MAMBATG: CANDIDATE-AWARE STATE SPACE MODELING FOR FUTURE LINK PREDICTION ON TEMPORAL GRAPHS

Anonymous authors
Paper under double-blind review

ABSTRACT

Future link prediction on temporal graphs involves both repeated interactions between previously connected node pairs and first-time interactions between pairs with no prior interaction. While repeated-edge prediction can exploit direct pairwise evidence, unseen-edge prediction must generalize beyond historical recurrence. Existing temporal graph models often exploit signals tied to recurring interactions or historical overlap, which are highly informative for repeated edges but become limited or unavailable for unseen edges, contributing to substantial performance degradation on unseen-edge prediction. Based on this observation, we identify two complementary signals for modeling sequential dynamics beyond historical recurrence: (1) *historical-sequence evolution*, which captures dependencies among the source node's chronologically ordered interactions, and (2) *candidate–history compatibility*, which captures how well a candidate fits the source node's evolving history. Based on the decomposition, we propose **MambaTG**, a candidate-aware Mamba-based framework that jointly models these two signals. Mamba provides a natural mechanism for this design: its selective recurrence captures the evolution of ordered interaction histories, while its input-dependent state transitions allow the history state to adapt to different candidate inputs. Extensive experiments on 13 temporal graph datasets against 13 representative baselines show that MambaTG achieves strong and consistent performance across both repeated and unseen edges while remaining computationally efficient.

1 INTRODUCTION

Temporal graphs represent entities as nodes and their interactions as time-stamped edges (Kazemi et al., 2020). They provide a natural framework for modeling evolving relationships in applications such as social networks (Alvarez-Rodriguez et al., 2021) and recommender systems (Yu et al., 2024). A fundamental task in temporal graph learning is *future link prediction*, which predicts interactions at a future query time from historical interactions (Divakaran & Mohan, 2020). Future interactions can be divided into two categories, (1) *repeated edges*, connecting previously interacting node pairs, and (2) *unseen edges*, connecting pairs with no prior interaction (Yi et al., 2025b). Both arise naturally in real-world systems: users on a social platform may repeatedly interact with existing connections while also forming new connections with previously unrelated users. Despite their coexistence, the two regimes present distinct predictive challenges. Repeated-edge prediction can directly exploit prior pairwise interactions, whereas unseen-edge prediction must infer future relationships from broader patterns in historical behaviors. A practical temporal graph model must therefore capture recurring relationships while also generalizing to new ones without prior pairwise interactions.

However, some existing temporal link prediction methods still rely heavily on signals associated with repeated interactions. The competitiveness of EdgeBank, a non-parametric baseline that predicts links by memorizing previously observed edges, shows that historical recurrence alone can provide a remarkably strong signal on conventional benchmarks (Poursafaei et al., 2022). This bias toward recurrence is reflected in representative temporal graph neural network (T-GNN) designs. Recent methods such as DyGFormer(Yu et al., 2023), FreeDyG(Tian et al., 2024), DyG-Mamba(Li et al., 2025), and DyGMamba(Ding et al., 2025) explicitly encode source–destination historical overlap through neighbor co-occurrence or interaction-frequency features. Such signals are highly

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

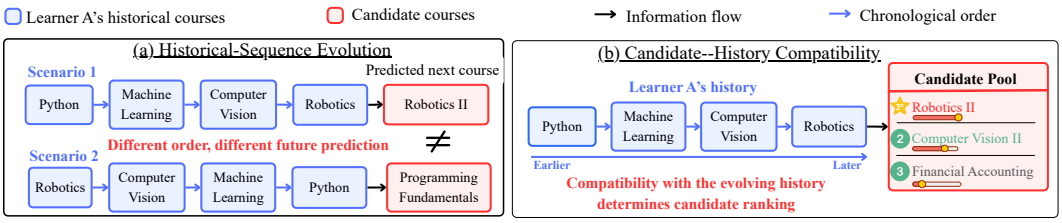


Figure 1: Illustration of two complementary signals for predicting Learner A's next course choice from previously taken courses in Python, Machine Learning, Computer Vision, and Robotics.

informative for repeated edges but may be sparse or unavailable for unseen edges, particularly in bipartite graphs. Consistent with this limitation, recent benchmarks report substantially lower performance on unseen edges than on repeated edges (Yi et al., 2025b). These observations suggest that effective future link prediction requires moving beyond recurrence-based evidence and learning historical patterns that can generalize to previously unseen interactions.

This raises a more fundamental question: ***What predictive information should a temporal graph model learn from historical interactions to generalize beyond direct pairwise recurrence?*** We argue that the answer lies in two complementary signals: *historical-sequence evolution* and *candidate–history compatibility*, as illustrated through Learner A's historical course interactions in Figure 1. **(1) Historical-sequence evolution** (Figure 1(a)) captures how historical interactions evolve in temporal order. The same interactions can imply different future behavior when ordered differently, as later interactions may reflect intentions developed through earlier ones. Capturing this signal therefore requires modeling dependencies among historical interactions as an ordered sequence, such that later interactions are interpreted in the context of the preceding history. **(2) Candidate–history compatibility** (Figure 1(b)) captures how well a candidate fits the source node's evolving history. Explicitly modeling this relationship allows predictions to adapt to each candidate. However, many T-GNNs construct source and candidate representations largely independently and combine them only at the prediction stage (Kumar et al., 2019; Cong et al., 2023), limiting direct modeling of the relationship between the candidate and the source history. CRAFT (Yi et al., 2025a) directly matches each candidate against the source node's individual historical interactions, but does not explicitly model dependencies arising from the history's sequential evolution before candidate matching. Overall, the two signals capture distinct but complementary aspects of future interaction prediction: *historical-sequence evolution* characterizes dependencies within the ordered interaction history, whereas *candidate–history compatibility* reflects how that evolving history informs the prediction of a candidate interaction.

Jointly modeling these complementary signals poses a non-trivial design challenge: historical-sequence evolution is shared across candidates, whereas candidate–history compatibility is inherently candidate-specific. To address this challenge, we propose MambaTG, an SSM-based framework that uses Mamba (Gu & Dao, 2024) as its sequential encoder. Its Candidate-Aware Mamba Encoder (CAME) first encodes the source's ordered historical interactions into a shared history state and then adapts this state to each candidate through Mamba's input-dependent state transitions. This design evaluates each candidate in the context of the evolved history while avoiding repeated history encoding. In addition, interaction histories may contain predictive patterns at different temporal scales, from slowly evolving preferences to rapid behavioral shifts. When modeled directly, dominant patterns at one scale may obscure weaker but predictive dynamics at another. We therefore introduce **Dual-Branch Representation Refinement (DRR)** to adaptively refine historical and candidate representations, providing CAME with more informative inputs for sequential encoding.

The main contributions of this work are summarized as follows:

- **A unified view of sequential dynamics.** We characterize sequential dynamics for future link prediction through two complementary signals: *historical-sequence evolution* and *candidate–history compatibility*. Through empirical analysis, we show that both signals contribute to strong performance across repeated- and unseen-edge prediction.
- **An efficient candidate-aware Mamba-based framework.** We propose **MambaTG**, which jointly captures these two signals through a Candidate-Aware Mamba Encoder and improves input representations through Dual-Branch Representation Refinement. By

reusing the encoded historical state across candidates, MambaTG avoids repeated history encoding and enables efficient candidate evaluation.

- **Extensive empirical validation.** We evaluate MambaTG on 13 future link prediction datasets. MambaTG achieves strong performance across repeated- and unseen-edge prediction and attains the best average rank among all baselines. Efficiency and ablation studies further demonstrate its scalability and validate its principal design choices.

2 Related Work

Temporal Graph Learning Methods for Future Link Prediction. Temporal graph models use diverse architectures to encode interaction histories, including memory-based methods such as JODIE (Kumar et al., 2019), DyRep (Trivedi et al., 2019), and TGN (Rossi et al., 2020), attention-based methods such as TGAT (Xu et al., 2020), TCL (Wang et al., 2021a), DyGFormer (Yu et al., 2023), and SimpleDyG (Wu et al., 2024), and alternative encoders such as CAWN (Wang et al., 2021b), GraphMixer (Cong et al., 2023), and FreeDyG (Tian et al., 2024). From the perspective of sequential dynamics, these methods differ in how they capture *historical-sequence evolution* and *candidate–history compatibility*. Many methods encode the source history and candidate largely independently and combine their representations only at the prediction stage (Kumar et al., 2019; Trivedi et al., 2019; Rossi et al., 2020; Xu et al., 2020; Cong et al., 2023; Wu et al., 2024), limiting direct candidate conditioning of the source history. Other approaches incorporate information from both endpoints through temporal motifs, cross-subgraph interactions, neighbor co-occurrence, or pairwise statistics (Wang et al., 2021b;a; Yu et al., 2023; Tian et al., 2024), but primarily characterize relationships between the source and destination histories rather than how a candidate relates to the source's evolving history. CRAFT (Yi et al., 2025a) directly matches each candidate against the source node's individual historical interactions, but does not explicitly model dependencies arising from the history's sequential evolution before candidate matching. In contrast, MambaTG jointly models *historical-sequence evolution* and *candidate–history compatibility* by first encoding the ordered source history to model its evolution and then adapting the resulting state to each candidate.

Mamba-Based Models for Continuous-Time Dynamic Graphs. Mamba (Gu & Dao, 2024) is a selective state-space model that efficiently models long sequences through input-dependent state transitions with linear sequence-length complexity. Recent studies have adapted Mamba to continuous-time dynamic graphs. DyGMamba (Ding et al., 2025) uses node- and time-level Mamba to encode long histories and identify temporally relevant interactions, while DyG-Mamba (Li et al., 2025) incorporates irregular inter-event times into state transitions for elapsed-time-aware forgetting and selective retention. These methods primarily focus on efficient long-history encoding and irregular temporal dynamics. In contrast, MambaTG uses candidate-aware continuation from the evolving source history to jointly model *historical-sequence evolution* and *candidate–history compatibility*, targeting effective prediction across both repeated and unseen edges.

3 Preliminaries

Temporal Graph and Future Link Prediction. A temporal graph is represented as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the node set and $\mathcal{E} = ((u_i, v_i, t_i))_{i=1}^{M}$ is a chronologically ordered stream of M interactions. Here, $u_i \in \mathcal{V}$ and $v_i \in \mathcal{V}$ denote the source and destination nodes of the i-th interaction, respectively, and t_i denotes its timestamp, with $t_1 \leq t_2 \leq \cdots \leq t_M$. At a query time t_q, the observed history is $\mathcal{E}_{<t_q} = \{(u_i, v_i, t_i) \in \mathcal{E} \mid t_i < t_q\}$. Given a source node u and a candidate set $\mathcal{C}_u^{t_q}$, future link prediction assigns each candidate destination $c_i \in \mathcal{C}_u^{t_q}$ a score conditioned on $\mathcal{E}_{<t_q}$ and ranks the true destination above negative candidates. A candidate edge (u, c_i) is *repeated* if the pair appears in $\mathcal{E}_{<t_q}$, and *unseen* otherwise (Yi et al., 2025b).

Discrete Fourier Transform. Let $\mathbf{X} = (\mathbf{x}_n)_{n=0}^{N-1} \in \mathbb{R}^{N \times d}$ be a sequence of d-dimensional representations. The discrete Fourier transform (DFT) and its inverse (IDFT), applied along the temporal dimension, are defined as

$$\widehat{\mathbf{x}}_m = \sum_{n=0}^{N-1} \mathbf{x}_n \exp\left(-\mathrm{i}\frac{2\pi mn}{N}\right), \qquad \mathbf{x}_n = \frac{1}{N}\sum_{m=0}^{N-1} \widehat{\mathbf{x}}_m \exp\left(\mathrm{i}\frac{2\pi mn}{N}\right), \tag{1}$$

where $\widehat{\mathbf{x}}_m \in \mathbb{C}^d$ is the m-th frequency component and $\mathrm{i}^2 = -1$. For real-valued inputs, the real FFT (rFFT) and its inverse (irFFT) are commonly used (Sorensen et al., 1987).

