Future Link Prediction Without Memory or Aggregation

Lu Yi[1], Runlin Lei[1], Fengran Mo[2], Yanping Zheng[1], Zhewei Wei[1]
[1]Renmin University of China, [2]Université de Montréal
{yilu, runlin_lei, zhengyanping, zhewei}@ruc.edu.cn
fengran.mo@umontreal.ca

Yuhang Ye[3]
[3]Huawei Poisson Lab, Huawei Technology Ltd.
yeyuhang@huawei.com

Abstract

Future link prediction on temporal graphs is a fundamental task with wide applicability in real-world dynamic systems. These scenarios often involve both recurring (seen) and novel (unseen) interactions, requiring models to generalize effectively across both types of edges. However, existing methods typically rely on complex memory and aggregation modules, yet struggle to handle unseen edges. In this paper, we revisit the architecture of existing temporal graph models and identify two essential but overlooked modeling requirements for future link prediction: representing nodes with unique identifiers and performing target-aware matching between source and destination nodes. To this end, we propose Cross-Attention based Future Link Predictor on Temporal Graphs (CRAFT), a simple yet effective architecture that discards memory and aggregation modules and instead builds on two components: learnable node embeddings and cross-attention between the destination and the source's recent interactions. This design provides strong expressive power and enables target-aware modeling of the compatibility between candidate destinations and the source's interaction patterns. Extensive experiments on diverse datasets demonstrate that CRAFT consistently achieves superior performance with high efficiency, making it well-suited for large-scale real-world applications.

1 Introduction

Dynamic systems are typically formulated as temporal graphs, where nodes and edges represent entities and their interactions, and each edge is annotated with a timestamp [9, 14, 43]. The prediction of future interactions between entities is defined as a future link prediction task in existing studies [8], which is crucial and widely applied in real-world systems [2, 24], such as social networks [7, 10] and collaboration platforms [29]. Future interactions can be categorized into repeated interactions with historical neighbors and new interactions with previously unconnected nodes [40]. For instance, in social networks, individuals often maintain regular contact with close friends while occasionally reaching out to other acquaintances. Similarly, in academic collaboration networks, researchers may frequently publish with long-term collaborators while also initiating projects with new coauthors. Thus, to precisely predict both seen and unseen interactions is important to enhance user experience in different dynamic systems and improve their utility. This dual nature poses a nontrivial challenge for future link prediction, demanding models that generalize across both seen and unseen edges.

Existing temporal graph learning methods perform future link predictions by adopting one or both of the memory and aggregation modules [26, 39], as illustrated in Figure 1. The memory module

Preprint. Under review.

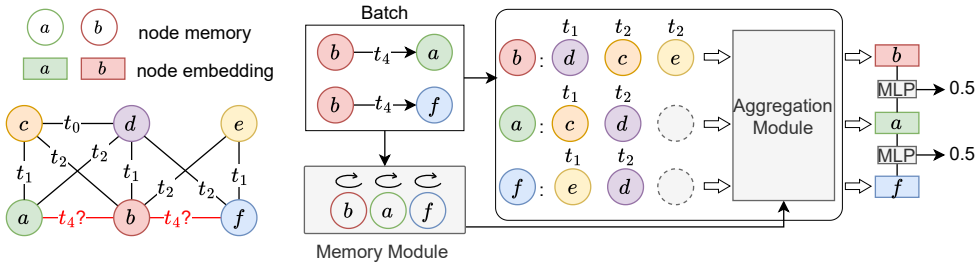

Figure 1: The general model architecture of existing temporal graph learning methods.

represents node interaction history in a compressed memory state, while the aggregation module generates node embeddings by aggregating historical neighbors. For example, TGN [26] uses a GRU [5] to update node memory and a graph attention layer for aggregation, whereas DyGFormer [42] omits the memory module and directly applies self-attention over historical neighbors. Although equipped with a sophisticated architecture, these methods suffer substantial performance degradation on recent benchmarks like TGB-Seq [40], which emphasize unseen edge prediction in real-world scenarios, thus limiting their practical utility. This consistent underperformance may stem from a deeper architectural misalignment with the core demands of future link prediction, prompting a critical question: *Are memory and aggregation modules truly effective for future link prediction, especially given the need to handle both seen and unseen edge scenarios?*

To investigate this question, we identify two essential model capabilities that are missing from the existing architecture but are key to addressing the challenges of future link prediction. **First**, the memory and aggregation modules have limited expressive power due to the absence of *unique node identifiers*. They typically operate over historical interaction data, which often lack distinctive node or edge features [12, 16, 23, 40]. Therefore, models often struggle to distinguish between nodes using only their edge timestamps [40]. This limitation becomes particularly severe in datasets with a low proportion of seen edges, where the model must rely on a more expressive representation of historical interactions to generalize to unseen edges. **Second**, both modules are generally applied to individual nodes in isolation, without modeling how well the candidate destination (i.e., the *target*) aligns with the source's recent interaction patterns, a capability we refer to as *target-aware matching*. Although some recent methods [42, 34, 19, 4] leverage correlation encodings to capture the relation between the source and destination, they primarily focus on the connections between their neighborhoods, rather than directly assessing semantic compatibility between the destination and the source's behavioral history. Therefore, the framework designs of these methods are not well aligned with the core demands of future link prediction – to evaluate how well a destination fits the source context.

In this paper, we introduce a minimalistic architecture that centers on these two essential modeling requirements: representing nodes with unique node identifiers and performing target-aware matching, while discarding the commonly used memory and aggregation modules. Specifically, we propose **CRoss-Attention based Future Link Predictor on Temporal Graphs (CRAFT)**, which employs learnable embeddings to serve as node identifiers and a cross-attention mechanism between each candidate destination and the source's recent neighbors to realize target-aware matching. By integrating these two components, our CRAFT provides strong expressive power and enables direct evaluation of the destination's compatibility with the source's interaction patterns, making it effective for both seen and unseen edge prediction. Overall, our main contributions are as follows:

- We propose CRAFT, a simple yet effective architecture for temporal graph learning. Unlike existing models that rely on memory and aggregation modules, CRAFT is built on two essential components tailored for future link prediction: unique node identifiers and target-aware matching.
- With strong expressiveness and destination-conditioned context modeling, CRAFT effectively addresses both seen and unseen edge prediction, addressing the limitations observed in prior work.
- Extensive experiments on 17 datasets, including the TGB and TGB-Seq benchmarks, demonstrate that CRAFT consistently delivers superior performance across both types of prediction tasks while maintaining high efficiency, making it well-suited for large-scale real-world applications.

2 Related Work

Temporal graph learning methods for future link prediction. Temporal graph learning focuses on modeling the evolving patterns of node interactions over time, with future link prediction serving as a core task [28]. TGN [26] initially propose the memory-aggregation architecture for temporal graph learning and categorize earlier methods within this architecture, including JODIE [16], DyRep [31], and TGAT [38]. Among them, JODIE, DyRep, and TGN adopt RNNs [36] in the memory module to update node memory when new edges arrive, and utilize time projection, identity function, and temporal graph attention to aggregate historical information, respectively. TGAT and many subseqent methods generate time-dependent node representations directly by aggregating historical neighbors without the memory module. These methods leverage various techniques to model dynamic interactions, including the MLP-Mixer [30] in GraphMixer [6], attention mechanisms in TGAT [38], DyGFormer [42], TCL [33], and SimpleDyG [37], as well as temporal point processes (e.g., Hawkes processes [11, 20]) in TREND [35] and LDG [15], etc.

Limitations of existing methods on predicting unseen edges. Poursafaei et al. [23] are the first to observe the imbalance between seen and unseen edges in existing datasets. They propose a simple heuristic method, EdgeBank, which memorizes historical edges without any learnable components, yet achieves surprisingly strong performance on many benchmarks due to their high ratio of seen edges. To make evaluation more challenging, Huang et al. [12] introduce the TGB benchmark, featuring large-scale datasets and rigorous multiple negative sampling strategy, while some datasets still exhibit excessive seen edges. To fill this gap, Yi et al. [40] propose the TGB-Seq benchmark, designed to evaluate performance on datasets with minimal seen edges. Their findings further revealed that most existing methods struggle to generalize to unseen edge. This persistent performance gap between seen and unseen edges motivates us to rethink the architecture of temporal graph learning.

