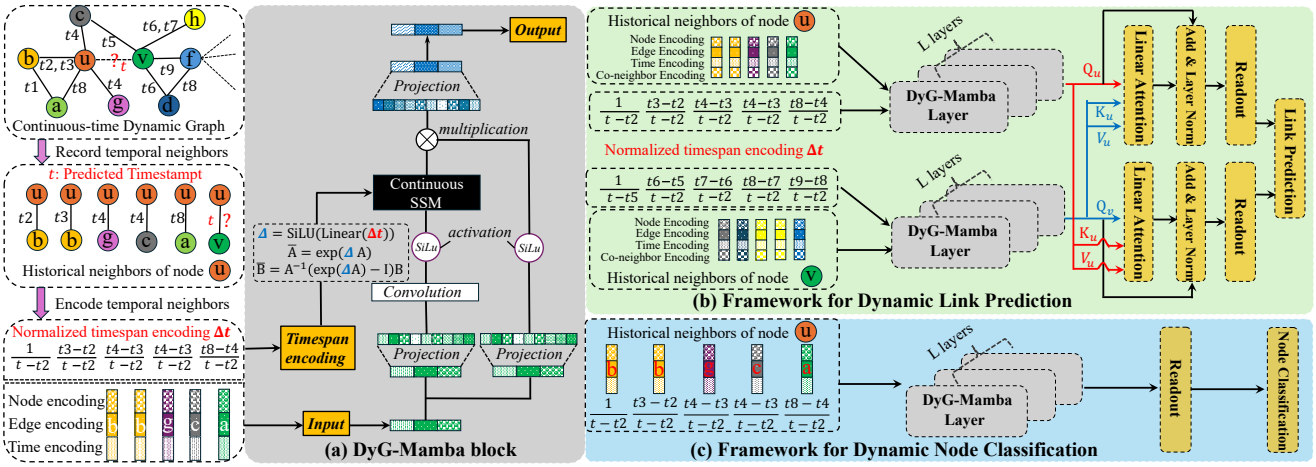

Figure 2: Overview of the DyG-Mamba (a) and its frameworks for downstream tasks (b, c).

where $\boldsymbol{X}^{\tau}_{*,C} \in \mathbb{R}^{n_* \times d_C}$, $* \in \{u, v\}$ d_C is the dimension of the co-occurrence encoding, and $\boldsymbol{W}_C$ and $\boldsymbol{b}_C$ are trainable parameters. We implement $f(\cdot)$ by two-layer perception with ReLU activation [35].

Encoding Alignment. We align the encodings to the same dimension d with trainable weight $\boldsymbol{W}_* \in \mathbb{R}^{d_* \times d}$ and $b_* \in \mathbb{R}^d$ to obtain $\boldsymbol{Z}^{\tau}_{u,*} \in \mathbb{R}^{n \times d}$, formulated by:

$$\boldsymbol{Z}^{\tau}_{u,*} = \boldsymbol{X}^{\tau}_{u,*}\boldsymbol{W}_* + \boldsymbol{b}_*, \quad \text{where } * \in \{N, E, T, C\}. \tag{4}$$

Finally, we concatenate $\boldsymbol{Z}^{\tau}_u = \boldsymbol{Z}^{\tau}_{u,V} \| \boldsymbol{Z}^{\tau}_{u,E} \| \boldsymbol{Z}^{\tau}_{u,T} \| \boldsymbol{Z}^{\tau}_{u,C}$ as the aligned embedding for u with $\boldsymbol{Z}^{\tau}_u \in \mathbb{R}^{n_u \times 4d}$.

4.2 SSMs for Dynamic Graph Modeling

In Eq.(2), DyG-Mamba has three core parameters: Δ, $\boldsymbol{B}$, and $\boldsymbol{C}$. Δ determines how much historical memory is forgotten, and $\boldsymbol{B}$ and $\boldsymbol{C}$ measure the similarity between previous inputs and the target output. In this section, we theoretically analysis their roles. We first introduce the definition of the parameter $\boldsymbol{\Delta}$ in Theorem 1.

Theorem 1. *For a general formulation of the linear time-invariant ODE of the form* $\frac{d}{dt}\boldsymbol{h}(t) = \boldsymbol{A}\boldsymbol{h}(t) + \boldsymbol{B}\boldsymbol{u}(t)$ *for some input function* $\boldsymbol{u}(t)$*, the general method for discretizing the ODE with each step size* Δ*, can be calculated in closed-form between t and t+*Δ *as* $\boldsymbol{h}(t+\Delta) = e^{\Delta\boldsymbol{A}}\boldsymbol{h}(t) + \left(\int_{s=0}^{\Delta} e^{s\boldsymbol{A}}\,\mathrm{d}s\right)\boldsymbol{B}\boldsymbol{u}(t)$ *if the control input* $\boldsymbol{u}(t)$ *remains constant between t and t+*Δ*. If* $\boldsymbol{A}$ *is invertible,* $\boldsymbol{h}(t+\Delta) = e^{\Delta\boldsymbol{A}}\boldsymbol{h}(t) + (\Delta\boldsymbol{A})^{-1}(e^{\Delta\boldsymbol{A}} - I)(\Delta\boldsymbol{B})\boldsymbol{u}(t)$.

Mamba [10] focuses on language sequences, which assumes that successive word pairs have the same interval, *e.g.*, $\Delta = 1$ in Theorem 1. To selectively copy some previous inputs, Mamba takes the current input $\boldsymbol{u}(t)$ as control signals to replace the previous fixed Δ with $\Delta = \text{SiLU}(\text{Linear}(\boldsymbol{u}(t)))$, where SiLU is an activation function. However, when applied to dynamic graphs, Mamba cannot effectively use irregular time information (Figure 1), and its data-dependent strategy leads to poor performance in inductive scenarios (w/o Time-Span vs. DyG-Mamba in Figure 3). The Ebbinghaus Forgetting Curve theory [7] suggests that historical memory is strongly correlated with time-spans between events rather than events themselves. Considering dynamic graphs naturally have irregular time intervals, we can use time-spans between any two timestamps t_{i-1} and t_i as control signals, formulated by:

$$\Delta_i = \text{SiLU}\left(\text{Linear}\left(\cos\left(\boldsymbol{\omega}\left(t_{i+1} - t_i\right) / \left(\tau - t_1\right)\right)\right)\right), \tag{5}$$

where Linear represents linear layers and we set the first normalized time span as $\Delta t_1 = 1/(\tau - t_1)$. We then take $\boldsymbol{Z}^{\tau}_u$ as input for DyG-Mamba and the time-span sequence $\{\Delta_i\}_{i=1}^{n}$ as control signals for continuous SSMs. $\boldsymbol{Z}^{\tau}_u$ first passes through a linear layer followed by a 1D convolution layer and SiLU activation function, formulated by

$$\boldsymbol{M}^{\tau}_u = \text{SiLU}\left(\text{Conv1D}\left(\text{Linear}\left(\boldsymbol{Z}^{\tau}_u\right)\right)\right). \tag{6}$$