State Space Models. State space models (SSMs) represent sequences through latent states that evolve over time. Given $x(t) \in \mathbb{R}$, an SSM produces $y(t) \in \mathbb{R}$ from a latent state $h(t) \in \mathbb{R}^N$:

$$h'(t) = Ah(t) + Bx(t), \qquad y(t) = Ch(t). \tag{2}$$

Here, $A \in \mathbb{R}^{N\times N}$ is the state-transition matrix, $B \in \mathbb{R}^{N\times 1}$ projects the input into the latent state, and $C \in \mathbb{R}^{1\times N}$ maps the latent state to the output. To process a discrete sequence, the continuous system is discretized using a step size Δ:

$$h_t = \bar{A}h_{t-1} + \bar{B}x_t, \qquad y_t = Ch_t. \tag{3}$$

Under zero-order-hold discretization, the discrete parameters are given by $\bar{A} = \exp(\Delta A)$ and $\bar{B} = (\Delta A)^{-1}\big(\exp(\Delta A) - I\big)\Delta B$. Unlike conventional SSMs with input-independent parameters, Mamba uses input-dependent B, C, and Δ with a hardware-aware selective scan for efficient recurrence computation (Gu & Dao, 2024).

4 METHOD

The overall MambaTG framework is illustrated in Figure 2. Given a source node u, a query timestamp t_q, and its chronologically ordered interaction history $\mathcal{H}_u^{t_q} = ((v_1, t_1), \ldots, (v_{\ell_u}, t_{\ell_u}))$, where $\ell_u \le L_{\max}$ and $t_1 \le \cdots \le t_{\ell_u} < t_q$, MambaTG assigns a link-prediction score to each candidate $c_i \in \mathcal{C}_u^{t_q}$. It first combines node embeddings with query-relative time encodings to construct historical and candidate representations. Dual-Branch Representation Refinement then enhances these representations through separate historical and candidate refinement branches, after which Candidate-Aware Mamba Encoder recurrently encodes the refined history and incorporates each candidate through an additional state update. Finally, a link classifier maps each history-conditioned candidate representation to a prediction score.

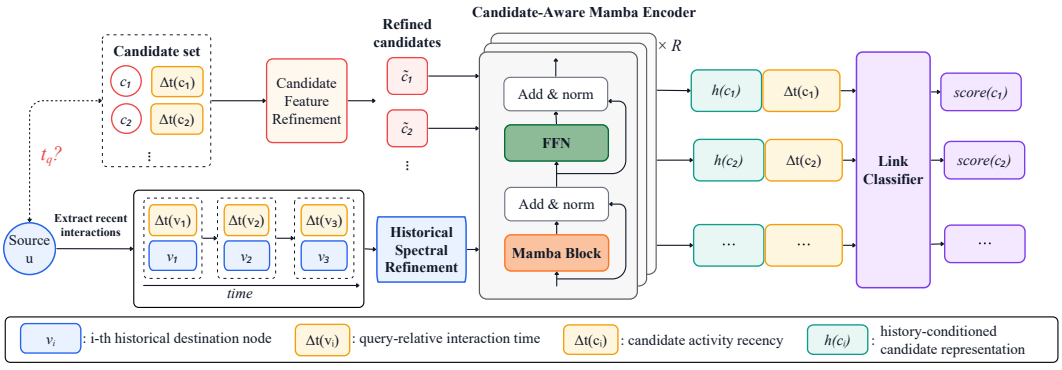


Figure 2: Framework of the proposed MambaTG model.

4.1 NODE AND TIME ENCODING.

Node embedding. Following Yi et al. (2025a), we associate node $v \in \mathcal{V}$ with a trainable identifier embedding $\mathrm{Emb}(v) \in \mathbb{R}^{d_e}$. Given a query time t_q, let $\mathcal{C}_u^{t_q}$ denote the set of $q = |\mathcal{C}_u^{t_q}|$ candidate destinations. We arrange the embeddings of the ℓ_u historical neighbors and q candidate destinations row-wise as $\mathbf{E}_H = [\mathrm{Emb}(v_i)]_{i=1}^{\ell_u} \in \mathbb{R}^{\ell_u \times d_e}$ and $\mathbf{E}_D = [\mathrm{Emb}(c_j)]_{j=1}^{q} \in \mathbb{R}^{q\times d_e}$, respectively.

Time encoding. To incorporate temporal information, we encode elapsed times relative to the query time t_q. For each historical interaction occurring at time t_i, its elapsed time is $\Delta t_i^H = t_q - t_i$. For candidate c_j, we instead use its activity recency $\Delta t_{c_j}^D = t_q - t_{c_j}^{\text{last}}$, where $t_{c_j}^{\text{last}} < t_q$ is its most recent interaction with any node in the global graph. For cold-start candidates with no prior

interactions, we set $\Delta t^D_{c_j} = 0$. Following Xu et al. (2020), each elapsed time is mapped to a d_T-dimensional representation using $\boldsymbol{\phi} : \mathbb{R}_{\geq 0} \to \mathbb{R}^{d_T}$. The resulting historical and candidate time encodings are $\mathbf{T}_H = [\boldsymbol{\phi}(\Delta t^H_i)]_{i=1}^{\ell_u} \in \mathbb{R}^{\ell_u \times d_T}$ and $\mathbf{T}_D = [\boldsymbol{\phi}(\Delta t^D_{c_j})]_{j=1}^{q} \in \mathbb{R}^{q \times d_T}$, respectively.

Feature fusion. To jointly capture node identity and temporal context, we combine the node and time representations: $\mathbf{Z}_H = \mathrm{Linear}([\mathbf{E}_H \| \mathbf{T}_H]) \in \mathbb{R}^{\ell_u \times d}$ and $\mathbf{Z}_D = \mathrm{Linear}([\mathbf{E}_D \| \mathbf{T}_D]) \in \mathbb{R}^{q \times d}$, where $\|$ denotes feature-wise concatenation and $\mathrm{Linear}(\cdot)$ is a shared linear projection.

4.2 Dual-Branch Representation Refinement (DRR).

An interaction history may simultaneously contain slowly evolving trends, recurring behavior, and abrupt short-term changes. Without prior refinement, stronger patterns at one scale may obscure subtler yet predictive dynamics at other scales. We therefore introduce **Dual-Branch Representation Refinement**, comprising **Historical Spectral Refinement (HSR)** and **Candidate Feature Refinement (CFR)**. HSR exposes and adaptively refines frequency components associated with these patterns, while CFR recalibrates candidate features for compatibility with the refined history.

Historical Spectral Refinement. We transform $\mathbf{Z}_H$ into the frequency domain along its temporal dimension, yielding $\widehat{\mathbf{Z}}_H = \mathrm{rFFT}_{L_{\max}}(\mathbf{Z}_H) \in \mathbb{C}^{(\lfloor L_{\max}/2 \rfloor + 1) \times d}$, where $L_{\max} \geq \ell_u$ is the transform length and zero-padding is applied when $\ell_u < L_{\max}$. Inspired by spectral filtering in FreeDyG (Tian et al., 2024), HSR modulates each frequency component using learnable complex-valued weights:

$$\widehat{\mathbf{Z}}^{\mathrm{HSR}}_H = \widehat{\mathbf{Z}}_H \odot \mathbf{W}_H, \qquad \mathbf{W}_H \in \mathbb{C}^{(\lfloor L_{\max}/2 \rfloor + 1) \times d}. \tag{4}$$

Here, $\odot$ denotes element-wise multiplication. The complex-valued weights $\mathbf{W}_H$ enable the model to selectively adjust the magnitude and phase of temporal patterns across different frequencies. Applying the inverse transform with output length $L_{\max}$ yields $\mathbf{Z}^{\mathrm{HSR}}_H = \left[\mathrm{irFFT}_{L_{\max}}\left(\widehat{\mathbf{Z}}^{\mathrm{HSR}}_H\right)\right]_{1:\ell_u} \in \mathbb{R}^{\ell_u \times d}$, where the first ℓ_u positions corresponding to the original history are retained.

Candidate Feature Refinement. To promote alignment between candidate representations and the spectrally refined historical representations, CFR learnably rescales candidate features through a lightweight feature-wise transformation. Specifically, for each candidate representation $\mathbf{z}_{c_i} = \mathbf{Z}_D[i] \in \mathbb{R}^d$, we compute $\mathbf{z}^{\mathrm{CFR}}_{c_i} = \mathbf{z}_{c_i} \odot \mathbf{w}_D$ where $\mathbf{w}_D \in \mathbb{R}^d$ is a learnable vector shared across candidates.

4.3 Candidate-Aware Mamba Encoder (CAME).

Modeling sequential dynamics requires capturing both *historical-sequence evolution* and *candidate–history compatibility*. Mamba (Gu & Dao, 2024) is well suited to this objective because its selective recurrence summarizes ordered sequences into recurrent states while its input-dependent state transitions allow the same historical state to respond differently to different candidates. CAME stacks R layers, each consisting of a Mamba block followed by a position-wise feed-forward network (FFN). Detailed formulations of these standard components are provided in Appendix C.1.

Historical-sequence encoding. CAME first encodes the source node's interaction history in chronological order to capture its sequential evolution. The refined historical representations $\mathbf{Z}^{\mathrm{HSR}}_H \in \mathbb{R}^{\ell_u \times d}$ are padded to $L_{\max}$ for batching. We initialize $\mathbf{X}^{(0)}_H = \mathrm{Pad}\big(\mathbf{Z}^{\mathrm{HSR}}_H, L_{\max}\big)$. At each layer r, CAME processes the complete historical sequence once and retains its terminal recurrent cache:

$$\begin{aligned} \left(\mathbf{M}^{(r)}_H, \mathcal{K}^{(r)}_H\right) &= \mathrm{MambaBlock}^{(r)}\left(\mathbf{X}^{(r-1)}_H\right), \qquad \widetilde{\mathbf{X}}^{(r)}_H = \mathrm{LN}^{(r)}_1\left(\mathbf{X}^{(r-1)}_H + \mathbf{M}^{(r)}_H\right), \\ \mathbf{X}^{(r)}_H &= \mathrm{LN}^{(r)}_2\left(\widetilde{\mathbf{X}}^{(r)}_H + \mathrm{FFN}^{(r)}\left(\widetilde{\mathbf{X}}^{(r)}_H\right)\right), \qquad r = 1, \ldots, R. \end{aligned} \tag{5}$$

Here, $\mathbf{M}^{(r)}_H$ denotes the Mamba output representations for the historical sequence, while $\mathcal{K}^{(r)}_H$ contains the terminal SSM and causal-convolution states required to continue the r-th Mamba layer with

one additional token. Through recurrent state propagation, historical representations incorporate information from earlier interactions, enabling CAME to capture *historical-sequence evolution*.

Candidate continuation. For candidate c_i, we initialize $\mathbf{x}_{c_i}^{(0)} = \mathbf{z}_{c_i}^{\mathrm{CFR}}$. At each layer, the candidate is processed as one additional input following the historical sequence by continuing from the corresponding terminal history cache:

$$\begin{aligned} \mathbf{m}_{c_i}^{(r)} &= \mathrm{MambaStep}^{(r)}\left(\mathbf{x}_{c_i}^{(r-1)}; \mathcal{K}_H^{(r)}\right), \qquad \widetilde{\mathbf{x}}_{c_i}^{(r)} = \mathrm{LN}_1^{(r)}\left(\mathbf{x}_{c_i}^{(r-1)} + \mathbf{m}_{c_i}^{(r)}\right), \\ \mathbf{x}_{c_i}^{(r)} &= \mathrm{LN}_2^{(r)}\left(\widetilde{\mathbf{x}}_{c_i}^{(r)} + \mathrm{FFN}^{(r)}\left(\widetilde{\mathbf{x}}_{c_i}^{(r)}\right)\right), \qquad r = 1, \ldots, R. \end{aligned} \tag{6}$$

Here, $\mathrm{MambaStep}^{(r)}$ shares its parameters with $\mathrm{MambaBlock}^{(r)}$ and continues the recurrence for one additional input using $\mathcal{K}_H^{(r)}$. Since the selective state-space parameters depend on the candidate input $\mathbf{x}_{c_i}^{(r-1)}$, each candidate induces a different state update from the shared historical cache. The resulting representation is therefore conditioned jointly on the candidate and the evolved historical state, capturing *candidate–history compatibility*. By reusing the terminal history cache across candidates, MambaTG encodes the history only once, while each candidate requires only one additional recurrent update per layer, avoiding redundant history encoding. After R layers, the history-conditioned candidate representation is $\mathbf{h}_{u,c_i}^{t_q} = \mathbf{x}_{c_i}^{(R)} \in \mathbb{R}^d$.