3 Problem Formulation

Temporal Graphs. We define a temporal graph as $G = (V, E)$, where V is the set of nodes, and E is a sequence of edges ordered by non-decreasing timestamps, i.e., $E = \{e_1, e_2, \cdots, e_m\}$, with each edge $e_i = (s_i, d_i, t_i)$ denoting an interaction from the source node s_i to the destination node d_i at time t_i. Throughout this paper, we use the terms *target* and *destination* interchangeably.

Future Link Prediction. Future link prediction aims to estimate the likelihood of an interaction between a source node s and a destination node d at a future time t, given all historical edges before t in the graph G. We follow prior works [12, 40] to frame this task as a ranking problem, as real-world applications often require identifying the most likely destination from a large pool of candidates. Accordingly, the objective is to rank the positive destination node d higher than sampled negative candidates, conditioned on the source node s and the prediction time t.

4 Proposed Methods

In this section, we first introduce how our CRAFT's architecture achieves two essential modeling requirements for future link prediction (Section 4.1): 1) representing nodes with unique identifiers to enhance the model expressive power and 2) performing target-aware matching to enable direct assessment of the temporal compatibility between the source and destination. Then, we details on the necessity of these two components in Section 4.2. In Section 4.3, we analyze the time complexity of CRAFT to demonstrate its superior efficiency and compare it with existing methods.

4.1 Model Architecture

The model architecture of CRAFT is shown in Figure 2. Given the source s and a set of candidate destinations $D = \{d_1, d_2, \cdots, d_j\}$ at time t, we first extract the k recent neighbors of s before t, and look up the node embeddings for these neighbors and the destinations. For the i-th neighbor n_i of s, we add positional encoding to the node embedding to capture the temporal order of these neighbors. Then we adopt a cross-attention module between the candidate destinations and the source's neighbors to compute the edge representations. To capture the temporal status of the destinations, we encode the elapsed time since last activity of these destinations and concatenate it with the edge representations. Finally, we use an MLP to predict the edge likelihoods. We detail the process as follows.

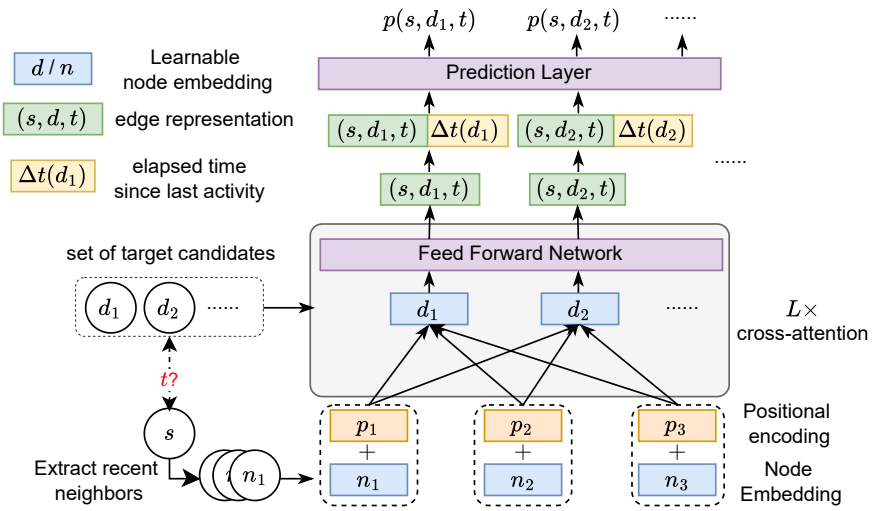

Figure 2: Model architecture of our proposed method, CRAFT: representing nodes with learnable embeddings and performing target-aware matching using cross-attention.

Learnable node embedding as unique identifiers. To represent nodes with unique identifiers, we introduce learnable node embeddings that are trained jointly with the model. Such trainable embeddings can encode the contextual information from interaction history of nodes, thereby serving as the unique latent identifiers. Specifically, we denote the embedding of node n as $\mathbf{e}(n) \in \mathbb{R}^d$. For the i-th neighbor n_i of the source s, we add positional encoding $\mathbf{p}(i)$ to the node embedding $\mathbf{e}(n_i)$ to represent the temporal order of these neighbors. The updated embedding is given by:

$$\bar{\mathbf{e}}(n_i) = \mathbf{e}(n_i) + \mathbf{p}(i), \tag{1}$$

where $\mathbf{p}(i)$ is also trainable. The positional encoding enable the model to capture the short-term and long-term interests of the source, which affect the future interactions crucially. Then, we obtain the embedding sequence of the source's neighbors: $\mathbf{S} = [\bar{\mathbf{e}}(n_1), \bar{\mathbf{e}}(n_2), \cdots, \bar{\mathbf{e}}(n_k)]$ and the embedding sequence of the candidate destinations: $\mathbf{D} = [\mathbf{e}(d_1), \mathbf{e}(d_2), \cdots, \mathbf{e}(d_j)]$. Note that these destination nodes are independent of each other, and we concatenate them only for parallel computation.

Target-aware matching via cross-attention. Considering the inherent requirement of future link prediction to assess the destination's compatibility with the source's historical interaction patterns, we implement target-aware matching through cross-attention between the candidate destinations and the source's neighbors. This mechanism allows each candidate destination to directly attend to the source's historical interactions, thus better evaluating the alignment between the destination and the source's temporal preferences. We define the $\mathrm{Attention}$ and FFN function [32] as follows:

$$\mathrm{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{softmax}(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}})\mathbf{V}, \tag{2}$$

$$\mathrm{FFN}(\mathbf{H}) = \mathrm{GELU}(\mathbf{H}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2. \tag{3}$$

Our cross-attention module is stacked with multiple layers, and each layer includes a multi-head attention network (MHA) and a feed-forward network (FFN). The ℓ-th layer calculation is given by:

$$\mathbf{Z}_i^{(\ell)} = \mathrm{Attention}(\mathbf{H}^{(\ell-1)}\mathbf{W}_{\ell,i}^Q, \mathbf{S}\mathbf{W}_{\ell,i}^K, \mathbf{S}\mathbf{W}_{\ell,i}^V), \tag{4}$$

$$\mathbf{Z}^{(\ell)} = \mathrm{MHA}(\mathbf{H}^{(\ell-1)}, \mathbf{S}) + \mathbf{H}^{(\ell-1)} = \mathrm{concat}\left(\mathbf{Z}_0^{(\ell-1)}, \mathbf{Z}_1^{(\ell-1)}, \cdots, \mathbf{Z}_{h-1}^{(\ell-1)}\right)\mathbf{W}_o + \mathbf{H}^{(\ell-1)}, \tag{5}$$

$$\mathbf{H}^{(\ell)} = \mathrm{FFN}(\mathbf{Z}^{(\ell)}) + \mathbf{Z}^{(\ell)}. \tag{6}$$

where $\mathbf{W}_{\ell,i}^Q \in \mathbb{R}^{d\times d_h}$, $\mathbf{W}_{\ell,i}^K \in \mathbb{R}^{d\times d_h}$, and $\mathbf{W}_{\ell,i}^V \in \mathbb{R}^{d\times d_h}$ are the projection weight for the i-th head in the ℓ-th layer, where $d_h = d/h$ and h is the number of heads. $\mathbf{H}^{(\ell)}$ denotes the hidden state of the ℓ-th layer, and $\mathbf{H}^{(0)} = \mathbf{D}$. Each layer takes the hidden state of the previous layer as the query, and the embedding sequence of the source's neighbors as the key and value. The output of the last layer $\mathbf{H}^{(L)}$ will be the edge representations of the source and candidate destinations.

Prediction. After the cross-attention module, we encode the elapsed time since last activity of the destination nodes, in order to capture the temporal status of the destinations. The elapsed time is projected to a time-context vector $\mathbb{R}^d$ by a linear layer and then concatenated with the edge representation. Finally, we feed them to an MLP to predict the edge likelihood between the source and the destination. Specifically, the predicted score of the i-th destination node d_i is computed as:

$$\hat{y}_i = \text{MLP}\left(\text{concat}\left(\mathbf{H}^{(L)}(i), \text{TimeProjection}\left(\Delta t(d_i)\right)\right)\right), \tag{7}$$

where $\Delta t(d_i) = t - t^-(d_i)$ is the elapsed time between the prediction time t and the last activity time of d_i, $t^-(d_i)$, and $\mathbf{H}^{(L)}(i)$ is the edge presentation of (s, d_i, t) generated by cross-attention module. Furthermore, for the scenarios involving excessive seen edges, we encode the repeat time of the candidate edge to capture the inherent repeat patterns in the temporal graph. Similar to the elapsed time encoding, we will first project the edge repeat time by a linear layer and then concatenate it with the edge representation before the prediction MLP.