Then, we initialize $\boldsymbol{A}$ as a dialogue matrix based on [15] and set $\{\Delta_i\}_{i=1}^{n}$ according to Eq.(5). We define $\boldsymbol{B} = \text{Linear}_B(\boldsymbol{M}^{\tau}_u)$ and $\boldsymbol{C} = \text{Linear}_C(\boldsymbol{M}^{\tau}_u)$. $\boldsymbol{M}^{\tau}_u$ can pass through the continuous SSM layer to learn new representations, which could be formulated by:

$$\widehat{\boldsymbol{Z}}^{\tau}_u = \text{SSM}(\boldsymbol{M}^{\tau}_u, \boldsymbol{B}, \boldsymbol{C}, \{\Delta\}_{i=1}^{n}), \tag{7}$$

where the k-th generated output of $\widehat{\boldsymbol{Z}}^{\tau}_u$ could be formulated by:

$$\boldsymbol{h}_k = \overline{\boldsymbol{A}}_k\boldsymbol{h}_{k-1} + \overline{\boldsymbol{B}}_k\boldsymbol{u}_k, \quad \widehat{\boldsymbol{z}}^{\tau}_k = \overline{\boldsymbol{C}}_k\boldsymbol{h}_k, \tag{8}$$

where $\overline{\boldsymbol{A}}_k = \exp(\Delta_k\boldsymbol{A})$, $\overline{\boldsymbol{B}}_k = (\Delta_k\boldsymbol{A})^{-1}(\exp(\Delta_k\boldsymbol{A}) - \boldsymbol{I})(\Delta_k\boldsymbol{B})$ and $\overline{\boldsymbol{C}}_k = \boldsymbol{C}_k$. To theoretically demonstrate that time-span encoding can control the importance between historical memory and current input, we first give the following Theorem 2.

Theorem 2. *Let* $\boldsymbol{A}$ *be diagonalizable as* $\boldsymbol{A} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{-1}$ *with eigenvalues* $\{\lambda_1, \ldots, \lambda_N\}$*. Given* Δ_k*,* $\overline{\boldsymbol{A}}_k = \text{diag}(e^{\lambda_1\Delta_k}, \ldots, e^{\lambda_N\Delta_k})$ *and* $\overline{\boldsymbol{B}}_k = \left(\lambda_1^{-1}(e^{\lambda_1\Delta_k} - 1), \ldots, \lambda_N^{-1}(e^{\lambda_N\Delta_k} - 1)\right)$*, the i-th coordinate of* $\boldsymbol{h}_k$ *in Eq.(8) can be denoted as* $h_{i,k} = e^{\lambda_i\Delta_k}h_{i,k-1} + \lambda_i^{-1}(e^{\lambda_i\Delta_k} - 1)u_{i,k}$.

Following [15], the real part of the elements of λ_i is restricted to be negative. According to Theorem 2, if Δ_k is small enough, we obtain $|\lambda_i|\Delta_k \approx 0$, *i.e.*, $h_{i,k} \approx h_{i,k-1}$, demonstrating that a small time-span Δ_k persists the historical state and ignores the current input. And a larger Δ_k makes $\lambda_i\Delta_k \ll 0$ and $h_{i,k} \approx -\lambda_i^{-1}u_k$, *i.e.*, the information from the previous timesteps would be forgotten, similar to a "forget" gate in LSTM [17].

We then show how the parameters $\boldsymbol{B}$ and $\boldsymbol{C}$ selectively copy previous inputs according to Theorem 3.

Theorem 3. *Parameters $\boldsymbol{B}$ and $\boldsymbol{C}$ can help selectively copy the previous input through the causal attention mechanism. Considering $\boldsymbol{u}_k$ as k-th column of $\boldsymbol{M}_u^\tau$, Eq.(8) can be further decomposed as follows:*

$$\hat{z}_k^\tau = \overline{C}_k \prod_{i=0}^{k-2} \overline{A}_{k-i} \overline{B}_1 \boldsymbol{u}_1 + \cdots + \overline{C}_k \prod_{i=0}^{k-j} \overline{A}_{k-i} \overline{B}_{j-1} \boldsymbol{u}_{j-1} + \cdots + \overline{C}_k \overline{B}_k \boldsymbol{u}_k,$$
$$= e^{(\sum_{i=0}^{k-2} \Delta_{k-i} \boldsymbol{A})} \overline{C}_k \overline{B}_1 \boldsymbol{u}_1 + \cdots + e^{(\sum_{i=0}^{k-j-1} \Delta_{k-i} \boldsymbol{A})} \overline{C}_k \overline{B}_j \boldsymbol{u}_j + \cdots + \overline{C}_k \overline{B}_k \boldsymbol{u}_k,$$

where $\overline{C}_k$ can be considered as the query of k-th input $\boldsymbol{u}_k$ and $\overline{B}_j$ can be considered as key of $\boldsymbol{u}_j$. Thus, $\overline{C}_k \overline{B}_j$ can measure the similarity.

Finally, after SSM layer, the output can be formulated by:

$$Z_{u,out}^\tau = (\widehat{Z}_u^\tau \odot \text{SiLU}(\text{Linear}(Z_u^\tau)))\boldsymbol{W}_{out} + \boldsymbol{b}_{out}, \tag{9}$$

where $\odot$ is element-wise dot product, and $\boldsymbol{W}_{out}$ and $\boldsymbol{b}_{out}$ are trainable parameters.

4.3 DyG-Mamba for Downstream Tasks

Temporal Link Prediction. In Figure 2(b), we employ a two-stream DyG-Mamba encoder to individually deal with each sequence. Inspired by linear attention [24], which reduces time and memory complexity, we then interact the two streams at the semantic level through a linear cross-attention layer, formulated by:

$$\text{Linear Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \frac{\phi(Q_i)^\top \sum_{j=1}^{i} \phi(K_j) V_j^\top}{\phi(Q_i)^\top \sum_{j=1}^{i} \phi(K_j)}, \tag{10}$$

$$\boldsymbol{Q}_*^\tau, \boldsymbol{K}_*^\tau, \boldsymbol{V}_*^\tau = Z_{*,out}^\tau \boldsymbol{W}_Q,\ Z_{*,out}^\tau \boldsymbol{W}_K,\ Z_{*,out}^\tau \boldsymbol{W}_V,\quad * \in \{u, v\}, \tag{11}$$

$$\boldsymbol{O}_*^\tau = \text{Linear Attention}(\boldsymbol{Q}_*^\tau, \boldsymbol{K}_*^\tau, \boldsymbol{V}_*^\tau), * \in \{u, v\}, \tag{12}$$

$$\boldsymbol{H}_*^\tau = \text{LayerNorm}\left(\text{Linear}\left(\boldsymbol{O}_*^\tau + \boldsymbol{Q}_*^\tau\right)\right), * \in \{u, v\}, \tag{13}$$

where $\phi(x) = \text{elu}(x) + 1$ and $\text{elu}(\cdot)$ is an activation function [4]. The co-attention operation in Eq.(12) allows the encoder to emphasize relevant shared semantics and suppress irrelevant ones. We use readout function $\mathbb{R}^{L \times 4d} \to \mathbb{R}^{1 \times 4d}$, *i.e.*, MEAN pooling, to obtain node embeddings $\boldsymbol{h}_*^\tau = \text{MEAN}(\boldsymbol{H}_*^\tau)$ with $* \in \{u, v\}$. Finally, we use an MLP to concatenate representations of two nodes as inputs and return the predicted probability $\hat{y}$ as output, formulated by:

$$\hat{y} = \text{Softmax}(\text{Linear}(\text{RELU}(\text{Linear}(\boldsymbol{h}_u^\tau \| \boldsymbol{h}_v^\tau)))). \tag{14}$$

We adopt binary cross-entropy loss for dynamic link prediction.