4.4 Link Classifier.

Following Yi et al. (2025a), we project the candidate's activity recency, $\Delta t_{c_i}^D = t_q - t_{c_i}^{\mathrm{last}}$, and concatenate it with the candidate representation $\mathbf{h}_{u,c_i}^{t_q}$. Prediction score is computed as:

$$\hat{y}_{u,c_i}^{t_q} = \mathrm{MLP}\left(\left[\mathbf{h}_{u,c_i}^{t_q} \,\|\, \mathrm{Linear}_\Delta\left(\Delta t_{c_i}^D\right)\right]\right), \tag{7}$$

where $\|$ denotes feature-wise concatenation. For repeated-edge-dominant datasets, we use **MambaTG-R**, which additionally projects and concatenates the historical repetition frequency $n_{u,c_i}^{t_q}$, defined as the number of interactions between u and c_i before t_q. Both variants are optimized using Bayesian Personalized Ranking (BPR) (Rendle et al., 2009), which encourages observed destinations to receive higher scores than sampled negatives.

4.5 Time Complexity Analysis

Table 1: Candidate-scoring time complexity. Entries omit $O(\cdot)$ and fixed architectural constants.

Method	Historical-Neighbor Extraction	Edge Likelihood Computation
TGN	$B(1+q)\log \bar{d}$	$B(1+q)\ell d^2$
TGAT	$B(1+q)(1+\ell)\log \bar{d}$	$B(1+q)\ell^2 d^2$
GraphMixer	$B(1+q)\log \bar{d}$	$B(1+q)\ell d^2$
DyGFormer	$B(1+q)\log \bar{d}$	$B(1+q)(\ell d^2 + \ell^2 d)$
DyG-Mamba	$B(1+q)\log \bar{d}$	$Bq\ell d^2$
DyGMamba	$B(1+q)\log \bar{d}$	$Bq\ell d^2$
CRAFT	$B\log \bar{d}$	$B(\ell+q)d^2 + Bq\ell d$
MambaTG	$B\log \bar{d}$	$B(\ell+q)d^2 + B\ell d\log \ell$

We analyze the time complexity of future link prediction for a batch of source nodes and their candidate destinations. Let B denote the number of source queries, q the number of candidates per source, ℓ the number of historical interactions considered per node, d the feature dimension, and $\bar{d}$ the average node degree. Table 1 summarizes the resulting complexities; detailed derivations and assumptions are provided in Appendix I.

MambaTG retrieves only the source history, requiring $O(B\log \bar{d})$ time for temporal-index lookup. For edge likelihood computation, DRR refines each history in $O(B\ell d\log \ell)$ time, while CAME encodes it once and reuses its layer-wise terminal caches across candidates, costing $O(B(\ell+q)d^2)$.

The total complexity is $O\big(B(\ell+q)d^2 + B\ell d \log \ell\big)$. Unlike methods that encode both endpoints, MambaTG uses only the source history, avoiding candidate-dependent neighborhood retrieval. The separation of history encoding from candidate evaluation further avoids repeated processing across candidates and yields more favorable asymptotic scaling than CRAFT when $q > \log \ell$, as in our evaluation, with $q = 101$ and ℓ up to 1000. Together, they enable scalable prediction over long histories and large candidate sets in applications such as recommendation and social-link prediction.

5 Experiments

5.1 Experimental Setup

Datasets. We evaluate MambaTG on 13 future link prediction datasets, including seven datasets from TGB-Seq (Yi et al., 2025b) and six commonly used temporal graph datasets. Following Yi et al. (2025a), we categorize them as repeated-edge-dominant or unseen-edge-dominant according to the relative prevalence of repeated and unseen edges. Dataset details are provided in Appendix B.2.

Baselines. We compare MambaTG with twelve representative temporal graph baselines: EdgeBank (Poursafaei et al., 2022), CRAFT (Yi et al., 2025a), JODIE (Kumar et al., 2019), DyRep (Trivedi et al., 2019), TGAT (Xu et al., 2020), TGN (Rossi et al., 2020), CAWN (Wang et al., 2021b), GraphMixer (Cong et al., 2023), DyGFormer (Yu et al., 2023), FreeDyG (Tian et al., 2024), DyG-Mamba (Li et al., 2025), and DyGMamba (Ding et al., 2025). Following Yi et al. (2025a), we additionally include SGNN-HN (Pan et al., 2020), a session-based recommendation method, as a sequential baseline. Together, these methods span non-parametric, memory-based, attention-based, walk-based, MLP-based, frequency-based, Mamba-based, and sequential recommendation architectures. Baseline details are provided in Appendix B.3.

Table 2: Future link prediction performance on (a) unseen-edge-dominant and (b) repeated-edge-dominant datasets. Results are reported as MRR (%), averaged over three random seeds. The **best**, second-best, and third-best results are highlighted. Avg.Rk. denotes the average rank over datasets with reported results (lower is better); parentheses indicate coverage in (a).

(a) Unseen-edge-dominant datasets

Model	ML-20M	Taobao	Yelp	GoogleLocal	Flickr	YouTube	WikiLink	Avg.Rk.
JODIE	21.16±0.73	48.36±2.18	69.88±0.31	41.86±1.49	46.21±0.83	41.67±2.86	57.94±1.33	6.43 (7/7)
DyRep	19.00±1.69	40.03±2.40	57.69±1.05	37.73±1.34	38.04±4.19	35.12±4.13	42.63±1.33	8.29 (7/7)
TGAT	10.47±0.20	–	–	19.78±0.24	23.53±3.35	43.56±2.53	–	10.75 (4/7)
TGN	23.99±0.20	60.28±0.54	69.79±0.24	54.13±1.97	46.03±6.78	55.16±5.89	62.94±2.16	5.29 (7/7)
CAWN	12.31±0.02	–	25.71±0.09	18.26±0.02	48.69±6.08	47.55±1.08	–	9.00 (5/7)
GraphMixer	21.97±0.17	31.54±0.02	33.96±0.19	21.31±0.14	45.01±0.08	58.87±0.12	48.57±0.02	6.86 (7/7)
DyGFormer	–	–	21.68±0.20	18.39±0.02	49.58±2.87	46.08±3.44	–	9.00 (4/7)
DyGMamba	5.86±0.21	–	–	17.64±0.06	52.07±0.43	54.09±1.68	–	9.25 (4/7)
DyG-Mamba	–	–	–	17.34±0.36	55.86±0.14	56.71±0.23	–	8.00 (3/7)
FreeDyG	20.67±0.70	–	–	20.49±0.13	56.22±0.33	60.35±0.21	–	5.00 (4/7)
SGNN-HN	33.12±0.01	68.58±0.21	69.34±0.44	62.88±0.51	60.15±0.20	59.64±0.22	69.37±0.39	3.14 (7/7)
CRAFT	35.91±0.65	70.68±0.42	72.69±0.61	62.35±0.46	62.34±0.84	58.92±0.16	75.48±0.84	2.29 (7/7)
MambaTG	37.98±0.11	73.06±0.21	75.85±0.17	64.62±0.45	63.94±0.72	59.81±0.74	75.38±0.13	1.29 (7/7)

(b) Repeated-edge-dominant datasets

Model	Wikipedia	Reddit	MOOC	LastFM	UCI	Flights	Avg.Rk.
EdgeBank	77.94±0.00	77.88±0.00	9.76±0.00	19.07±0.00	39.43±0.00	54.46±0.00	11.50
JODIE	76.48±1.72	77.16±1.27	19.91±3.06	18.49±2.87	51.43±1.74	20.37±4.18	11.83
DyRep	67.42±4.21	72.33±2.14	17.71±1.27	17.76±6.56	14.00±2.71	15.62±0.47	13.67
TGAT	72.72±1.50	78.03±0.25	31.55±4.14	24.10±1.26	31.50±0.48	53.90±1.07	11.17
TGN	82.22±0.39	79.25±0.24	39.03±4.53	25.59±4.43	45.29±6.43	64.24±2.11	9.25
CAWN	86.07±0.08	87.66±0.04	27.02±0.57	35.10±0.34	66.96±0.48	80.06±2.10	8.00
GraphMixer	73.57±1.48	70.87±0.38	29.02±0.28	26.76±1.42	59.58±1.00	42.26±0.49	10.83
DyGFormer	88.69±0.04	88.70±0.05	42.20±1.31	46.50±0.51	76.61±0.26	84.13±3.46	4.83
DyGMamba	89.10±0.17	89.00±0.03	47.17±1.43	46.39±0.49	76.98±0.46	85.02±0.28	3.50
DyG-Mamba	88.50±0.21	88.77±0.14	47.33±0.23	53.42±0.19	78.25±0.60	81.47±0.04	4.00
FreeDyG	90.18±0.12	89.16±0.05	44.51±0.37	49.19±0.55	74.55±1.12	81.10±0.06	4.33
SGNN-HN	82.22±0.35	87.42±0.04	57.25±0.17	40.72±0.26	57.23±0.39	80.80±0.14	7.08
CRAFT-R	88.25±0.26	89.33±0.11	62.32±0.39	55.21±0.23	75.11±0.09	83.16±0.16	3.50
MambaTG-R	88.99±0.02	89.71±0.10	64.77±0.22	56.98±0.24	79.31±0.26	84.51±0.03	1.50

Evaluation Settings and Metrics. Following TGB-Seq (Yi et al., 2025b), we use Mean Reciprocal Rank (MRR) (Craswell, 2009) as the evaluation metric and adopt official negative samples

when available. Otherwise, we randomly sample 100 negative destinations per positive interaction. All datasets are split chronologically into 70%/15%/15% for training, validation, and testing, respectively. We report the average performance over three random seeds. Further evaluation and implementation details are provided in Appendix B.4.

5.2 Predictive Effectiveness and Computational Efficiency

Overall Predictive Performance. Table 2 reports MRR across all datasets. We use MambaTG for unseen-edge-dominant datasets and MambaTG-R for repeated-edge-dominant datasets, with the latter additionally incorporating source–candidate repetition frequency. EdgeBank, FreeDyG, DyGMamba, and DyGMamba results are from our experiments, while the remaining baseline results are taken from prior studies (Yi et al., 2025a;b). Missing entries indicate experiments exceeding the specified runtime or GPU memory budget. Further details are provided in Appendix B.4.

MambaTG and MambaTG-R achieve the best performance on 9 of the 13 datasets, with strong performance across the remaining datasets, and attain the best average rank in both regimes: 1.29 on unseen-edge-dominant datasets and 1.50 on repeated-edge-dominant datasets. We attribute these gains primarily to two design choices: (i) CAME jointly captures *historical-sequence evolution* and *candidate–history compatibility*, supporting prediction across repeated and unseen edges, as validated in Section 5.3 and Appendix G; and (ii) DRR refines historical patterns associated with different temporal scales while promoting compatibility between the refined history and candidate representations, as examined in Section 5.4 and Appendix F. Moreover, layer-wise caching encodes each source history once and reuses its terminal states across candidates, enabling MambaTG to scale to large datasets on which several competing methods exceed the computational budget, as analyzed in Figure 4 and Appendix J.1.

Long-Sequence Effectiveness. To assess MambaTG's ability to exploit long interaction histories, we evaluate it on LastFM, which benefits from longer historical sequences. We vary the maximum source-history length ℓ and compare MambaTG-R with other representative baselines.[1] As shown in Figure 3, MambaTG-R maintains strong performance as ℓ increases, while CRAFT-R and DyGFormer degrade with longer histories. These results demonstrate MambaTG's robustness to increasing history lengths and its ability to leverage extended temporal context effectively.