Loss function. We adopt the Bayesian Personalized Ranking loss [25], which is widely used and well-suited for ranking tasks [17]. Given a source node s interacting with a positive destination node d_i and a negative destination node d_j at time t, the loss is defined as:$\mathcal{L}(s, t) = -\log \sigma(\hat{y}_i - \hat{y}_j)$, where $\hat{y}_i$ and $\hat{y}_j$ denote the predicted scores for the positive and negative samples, respectively.

4.2 On the Necessity of Learnable Embeddings and Cross-Attention

4.2.1 Learnable Node Embeddings Enable Powerful Expressiveness

Most existing methods generate node representations from interaction data, such as edge timestamps and node or edge features. However, in many real-world temporal graphs, including well-established benchmarks [12, 40, 16, 23], these features are often missing or extremely sparse. This is largely because node and edge features are often hard to collect in practice and are not easily aligned across heterogeneous node types [40]. Therefore, many models rely purely on edge timestamps, which significantly limits their ability to distinguish between nodes. In the example shown in Figure 1, existing methods are unable to distinguish nodes a and f because they have identical interaction timestamps and lack unique identifiers.

Recent approaches [4, 19, 34, 42] attempt to mitigate this issue by introducing correlation encodings to capture the relation between the k-hop neighborhoods of the source and destination nodes. However, the model's perception is inherently constrained by the choice of k. For example, DyGFormer uses neighbor co-occurrence frequency to model correlations between nodes. But in the example shown in Figure 1, it cannot distinguish the candidate edge (b, a) from (b, f), since both pairs share two common neighbors. It fails to capture that only the co-neighbors of b and a have prior interactions (c, d, t_0), which is a critical signal of a tightly-knit group structure and essential for making accurate predictions. While increasing k may offer broader context, it leads to exponential neighborhood expansion, introducing excessive noise [4] or resulting in memory overhead [40].

These limitations emphasize the necessity of representing nodes with unique identifiers. Similar insights have been observed in the study of the expressiveness of static Graph Neural Networks (GNNs) [3, 18], where the lack of distinctive node embeddings hampers model performance. Notably, Abboud et al. [1] demonstrates that GNNs with randomly initialized node embeddings, which serve as implicit identifiers, can outperform models that rely solely on structural information. Therefore, we introduce learnable node embeddings that serve as latent identifiers and encode contextual information, enabling the model to distinguish between nodes even in the absence of explicit features.

4.2.2 Cross-attention Mechanism Enables Target-aware Matching

Most existing methods [26, 38, 6, 16, 31] do not explicitly model the relationship between source and destination nodes. As illustrated in Figure 1, they typically follow a common paradigm: first, the memory and aggregation modules are used to encode interaction histories into individual node representations; then, an MLP predicts the edge likelihood based on these node representations. This approach models each node's local dynamics independently and neglects the direct matching between node pairs. An exception is DyGFormer [42], which applies self-attention over the combined historical neighbors of both the source and destination nodes, aiming to learn temporal dependencies within and cross the two sets together. While it introduces a degree of cross-node awareness, it focuses

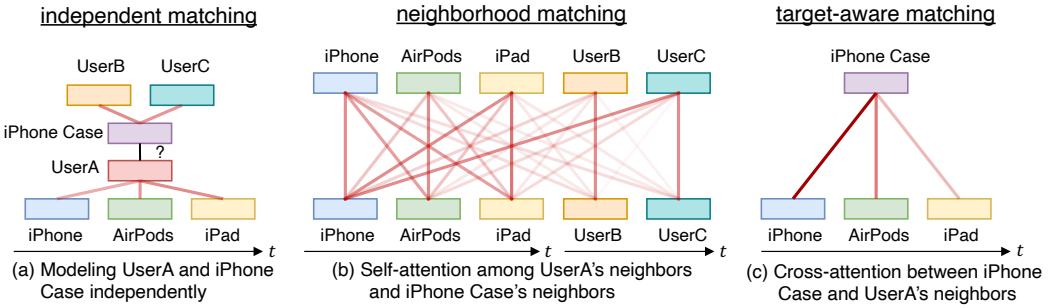

Figure 3: Illustration of different source-destination matching schemes for predicting whether *UserA* is likely to purchase *iPhone Case* at time t. Before time t, UserA has purchased *iPhone*, *AirPods*, and *iPad* in chronological order, while *UserB* and *UserC* have previously purchased iPhone Case.

on the relation between the source and destination neighborhoods and is not explicitly conditioned on the candidate destination. As a result, when the source and destination neighborhoods lack semantic overlap or are heterogeneous in type, the attention signal becomes diluted or misaligned, leading to weak modeling of compatibility between the destination and the source context.

To better align with the objective of future link prediction, which involves assessing how well a target node fits the source context, we implement target-aware matching through a simple yet effective design: cross-attention between the destination and the source's recent neighbors. This allows each candidate destination to directly attend to the source node's historical interactions and enables the model to capture fine-grained behavioral patterns as well as semantic alignment across entities. Consider the example in Figure 3, which demonstrates how different matching schemes perform in predicting whether *UserA* is likely to purchase *iPhone Case* at time t. Subfigure (a) illustrates the independent matching mechanism adopted by most existing methods, where the source (UserA) and the destination (iPhone Case) are modeled independently with their respective neighbors. This hinders the model from recognizing the strong connection between iPhone and iPhone case. In (b), the neighborhood matching mechanism adopted by DyGFormer includes both source and destination neighborhoods, but treats all nodes uniformly. If UserB or UserC has purchased many unrelated items besides iPhone Case, the semantic connection between iPhone and iPhone Case may be overwhelmed. In contrast, our target-aware matching mechanism, shown in (c), enables the candidate iPhone Case to attend directly to iPhone, effectively capturing their semantic relationship. We empirically compare these matching strategies in Section 5.2, demonstrating the advantages of our cross-attention design.

4.3 Time Complexity Analysis

We analyze the time complexity of performing future link predictions for a batch of source nodes and their corresponding candidate destinations. Let B denote the batch size, k the number of historical neighbors considered per node, q the number of candidate destinations per source, F the embedding or feature dimension, and d_{avg} the average node degree in the temporal graph. We compare CRAFT with four state-of-the-art methods: TGAT [38], TGN [26], GraphMixer [6], and DyGFormer [42]. For TGN and TGAT, we follow common practice and consider their standard configurations: TGN with one graph attention layer and TGAT with two. The total time complexity generally consists of two dominant parts: 1) extracting historical neighbors and 2) computing edge likelihoods.

For the extraction of historical neighbors, we consider the common strategy of retrieving the k most recent neighbors [39, 26], following the widely adopted implementation in DyGLib [42]. Alternative neighbor extraction implementation are discussed in the appendix. The dominant cost arises from locating the most recent neighbor among all existing neighbors via binary search, taking $O(\log d_{\text{avg}})$ per query. Since existing methods typically gather neighbors for both source and destination nodes,

Table 1: The big-O notation of the time complexity of different temporal graph learning methods, where only the dominant term is retained for clarity.

Methods	**Extract historical neighbors**	**Compute edge likelihoods**
TGAT [38]	$B(1+q)(1+k)\log d_{\text{avg}}$	$B(1+q)k^2F^2$
TGN [26]	$B(1+q)\log d_{\text{avg}}$	$B(1+q)kF^2$
GraphMixer [6]	$B(1+q)\log d_{\text{avg}}$	$B(1+q)kF^2$
DyGFormer [42]	$B(1+q)\log d_{\text{avg}}$	$B(1+q)(kF^2+k^2F)$
CRAFT	$B\log d_{\text{avg}}$	BqF^2+BkF^2+BqkF

Table 2: MRR scores (%) of CRAFT and baselines on unseen-dominant datasets. The first, second, and third place rankings are highlighted, accordingly.