Temporal Node Classification. After obtaining sequential node representations $\boldsymbol{H}_{u,out}^\tau \in \mathbb{R}^{L \times 3d}$ without co-neighbor encoding, node embeddings are calculated by $\boldsymbol{h}_u^\tau = \text{MEAN}(\boldsymbol{H}_{u,out})$, and we use the cross-entropy loss function for node classification.

Computational Efficiency. Assuming the batch size is b, the feature dimension is d, and the sequence length is L, the memory and time complexities of DyG-Mamba are $O(bLd)$. DyGFormer exhibits quadratic memory and time complexities as $O(bL^2d)$, showing the efficiency of DyG-Mamba.

5 Experiments

Datasets and Baselines. We evaluate model's performance on 12 datasets, ranging from social networks to transportation networks [38]. We split each dataset with the ratio of 70%/15%/15% for training/validation/testing. We select nine best-performing dynamic graph learning baselines, including RNN-based methods: JODIE, DyRep, TGN and CAWN, a GNN-based method: TGAT, a memory-based method: EdgeBank, a MLP-based method: GraphMixer, and Transformer-based methods: TCL and DyGFormer.

Evaluation Details and Metrics. *For dynamic link prediction*, following [38, 45, 53], we evaluate baselines with two settings: the *transductive setting* aims to predict future links between nodes observed during training, and the *inductive setting* predicts future links between unseen nodes. We use *AP* and *AUC-ROC* as the evaluation metrics. [38] found that random negative sampling may not provide a complete evaluation. Thus, following their studies, we adopt random (*rnd*), historical (*hist*), and inductive (*ind*) negative sampling for evaluation. *For dynamic node classification*, we estimate the state of a node in a given interaction at a specific time and use *AUC-ROC* as the evaluation metric.

Implementation Details. For a fair comparison, we use DyGLib [53] that reproduce all baselines via the same training/inference pipeline. We use the Adam optimizer [25] with a learning rate of 0.0001.We train the models for 100 epochs and use the early stopping strategy with a patience of 20. We select the model that achieves the best performance in the validation set for testing. We run methods five times and report the average performance to eliminate deviations.

Table 2: Performance on dynamic node classification.

Methods	Wikipedia	Reddit	Avg. Rank
JODIE	**88.99±1.05**	60.37±2.58	5.00
DyRep	86.39±0.98	63.72±1.32	6.00
TGAT	84.09±1.27	70.04±1.09	5.00
TGN	86.38±2.34	63.27±0.90	7.00
CAWN	84.88±1.33	66.34±1.78	6.00
EdgeBank	N/A	N/A	N/A
TCL	77.83±2.13	68.87±2.15	6.00
GraphMixer	86.80±0.79	64.22±3.32	5.00
DyGFormer	87.44±1.08	68.00±1.74	3.50
DyG-Mamba	88.58±0.92	**70.79±1.97**	**1.50**

5.1 Quantitative Evaluation

Performance on Dynamic Node Classification. Table 2 shows the AUC-ROC results on dynamic node classification. DyG-Mamba achieves SOTA performance on the Reddit dataset and second-best performance on the Wikipedia dataset. In addition, DyG-Mamba achieves the best average rank of 1.5 compared to the second-best DyGFormer with 3.5 AUC-ROC results for all baselines in Table 2.

Performance on Dynamic Link Prediction. To demonstrate the effectiveness and generalizability of DyG-Mamba, we perform evaluations in two different experimental settings: transductive and inductive settings. Table 3 reports the results in the inductive setting with *rnd* negative sampling. From these tables, we observe that DyG-Mamba achieves the best performance on 11 datasets and achieves a best average rank of 1.91/1.69 on AP/AUC across three negative sampling strategies, while the best-performing baseline DyGFormer achieves only 3.47/3.28. Table 4 shows the results in the transductive setting. Among ten models, DyG-Mamba achieves the best/second-best performance on 11, eight, and ten datasets in three

Table 3: AP for *inductive dynamic link prediction* with random negative sampling strategies. The best and second-best results are emphasized by bold and underlined fonts. Please note that N/A means Not Applicable.

Datasets	JODIE	DyRep	TGAT	TGN	CAWN	EdgeBank	TCL	GraphMixer	DyGFormer	DyG-Mamba
Wikipedia	94.82±0.20	92.43±0.37	96.22±0.07	97.83±0.04	98.24±0.03	N/A	96.22±0.17	96.65±0.02	98.59±0.03	**98.65±0.03**
Reddit	96.50±0.13	96.09±0.11	97.09±0.04	97.50±0.07	98.62±0.01	N/A	94.09±0.07	95.26±0.02	98.84±0.02	**98.88±0.00**
MOOC	79.63±1.92	81.07±0.44	85.50±0.19	89.04±1.17	81.42±0.24	N/A	80.60±0.22	81.41±0.21	86.96±0.43	**90.20±0.06**
LastFM	81.61±3.82	83.02±1.48	78.63±0.31	81.45±4.29	89.42±0.07	N/A	73.53±1.66	82.11±0.42	94.23±0.09	**95.13±0.08**
Enron	80.72±1.39	74.55±3.95	67.05±1.51	77.94±1.02	86.35±0.51	N/A	76.14±0.79	75.88±0.48	89.76±0.34	**91.14±0.07**
Social Evo.	91.96±0.48	90.04±0.47	91.41±0.16	90.77±0.86	79.94±0.18	N/A	91.55±0.09	91.86±0.06	93.14±0.04	**93.23±0.01**
UCI	79.86±1.48	57.48±1.87	79.54±0.48	88.12±2.05	92.73±0.06	N/A	87.36±2.03	91.19±0.42	**94.54±0.12**	94.15±0.04
Can. Parl.	53.92±0.94	54.02±0.76	55.18±0.79	54.10±0.93	55.80±0.69	N/A	54.30±0.66	55.91±0.82	87.74±0.71	**90.05±0.86**
US Legis.	54.93±2.29	57.28±0.71	51.00±3.11	58.63±0.37	53.17±1.20	N/A	52.59±0.97	50.71±0.76	54.28±2.87	**59.52±0.54**
UN Trade	59.65±0.77	57.02±0.69	61.03±0.18	58.31±3.15	65.24±0.21	N/A	62.21±0.12	62.17±0.31	64.55±0.62	**65.87±0.40**
UN Vote	56.64±0.96	54.62±2.22	52.24±1.46	58.85±2.51	49.94±0.45	N/A	51.60±0.97	50.68±0.44	55.93±0.39	**59.89±1.04**
Contact	94.34±1.45	92.18±0.41	95.87±0.11	93.82±0.99	89.55±0.30	N/A	91.11±0.12	90.59±0.05	98.03±0.02	**98.12±0.04**
Avg. Rank	5.83	6.91	6.20	4.83	4.91	N/A	6.70	6.00	2.50	**1.08**

Table 4: AP for *transductive dynamic link prediction* with random, historical, and inductive negative sampling strategies.