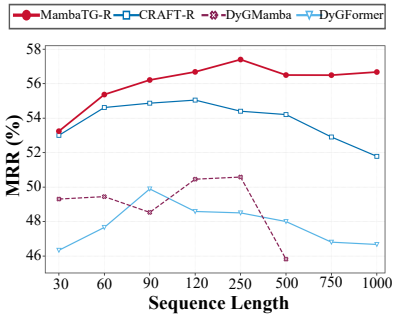


Figure 3: Long-sequence effectiveness on LastFM.

Performance–Efficiency Trade-off. Figure 4 illustrates the trade-off between predictive performance and computational efficiency on MOOC. MambaTG-R consistently achieves the highest test MRR while remaining competitive in parameter size and per-epoch training time. In particular, MambaTG-R achieves up to 8.9× faster per-epoch training and a 6.3× smaller parameter footprint than competing methods. Overall, MambaTG-R offers the most favorable trade-off between predictive performance and computational efficiency across the evaluated datasets. Results on UCI and LastFM in Appendix J.2.

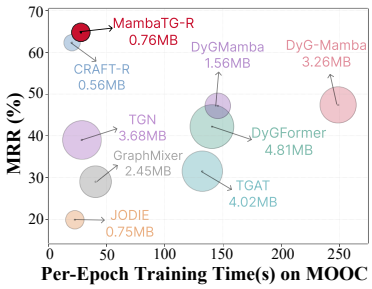


Figure 4: Performance versus efficiency on MOOC. Bubble size indicates model size.

5.3 Validating Historical-Sequence Evolution and Candidate–History Compatibility

[1] DyGMamba results at $\ell = 750$ and $\ell = 1000$ are omitted because training became numerically unstable and produced NaN prediction scores.

Our hypothesis is that MambaTG's effectiveness arises from jointly modeling *historical-sequence evolution* and *candidate–history compatibility*. To validate this hypothesis, we examine whether performance decreases when (i) *the chronological order of the source node's interaction history is disrupted* and (ii) *candidate-conditioned state updating is replaced with separate history–candidate fusion*.

Table 3: Macro-average MRR change (pp) relative to full MambaTG under signal ablations.

Variant	Repeated	Unseen
ShuffledTrain	-8.23	-1.56
ShuffledInference	-27.21	-3.53
ShuffledBoth	-4.22	-1.78
LateFusion	-2.88	-3.98

For (i), following Kim et al. (2025), we adopt *ShuffledTrain* and *ShuffledInference*, which permute the source node's interaction history only during training or evaluation, respectively. We additionally introduce *ShuffledBoth*, which shuffles histories during both training and evaluation. Table 3 shows lower macro-average MRR for all three variants in both dataset groups. In particular, the degradation under *ShuffledBoth*, which preserves neighbor identities while disrupting chronological order, shows that chronological order provides predictive information beyond neighbor identity. Degradation under *ShuffledInference* indicates that MambaTG's predictions depend on the chronological ordering of historical interactions. Together, these results show that MambaTG leverages *historical-sequence evolution* rather than treating the history as an unordered set.

For (ii), we introduce *LateFusion*, which retains the original history encoder but replaces the candidate-conditioned Mamba continuation with MLP-based fusion of the separately encoded history and candidate representations, a common prediction design in T-GNNs. The resulting degradation indicates that late-stage fusion does not fully recover the benefit of candidate continuation from the evolved historical state. The larger degradation on GoogleLocal and YouTube further suggests that *candidate–history compatibility* is particularly important for unseen-edge prediction. Overall, these findings support our hypothesis that jointly modeling *historical-sequence evolution* and *candidate–history compatibility* yields MambaTG's strongest performance. Detailed settings and per-dataset results are provided in Appendix D.

5.4 Ablation Study

We ablate five components of MambaTG: (1) Historical Spectral Refinement (**HSR**) and (2) Candidate Feature Refinement (**CFR**) within DRR, (3) **Query-relative time encoding**, (4) **Candidate activity recency**, and (5) **Historical repetition frequency**. Table 4 reports the macro-average MRR (%) reduction across two repeated-edge-dominant (UCI, MOOC) and two unseen-edge-dominant (GoogleLocal, Flickr) datasets. Variant definitions and per-dataset results are provided in Appendix E.

Table 4: Macro-average MRR change (pp) under component ablations. Repetition frequency applies only to repeated-edge-dominant datasets.

Removed component	**Repeated**	**Unseen**
HSR	-0.21	-1.74
CFR	-0.57	-1.28
Query-relative time encoding	-2.87	-1.77
Candidate activity recency	-2.35	-1.03
Historical repetition freq.	-5.67	N/A

Overall, all components contribute to MambaTG, but their importance differs across prediction regimes. Query-relative time encoding, candidate activity recency, and historical repetition frequency contribute more strongly to repeated-edge prediction, where the timing and frequency of previous source–candidate interactions provide direct evidence of recurrence. In contrast, HSR and CFR show more significant contributions to unseen-edge prediction, where prediction relies more heavily on patterns extracted from the source history and informative candidate representations.

6 Conclusion

In this paper, we identified *historical-sequence evolution* and *candidate–history compatibility* as complementary sources of predictive information beyond direct pairwise recurrence. We proposed MambaTG, a candidate-aware Mamba-based framework that jointly models these signals to capture how ordered interaction histories evolve and relate to each candidate. Across 13 temporal graph datasets, MambaTG achieves strong performance in repeated-edge-dominant and unseen-edge-dominant settings while efficiently handling long histories and large candidate sets.

AI USE STATEMENT

We used generative AI tools to assist with code implementation, provide writing feedback, identify relevant literature, and provide feedback on experimental methodology. We reviewed the AI-assisted work and take responsibility for the final content.

REFERENCES

Unai Alvarez-Rodriguez, Federico Battiston, Guilherme Ferraz De Arruda, Yamir Moreno, Matjaž Perc, and Vito Latora. Evolutionary dynamics of higher-order interactions in social networks. *Nature Human Behaviour*, 5(5):586–595, January 2021. ISSN 2397-3374. doi: 10.1038/s41562-020-01024-1. URL https://www.nature.com/articles/s41562-020-01024-1.

Weilin Cong, Si Zhang, Jian Kang, Baichuan Yuan, Hao Wu, Xin Zhou, Hanghang Tong, and Mehrdad Mahdavi. Do We Really Need Complicated Model Architectures For Temporal Networks? 2023. URL https://openreview.net/forum?id=ayPPc0SyLv1.

Nick Craswell. Mean Reciprocal Rank. In Ling Liu and M. Tamer Özsu (eds.), *Encyclopedia of Database Systems*, pp. 1703–1703. Springer US, Boston, MA, 2009. ISBN 978-0-387-35544-3 978-0-387-39940-9. doi: 10.1007/978-0-387-39940-9_488. URL http://link.springer.com/10.1007/978-0-387-39940-9_488.

Zifeng Ding, Yifeng Li, Yuan He, Antonio Norelli, Jingcheng Wu, Volker Tresp, Michael M. Bronstein, and Yunpu Ma. DyGMamba: Efficiently Modeling Long-Term Temporal Dependency on Continuous-Time Dynamic Graphs with State Space Models. *Transactions on Machine Learning Research*, February 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=sq5AJvVuha.

Aswathy Divakaran and Anuraj Mohan. Temporal Link Prediction: A Survey. *New Generation Computing*, 38(1):213–258, March 2020. ISSN 0288-3635, 1882-7055. doi: 10.1007/s00354-019-00065-z. URL http://link.springer.com/10.1007/s00354-019-00065-z.

Albert Gu and Tri Dao. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. 2024. URL https://openreview.net/forum?id=tEYskw1VY2.

Shenyang Huang, Farimah Poursafaei, Jacob Danovitch, Matthias Fey, Weihua Hu, Emanuele Rossi, Jure Leskovec, Michael Bronstein, Guillaume Rabusseau, and Reihaneh Rabbany. Temporal Graph Benchmark for Machine Learning on Temporal Graphs. In *Advances in Neural Information Processing Systems 36*, pp. 2056–2073, New Orleans, Louisiana, USA, 2023. Neural Information Processing Systems Foundation, Inc. (NeurIPS). ISBN 978-1-7138-9911-2. doi: 10.52202/075280-0099. URL http://www.proceedings.com/075280-0099.html.

Wang-Cheng Kang and Julian McAuley. Self-Attentive Sequential Recommendation. *2018 IEEE International Conference on Data Mining (ICDM)*, pp. 197–206, November 2018. doi: 10.1109/ICDM.2018.00035. URL https://ieeexplore.ieee.org/document/8594844/. Conference Name: 2018 IEEE International Conference on Data Mining (ICDM) ISBN: 9781538691595.

Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. Representation Learning for Dynamic Graphs: A Survey. *Journal of Machine Learning Research*, 21(70):1–73, 2020. ISSN 1533-7928. URL http://jmlr.org/papers/v21/19-447.html.

Sein Kim, Hongseok Kang, Kibum Kim, Jiwan Kim, Donghyun Kim, Minchul Yang, Kwangjin Oh, Julian McAuley, and Chanyoung Park. Lost in Sequence: Do Large Language Models Understand Sequential Recommendation? In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, KDD '25, pp. 1160–1171, New York, NY, USA, August 2025. Association for Computing Machinery. ISBN 979-8-4007-1454-2. doi: 10.1145/3711896.3737035. URL https://dl.acm.org/doi/10.1145/3711896.3737035.

Srijan Kumar, Xikun Zhang, and Jure Leskovec. Predicting Dynamic Embedding Trajectory in Temporal Interaction Networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1269–1278, Anchorage AK USA, July 2019. ACM. ISBN 978-1-4503-6201-6. doi: 10.1145/3292500.3330895. URL https://dl.acm.org/doi/10.1145/3292500.3330895.

Dongyuan Li, Shiyin Tan, Ying Zhang, Ming Jin, Shirui Pan, Manabu Okumura, and Renhe Jiang. DyG-Mamba: Continuous State Space Modeling on Dynamic Graphs. In *Advances in Neural Information Processing Systems 38*, pp. 143195–143224, San Diego, California, USA and Mexico City, Mexico, 2025. Neural Information Processing Systems Foundation, Inc. (NeurIPS). ISBN 979-8-3313-3827-5. doi: 10.52202/085713-4299. URL http://www.proceedings.com/085713-4299.html.

Chengkai Liu, Jianghao Lin, Jianling Wang, Hanzhou Liu, and James Caverlee. Mamba4Rec: Towards Efficient Sequential Recommendation with Selective State Space Models, June 2024. URL http://arxiv.org/abs/2403.03900. arXiv:2403.03900 [cs.IR].

Zhiqiang Pan, Fei Cai, Wanyu Chen, Honghui Chen, and Maarten De Rijke. Star Graph Neural Networks for Session-based Recommendation. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pp. 1195–1204, Virtual Event Ireland, October 2020. ACM. ISBN 978-1-4503-6859-9. doi: 10.1145/3340531.3412014. URL https://dl.acm.org/doi/10.1145/3340531.3412014.

Farimah Poursafaei, Andy Huang, Kellin Pelrine, and Reihaneh Rabbany. Towards Better Evaluation for Dynamic Link Prediction. June 2022. URL https://openreview.net/forum?id=1GVpwr2Tfdg.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pp. 452–461, Arlington, Virginia, USA, June 2009. AUAI Press. ISBN 978-0-9749039-5-8. URL https://dl.acm.org/doi/10.5555/1795114.1795167.

Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal Graph Networks for Deep Learning on Dynamic Graphs, October 2020. URL http://arxiv.org/abs/2006.10637. arXiv:2006.10637 [cs.LG].

H. Sorensen, D. Jones, M. Heideman, and C. Burrus. Real-valued fast Fourier transform algorithms. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35(6):849–863, June 1987. ISSN 0096-3518. doi: 10.1109/TASSP.1987.1165220. URL http://ieeexplore.ieee.org/document/1165220/.