Datasets	ML-20M	Taobao	Yelp	GoogleLocal	Flickr	YouTube	WikiLink	tgbl-review	tgbl-comment
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	41.43±0.15	-
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	40.06±0.59	28.90±3.30
TGAT	10.47±0.20	-	-	19.78±0.24	23.53±3.35	43.56±2.53	-	19.64±0.23	56.20±2.11
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	37.48±0.23	37.90±2.10
CAWN	12.31±0.02	-	25.71±0.09	18.26±0.02	48.69±6.08	47.55±1.08	-	19.30±0.10	-
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	36.89±1.50	76.17±0.17
DyGFormer	-	-	21.68±0.20	18.39±0.02	49.58±2.87	46.08±3.44	-	22.39±1.52	67.03±0.14
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	35.62±0.53	55.24±0.22
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	41.77±0.06	91.72±0.59
Rel.Imprv.	49.69%	17.25%	4.02%	15.19%	25.74%	0.08%	19.92%	0.82%	20.41%
Abs.Imprv.	11.92	10.40	2.81	8.22	12.76	0.05	12.54	0.34	15.55

Table 3: MRR scores (%) of CRAFT-R and baselines on seen-dominant datasets. The first, second, and third place rankings are highlighted accordingly.

Datasets	wikipedia	reddit	mooc	lastfm	uci	Flights	tgbl-coin	tgbl-flights
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	-	-
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	45.20±4.60	55.60±1.40
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	60.92±0.57	-
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	58.60±3.70	70.50±2.00
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	-	-
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	75.57±0.27	-
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	75.17±0.38	-
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	78.46±0.40	89.03±0.03
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	88.47±0.25	91.39±0.03
Rel.Imprv.	-0.50%	0.71%	47.68%	18.73%	-1.96%	-1.15%	17.07%	29.63%
Abs.Imprv.	-0.44	0.63	20.12	8.71	-1.50	-0.97	12.90	20.89

this step takes $O(B(1+q)\log d_{\text{avg}})$. TGAT, which aggregates two-hop neighborhoods via two attention layers, incurs $O(B(1+q)(1+k)\log d_{\text{avg}})$. In contrast, CRAFT only considers neighbors of the source, reducing the complexity to $O(B\log d_{\text{avg}})$. This reduction is particularly beneficial in real-world scenarios, where it is common to evaluate many candidate destinations per source, making efficient neighbor retrieval critical for scalable inference.

To compute edge likelihoods, most existing methods compute node representations by aggregating each node's historical neighbors, followed by an MLP for prediction. The aggregation step dominates computation. TGN, TGAT, GraphMixer, and DyGFormer employ one graph attention layer, two graph attention layers, an MLP-Mixer, and a Transformer encoder, respectively, each incurring various computational complexities per node: $O(kF^2)$, $O(k^2F^2)$, $O(kF^2)$, and $O(kF^2+k^2F)$. In contrast, CRAFT computes edge likelihoods directly using cross-attention between destinations and the source's neighbors, requiring $O(BqF^2+BkF^2)$ for projections and $O(BqkF)$ for attention per batch. This approach eliminates the need for neighbor feature projections for each node, significantly improving efficiency, especially when q or k is large.

Table 1 summarizes the results, showing that CRAFT offers superior efficiency in both neighbor extraction and edge likelihood computation. These findings highlight the advantage of CRAFT's simple architecture, making it well-suited for large-scale dynamic systems in practical applications.

5 Experiments

In this section, we investigate the effectiveness and efficiency of CRAFT on future link prediction. To fully evaluate the performance of CRAFT on predicting seen and unseen edges, we conduct experiments on 17 datasets, including the TGB-Seq benchmark [40] (ML-20M, Taobao, Yelp, GoogleLocal, Flickr, YouTube, WikiLink), the TGB benchmark [12] (tgbl-review, tgbl-comment, tgbl-coin, tgbl-flights), and six commonly used datasets (wikipedia, reddit, mooc, lastfm, uci, Flights). We compare CRAFT with seven state-of-the-art continuous-time temporal graph models, including JODIE [16], DyRep [31], TGAT [38], TGN [26], CAWN [34], GraphMixer [6], and DyGFormer [42]. The details of datasets and baselines are provided in the appendix.

Experimental Settings. We follow the evaluation protocol of TGB-Seq and TGB benchmarks for all datasets. We use MRR as the evaluation metric, follow the original split of datasets, or chronologically split the datasets into training, validation, and test sets with 75%, 15%, and 15% of the edges, respectively. We adopt the negative samples from the benchmarks or randomly sample 100 negatives from all potential destinations. Importantly, we perform collision checks to ensure that no

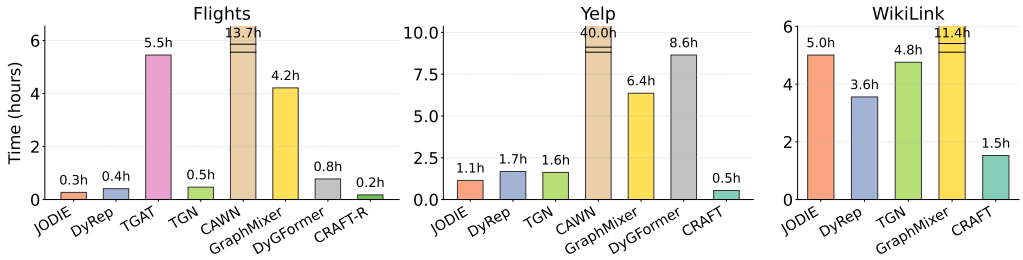

Figure 4: Test set inference time of CRAFT and baselines on Flights, Yelp and WikiLink.

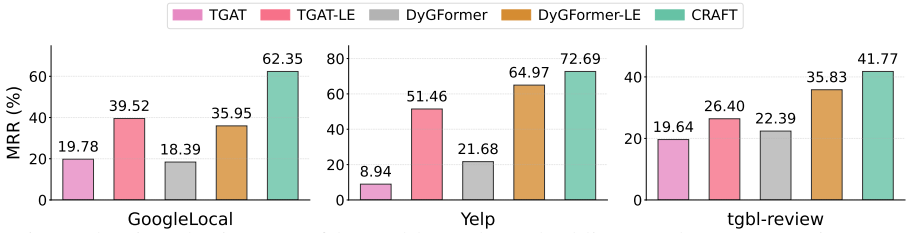

Figure 5: Evaluating the impact of learnable node embeddings and cross-attention: CRAFT vs. TGAT/DyGFormer with and without learnable embeddings (LE).

positive edges are sampled as negative edges for small datasets, which may lead to different results compared to prior works. We follow the configurations in [40, 12, 42] to use the same batch size (200 or 400, accordingly) and learning rate (1e-4) across all methods. The baseline implementation is based on the DyGLib library [42]. For fair comparison, the average results of 3 runs are reported. Other implementation details are provided in the appendix.

5.1 Effectiveness and Efficiency of CRAFT on Future Link Prediction

Experimental Setup. We categorize the datasets into two groups based on their seen edge ratio. For datasets with a high ratio of seen edges (referred to as *seen-dominant datasets*), we apply repeat time encoding, and denote the variant as CRAFT-R. For datasets with minimal seen edges (referred to as *unseen-dominant datasets*), we use the default CRAFT model without repeat time encoding. The results for all baselines on the TGB-Seq and TGB datasets are directly taken from previous studies [40, 41]. Missing entries indicate cases where the model failed due to either out-of-time (unable to complete a single training epoch within 24 hours) or out-of-memory on a 32GB GPU.

Effectiveness Evaluation. Table 2 and Table 3 present the MRR scores on unseen-dominant and seen-dominant datasets, respectively, with *Rel.Imprv.* and *Abs.Imprv.* indicating the relative and absolute improvements over the second-best baseline. Overall, CRAFT achieves the best results on a majority of benchmarks, with over 10% relative improvements on 10 out of 17 datasets compared to the second-best baseline. These results confirm the effectiveness of our architecture and highlight the value of its two key components: unique node identifiers and target-aware matching, which is opposed to the traditional memory and aggregation modules in the baselines. Specifically, CRAFT exhibits superior performance on unseen-dominant datasets, demonstrating that it successfully addresses a key limitation of prior models, which often fail on unseen edge prediction. For seen-dominant datasets, CRAFT-R still performs best on most benchmarks, with only minor underperformance (less than 2%) compared to DyGFormer on three datasets. The reason may be attributed to DyGFormer's use of additional heuristics, such as neighbor co-occurrence frequency, which could provide additional gains in these datasets. However, such design increases computational complexity and lead to out-of-time or out-of-memory failures on large-scale datasets such as WikiLink and tgbl-flights. In contrast, CRAFT maintains high performance with a simpler and more generalizable architecture.