NSS	Datasets	JODIE	DyRep	TGAT	TGN	CAWN	EdgeBank	TCL	GraphMixer	DyGFormer	DyG-Mamba
rnd	Wikipedia	96.50 ± 0.14	94.86 ± 0.06	96.94 ± 0.06	98.45 ± 0.06	98.76 ± 0.03	90.37 ± 0.00	96.47 ± 0.16	97.25 ± 0.03	99.03 ± 0.02	**99.08 ± 0.09**
	Reddit	98.31 ± 0.14	98.22 ± 0.04	98.52 ± 0.02	98.63 ± 0.06	99.11 ± 0.01	94.86 ± 0.00	97.53 ± 0.02	97.31 ± 0.01	99.22 ± 0.01	**99.27 ± 0.00**
	MOOC	80.23 ± 2.44	81.97 ± 0.49	85.84 ± 0.15	89.15 ± 1.60	80.15 ± 0.25	57.97 ± 0.00	82.38 ± 0.24	82.78 ± 0.15	87.52 ± 0.49	**90.25 ± 0.09**
	LastFM	70.85 ± 2.13	71.92 ± 2.21	73.42 ± 0.21	77.07 ± 3.97	86.99 ± 0.06	79.29 ± 0.00	67.27 ± 2.16	75.61 ± 0.24	93.00 ± 0.12	**94.23 ± 0.01**
	Enron	84.77 ± 0.30	82.38 ± 3.36	71.12 ± 0.97	86.53 ± 1.11	89.56 ± 0.09	83.53 ± 0.00	79.70 ± 0.71	82.25 ± 0.16	92.47 ± 0.12	**93.14 ± 0.08**
	Social Evo.	89.89 ± 0.55	88.87 ± 0.30	93.16 ± 0.17	93.57 ± 0.17	84.96 ± 0.09	74.95 ± 0.00	93.13 ± 0.16	93.37 ± 0.07	94.73 ± 0.01	**94.77 ± 0.01**
	UCI	89.43 ± 1.09	65.14 ± 2.30	79.63 ± 0.70	92.34 ± 1.04	95.18 ± 0.06	76.20 ± 0.00	89.57 ± 1.63	93.25 ± 0.57	95.79 ± 0.17	**96.14 ± 0.14**
	Can. Parl.	69.26 ± 0.31	66.54 ± 2.76	70.73 ± 0.72	70.88 ± 2.34	69.82 ± 2.34	64.55 ± 0.00	68.67 ± 2.67	77.04 ± 0.46	97.36 ± 0.45	**98.20 ± 0.52**
	US Legis.	75.05 ± 1.52	75.34 ± 0.39	68.52 ± 3.16	**75.99 ± 0.58**	70.58 ± 0.48	58.39 ± 0.00	69.59 ± 0.48	70.74 ± 1.02	71.11 ± 0.59	73.66 ± 1.13
	UN Trade	64.94 ± 0.31	63.21 ± 0.93	61.47 ± 0.18	65.03 ± 1.37	65.39 ± 0.12	60.41 ± 0.00	62.21 ± 0.03	62.61 ± 0.27	66.46 ± 1.29	**68.51 ± 0.17**
	UN Vote	63.91 ± 0.81	62.81 ± 0.80	52.21 ± 0.98	**65.72 ± 2.17**	52.84 ± 0.10	58.49 ± 0.00	51.90 ± 0.30	52.11 ± 0.16	55.55 ± 0.42	64.74 ± 1.48
	Contact	95.31 ± 1.33	95.98 ± 0.15	96.28 ± 0.09	96.89 ± 0.56	90.26 ± 0.28	92.58 ± 0.00	92.44 ± 0.12	91.92 ± 0.03	98.29 ± 0.01	**98.38 ± 0.01**
	Avg. Rank	6.08	6.83	6.67	3.33	5.50	8.42	7.92	6.25	2.67	**1.33**
hist	Wikipedia	83.01 ± 0.66	79.93 ± 0.56	87.38 ± 0.22	86.86 ± 0.33	71.21 ± 1.67	73.35 ± 0.00	89.05 ± 0.39	**90.90 ± 0.10**	82.23 ± 2.54	82.35 ± 1.25
	Reddit	80.03 ± 0.36	79.83 ± 0.31	79.55 ± 0.20	81.22 ± 0.61	80.82 ± 0.45	73.59 ± 0.00	77.14 ± 0.16	78.44 ± 0.18	**81.57 ± 0.67**	81.02 ± 0.19
	MOOC	78.94 ± 1.25	75.60 ± 1.12	82.19 ± 0.62	87.06 ± 1.93	74.05 ± 0.95	60.71 ± 0.00	77.06 ± 0.41	77.77 ± 0.92	85.85 ± 0.66	**87.42 ± 1.57**
	LastFM	74.35 ± 3.81	74.92 ± 2.46	71.59 ± 0.24	76.87 ± 4.64	69.86 ± 0.43	73.03 ± 0.00	59.30 ± 2.31	72.47 ± 0.49	81.57 ± 0.48	**84.08 ± 0.45**
	Enron	69.85 ± 2.70	71.19 ± 2.76	64.07 ± 1.05	73.91 ± 1.76	64.73 ± 0.36	76.53 ± 0.00	70.66 ± 0.39	**77.98 ± 0.92**	75.63 ± 0.73	77.85 ± 1.20
	Social Evo.	87.44 ± 6.78	93.29 ± 0.43	95.01 ± 0.44	94.45 ± 0.56	85.53 ± 0.38	80.57 ± 0.00	94.74 ± 0.31	94.93 ± 0.31	**97.38 ± 0.14**	97.35 ± 0.18
	UCI	75.24 ± 5.80	55.10 ± 3.14	68.27 ± 1.37	80.43 ± 2.12	65.30 ± 0.43	65.50 ± 0.00	80.25 ± 2.74	**84.11 ± 1.35**	82.17 ± 0.82	81.36 ± 0.14
	Can. Parl.	51.79 ± 0.63	63.31 ± 1.23	67.13 ± 0.84	68.42 ± 3.07	66.53 ± 2.77	63.84 ± 0.00	65.93 ± 3.00	74.34 ± 0.87	97.00 ± 0.31	**97.39 ± 0.29**
	US Legis.	51.71 ± 5.76	86.88 ± 2.25	62.14 ± 6.60	74.00 ± 7.57	68.82 ± 8.23	63.22 ± 0.00	80.53 ± 3.95	81.65 ± 1.02	85.30 ± 3.88	**88.86 ± 1.40**
	UN Trade	61.39 ± 1.83	59.19 ± 1.07	55.74 ± 0.91	58.44 ± 5.51	55.71 ± 0.38	**81.32 ± 0.00**	55.90 ± 1.17	57.05 ± 1.22	64.41 ± 1.40	65.11 ± 0.19
	UN Vote	70.02 ± 0.81	69.30 ± 1.12	52.96 ± 2.14	69.37 ± 3.93	51.26 ± 0.04	**84.89 ± 0.00**	52.30 ± 2.35	51.20 ± 1.60	60.84 ± 1.58	61.17 ± 2.64
	Contact	95.31 ± 2.13	96.39 ± 0.20	96.05 ± 0.52	93.05 ± 2.35	84.16 ± 0.49	88.81 ± 0.00	93.86 ± 0.21	93.36 ± 0.41	97.57 ± 0.06	**97.76 ± 0.05**
	Avg. Rank	6.08	6.00	6.33	4.42	8.42	6.92	6.58	4.92	3.00	**2.33**
ind	Wikipedia	75.65 ± 0.79	70.21 ± 1.58	87.00 ± 0.16	85.62 ± 0.44	74.06 ± 2.62	80.63 ± 0.00	86.76 ± 0.72	**88.59 ± 0.17**	78.29 ± 5.38	87.06 ± 0.86
	Reddit	86.98 ± 0.16	86.30 ± 0.26	89.59 ± 0.24	88.10 ± 0.24	91.67 ± 0.24	85.48 ± 0.00	87.45 ± 0.29	85.26 ± 0.11	91.11 ± 0.40	**91.77 ± 0.46**
	MOOC	65.23 ± 2.19	61.66 ± 0.95	75.95 ± 0.64	77.50 ± 2.91	73.51 ± 0.94	49.43 ± 0.00	74.65 ± 0.54	74.27 ± 0.92	**81.24 ± 0.69**	81.19 ± 2.02
	LastFM	62.67 ± 4.49	64.41 ± 2.70	71.13 ± 0.17	65.95 ± 5.98	67.48 ± 0.77	**75.49 ± 0.00**	58.21 ± 0.89	68.12 ± 0.33	73.97 ± 0.50	75.05 ± 0.40
	Enron	68.96 ± 0.98	67.79 ± 1.53	63.94 ± 1.36	70.89 ± 2.72	75.15 ± 0.58	73.89 ± 0.00	71.29 ± 0.32	75.01 ± 0.79	77.41 ± 0.89	**77.46 ± 0.90**
	Social Evo.	89.82 ± 4.11	93.28 ± 0.48	94.84 ± 0.44	95.13 ± 0.56	88.32 ± 0.27	83.69 ± 0.00	94.90 ± 0.36	94.72 ± 0.33	97.68 ± 0.10	**97.78 ± 0.15**
	UCI	65.99 ± 1.40	54.79 ± 1.76	68.67 ± 0.84	70.94 ± 0.71	64.61 ± 0.48	57.43 ± 0.00	76.01 ± 1.11	**80.10 ± 0.51**	72.25 ± 1.71	77.75 ± 1.56
	Can. Parl.	48.42 ± 0.66	58.61 ± 0.86	68.82 ± 1.21	65.34 ± 2.87	67.75 ± 1.00	62.16 ± 0.00	65.85 ± 1.75	69.48 ± 0.63	95.44 ± 0.57	**97.29 ± 0.96**
	US Legis.	50.27 ± 5.13	83.44 ± 1.16	61.91 ± 5.82	67.57 ± 6.47	65.81 ± 8.52	64.74 ± 0.00	78.15 ± 3.34	79.63 ± 0.84	81.25 ± 3.62	**85.61 ± 1.66**
	UN Trade	60.42 ± 1.48	60.19 ± 1.24	60.61 ± 1.24	61.04 ± 6.01	62.54 ± 0.67	**72.97 ± 0.00**	61.06 ± 1.74	60.15 ± 1.29	55.79 ± 1.02	60.43 ± 1.59
	UN Vote	67.79 ± 1.46	**67.53 ± 1.98**	52.89 ± 1.61	67.63 ± 2.67	52.19 ± 0.34	66.30 ± 0.00	50.62 ± 0.82	51.60 ± 0.73	51.91 ± 0.84	60.05 ± 2.41
	Contact	93.43 ± 1.78	94.18 ± 0.10	94.35 ± 0.48	90.18 ± 3.28	89.31 ± 0.27	85.20 ± 0.00	91.35 ± 0.21	90.87 ± 0.35	**94.75 ± 0.28**	94.63 ± 0.06
	Avg. Rank	7.33	7.25	5.25	5.17	6.17	6.75	5.67	5.42	3.83	**2.17**