Yuxing Tian, Yiyan Qi, and Fan Guo. FreeDyG: Frequency Enhanced Continuous-Time Dynamic Graph Model for Link Prediction. 2024. URL https://openreview.net/forum?id=82Mc5ilInM.

Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. DyRep: Learning Representations over Dynamic Graphs. 2019. URL https://openreview.net/forum?id=HyePrhR5KX.

Lu Wang, Xiaofu Chang, Shuang Li, Yunfei Chu, Hui Li, Wei Zhang, Xiaofeng He, Le Song, Jingren Zhou, and Hongxia Yang. TCL: Transformer-based Dynamic Graph Modelling via Contrastive Learning. *ArXiv*, May 2021a. URL https://www.semanticscholar.org/paper/TCL%3A-Transformer-based-Dynamic-Graph-Modelling-via-Wang-Chang/d37a8e9cea2f1e1b45b4912d243eee1b9dd5f756.

Yanbang Wang, Yen-Yu Chang, Yunyu Liu, Jure Leskovec, and Pan Li. Inductive Representation Learning in Temporal Networks via Causal Anonymous Walks. 2021b. URL https://openreview.net/forum?id=KYPz4YsCPj.

Yuxia Wu, Yuan Fang, and Lizi Liao. On the Feasibility of Simple Transformer for Dynamic Graph Modeling. In *Proceedings of the ACM Web Conference 2024*, pp. 870–880, Singapore Singapore, May 2024. ACM. ISBN 979-8-4007-0171-9. doi: 10.1145/3589334.3645622. URL https://dl.acm.org/doi/10.1145/3589334.3645622.

Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Inductive representation learning on temporal graphs. 2020. URL https://openreview.net/forum?id=rJeW1yHYwH.

Lu Yi, Runlin Lei, Fengran Mo, Yanping Zheng, Zhewei Wei, and Yuhang Ye. Future Link Prediction Without Memory or Aggregation. In *Advances in Neural Information Processing Systems 38*, pp. 2672–2700, San Diego, California, USA and Mexico City, Mexico, 2025a. Neural Information Processing Systems Foundation, Inc. (NeurIPS). ISBN 979-8-3313-3827-5. doi: 10.52202/085713-0084. URL http://www.proceedings.com/085713-0084.html.

Lu Yi, Jie Peng, Yanping Zheng, Fengran Mo, Zhewei Wei, Yuhang Ye, Yue Zixuan, and Zengfeng Huang. TGB-Seq Benchmark: Challenging Temporal GNNs with Complex Sequential Dynamics. *International Conference on Learning Representations*, 2025:88022–88042, May 2025b. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/db5ca61dbc08cf5143c05ad2d1b0b2ca-Abstract-Conference.html.

Le Yu. An Empirical Evaluation of Temporal Graph Benchmark, July 2023. URL https://arxiv.org/abs/2307.12510v5.

Le Yu, Leilei Sun, Bowen Du, and Weifeng Lv. Towards Better Dynamic Graph Learning: New Architecture and Unified Library. In *Advances in Neural Information Processing Systems 36*, pp. 67686–67700, New Orleans, Louisiana, USA, 2023. Neural Information Processing Systems Foundation, Inc. (NeurIPS). ISBN 978-1-7138-9911-2. doi: 10.52202/075280-2960. URL http://www.proceedings.com/075280-2960.html.

Le Yu, Zihang Liu, Leilei Sun, Bowen Du, Chuanren Liu, and Weifeng Lv. Continuous-Time User Preference Modelling for Temporal Sets Prediction. *IEEE Transactions on Knowledge and Data Engineering*, 36(4):1475–1488, April 2024. ISSN 1041-4347, 1558-2191, 2326-3865. doi: 10.1109/TKDE.2023.3309982. URL https://ieeexplore.ieee.org/document/10234655/.

A ABBREVIATIONS

Table 5: Key abbreviations and shorthand.

Term	Meaning
CAME	Candidate-Aware Mamba Encoder
DRR	Dual-Branch Representation Refinement
HSR	Historical Spectral Refinement
CFR	Candidate Feature Refinement
MRR	Mean Reciprocal Rank
-R	Suffix indicating historical repetition frequency
w/o	Without the specified component
pp	Percentage points
OOM	Exceeds the memory budget
OOT	Exceeds the time budget

B EXPERIMENTAL DETAILS

B.1 HARDWARE AND SOFTWARE ENVIRONMENT

All experiments were conducted on NVIDIA RTX PRO 6000 Blackwell GPUs with 96GB of memory. Our implementation uses Python 3.11, PyTorch 2.7.1, and CUDA 12.8, together with `mamba-ssm` 2.3.1 and `causal-conv1d` 1.6.2.

B.2 DATASET DETAILS

Our main evaluation includes 13 future link prediction datasets. Seven are from TGB-Seq (Yi et al., 2025b): ML-20M, Taobao, Yelp, GoogleLocal, Flickr, YouTube, and WikiLink. The remaining six are commonly used temporal graph datasets: Wikipedia, Reddit, MOOC, LastFM, UCI, and Flights. We additionally evaluate MambaTG on three datasets from TGB (Huang et al., 2023): `tgbl-comment`, `tgbl-coin`, and `tgbl-flight`. These additional results are reported in Appendix J.3.

Following Yi et al. (2025a), we classify datasets as *unseen-edge-dominant* if most target interactions are previously unseen edges, and as *repeated-edge-dominant* if most are repeated edges. The datasets are obtained from the TGB-Seq collection, the TGB repository, and the public Zenodo archive. Table 6 summarizes their statistics; for bipartite datasets, the two node counts correspond to the source and destination node types.

B.3 BASELINE DETAILS

We compare MambaTG with established temporal graph models, recent Mamba-based architectures, a sequential recommendation method, and a candidate-aware future link predictor. Their main designs are summarized below:

EdgeBank. EdgeBank (Poursafaei et al., 2022) is a non-parametric baseline that stores previously observed edges and scores candidate links according to whether they have appeared in the interaction history. It captures edge recurrence without learning node representations or model parameters. Given its explicit focus on edge recurrence, we report EdgeBank only on repeated-edge-dominant datasets.

JODIE. JODIE (Kumar et al., 2019) maintains dynamic source and destination embeddings using two coupled recurrent neural networks. After each interaction, it jointly updates the embeddings of both endpoints and projects the source embedding forward to the query time for future interaction prediction.

DyRep. DyRep (Trivedi et al., 2019) models communication and association events as two interacting temporal processes. It updates persistent node representations using temporally attentive neighborhood information and parameterizes event occurrence through a temporal point process.

TGAT. TGAT (Xu et al., 2020) computes time-dependent node representations by applying self-attention over sampled temporal neighborhoods. It uses a functional time encoding to represent the elapsed time of historical interactions and supports inductive inference without maintaining persistent node memories.

TGN. TGN (Rossi et al., 2020) maintains a persistent memory for each node. Incoming interactions generate messages that are aggregated and used to update the endpoint memories, after which a graph-based embedding module combines the memories with temporal-neighborhood information for prediction.

CAWN. CAWN (Wang et al., 2021b) samples causal temporal walks from the two endpoints of a query edge. It anonymizes node identities through walk-based occurrence patterns and encodes the resulting walks to capture temporal motifs and structural relationships relevant to the queried pair.

GraphMixer. GraphMixer (Cong et al., 2023) uses an MLP-Mixer-based link encoder to summarize recent temporal interactions, a mean-pooling node encoder to aggregate node features, and an MLP classifier for link prediction. It provides a memory-free alternative to recurrent and attention-based temporal graph models.

DyGFormer. DyGFormer (Yu et al., 2023) extracts the first-hop interaction histories of both endpoints and uses neighbor co-occurrence encoding to represent correlations between them. The histories are divided into patches and processed by a Transformer to capture long-range temporal dependencies efficiently.

FreeDyG. FreeDyG (Tian et al., 2024) is a frequency-enhanced continuous-time dynamic graph model for link prediction. It introduces node interaction frequency encoding to explicitly capture the proportion of common neighbors and the interaction frequency of each node pair. By modeling interaction patterns in the frequency domain, FreeDyG aims to capture periodic behavior and temporal shifts in node interactions.

DyGMamba. DyGMamba (Ding et al., 2025) employs two state-space components. A node-level SSM encodes historical node interactions, while a time-level SSM captures temporal patterns and dynamically selects relevant information from the extended interaction history.

DyG-Mamba. DyG-Mamba (Li et al., 2025) formulates dynamic graph learning using a continuous state-space architecture inspired by Mamba. It uses irregular time intervals as control signals for the state transitions, allowing the model to adjust how historical information is retained or forgotten.

SGNN-HN. SGNN-HN (Pan et al., 2020) was originally proposed for session-based recommendation. It represents an ordered interaction sequence as a star graph, allowing the star node to exchange information with all items in the sequence. A highway network combines item representations before and after graph propagation, and an attention mechanism aggregates them into a sequence representation for candidate prediction.

SASRec. SASRec (Kang & McAuley, 2018) is a self-attention-based sequential recommendation model that encodes a user's ordered item-interaction history to predict the next item, capturing dependencies among past interactions.

Mamba4Rec. Mamba4Rec (Liu et al., 2024) is a Mamba-based sequential recommendation model that encodes a user's ordered item-interaction history for next-item prediction through selective state-space updates.

CRAFT. CRAFT (Yi et al., 2025a) represents nodes using trainable identifier embeddings and applies cross-attention from each candidate destination to the source node's recent interactions. It therefore performs direct candidate-to-history matching without maintaining persistent node memories or aggregating the candidate's temporal neighborhood. Following its original design, the repetition-enhanced CRAFT-R variant additionally supplies historical repetition frequency to the predictor on repeated-edge-dominant datasets.

Table 6: Statistics of the 16 datasets used in our experiments. Regime indicates whether a dataset is unseen-edge-dominant (Unseen) or repeated-edge-dominant (Repeated).

Dataset	Nodes	Edges	Timestamps	Repeat Ratio (%)	Density (%)	Bipartite	Domain	Regime
ML-20M	100,785/9,646	14,494,325	9,993,250	0	1.49×10^{0}	Yes	Movie rating	Unseen
Taobao	760,617/863,016	18,853,792	139,171	16.58	2.87×10^{-3}	Yes	E-commerce interaction	Unseen
Yelp	1,338,688/405,081	19,760,293	14,646,734	25.18	3.64×10^{-3}	Yes	Business review	Unseen
GoogleLocal	206,244/267,336	1,913,967	1,771,060	0	3.47×10^{-3}	Yes	Business review	Unseen
Flickr	233,836	7,223,559	134	0	1.32×10^{-2}	No	Who-to-follow	Unseen
YouTube	402,422	3,288,028	203	0	2.03×10^{-3}	No	Who-to-follow	Unseen
WikiLink	1,361,972	34,163,774	2,198	0	1.84×10^{-3}	No	Web link	Unseen
Wikipedia	8,227/1,000	157,474	152,757	88.41	1.91×10^{0}	Yes	Interaction	Repeated
Reddit	10,000/984	672,447	669,065	88.32	6.83×10^{0}	Yes	Social	Repeated
MOOC	7,047/97	411,749	345,600	56.66	6.02×10^{1}	Yes	Interaction	Repeated
LastFM	980/1,000	1,293,103	1,283,614	88.01	1.32×10^{2}	Yes	Interaction	Repeated
UCI	1,899	59,835	58,911	66.06	1.66×10^{0}	No	Social	Repeated
Flights	13,169	1,927,145	122	79.50	1.11×10^{0}	No	Transport	Repeated
tgbl-comment	994,790	44,314,507	30,998,030	19.81	4.48×10^{-3}	No	Social	Unseen
tgbl-coin	638,486	22,809,486	1,295,720	82.93	5.60×10^{-3}	No	Transaction	Repeated
tgbl-flight	18,143	67,169,570	1,385	96.48	2.04×10^{1}	No	Transport	Repeated

B.4 Evaluation and Implementation Details

Following TGB-Seq (Yi et al., 2025b) and DyGLib (Yu et al., 2023), we use the Adam optimizer with a learning rate of 10^{-4} and a batch size of 200, which is increased to 400 for large datasets. Models are trained for up to 100 epochs with early stopping using a patience of 20 epochs. We impose a computational budget of 1.5 hours per epoch and 32 GB of GPU memory; exceeding these limits is denoted as out of time (OOT) and out of memory (OOM), respectively. We use official negative samples whenever available. TGB (Huang et al., 2023) provides either 20 or 100 negative destinations per positive edge, depending on the dataset, while TGB-Seq provides 100 randomly sampled negative destinations. For datasets without official negative samples, we follow TGB-Seq and randomly sample 100 negative destinations per positive edge. For small datasets, we additionally follow CRAFT (Yi et al., 2025a) and perform collision checks to exclude positive edges from the sampled negatives. To ensure fair comparison, we use published baseline results when they follow the same evaluation protocol and closely matched training settings; otherwise, we reproduce the methods using our unified pipeline.