Efficiency Evaluation. We evaluate the efficiency of CRAFT by measuring both inference and training times on three representative datasets of increasing scale: Flights (1M edges), Yelp (19M edges), and WikiLink (34M edges). Figure 4 presents the inference time for CRAFT and baseline methods, showing that CRAFT completes prediction on all test edges (each with 100 negative samples) significantly faster than all baselines. Notably, on the larger datasets Yelp and WikiLink, CRAFT requires less than half the time of the most efficient baseline, JODIE. This highlights CRAFT's practical suitability for real-world deployment, where temporal graphs are large and fast inference is essential. Training time comparisons are included in the appendix.

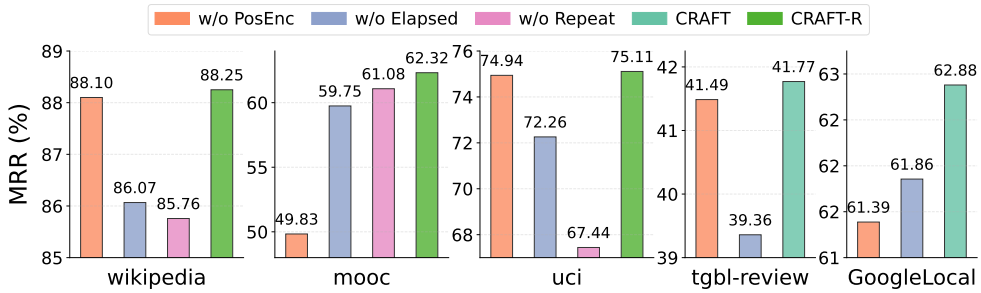

Figure 6: Evaluating the impact of positional encoding, elapsed time encoding, repeat time encoding.

5.2 Effect of Learnable Node Embeddings and Cross-attention Mechanism

Experimental Setup. To verify that unique node identifiers and target-aware matching are critical for future link prediction, we incorporate our designed learnable node embeddings into existing methods, to explore whether it can results in performance improvements. We choose TGAT and DyGFormer as representative methods because they also rely on attention mechanisms. The only difference is the matching mechanisms, where TGAT adopts independent matching, while DyGFormer adopts neighborhood matching. After replacing the original node features with learnable node embeddings, we denotes the modified variants as *TGAT-LE* and *DyGFormer-LE*. For fair comparison, we also use learnable positional encodings for the source and destination neighborhoods in DyGFormer-LE, which is consistent with CRAFT, and we remove the neighbor co-occurrence frequency encoding for DyGFormer-LE. Detailed implementations are provided in the appendix. We evaluate all methods on three large and challenging datasets with minimal seen edges, GoogleLocal, Yelp, and tgbl-review.

Results. As shown in Figure 5, comparing each method with its LE variant, we observe significant performance improvements, particularly for TGAT-LE, which achieves up to fivefold gains on Yelp. These results show that learnable node embeddings improve the expressive power of these models and are essential for generalizing to unseen edges in real-world applications. Despite these improvements, both TGAT-LE and DyGFormer-LE still underperform our CRAFT. The performance gap indicates again the importance of target-aware matching, which enables the model to directly assess whether the destination node is compatible with the source context. These findings confirm that the two crucial components in our CRAFT, learnable node embeddings and cross-attention-based target-aware matching, are essential for future link prediction.

5.3 Ablation Study: Effect of Positional, Elapsed Time, and Repeat Time Encoding

Experimental Setup. We conduct ablation studies to evaluate the impact of three encoding components in CRAFT: positional encoding, elapsed time encoding, and repeat time encoding. For each study, we remove one component while keeping the others intact, and denote the resulting variants as *w/o PosEnc*, *w/o Elapsed*, and *w/o Repeat*, respectively. We evaluate these variants on five representative datasets: wikipedia, mooc, uci, tgbl-review, and GoogleLocal.

Results. We find that all three encodings can positively impact the performance of CRAFT. Among them, the elapsed time encoding contributes significantly across all datasets, as it effectively captures the temporal status of each destination node. Positional encoding is especially critical on mooc and GoogleLocal, demonstrating its ability to capture fine-grained temporal order and enable the model to effectively learn from sequential dynamics in these domains. Repeat time encoding plays a prominent role on wikipedia and uci, which exhibit a high proportion of seen edges. While mooc also contains many seen edges, the gain from repeat time encoding is relatively small, possibly because learnable node embeddings and target-aware matching already capture the relationships between the source and destination in this case. Overall, these results confirm the necessity of all three encodings in enabling CRAFT to handle a broad range of temporal dynamics effectively.

6 Conclusion

In this paper, we revisit the architectural foundations of temporal graph learning and identify two essential modeling requirements for future link prediction: representing nodes with unique identifiers and performing target-aware matching between the source and destination nodes, which are over-

looked in existing methods. Motivated by these observations, we proposed CRAFT, a simple yet effective framework that replaces memory and aggregation modules with learnable node embeddings and a cross-attention mechanism. This design equips CRAFT with strong expressive power and the capacity to explicitly capture destination-conditioned relevance, allowing it to generalize effectively to both seen and unseen edge prediction. Extensive evaluations demonstrate the superiority of CRAFT, establishing it as a practical and scalable solution for real-world future link prediction tasks.

References

[1] Ralph Abboud, Ismail Ilkan Ceylan, Martin Grohe, and Thomas Lukasiewicz. The surprising power of graph neural networks with random node initialization. *arXiv preprint arXiv:2010.01179*, 2020.

[2] Ting Bai, Youjie Zhang, Bin Wu, and Jian-Yun Nie. Temporal graph neural networks for social recommendation. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 898–903. IEEE, 2020.

[3] Maya Bechler-Speicher, Moshe Eliasof, Carola-Bibiane Schonlieb, Ran Gilad-Bachrach, and Amir Globerson. Towards invariance to node identifiers in graph neural networks. *arXiv preprint arXiv:2502.13660*, 2025.

[4] Maciej Besta, Afonso Claudino Catarino, Lukas Gianinazzi, Nils Blach, Piotr Nyczyk, Hubert Niewiadomski, and Torsten Hoefler. Hot: Higher-order dynamic graph representation learning with efficient transformers. In *Learning on Graphs Conference*, pages 15–1. PMLR, 2024.

[5] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[6] 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? In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*, 2023. URL `https://openreview.net/forum?id=ayPPc0SyLv1`.

[7] Nur Nasuha Daud, Siti Hafizah Ab Hamid, Muntadher Saadoon, Firdaus Sahran, and Nor Badrul Anuar. Applications of link prediction in social networks: A review. *Journal of Network and Computer Applications*, 166:102716, 2020.

[8] Aswathy Divakaran and Anuraj Mohan. Temporal link prediction: A survey. *New Generation Computing*, 38(1):213–258, 2020.

[9] Sanaz Hasanzadeh Fard. Machine learning on dynamic graphs: a survey on applications. *2023 IEEE Ninth Multimedia Big Data (BigMM)*, pages 32–39, 2023.

[10] Sogol Haghani and Mohammad Reza Keyvanpour. A systemic analysis of link prediction in social network. *Artificial Intelligence Review*, 52:1961–1995, 2019.

[11] Alan G. Hawkes. Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, 58:83–90, 1971.

[12] 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. *Advances in Neural Information Processing Systems*, 36, 2024.

[13] Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*, pages 197–206. IEEE, 2018.

[14] 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.

[15] Boris Knyazev, Carolyn Augusta, and Graham W Taylor. Learning temporal attention in dynamic graphs with bilinear interactions. *Plos one*, 16(3):e0247936, 2021.

[16] 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, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pages 1269–1278, 2019. doi: 10.1145/3292500.3330895. URL https://doi.org/10.1145/3292500.3330895.

[17] Dong Li, Ruoming Jin, and Bin Ren. Revisiting recommendation loss functions through contrastive learning (technical report). *arXiv preprint arXiv:2312.08520*, 2023.

[18] Andreas Loukas. What graph neural networks cannot learn: depth vs width. *arXiv preprint arXiv:1907.03199*, 2019.

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

[20] Hongyuan Mei and Jason M Eisner. The neural hawkes process: A neurally self-modulating multivariate point process. *Advances in neural information processing systems*, 30, 2017.

[21] Zhiqiang Pan, Fei Cai, Wanyu Chen, Honghui Chen, and M. de Rijke. Star graph neural networks for session-based recommendation. *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020. URL https://api.semanticscholar.org/CorpusID:221339954.

[22] Pietro Panzarasa, Tore Opsahl, and Kathleen M Carley. Patterns and dynamics of users' behavior and interaction: Network analysis of an online community. *Journal of the American Society for Information Science and Technology*, 60(5):911–932, 2009.