negative sampling strategies, respectively. DyG-Mamba achieves the best average rank of 1.94/2.05 on AP/AUC, while the best-performing baseline DyGFormer achieves 3.16/3.30. Furthermore, as Figure 6(A) shows DyG-Mamba outperforms DyGFormer in terms of averaged AUC scores, achieving improvements of 3.64% and 1.83% in dynamic link prediction and node classification.

Comparison of Time and Memory. To demonstrate the efficiency of DyG-Mamba, we present a comprehensive comparison with the best-performing Transformer-based backbone, DyGFormer. As

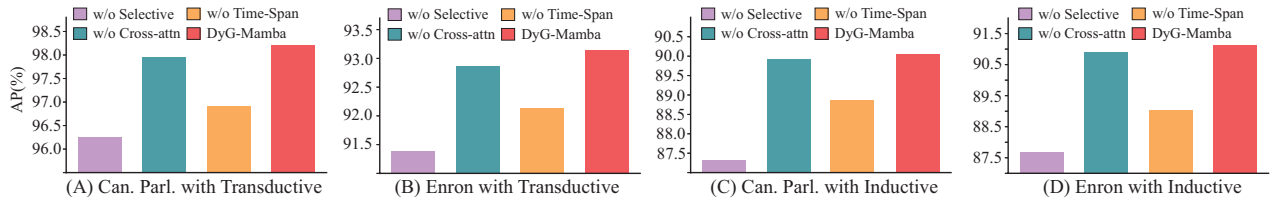

Figure 3: Ablation Studies for DyG-Mamba in both (A-B) transductive and (C-D) inductive settings.

shown in Figures 4(A)-(B), DyG-Mamba requires significantly less training time and memory usage than DyGFormer when handling long sequences. *Specifically, DyG-Mamba is 8.9 times faster than DyGFormer and saves 77.2% GPU memory with a sequence length of 2,048.* For a fair comparison, we do not use the patching technique for both DyGFormer and DyG-Mamba.