B.5 Hyperparameter Settings

Baseline configurations. For each baseline we reproduce, we use the best configuration reported for the corresponding baseline–dataset pair in the original paper or benchmark study, including TGB-Seq (Yi et al., 2025b), CRAFT (Yi et al., 2025a), or DyGLib (Yu et al., 2023). When no dataset-specific configuration is reported, we use the default configuration recommended by the baseline's official implementation.

MambaTG configurations. We select MambaTG hyperparameters according to validation MRR. Specifically, we tune the maximum history length $L_{\max}$, number of CAME layers R, hidden-layer dropout rate p_{h}, and node-embedding dropout rate p_{e} using the search space in Table 7. Across all datasets, we fix the Mamba state dimension to $d_{\mathrm{state}} = 32$, the expansion factor to 2, and the causal-convolution width to 4. Table 8 reports the selected configurations, where d_e denotes the node-embedding dimension and B the training batch size. Following Yi et al. (2025a), we adopt the same dataset-specific settings for the node-embedding dimension d_e and training batch size B to ensure a fair comparison.

Table 7: Hyperparameter search space for MambaTG.

Hyperparameter	**Search space**
Maximum history length, $L_{\max}$	$\{30, 60, 90, 120, 250\}$
Number of CAME layers, R	$\{1, 2\}$
Hidden-layer dropout, p_{h}	$\{0.1, 0.2\}$
Node-embedding dropout, p_{e}	$\{0.1, 0.2\}$

Table 8: Selected MambaTG hyperparameters for each dataset.

Dataset	$L_{\max}$	R	p_{h}	p_{e}	d_e	B
UCI	30	1	0.1	0.1	64	200
Wikipedia	90	2	0.2	0.2	64	200
Reddit	120	1	0.1	0.1	64	200
MOOC	30	2	0.1	0.1	64	200
Flights	250	1	0.1	0.1	64	200
LastFM	250	2	0.1	0.1	64	200
GoogleLocal	250	2	0.1	0.2	128	200
Flickr	90	1	0.1	0.1	128	400
YouTube	120	2	0.1	0.1	128	400
Taobao	60	2	0.1	0.1	128	400
Yelp	60	2	0.1	0.1	128	400
ML-20M	60	1	0.1	0.1	128	400
WikiLink	250	1	0.1	0.1	128	400
tgbl-comment	30	1	0.1	0.1	128	200
tgbl-coin	250	1	0.1	0.1	128	200
tgbl-flight	90	1	0.1	0.1	128	200

C Method Details

C.1 Mamba Block and Feed-Forward Network

Mamba block. Each Mamba block takes an input sequence $\mathbf{X} \in \mathbb{R}^{m \times d}$. Following Gu & Dao (2024), it applies an input projection and a causal one-dimensional convolution, processes the resulting sequence with a selective SSM, and gates the SSM output:

$$\begin{aligned} \mathbf{U} &= \mathrm{SiLU}(\mathrm{Conv1D}(\mathrm{Linear}_{\mathrm{in}}(\mathbf{X}))), \qquad \mathbf{Y} = \mathrm{SSM}_{\overline{\mathbf{A}},\overline{\mathbf{B}},\mathbf{C}}(\mathbf{U}), \\ \mathrm{MambaBlock}(\mathbf{X}) &= \mathrm{Linear}_{\mathrm{out}}(\mathbf{Y} \odot \mathrm{SiLU}(\mathrm{Linear}_{\mathrm{gate}}(\mathbf{X}))). \end{aligned} \tag{8}$$

The selective SSM processes $\mathbf{U}$ recurrently. At position t, it maps $\mathbf{u}_t$ to $\mathbf{y}_t$ through the hidden state $\mathbf{h}_t$:

$$\mathbf{h}_t = \overline{\mathbf{A}}_t \mathbf{h}_{t-1} + \overline{\mathbf{B}}_t \mathbf{u}_t, \qquad \mathbf{y}_t = \mathbf{C}_t \mathbf{h}_t. \tag{9}$$

Here, $\overline{\mathbf{A}}_t$ controls how the preceding state is retained, $\overline{\mathbf{B}}_t$ determines how the current input updates the state, and $\mathbf{C}_t$ maps the hidden state to the output. Mamba derives $\mathbf{B}_t$, $\mathbf{C}_t$, and the discretization step $\boldsymbol{\Delta}_t$ from the current input:

$$\mathbf{B}_t = \mathbf{u}_t \mathbf{W}_B, \qquad \mathbf{C}_t = \mathbf{u}_t \mathbf{W}_C, \qquad \boldsymbol{\Delta}_t = \mathrm{softplus}(\mathrm{Broadcast}_{d_{\mathrm{inner}}}(\mathbf{u}_t \mathbf{W}_\Delta) + \mathbf{b}_\Delta). \tag{10}$$

Under zero-order-hold discretization, the shared continuous state-transition parameter $\mathbf{A}$ and input-dependent parameter $\mathbf{B}_t$ are converted into their discrete forms:

$$\overline{\mathbf{A}}_t = \exp(\boldsymbol{\Delta}_t \mathbf{A}), \qquad \overline{\mathbf{B}}_t = (\boldsymbol{\Delta}_t \mathbf{A})^{-1} [\exp(\boldsymbol{\Delta}_t \mathbf{A}) - \mathbf{I}] \, \boldsymbol{\Delta}_t \mathbf{B}_t. \tag{11}$$

Feed-forward network. Each Mamba block is followed by a position-wise FFN:

$$\mathrm{FFN}(\mathbf{X}) = \mathrm{GELU}(\mathbf{X}\mathbf{W}_1 + \mathbf{b}_1)\, \mathbf{W}_2 + \mathbf{b}_2, \tag{12}$$

where $\mathbf{W}_1 \in \mathbb{R}^{d \times 4d}$ and $\mathbf{W}_2 \in \mathbb{R}^{4d \times d}$.

C.2 CAME IMPLEMENTATION.

At a given query time, CAME encodes each source history once and retains its layer-wise terminal recurrent caches $K_H^{(r)}$ for candidate continuation during both training and inference. Each candidate then continues independently from the same terminal history caches, so candidates share the encoded history without affecting one another.

For training, however, the terminal SSM state returned by the standard `mamba-ssm` selective-scan interface does not support gradient propagation. To allow the BPR objective to propagate gradients through candidate continuation to the history encoder, we extend each history scan from length $L_{\max}$ to $L_{\max} + d_{\text{state}}$, and append d_{state} auxiliary positions to expose the terminal SSM state through the scan outputs. For each auxiliary position $t = L_{\max} + j$, we set $\mathbf{u}_t = \mathbf{0}$, $\mathbf{B}_t = \mathbf{0}$, and $\widetilde{\mathbf{\Delta}}_t = \Delta_{\text{off}}\mathbf{1}$, where Δ_{off} is chosen such that $\widetilde{\mathbf{\Delta}}_t + \mathbf{b}_\Delta$ is componentwise sufficiently negative. Thus, $\mathbf{\Delta}_t = \text{softplus}(\widetilde{\mathbf{\Delta}}_t + \mathbf{b}_\Delta) \approx \mathbf{0}$, giving $\overline{\mathbf{A}}_t = \exp(\mathbf{\Delta}_t\mathbf{A}) \approx \mathbf{I}$ and $\overline{\mathbf{B}}_t\mathbf{u}_t = \mathbf{0}$, and hence $\mathbf{h}_{L_{\max}+j} \approx \mathbf{h}_{L_{\max}}$. Although the softplus parameterization keeps $\mathbf{\Delta}_t > 0$, the sufficiently negative Δ_{off} makes the resulting state decay numerically negligible. We then read out the terminal state from the auxiliary outputs by setting $\mathbf{C}_{L_{\max}+j} = \mathbf{e}_j^\top$ at auxiliary position $t = L_{\max} + j$, where $\mathbf{e}_j \in \mathbb{R}^{d_{\text{state}}}$ is the j-th standard basis vector. Since $\mathbf{u}_{L_{\max}+j} = \mathbf{0}$ and $\mathbf{h}_{L_{\max}+j} \approx \mathbf{h}_{L_{\max}}$:

$$\mathbf{y}_{L_{\max}+j} = \mathbf{C}_{L_{\max}+j}\mathbf{h}_{L_{\max}+j} \approx \mathbf{e}_j^\top \mathbf{h}_{L_{\max}}. \tag{13}$$

The d_{state} auxiliary outputs therefore jointly recover the terminal SSM state through the standard selective-scan interface, without modifying its CUDA kernel. The SiLU gate is applied only to the original history outputs $\mathbf{y}_{1:L_{\max}}$, while the auxiliary outputs are used directly for state readout.

For batched histories with different true lengths, each history is right-padded to $L_{\max}$. At padded positions, we set $\mathbf{u}_t = \mathbf{0}$, $\mathbf{B}_t = \mathbf{0}$, and $\tilde{\mathbf{\Delta}}_t = \Delta_{\text{off}}\mathbf{1}$, following the approximate-identity construction above. Consequently, the scan can operate on the common length $L_{\max}$ without introducing state decay after the last valid interaction. The terminal causal-convolution state is gathered from the last valid positions of each history rather than from the padded positions.

D ADDITIONAL DETAILS AND RESULTS FOR SIGNAL ABLATIONS

We provide the experimental settings for ShuffledTrain, ShuffledInference, ShuffledBoth, and Late-Fusion, which examine *historical-sequence evolution* and *candidate–history compatibility*. Unless otherwise stated, all experiments follow the same data splits, historical sampling strategy, negative samples, model configuration, optimization settings, and evaluation protocol as MambaTG.

D.1 SHUFFLEDTRAIN

ShuffledTrain examines how disrupting chronological order during training affects predictive performance. We randomly permute the source node's historical interactions for every training query. Each permutation is generated once before training and remains fixed across epochs, preventing repeated shuffling from acting as stochastic data augmentation. All attributes associated with an interaction, including its timestamp, are permuted together, while the candidate remains the final input token. The model is trained from scratch on these shuffled histories and evaluated using the original chronologically ordered test histories. This variant probes the role of learning from ordered interactions in capturing *historical-sequence evolution*.

D.2 SHUFFLEDINFERENCE

ShuffledInference examines whether the trained model relies on chronological ordering when making predictions. We train MambaTG normally and randomly permute the source histories only during test-time inference, without retraining or modifying its parameters. As in ShuffledTrain, all attributes associated with each interaction are permuted together, and the candidate remains the final token. A performance reduction relative to standard chronological inference indicates that the model's predictions are sensitive to historical order.

D.3 SHUFFLEDBOTH

ShuffledBoth examines the benefit of chronological ordering under matched training and evaluation ordering conditions. Since ShuffledTrain and ShuffledInference use different ordering conditions during training and evaluation, their performance changes may also reflect sensitivity to this mismatch. To address this, ShuffledBoth uses shuffled histories during both training and evaluation.

The model is trained from scratch following the ShuffledTrain procedure and evaluated on shuffled test histories. This comparison tests whether the chronological order of a source node's interaction history provides predictive information for future link prediction beyond historical-neighbor identities.