[23] Farimah Poursafaei, Shenyang Huang, Kellin Pelrine, and Reihaneh Rabbany. Towards better evaluation for dynamic link prediction. *Advances in Neural Information Processing Systems*, 35:32928–32941, 2022.

[24] Nitendra Rajput and Karamjit Singh. Temporal graph learning for financial world: Algorithms, scalability, explainability & fairness. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4818–4819, 2022.

[25] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*, 2012.

[26] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal graph networks for deep learning on dynamic graphs. In *ICML 2020 Workshop on Graph Representation Learning*, 2020.

[27] Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260, 2002.

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

[29] Yizhou Sun, Rick Barber, Manish Gupta, Charu C Aggarwal, and Jiawei Han. Co-author relationship prediction in heterogeneous bibliographic networks. In *2011 international conference on advances in social networks analysis and mining*, pages 121–128. IEEE, 2011.

[30] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in neural information processing systems*, 34: 24261–24272, 2021.

[31] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. Dyrep: Learning representations over dynamic graphs. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019. URL https://openreview.net/forum?id=HyePrhR5KX.

[32] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[33] 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 preprint arXiv:2105.07944*, 2021.

[34] Yanbang Wang, Yen-Yu Chang, Yunyu Liu, Jure Leskovec, and Pan Li. Inductive representation learning in temporal networks via causal anonymous walks. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021. URL `https://openreview.net/forum?id=KYPz4YsCPj`.

[35] Zhihao Wen and Yuan Fang. Trend: Temporal event and node dynamics for graph representation learning. In *Proceedings of the ACM Web Conference 2022*, pages 1159–1169, 2022.

[36] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. Recurrent recommender networks. In *Proceedings of the tenth ACM international conference on web search and data mining*, pages 495–503, 2017.

[37] Yuxia Wu, Yuan Fang, and Lizi Liao. On the feasibility of simple transformer for dynamic graph modeling. In *Proceedings of the ACM on Web Conference 2024*, pages 870–880, 2024.

[38] Da Xu, Chuanwei Ruan, Evren Körpeoglu, Sushant Kumar, and Kannan Achan. Inductive representation learning on temporal graphs. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020. URL `https://openreview.net/forum?id=rJeW1yHYwH`.

[39] Yuxin Yang, Hongkuan Zhou, Rajgopal Kannan, and Viktor Prasanna. Towards ideal temporal graph neural networks: Evaluations and conclusions after 10,000 gpu hours. *arXiv preprint arXiv:2412.20256*, 2024.

[40] 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. In *The Thirteenth International Conference on Learning Representations*.

[41] Le Yu. An empirical evaluation of temporal graph benchmark. *arXiv preprint arXiv:2307.12510*, 2023.

[42] 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: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL `http://papers.nips.cc/paper_files/paper/2023/hash/d611019afba70d547bd595e8a4158f55-Abstract-Conference.html`.

[43] Yanping Zheng, Lu Yi, and Zhewei Wei. A survey of dynamic graph neural networks. *arXiv preprint arXiv:2404.18211*, 2024.

Table 4: Datasets statistics: seven TGB-Seq datasets, four TGB datasets, and six commonly used datasets. This table is partially adopted from TGB-Seq [40].

Dataset	Nodes (users/items)	Edges	Timestamps	Repeat ratio(%)	Bipartite	Domain
ML-20M	100,785/9,646	14,494,325	9,993,250	0	✓	Movie rating
Taobao	760,617/863,016	18,853,792	139,171	16.58	✓	E-commerce interaction
Yelp	1,338,688/405,081	19,760,293	14,646,734	25.18	✓	Business review
GoogleLocal	206,244/267,336	1,913,967	1,771,060	0	✓	Business review
Flickr	233,836	7,223,559	134	0	×	Who-To-Follow
YouTube	402,422	3,288,028	203	0	×	Who-To-Follow
WikiLink	1,361,972	34,163,774	2,198	0	×	Web link
tgbl-review	352,636/298,590	4,873,540	6,865	0.19	✓	E-commerce review
tgbl-coin	638,486	22,809,486	1,295,720	82.93	×	Transaction
tgbl-comment	994,790	44,314,507	30,998,030	19.81	×	Reply network
tgbl-flight	18,143	67,169,570	1,385	96.48	×	Transport
Wikipedia	8,227/1,000	157,474	152,757	88.41	✓	Interaction
Reddit	10,000/984	672,447	669,065	88.32	✓	Reply network
MOOC	7,047/97	411,749	345,600	56.66	✓	Interaction
LastFM	980/1,000	1,293,103	1,283,614	88.01	✓	Interaction
UCI	1,899	59,835	58,911	66.06	×	Social contact
Flights	13,169	1,927,145	122	79.50	×	Transport

A Time Complexity Analysis

A.1 Other strategies for extracting historical neighbors

An alternative approach for extracting historical neighbors is to maintain a fixed-size window of k most recent neighbors, as implemented in TGB [12]. This strategy necessitates updating the window by removing the oldest neighbor and adding the new one, incurring a cost of $O(B \log B)$ per batch for any temporal graph learning method [12]. For existing methods except CRAFT, this approach is more efficient than maintaining all historical neighbors (Section 4.3) when $(1+q)\log d_{\text{avg}} < \log B$. However, this window-based strategy presents two significant challenges: 1) It requires processing edges in strict chronological order, preventing edge shuffling for sequence-based methods, potentially leading to performance degradation. (We shuffle the training set during CRAFT's training process to learn more robust embeddings for CRAFT.) 2) For datasets with multiple edges occurring simultaneously, this strategy may be infeasible, as the neighbor window could include interactions at the prediction time, rather than before the prediction time, resulting in data leakage.

B Experiment Details

B.1 Code Availability

The implementation of CRAFT and baselines are based on DyGLib [42]. The implementation of CRAFT and all commands to reproduce the experimental results are publicly available at `https://github.com/luyi256/CRAFT`.

B.2 Datasets

In our experiments, we adopt a diverse set of 17 datasets: seven from TGB-Seq [40], four from TGB [12], and six commonly used datasets in the field. Table 4 presents the key statistics of these datasets for reference. Detailed descriptions of the datasets are available in their respective original papers: TGB-Seq datasets in [40], TGB datasets in [12], wikipedia, reddit, mooc, lastfm in [16], uci in [22], Flights in [23]. While other datasets have been proposed in the literature [23], we exclude those with an insufficient number of nodes to ensure robust evaluation.

B.3 Baselines

We provide a concise overview of the baselines used in our experiments, focusing on their memory and aggregation modules.

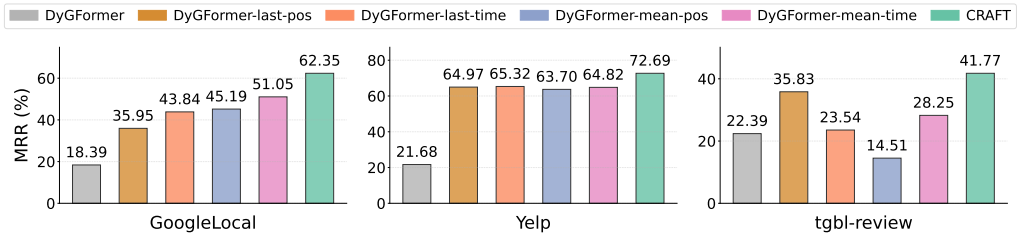

Figure 7: Evaluating the impact of learnable node embeddings and cross-attention: CRAFT and more variants of DyGFormer.

JODIE incorporates both memory and aggregation modules. The memory module utilizes an RNN to encode historical interactions into a compressed state. The aggregation (embedding) module generates current node representations by projecting the memory states with a time-context vector.

DyRep employs an RNN for memory state updates, similar to JODIE. It leverages temporal graph attention to transform interaction history, using these transformed representations as input for the memory module. The aggregation module is a simple identity mapping, directly outputting nodes' memory states.

TGN uses a GRU for memory state updates and temporal graph attention to aggregate historical neighbor information for node embeddings. This information encompasses neighbors' memory states, features, edge features, and timestamps.

TGAT pioneered temporal graph attention, employing multiple layers to aggregate historical neighbors without a memory module.

CAWN extracts temporal random walks for each node and applies an anonymization strategy to ensure inductive learning. These causal anonymous walks are then encoded by an RNN to generate node embeddings.