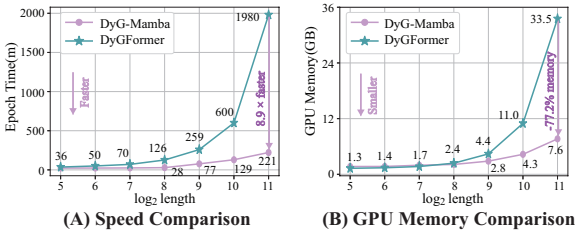

Figure 4: Comparison of two layers DyGFormer and two layers DyG-Mamba with varying lengths. (A) running time consumption and (B) GPU memory consumption.

Comparison of Training Time and Parameters. By setting the sequence length to 256, we conduct a comparative analysis of performance, training time per epoch (measured in seconds), and size of trainable parameters (measured in Megabyte, *i.e.*, MB) between DyG-Mamba and all baselines on the Enron dataset, as shown in Figure 5. Obviously, CAWN requires the longest training time and a substantial number of parameters, because it conducts random walks on dynamic graphs to collect time-aware sequences for dynamic graph learning. On the other hand, simpler methods, *e.g.*, the MLP-based GraphMixer and the RNN-based JODIE, have fewer parameters, but exhibit a significant performance gap compared to the best-performing DyGFormer and DyG-Mamba. Overall, DyG-Mamba achieves the best performance with a small size of trainable parameters and a moderate training time required per epoch.

Discussion and Analysis. From the above experimental results, we obtain the following three conclusions: *(i) Higher Effectiveness.* DyG-Mamba can achieve the best performance in temporal link prediction and node classification tasks, demonstrating that DyG-Mamba is more suitable for dealing with dynamic graph learning. The reason is that DyG-Mamba can better track long-term temporal dependencies, as detailed in Figure 7. Compared with the best-performing baseline DyGFormer, based on self-attention aggregation, DyG-Mamba based on the causal attention mechanism, which can capture the evolving nature of node representations and remove irrelevant historical neighbors for each timestamp (Figure 6). Finally, the cross-attention scheme also helps DyG-Mamba exploit

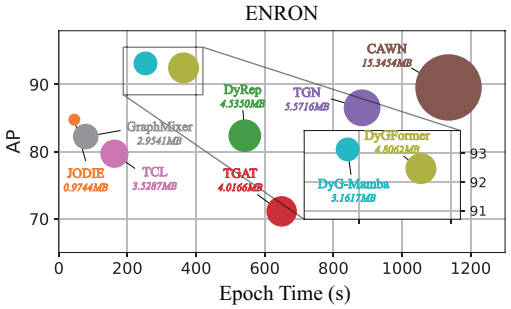

Figure 5: Comparison of model performance (AP), parameter size (size of the circle) and training time (seconds) per epoch.

the correlations between nodes, which are often predictive of future links (please refer to Figure 3). *(ii) Better Generalizability.* By incorporating time-spans as control signals for SSMs, DyG-Mamba can establish a strong correlation between time-spans and evolution laws of dynamic networks, *i.e.*, how to balance historical memory with the current input. In Figure 3, by removing time-span information, we observe a significant drop in the inductive setting. On the other hand, compared to data-independent Mamba, time information is one kind of anonymous information, and DyG-Mamba can improve generalization and easily apply to unseen nodes (w/o Time-Span vs. DyG-Mamba in Figure 3). *(iii) Greater Efficiency.* DyG-Mamba leverages the advantages of SSMs, employing a limited number of parameters to remember or forget historical information, thus achieving memory efficiency. Using parallel scanning and hardware-aware training strategies, the time complexity of DyG-Mamba approaches linearity (Figure 4).

Ablation Study. To evaluate the effectiveness of irregular time-aware continuous SSMs, the selective mechanism, and the cross-attention layer, we conduct ablation studies with the following three variants: (i) w/o Time-span: we use the input data as control signals instead of time-spans, *i.e.*, $\Delta = \text{SiLU}(\text{Linear}(\boldsymbol{u}(t)))$; (ii) w/o Selective: following the same parameter settings of S4 [12], *i.e.*, all parameters are irrelevant with input data or time-spans; and (iii) w/o Cross-attn: we remove the linear cross-attention layer. In Figure 3, the removal of either component of DyG-Mamba negatively impacts the ability of dynamic graph learning. Compared to DyG-Mamba, the performance of data-dependent w/o Time-span decreases significantly, especially in the inductive setting, showing the effectiveness of time-spans as control signals. Secondly, w/o Selective decreases the performance since the parameters are fixed and cannot fit to dynamic networks well. Finally, the cross-attention layer also brings a certain degree of performance gain.

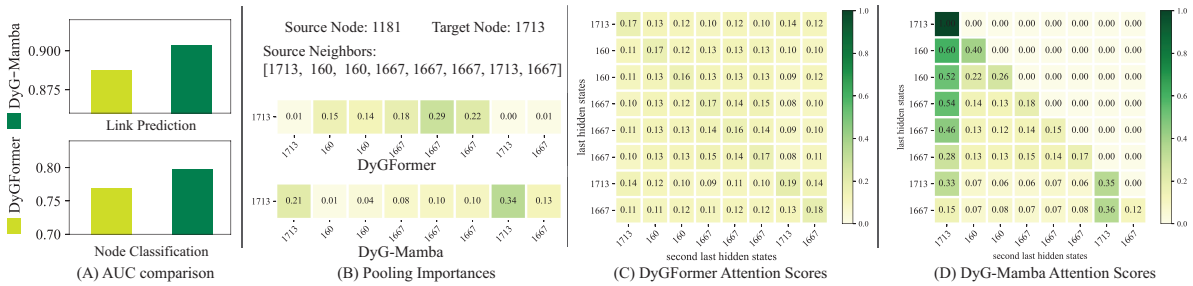

Figure 6: (A) Average AUC comparison. (B-D) An case study of historical attention mapping in DyG-Mamba and DyGFormer.

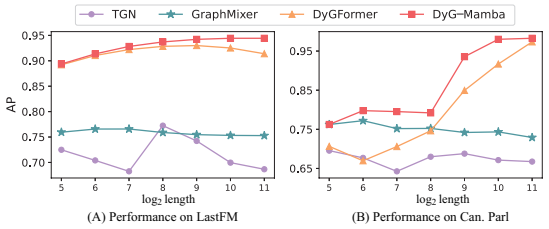

Figure 7: Performance with varying sequence lengths.

5.2 Qualitative Evaluation

Sensitivity to Sequence Length. To show the ability to capture long-term dependencies of DyG-Mamba, we select three best-performing baselines and depict their performance changing with varying sequence lengths in Figure 7. We observe that the performance of DyG-Mamba improves significantly with increasing sequence length, showing its ability to model long sequences. DyG-Mamba achieves better performance even with short sequence lengths, showing its ability to learn dynamic graphs.