D.4 LATEFUSION

LateFusion examines whether candidate–history compatibility is better captured by incorporating the candidate into the state transition rather than combining independently encoded representations at the prediction stage. In MambaTG, the complete history encoder processes the historical sequence $H_u(t)$ and retains the layer-wise history caches. The candidate then continues through all R CAME layers using these caches, following Equation (6):

$$\begin{aligned}(\mathbf{h}_H, \mathcal{K}_H) &= \mathrm{HistoryEncoder}(H_u(t))\,,\\ \mathbf{h}_c &= \mathrm{CandidateContinuation}\big(\mathbf{z}_{H,c}^{\mathrm{CFR}}; \mathcal{K}_H\big)\,,\\ \hat{y}_{u,c} &= \mathrm{MLP}(\mathbf{h}_c)\,.\end{aligned} \tag{14}$$

Here, $\mathbf{h}_H$ denotes the final history representation, $\mathcal{K}_H$ contains the recurrent and convolutional states retained at each layer, and $\mathbf{z}_{H,c}^{\mathrm{CFR}}$ denotes the refined candidate representation.

LateFusion removes this candidate continuation while keeping the complete chronological history encoder unchanged. Instead, it independently refines the candidate with a residual MLP and combines it with the history representation only at the prediction stage:

$$\begin{aligned}\mathbf{h}_H &= \mathrm{HistoryEncoder}(H_u(t))\,,\\ \widetilde{\mathbf{z}}_{H,c}^{\mathrm{CFR}} &= \mathrm{ResMLP}\big(\mathbf{z}_{H,c}^{\mathrm{CFR}}\big)\,,\\ \hat{y}_{u,c} &= \mathrm{MLP}\big([\mathbf{h}_H \,\|\, \widetilde{\mathbf{z}}_{H,c}^{\mathrm{CFR}}]\big)\,.\end{aligned} \tag{15}$$

For LateFusion, only the history representation is used at the prediction stage, and the history caches are not used for candidate processing. The residual MLP is sized to approximately match the parameter count of one CAME layer. MambaTG reuses its history-encoder parameters for candidate continuation, while LateFusion adds a separate residual MLP and uses a wider prediction head. As a result, LateFusion has more total trainable parameters. Thus, the comparison evaluates candidate-conditioned continuation against late fusion with a larger parameter budget, ensuring that any advantage of MambaTG cannot be attributed simply to a higher total parameter count. LateFusion is trained from scratch using the same training and evaluation settings as MambaTG. The explicit features used by the MambaTG link classifier are retained but omitted from both equations for clarity.

D.5 PER-DATASET RESULTS

Figure 5 reports the per-dataset results for all four signal ablations.

E ABLATION STUDY DETAILS

We conduct an ablation study on four datasets to evaluate the contributions of MambaTG's key components. Specifically, we consider five variants: (1) **w/o HSR**, which removes the Historical Spectral Refinement (HSR) applied to the source node's historical interaction sequence; (2) **w/o CFR**, which removes Candidate Feature Refinement (CFR), preventing the candidate representations from being adaptively aligned with the refined historical representation space; (3) **w/o query-rel**, which removes the query-relative time encoding from the input representations; (4) **w/o recency**, which removes the candidate activity recency feature concatenated at the prediction layer; and (5) **w/o repeat**, which removes the candidate repetition-frequency feature from the prediction layer. The

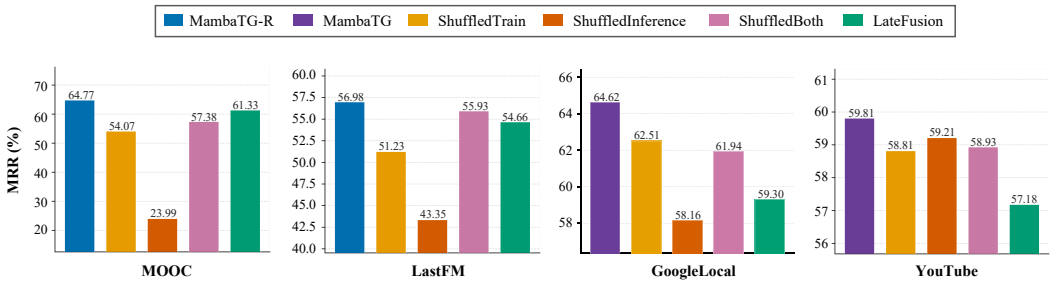


Figure 5: Per-dataset signal-ablation results on the repeated-edge-dominant datasets (MOOC and LastFM) and the unseen-edge-dominant datasets (GoogleLocal and YouTube). Results are reported as MRR (%), averaged over three random seeds.

repetition-frequency feature is evaluated only on repeated-edge datasets. Figure 6 presents the complete per-dataset results.

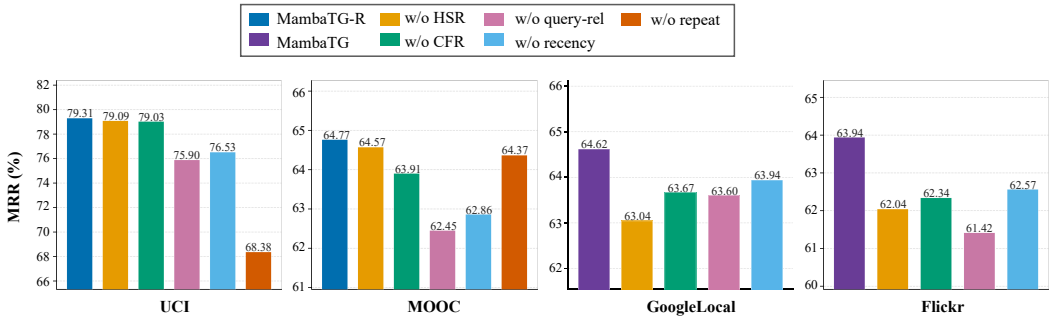


Figure 6: Per-dataset component-ablation results on the repeated-edge-dominant datasets (UCI and MOOC) and the unseen-edge-dominant datasets (GoogleLocal and Flickr). MambaTG-R denotes MambaTG augmented with historical repetition frequency. Each ablation variant removes one component from the corresponding complete model. Results are reported as MRR (%), averaged over three random seeds.

F Effectiveness and Transferability of Dual-Branch Representation Refinement (DRR)

DRR improves the inputs to candidate-aware temporal modeling through two complementary branches: HSR refines historical patterns at different temporal scales, while CFR recalibrates candidate features to promote compatibility with the refined history. To examine whether its benefits extend beyond MambaTG, we incorporate DRR into CRAFT and compare the resulting **CRAFT-DRR** with the original CRAFT. Specifically, we construct CRAFT-DRR by applying HSR to the history embeddings after positional encoding and CFR to the candidate embeddings, with the resulting representations passed to CRAFT's cross-attention module. We also compare **MambaTG** with **MambaTG without DRR** to isolate the contribution of DRR within MambaTG. We conduct these comparisons on UCI and GoogleLocal, representing repeated- and unseen-edge-dominant settings, respectively. Within each framework, the variants with and without DRR use the same model configuration and training settings, differing only in the inclusion of DRR. On UCI, all variants additionally use historical repetition frequency and are denoted by the suffix "-R." The CRAFT results in this analysis are obtained from our own experiments and may therefore differ from the originally reported results presented in Table 2.

Results. Figure 7 shows that DRR improves both CRAFT and MambaTG on both datasets, indicating that its benefits are not specific to either architecture. Comparing the two frameworks without DRR, MambaTG consistently outperforms CRAFT, showing that its candidate-aware sequen-

tial modeling provides an advantage independently of DRR. This advantage is retained when both frameworks use DRR, with MambaTG still outperforming CRAFT-DRR on both datasets. Thus, the gains from MambaTG cannot be attributed solely to representation refinement. Instead, the results support complementary contributions from DRR and MambaTG's joint modeling of *historical-sequence evolution* and *candidate–history compatibility*. The consistent improvement of CRAFT with DRR further suggests that the refined representations are independently useful for candidate–history compatibility, particularly when predicting unseen edges where the source and candidate have no previous direct interaction.

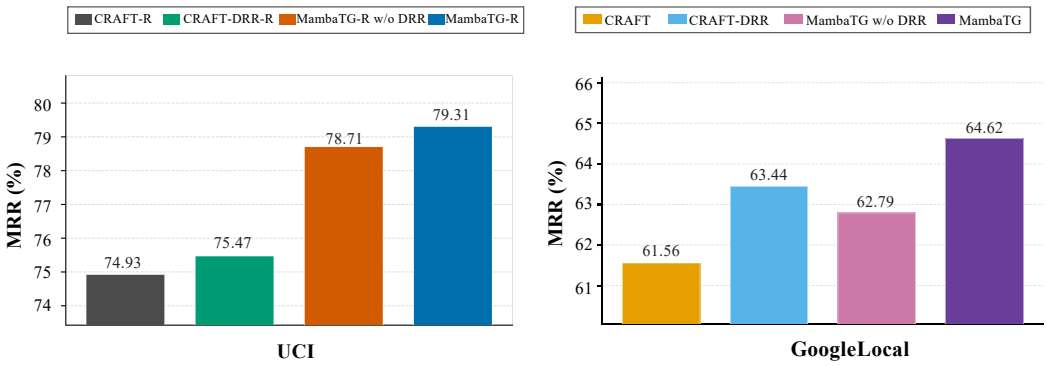


Figure 7: Impact of DRR on CRAFT and MambaTG on UCI (repeated-edge-dominant) and GoogleLocal (unseen-edge-dominant) datasets. Results are reported as MRR (%), averaged over three random seeds.

G Effectiveness of Candidate-Aware Mamba Encoder (CAME)

CAME jointly models *historical-sequence evolution* and *candidate–history compatibility* by treating each candidate as a continuation of the source node's interaction sequence. To assess the role of Mamba-based sequence modeling within CAME, we compare MambaTG with GRUTG. GRUTG replaces each Mamba block with a Gated Recurrent Unit (GRU) of the same hidden dimension while retaining all other model components, training settings, and the candidate-continuation procedure. We evaluate both models on UCI and GoogleLocal, representing repeated-edge-dominant and unseen-edge-dominant settings, respectively. On UCI, both models additionally use historical repetition frequency and are denoted as GRUTG-R and MambaTG-R.

Results. As shown in Figure 8, MambaTG consistently outperforms GRUTG on both datasets, indicating that Mamba provides a more effective sequential backbone for CAME. This advantage may stem from Mamba's structured SSM state and selective recurrence, which capture different aspects of the interaction history over different timescales and selectively retain informative events. The larger gain on the unseen-edge-dominant GoogleLocal dataset further suggests that Mamba is particularly useful when direct source–candidate evidence is limited and prediction relies more heavily on patterns in the source history. Overall, these results support the use of Mamba as the sequential encoder for jointly modeling *historical-sequence evolution* and *candidate–history compatibility* in CAME.

H Comparison with Recommendation Models

MambaTG targets future link prediction on temporal graphs, a general task arising in diverse applications such as recommendation, social networks, communication systems, and transportation networks. It supports both bipartite and non-bipartite graphs and covers both repeated and unseen future edges. Recommendation can be viewed as a specific application of this task, where temporal interactions typically occur between users and items in a bipartite graph.

To further examine the effectiveness of MambaTG, we compare it with two representative sequential recommendation models: the Transformer-based SASRec (Kang & McAuley, 2018) and the

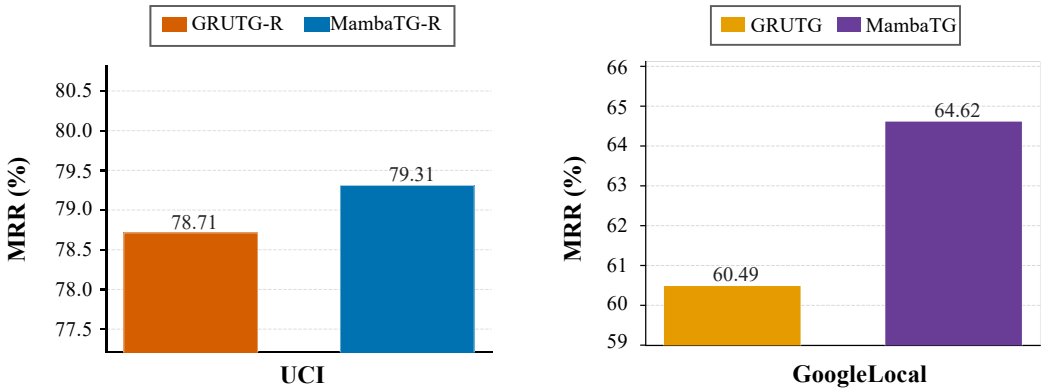


Figure 8: Effect of CAME on the repeated-edge-dominant UCI dataset and the unseen-edge-dominant GoogleLocal dataset. MambaTG is compared with GRUTG. Results are reported as MRR (%), averaged over three random seeds.