GraphMixer is a sequence-based method that simply aggregates historical neighbor information using MLP-Mixer without a memory module. It introduces a fixed time encoding scheme, demonstrating superior effectiveness compared to trainable time encoding.

DyGFormer, another sequence-based method, utilizes a transformer encoder in its aggregation module. It computes neighbor co-occurrence frequency between source and destination nodes to capture their relationship in terms of common neighbors.

B.4 Collision Check for Small Datasets

For datasets with a small number of nodes (wikipedia, reddit, mooc, lastfm, uci and Flights), we implement a collision check throughout the entire training process, including training, validation, and testing phases. This approach enhances the performance of most temporal graph learning methods compared to the results of the wikipedia and reddit datasets reported in [40]. This improvement is attributed to the elimination of false positive samples, which prevents model confusion during training and validation.

B.5 Implementation of TGAT-LE and DyGFormer-LE

In Section 5.2, we introduce TGAT-LE (TGAT with learnable embeddings) and DyGFormer-LE (DyGFormer with learnable embeddings) to demonstrate the effectiveness of learnable embeddings and the cross-attention mechanism. We provide detailed implementations of these variants below.

TGAT-LE. We replace node features with learnable embeddings without further modifications. While the standard TGAT configuration uses two layers of temporal graph attention, TGAT-LE employs only one layer. Due to the out-of-time issue, TGAT cannot run on the Yelp dataset with two layers [40]. Therefore, we evaluate TGAT with one layer for Yelp, while other TGAT results are directly adopted from Table 2 in the main paper. Notably, TGAT-LE outperforms TGAT even with a single layer, demonstrating the strong expressiveness of learnable embeddings.

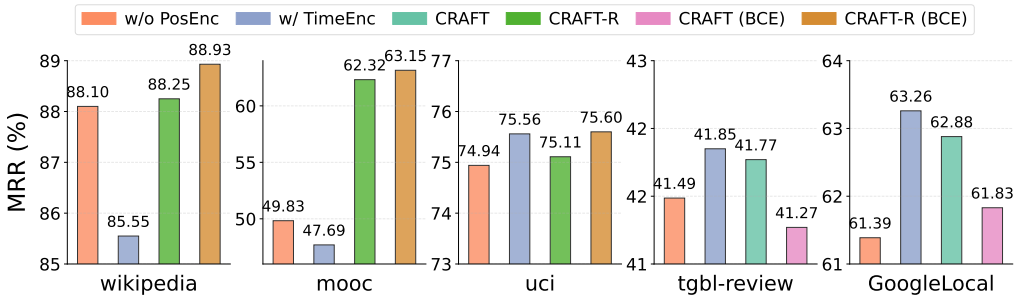

Figure 8: Comparisons of positional encoding vs. time encoding and BPR loss vs. BCE loss.

DyGFormer-LE. In this variant, we replace the original node features with learnable node embeddings and remove the neighbor co-occurrence frequency encoding scheme. Additionally, we replace the time encoding with learnable positional encoding, similar to the approach used in CRAFT. After passing through the transformer encoder, we use the output for the last neighbor of both the source and destination nodes as their respective node representations, following a strategy commonly used in Transformer-based sequence models [13]. To better differentiate this variant, we refer to it as *DyGFormer-last-pos*, as it uses the output of the last neighbor from the transformer encoder and adopts positional encoding.

More Variants of DyGFormer. An alternative strategy for producing node representations is to use *mean-pooling* over the outputs of all neighbors from the transformer encoder. Additionally, temporal information captured by positional encoding can, in principle, also be conveyed through time encoding. We explore these options and introduce three variants of DyGFormer: 1) *DyGFormer-last-time*, which uses the last neighbor output with time encoding instead of positional encoding; 2) *DyGFormer-mean-pos*, which uses mean-pooling with positional encoding; and 3) *DyGFormer-mean-time*, which uses mean-pooling with time encoding. For time encoding, we concatenate a time-context vector — obtained by projecting the time interval between the prediction time and each historical interaction through a linear layer — with the corresponding node embedding. The results of these variants are shown in Table 7. The performance of these variants varies across different datasets, suggesting that different datasets exhibit varying interaction patterns. For instance, the GoogleLocal dataset benefits from the mean-pooling strategy, possibly due to the consistent preferences of users in this dataset. Nevertheless, we observe that all variants underperform CRAFT, highlighting the superiority of our cross-attention mechanism.

B.6 Empirical Study: Positional Encoding vs. Time Encoding

In CRAFT, we use positional encoding to capture the temporal order of the source's neighbors. An alternative approach is time encoding, which encodes the time interval between the prediction time and the timestamps of historical interactions. We compare the performance of CRAFT using positional encoding and time encoding in Figure 8. For time encoding, we concatenate a time-context vector, which is obtained by projecting the time interval through a linear layer, with the node embedding of each neighbor of the source. These concatenated vectors serve as the key and value inputs to the cross-attention module. In the figure, *w/o PosEnc* refers to models without either positional or time encoding (same as in Figure 6), *w/o TimeEnc* refers to models using time encoding instead of positional encoding, and the original CRAFT (or CRAFT-R) uses positional encoding. We observe that CRAFT with positional encoding and CRAFT with time encoding perform comparably across most datasets. However, CRAFT with positional encoding significantly outperforms CRAFT with time encoding on the wikipedia and mooc datasets. This suggests that using time encoding introduces excessive noise in these datasets, whereas the temporal order provided by positional encoding is sufficient for capturing the sequential dynamics.

B.7 Empirical Study: BPR loss vs. BCE loss

Most temporal graph learning methods are trained using the Binary Cross Entropy (BCE) loss, whereas CRAFT adopts the BPR loss. To assess the impact of the loss function, we compare the

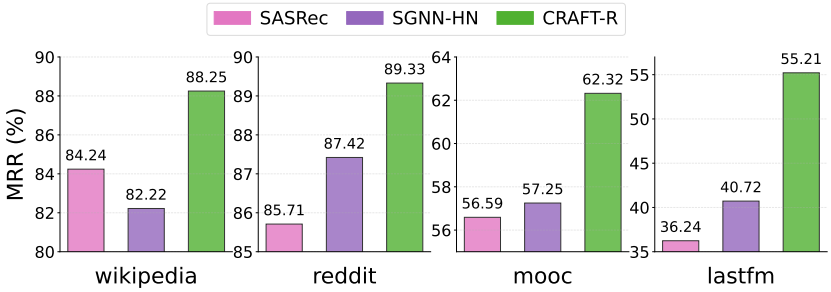

Figure 9: MRR scores of CRAFT-R, SASRec and SGNN-HN on seen-dominant bipartite datasets.

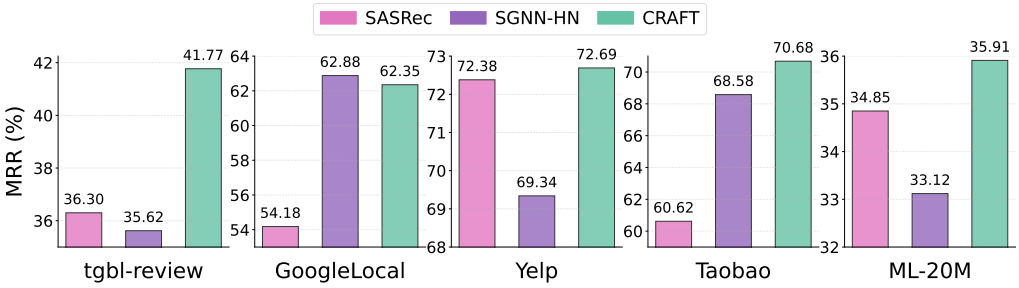

Figure 10: MRR scores of CRAFT, SASRec and SGNN-HN on unseen-dominant bipartite datasets.

performance of CRAFT trained with BPR loss and BCE loss in Figure 8. The results show that both versions perform comparably across most datasets. Among the five evaluated datasets, only GoogleLocal exhibits a performance gap greater than 1% between the two losses. These findings highlight the robustness of CRAFT and indicate that its performance is largely unaffected by the choice between BPR and BCE loss functions.