Robustness against Noise. We conduct robust tests by randomly inserting 10% to 60% noisy edges with chronological timestamps during the evaluation step. In Figure 8, as the noisy edges increase, DyG-Mamba performance decreases to a minimum extent, indicating that DyG-Mamba has stronger robustness than baselines. This is because DyG-Mamba adopts a selective mechanism that helps select the most relevant content and remove irrelevant information.

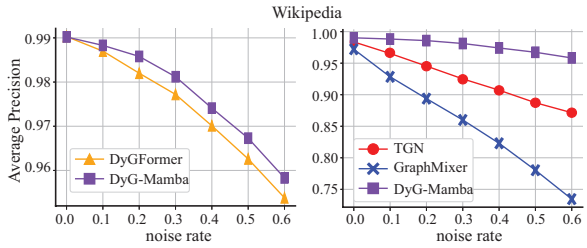

Figure 8: Robustness against noise, ranging from 0.1 to 0.6.

Time information Analysis. To evaluate the effectiveness of time information in DyG-Mamba, we conduct additional ablation studies, as shown in Table 5. There are four settings of DyG-Mamba: (i) [w/o Time-Encoding] refers to removing time encoding $Z^{\tau}_{u,T}$; (ii) [w/o Time-Span] replaces time-span with data-dependent control signals; (iii) [w/o both Time] refers to removing both time-encoding and time-span; (iv) [all Time-dependent] set parameters B and C from data-dependent to time-span dependent. From the above Table 5, we observe that time information plays a crucial role in DyG-Mamba. Specifically, the Time-span is key to capturing irregular temporal patterns, which is essential for the overall performance. Additionally, the time-encoding, which captures the relative time to the current time, also contributes to performance improvement, although to a lesser extent. Finally, although the time-span is crucial for capturing temporal information, the model entirely time-dependent will significantly reduce performance.

Table 5: Results (AP score) of time information ablations.

Settings	Can. Parl.	Enron	UCI	USLegis.
w/o Time-encoding	97.80±0.43	92.83±0.06	95.92±0.11	73.33±1.15
w/o Time-span	96.90±0.18	92.14±0.12	95.62±0.05	72.26±0.76
w/o both Time	96.87±0.18	92.08±0.12	95.54±0.08	72.19±0.72
all Time-dependent	79.24±0.58	82.25±0.16	94.16±0.66	70.57±0.84
DyG-Mamba	**98.20±0.52**	**93.14±0.08**	**96.14±0.14**	**73.66±1.13**

Attention Investigation. We conduct a case study to reveal the attention mapping in DyG-Mamba. We aim to predict the link between nodes 1181 and 1713. Figure 6(B) show the normalized cosine similarity between the target node encoding and the last hidden state in each model with respect to source node's neighbors. We observe that DyG-Mamba can better balance historical memory with current input, *i.e.*, it has higher similarity when input node 1713 is the source node's neighbor. Figures 6(C)-(D) depict the normalized cosine similarity between the last hidden state and the second-to-last hidden state in each model. DyGFormer in Figures 6(C)-(D) exhibits the highest similarity along the diagonal and an almost uniform distributed similarity across all neighbors, indicating that it struggles to distinguish the correct node. In contrast, DyG-Mamba prioritizes important information and assigns higher scores to the target node, showing the effectiveness of its selective mechanism.

6 Conclusion

In this work, we proposed a new SSM-based framework, DyG-Mamba, designed to effectively and efficiently capture long-term temporal dependencies on dynamic graphs. To achieve this goal, we directly used irregular time-spans as controllable signals to establish a strong correlation between dynamic evolution laws and time information, further improving the model's generalization and robustness. By evaluating on various downstream tasks, DyG-Mamba showed strong performance and applicability. We plan to apply DyG-Mamba to real-world applications in the future.

References

[1] Unai Alvarez-Rodriguez, Federico Battiston, Guilherme Ferraz de Arruda, Yamir Moreno, Matjaž Perc, and Vito Latora. 2021. Evolutionary dynamics of higher-order interactions in social networks. *Nature Human Behaviour* (2021), 586–595.

[2] Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. 2020. Adaptive Graph Convolutional Recurrent Network for Traffic Forecasting. In *Proc. of NeurIPS*.

[3] Ali Behrouz and Farnoosh Hashemi. 2024. Graph Mamba: Towards Learning on Graphs with State Space Models. In *Proc. of KDD*. 119–130.

[4] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2016. Fast and Accurate Deep Network Learning by Exponential Linear Units. In *Proc. of ICLR*.

[5] Weilin Cong, Yanhong Wu, Yuandong Tian, Mengting Gu, Yinglong Xia, Mehrdad Mahdavi, and Chun-cheng Jason Chen. 2021. Dynamic Graph Representation Learning via Graph Transformer Networks. *CoRR* (2021).

[6] 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?. In *Proc. of ICLR*.

[7] Hermann Ebbinghaus. 1885. *Über das gedächtnis: untersuchungen zur experimentellen psychologie.* Duncker & Humblot.

[8] Yuan Feng, Yukun Cao, Wang Hairu, Xike Xie, and S Kevin Zhou. 2024. Mayfly: a Neural Data Structure for Graph Stream Summarization. In *Proc. of ICLR*.

[9] Daniel Y Fu, Tri Dao, Khaled Kamal Saab, Armin W Thomas, Atri Rudra, and Christopher Re. 2022. Hungry Hungry Hippos: Towards Language Modeling with State Space Models. In *ICLR*.

[10] Albert Gu and Tri Dao. 2023. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752* (2023).

[11] Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. 2020. HiPPO: Recurrent Memory with Optimal Polynomial Projections. In *Proc. of NeurIPS*.

[12] Albert Gu, Karan Goel, and Christopher Ré. 2022. Efficiently Modeling Long Sequences with Structured State Spaces. In *The Tenth International Conference on Learning Representations*. OpenReview.net. https://openreview.net/forum?id=uYLFoz1vlAC

[13] Albert Gu, Isys Johnson, Karan Goel, Khaled Saab, Tri Dao, Atri Rudra, and Christopher Ré. 2021. Combining Recurrent, Convolutional, and Continuous-time Models with Linear State Space Layers. In *Proc. of NeurIPS*. 572–585.

[14] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. 2019. Attention Based Spatial-Temporal Graph Convolutional Networks for Traffic Flow Forecasting. In *Proc. of AAAI*. 922–929.

[15] Ankit Gupta, Albert Gu, and Jonathan Berant. 2022. Diagonal State Spaces are as Effective as Structured State Spaces. In *Proc. of NeurIPS*.

[16] Ramin Hasani, Mathias Lechner, Tsun-Hsuan Wang, Makram Chahine, Alexander Amini, and Daniela Rus. 2022. Liquid Structural State-Space Models. In *ICLR*.

[17] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[18] Shenyang Huang, Samy Coulombe, Yasmeen Hitti, Reihaneh Rabbany, and Guillaume Rabusseau. 2024. Laplacian Change Point Detection for Single and Multi-view Dynamic Graphs. *ACM Trans. Knowl. Discov. Data* (2024), 63:1–63:32.