Mamba-based Mamba4Rec (Liu et al., 2024). The comparison covers MOOC and GoogleLocal, representing repeated-edge-dominant and unseen-edge-dominant settings, respectively. As shown in Figure 9, MambaTG or MambaTG-R consistently outperforms both recommendation models across both datasets. These results suggest that models designed specifically for sequential user–item recommendation may not fully capture the broader characteristics of future link prediction, particularly the distinction between repeated and unseen edges.

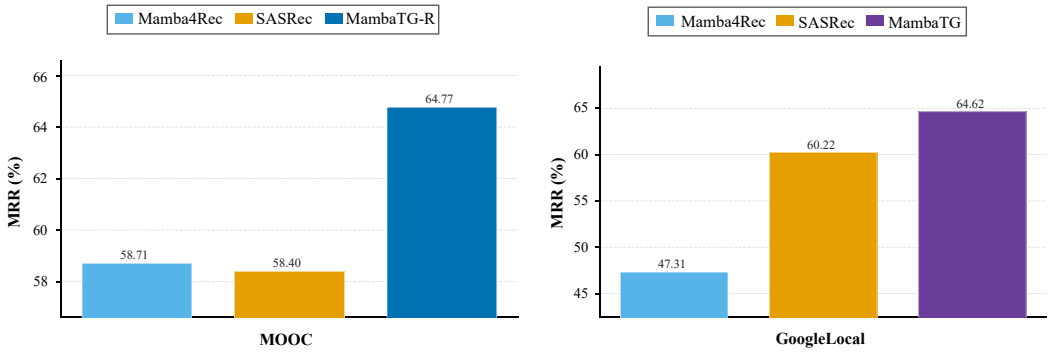


Figure 9: Comparison with sequential recommendation models on MOOC (repeated-edge-dominant) and GoogleLocal (unseen-edge-dominant) datasets. Results are reported as MRR (%), averaged over three random seeds.

I Time Complexity Analysis

We analyze a batch of B source nodes, each evaluated against q candidate destinations. Let ℓ denote the number of historical interactions per node, d the feature dimension, and $\bar{d}$ the average node degree. We report only dominant terms and treat architectural hyperparameters, including the number of layers and attention heads, as well as the Mamba expansion factors and state dimensions, as constants. For TGAT, however, we explicitly account for its two-layer neighborhood expansion, which introduces two-hop histories and changes the dependence on ℓ.

Historical-neighbor extraction. We use neighbor extraction to refer to the temporal-index lookup operations required to locate the relevant histories in node adjacency lists. Following the common implementation in DyGLib (Yu et al., 2023), retrieving the most recent historical interactions before the query time requires $O(\log \bar{d})$ per node. TGN (Rossi et al., 2020), GraphMixer (Cong et al., 2023), DyGFormer, DyG-Mamba (Li et al., 2025), and DyGMamba (Ding et al., 2025) retrieve his-

tories for both the source and all candidate destinations, resulting in $O(B(1+q)\log \bar{d})$. TGAT (Xu et al., 2020) additionally performs temporal-index lookups for up to ℓ first-hop neighbors in its two-layer architecture. Thus, it performs $O(B(1+q)(1+\ell))$ lookup operations, each costing $O(\log \bar{d})$, resulting in a neighbor-extraction cost of $O(B(1+q)(1+\ell)\log \bar{d})$. In contrast, CRAFT (Yi et al., 2025a) and MambaTG retrieve only the source history, reducing the cost to $O(B \log \bar{d})$.

Edge-likelihood computation. TGN and GraphMixer process the ℓ historical interactions of both the source and every candidate destination, resulting in $O(B(1+q)\ell d^2)$. TGAT aggregates two-hop histories with two graph-attention layers, where ℓ is the sampling size at each hop, requiring $O(B(1+q)\ell^2 d^2)$. DyGFormer applies feature projection and self-attention to each endpoint, with costs of $O(\ell d^2)$ and $O(\ell^2 d)$, respectively. Its total complexity is therefore $O(B(1+q)(\ell d^2 + \ell^2 d))$. DyG-Mamba and DyGMamba process the full history for every source–candidate pair using state-space models, resulting in $O(Bq\ell d^2)$. CRAFT projects the source history and candidates in $O(B(\ell+q)d^2)$, followed by candidate-to-history attention with complexity $O(Bq\ell d)$. Its total cost is thus $O(B(\ell+q)d^2 + Bq\ell d)$. MambaTG incurs $O(B\ell d \log \ell)$ time for DRR, whose dominant cost arises from frequency-domain refinement of each source history. It then encodes each history once in $O(B\ell d^2)$ and evaluates all candidates using the cached states in $O(Bqd^2)$. Therefore, its total edge-likelihood computation is $O(B\ell d \log \ell + B(\ell+q)d^2)$.

J Additional Experimental Results

J.1 Long Sequence Efficiency

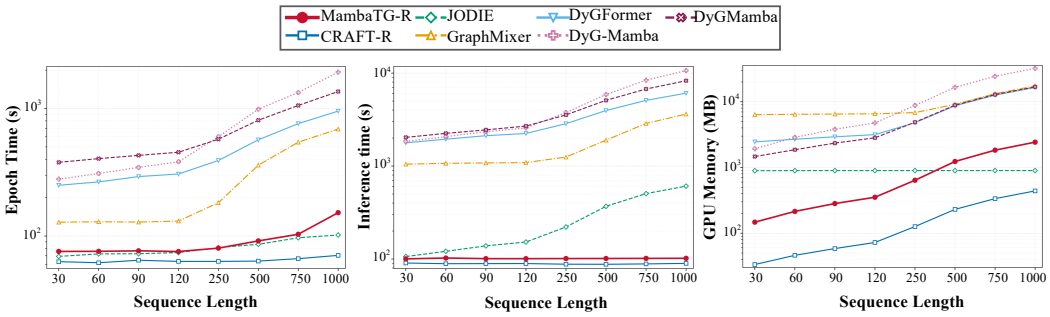


Figure 10: Computational efficiency on LastFM Dataset as the maximum source-history length ℓ increases. Training time per epoch, inference time, and peak GPU memory usage.

We evaluate the scalability of MambaTG as the source-node history length increases on LastFM, measuring per-epoch training time, inference time, and peak GPU memory usage. Training time measures the wall-clock time of one training epoch, excluding validation and data loading before the epoch. Inference time measures the wall-clock time of the 100-negative test pass, including the operations required for each method to produce predictions. Peak GPU memory is measured as the maximum allocated GPU memory over the full run, including training and evaluation.

Results are shown in Figure 10. (1) *Training efficiency.* MambaTG scales efficiently with increasing history length and remains among the faster methods. At $\ell = 750$, it achieves up to $12.91\times$ speedup over DyG-Mamba while remaining competitive with CRAFT and JODIE. (2) *Inference efficiency.* Although inference time scales near-linearly with ℓ in theory, it remains nearly constant empirically because candidate evaluation dominates wall-clock time. MambaTG caches historical states and reuses them across candidates, avoiding repeated history encoding. At $\ell = 1000$, it achieves up to $104.54\times$ speedup over the compared baselines. (3) *Memory efficiency.* MambaTG exhibits approximately linear memory growth with history length and remains substantially more memory-efficient than several baselines. Although JODIE becomes more memory-efficient for longer histories due to its fixed-size node states, MambaTG maintains a competitive memory footprint across the evaluated sequence lengths. Overall, these results demonstrate that MambaTG can effectively scale to long source histories while maintaining low computational overhead, making it well suited to large-scale

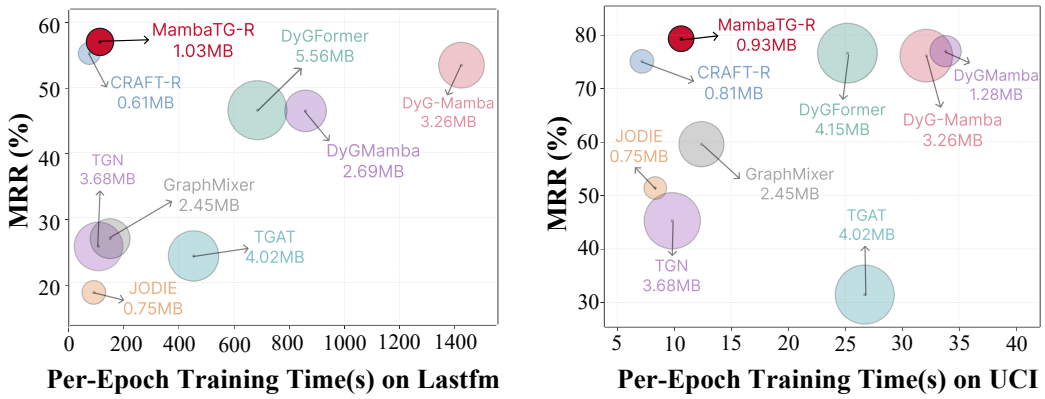


Figure 11: Performance–efficiency trade-off on (a) LastFM and (b) UCI. Test MRR is plotted against per-epoch training time, with bubble size representing parameter footprint.

recommendation scenarios where a user's extensive interaction history must be evaluated against many candidate items.

J.2 Performance–Efficiency Trade-off

We additionally evaluate the performance–efficiency trade-off on LastFM and UCI. Figure 11 plots test MRR against per-epoch training time, with bubble size representing parameter footprint.

J.3 Additional results on other datasets

We further evaluate MambaTG on three datasets from the Temporal Graph Benchmark (TGB) (Huang et al., 2023): `tgbl-comment`, `tgbl-coin`, and `tgbl-flight`. We compare against DyGFormer (Yu et al., 2023), SGNN-HN (Pan et al., 2020), and CRAFT (Yi et al., 2025a). The DyGFormer results are taken from Yu (2023), whereas the SGNN-HN and CRAFT results are obtained from our experiments. The two prior Mamba-based baselines, DyGMamba and DyG-Mamba, are not reported because they exceed the available computational budget on all three datasets. As shown in Table 9, MambaTG achieves superior performance across all three datasets. These results further demonstrate the effectiveness and generalization of MambaTG across diverse temporal graph datasets.

Table 9: Future link prediction results on three additional TGB datasets. Results are reported as MRR (%), averaged over three random seeds. The **best** and second-best results in each column are highlighted. Missing entries indicate experiments that exceeded computational budget.

Unseen-edge-dominant		**Repeated-edge-dominant**		
Model	**tgbl-comment**	**Model**	**tgbl-coin**	**tgbl-flight**
DyGFormer	67.03 ± 0.14	DyGFormer	75.17 ± 0.38	–
SGNN-HN	40.38 ± 0.18	SGNN-HN	66.25 ± 0.41	81.86 ± 0.07
CRAFT	75.69 ± 0.75	CRAFT-R	74.43 ± 0.30	86.42 ± 0.05
MambaTG	75.93 ± 0.21	**MambaTG-R**	75.74 ± 0.28	86.53 ± 0.08

K Discussion

K.1 Limitations and Future Work

Although this work focuses on future link prediction, we have not examined whether the two signals modeled by MambaTG generalize to other temporal-graph prediction tasks. Extending MambaTG

to tasks such as temporal edge classification and temporal anomaly detection, while investigating the roles of historical-sequence evolution and candidate–history compatibility in these settings, is a promising direction for future work.