B.8 Comparing CRAFT with Recommendation Methods

As shown in [40], SGNN-HN, a sequential recommendation method, outperforms various temporal graph learning methods on bipartite datasets. To further assess the effectiveness of CRAFT, we compare it with two widely recognized recommendation methods: SASRec [13] and SGNN-HN [21], on bipartite datasets. Figures 9 and 10 display the MRR scores of CRAFT (and CRAFT-R), SASRec, and SGNN-HN on four small bipartite datasets and five large bipartite datasets, respectively. We observe that CRAFT achieves the best performance on most datasets, except for the GoogleLocal dataset, where it slightly trails behind SGNN-HN. Generally, CRAFT-R outperforms both SASRec and SGNN-HN on seen-dominant datasets with a larger margin, compared to the results on unseen-dominant datasets. These results suggest that existing recommendation methods perform worse than temporal graph learning methods in seen-dominant scenarios, while our proposed CRAFT achieves the best of both worlds.

Table 5: The tuning ranges of hyperparameters for CRAFT.

Hyperparameter	Tuning Range	Descriptions
$N_{\text{neighbors}}$	[30, 60, 90, 120]	the number of neighbors used for each source
$p_{\text{hidden\_dropout}}$	[0.1, 0.2, 0.3, 0.5]	the dropout rate for FFNs and MLPs
$p_{\text{attn\_dropout}}$	[0.1, 0.2]	the dropout rate for attention scores
$p_{\text{emb\_dropout}}$	[0.1, 0.2]	the dropout rate for node embeddings
N_{layers}	[1, 2]	the number of layers in cross-attention

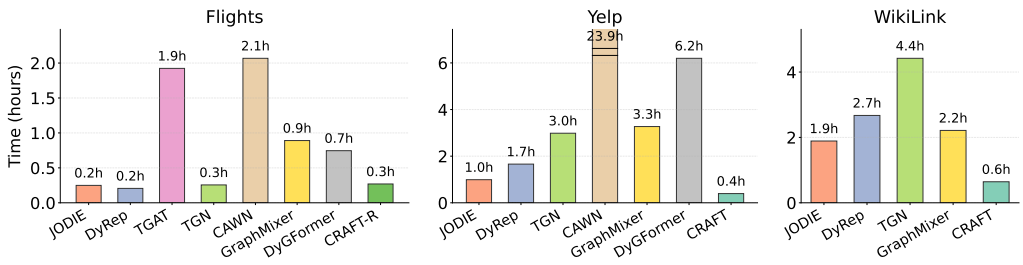

Figure 11: Training time per epoch of CRAFT and baselines on Flights, Yelp and WikiLink.

Table 6: Optimal hyperparameters and embedding size F for CRAFT, and the fixed batch size B on various datasets.

Dataset	$N_{\text{neighbors}}$	$p_{\text{hidden\_dropout}}$	$p_{\text{attn\_dropout}}$	$p_{\text{emb\_dropout}}$	N_{layers}	B	F
uci	30	0.3	0.2	0.2	1	200	64
wikipedia	120	0.1	0.1	0.2	1	200	64
reddit	120	0.1	0.2	0.1	1	200	64
mooc	30	0.1	0.1	0.1	2	200	64
Flights	90	0.2	0.2	0.2	1	200	64
lastfm	120	0.1	0.1	0.1	1	200	64
GoogleLocal	60	0.2	0.1	0.2	2	200	128
Flickr	90	0.1	0.1	0.2	1	400	128
YouTube	90	0.2	0.2	0.2	2	400	128
Taobao	60	0.2	0.1	0.2	2	400	128
Yelp	60	0.2	0.2	0.2	2	400	128
ML-20M	60	0.2	0.2	0.2	1	400	128
Wikilink	60	0.2	0.2	0.2	1	400	128
tgbl-review	120	0.1	0.2	0.1	1	200	128
tgbl-coin	60	0.2	0.2	0.2	1	200	128
tgbl-comment	60	0.2	0.1	0.2	1	200	128
tgbl-flight	90	0.2	0.2	0.2	1	200	128

B.9 Training Time Comparison

Figure 11 illustrates the training time per epoch of CRAFT and the baselines on the Flights, Yelp, and WikiLink datasets. CRAFT is the most efficient method overall, with the exception of being slightly slower than JODIE and DyRep on the smallest dataset, Flights. As the dataset size increases, the efficiency advantage of CRAFT becomes more pronounced. For instance, on the WikiLink dataset, many temporal graph learning methods fail to complete a single training epoch within 24 hours, whereas CRAFT finishes one epoch in just 0.6 hours. Compared to the most efficient baseline, JODIE, CRAFT is 1.5x faster on the WikiLink dataset.

B.10 Hyperparameters

The hyperparameters for the baselines on the TGB-Seq benchmark, TGB benchmark, and six commonly used datasets (wikipedia, reddit, mooc, lastfm, uci, and Flights) are selected based on the best configurations reported in [40],[12], and[42], respectively. Thanks to CRAFT's simple architecture, it involves only a limited number of hyperparameters. We fix the batch size across all methods, and set the embedding size to 64 for small datasets and 128 for large ones. Table 5 presents the hyperparameters and their tuning ranges. We perform grid search over these ranges using the validation set to determine the optimal settings. The final hyperparameters for CRAFT, along with the batch size and embedding size per dataset, are summarized in Table 6.

C Discussion

C.1 Broader Impact

This paper introduces CRAFT, an efficient and effective temporal graph learning method for future link prediction. CRAFT can be applied across a range of real-world scenarios, including user recommendation on social media, product recommendation on e-commerce platforms, and interaction forecasting in social, financial, and gaming networks. By enabling more accurate and scalable future link prediction, CRAFT has the potential to enhance user experience and improve overall system utility in these domains.

C.2 Transductive and Inductive Future Link Prediction

The transductive setting for future link prediction focuses on forecasting future links for nodes observed during training, while the inductive setting targets predicting links involving previously unseen nodes [26]. In this paper, we do not explicitly separate the transductive and inductive tasks. All datasets follow the original training, validation, and test splits provided by existing benchmarks or are split chronologically. As a result, our evaluation naturally encompasses both settings.

CRAFT is primarily designed for transductive future link prediction, as it relies on historical interactions to learn meaningful node embeddings. In the inductive case, where the source or destination node has little to no interaction history, the task becomes significantly more challenging. Without prior interactions, it is difficult for any method to infer a new node's behavioral preferences. This scenario aligns with the well-known *cold-start* problem in recommendation systems [27], where common solutions involve incorporating node profile features and estimating similarity with known nodes. We view this inductive scenario as a distinct challenge from standard future link prediction, which focuses on leveraging interaction history to model node preferences and predict future links based on dynamic behavioral patterns. Addressing the inductive case effectively may require additional assumptions or auxiliary data beyond the scope of this work.

Another perspective on inductive learning is to eliminate training costs by expecting the model to generalize directly to unobserved nodes, assuming these nodes have sufficient historical interactions to infer their preferences. However, as demonstrated in Section 5.2, incorporating learnable node embeddings substantially improves the performance of temporal graph learning methods by enhancing their expressive power. Moreover, the experimental results in Table 2 and Table 3 show that CRAFT consistently outperforms baselines designed for inductive learning, despite their more complex architectures. While inductive learning remains a valuable research direction, we argue that learning from historical interactions is a fundamental requirement for future link prediction, which depends on accurately modeling node-specific interaction patterns. Therefore, training CRAFT on all available historical interactions to derive meaningful node embeddings offers a more effective approach.

C.3 Negative Sampling Strategies

Poursafaei et al. [23] proposed two challenging negative sampling strategies for future link prediction: historical negative sampling and inductive negative sampling. In this work, we adopt multiple negative samples per positive instance and follow the random negative sampling strategy used in prior benchmarks [12, 40], as using multiple random negatives per sample is sufficiently challenging for the models and provides a fair basis for comparison.

C.4 Features

The current version of CRAFT does not incorporate any node or edge features. This decision is motivated by the fact that most benchmark datasets lack such features, and even when available, they often provide limited additional value for future link prediction, as observed in prior work [6]. Nevertheless, CRAFT is flexible and can easily accommodate features if needed. Node features can be concatenated with learnable node embeddings, while edge features can be appended to the source's neighbor embeddings or combined with positional encodings to indicate edge types.

C.5 Limitations and Future Work

CRAFT is specifically designed for future link prediction and does not directly support other tasks such as dynamic node classification. Our design focuses on modeling the relationship between the destination node and the source node's interaction history, without maintaining dynamic node representations over time. To extend CRAFT to support dynamic node classification, a straightforward approach is to update node embeddings continuously via gradient descent. This has the potential advantage of enabling a single model to support multiple tasks, eliminating the need for training separate models as required in prior work. Exploring such unified and flexible modeling capabilities is a promising direction for future research.