[19] Shenyang Huang, Farimah Poursafaei, Jacob Danovitch, Matthias Fey, Weihua Hu, Emanuele Rossi, Jure Leskovec, Michael M. Bronstein, Guillaume Rabusseau, and Reihaneh Rabbany. 2023. Temporal Graph Benchmark for Machine Learning on Temporal Graphs. In *Proc. of NeurIPS*.

[20] Zijie Huang, Yizhou Sun, and Wei Wang. 2020. Learning Continuous System Dynamics from Irregularly-Sampled Partial Observations. In *Proc. of NeurIPS*.

[21] Zihang Jiang, Weihao Yu, Daquan Zhou, Yunpeng Chen, Jiashi Feng, and Shuicheng Yan. 2020. ConvBERT: Improving BERT with Span-based Dynamic Convolution. In *Proc. of NeurIPS*.

[22] Ming Jin, Yuan-Fang Li, and Shirui Pan. 2022. Neural Temporal Walks: Motif-Aware Representation Learning on Continuous-Time Dynamic Graphs. In *Proc. of NeurIPS*.

[23] Rudolph Emil Kalman. 1960. A new approach to linear filtering and prediction problems. (1960).

[24] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *Proc. of ICML*. 5156–5165.

[25] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proc. of ICLR*.

[26] Srijan Kumar, Xikun Zhang, and Jure Leskovec. 2019. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proc. of KDD*.

[27] Jay H Lee, Manfred Morari, and Carlos E Garcia. 1994. State-space interpretation of model predictive control. *Automatica* (1994), 707–717.

[28] Lincan Li, Hanchen Wang, Wenjie Zhang, and Adelle Coster. 2024. STG-Mamba: Spatial-Temporal Graph Learning via Selective State Space Model. arXiv:2403.12418 [cs.LG]

[29] Meng Liu, Yue Liu, KE LIANG, Wenxuan Tu, Siwei Wang, sihang zhou, and Xinwang Liu. 2024. Deep Temporal Graph Clustering. In *Proc. of ICLR*.

[30] Antonio Longa, Veronica Lachi, Gabriele Santin, Monica Bianchini, Bruno Lepri, Pietro Lio, franco scarselli, and Andrea Passerini. 2023. Graph Neural Networks for Temporal Graphs: SOTA, Open Challenges, and Opportunities. *TMLR* (2023).

[31] Linhao Luo, Gholamreza Haffari, and Shirui Pan. 2023. Graph Sequential Neural ODE Process for Link Prediction on Dynamic and Sparse Graphs. In *Proc. of WSDM*. 778–786.

[32] Xiao Luo, Haixin Wang, Zijie Huang, Huiyu Jiang, Abhijeet Sadashiv Gangan, Song Jiang, and Yizhou Sun. 2023. CARE: Modeling Interacting Dynamics Under Temporal Environmental Variation. In *Proc. of NeurIPS*.

[33] Chen Ma, Liheng Ma, Yingxue Zhang, Jianing Sun, Xue Liu, and Mark Coates. 2020. Memory Augmented Graph Neural Networks for Sequential Recommendation. In *Proc. of AAAI*. 5045–5052.

[34] Prabhaker Mishra, Uttam Singh, Chandra M Pandey, Priyadarshni Mishra, and Gaurav Pandey. 2019. Application of student's t-test, analysis of variance, and covariance. *Annals of cardiac anaesthesia* (2019), 407–411.

[35] Vinod Nair and Geoffrey E. Hinton. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proc. of ICML*. 807–814.

[36] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao B. Schardl, and Charles E. Leiserson. 2020. EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs. In *Proc. of AAAI*. 5363–5370.

[37] Jong Ho Park, Jaden Park, Zheyang Xiong, Nayoung Lee, Jaewoong Cho, Samet Oymak, Kangwook Lee, and Dimitris Papailiopoulos. 2024. Can Mamba Learn How To Learn? A Comparative Study on In-Context Learning Tasks. In *Proc. of ICML*.

[38] Farimah Poursafaei, Andy Huang, Kellin Pelrine, and Reihaneh Rabbany. 2022. Towards Better Evaluation for Dynamic Link Prediction. In *Proc. of NeurIPS*.

[39] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. 2020. Temporal Graph Networks for Deep Learning on Dynamic Graphs. In *ICML 2020 Workshop on Graph Representation Learning*.

[40] 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 *Proc. of WSDM*. 519–527.

[41] Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. 2018. Structured Sequence Modeling with Graph Convolutional Recurrent Networks. In *Neural Information Processing - 25th International Conference*. 362–373.

[42] Jimmy TH Smith, Andrew Warrington, and Scott Linderman. 2022. Simplified State Space Layers for Sequence Modeling. In *ICLR*.

[43] Weiping Song, Zhiping Xiao, Yifan Wang, Laurent Charlin, Ming Zhang, and Jian Tang. 2019. Session-Based Social Recommendation via Dynamic Graph Attention Networks. In *Proc. of WSDM*. 555–563.

[44] Yujin Tang, Peijie Dong, Zhenheng Tang, Xiaowen Chu, and Junwei Liang. 2024. VMRNN: Integrating Vision Mamba and LSTM for Efficient and Accurate Spatiotemporal Forecasting. *arXiv preprint arXiv:2403.16536* (2024).

[45] Yuxing Tian, Yiyan Qi, and Fan Guo. 2024. FreeDyG: Frequency Enhanced Continuous-Time Dynamic Graph Model for Link Prediction. In *Proc. of ICLR*.

[46] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. 2019. DyRep: Learning Representations over Dynamic Graphs. In *Proc. of ICLR*.

[47] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Proc. of NeurIPS*. 5998–6008.

[48] 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* (2021).

[49] Yanbang Wang, Yen-Yu Chang, Yunyu Liu, Jure Leskovec, and Pan Li. 2021. Inductive Representation Learning in Temporal Networks via Causal Anonymous Walks. In *Proc. of ICLR*.

[50] Yuxia Wu, Yuan Fang, and Lizi Liao. 2024. On the Feasibility of Simple Transformer for Dynamic Graph Modeling. In *Proceedings of the ACM on Web Conference 2024*. 870–880.

[51] Jiaxuan You, Tianyu Du, and Jure Leskovec. 2022. ROLAND: Graph Learning Framework for Dynamic Graphs. In *Proc. of KDD*. 2358–2366.

[52] Le Yu, Zihang Liu, Leilei Sun, Bowen Du, Chuanren Liu, and Weifeng Lv. 2023. Continuous-Time User Preference Modelling for Temporal Sets Prediction. *IEEE Transactions on Knowledge and Data Engineering* (2023).

[53] Le Yu, Leilei Sun, Bowen Du, and Weifeng Lv. 2023. Towards Better Dynamic Graph Learning: New Architecture and Unified Library. In *Proc. of NeurIPS*.

[54] 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* (2022).

[55] Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. 2024. Vision mamba: Efficient visual representation learning with bidirectional state space model. *arXiv preprint arXiv:2401.09417* (2024).

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009