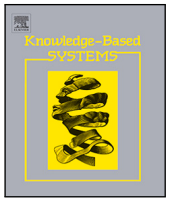
Knowledge-Based Systems 300 (2024) 112245

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys



MTdyg: Multi-scale transformers with continuous time dynamic graph model for link prediction

Long Xu, Zhiqiang Pan, Honghui Chen *, Shen Wang

National Key Laboratory of Information Systems Engineering, Changsha, 410000, China

ARTICLE INFO

Keywords:
Link prediction
Continuous time dynamic graph
Multi-scale patches
Transformer

ABSTRACT

Link prediction through continuous dynamic graph neural networks is a challenging endeavour. Previous studies have considered historic interaction sequences among pairs of nodes. However, this approach does not sufficiently model the links between them. Thus, the frequency of historical common neighbours was proposed to obtain more information about node pairs. Nonetheless, this measure fails to consider the timing of interactions because the relative importance of different interactions shifts over time.

Multi-scale transformers with a continuous time dynamic graph model (MTdyg) are our response to this challenge to enhance link prediction. The novel MTdyg model employs a multi-scale patching strategy that dynamically adjusts based on the interaction frequency, feeding segmented interaction sequences into the transformer. This methodology significantly improves the model's ability to assimilate historical data. Furthermore, we developed a temporal attention-based historical common neighbour encoding technique to effectively identify linkages between source and target nodes in interaction sequences. Extensive testing on eight public datasets confirmed that MTdyg delivers superior performance on most datasets, demonstrating its effectiveness in capturing the nuances of node interactions over time.

1. Introduction

Dynamic graph neural networks (GNNs) have become an essential research area in link prediction because they can effectively capture temporal evolutionary patterns within graph structures. Link prediction identifies potential future links between nodes, which is critical for social network analysis, bioinformatics and recommendation systems. Although traditional static graph methods such as graph convolutional networks (GCNs) [1] and graph attention networks (GATs) [2] have made significant strides in static relationship representation, they often fail to account for the temporal dynamics of graph structures. Recent studies have integrated temporal information into GNNs, such as continuous time dynamic graph networks (CTDGNs) [3]. However, there is significant room for improvement relative to leveraging historical interaction information between node pairs.

Current link prediction frameworks that leverage on CTDGNs, such as DYGlib [4], use a simplistic encoding strategy for historical common neighbour information between the source and target nodes, relying only on frequency occurrences. However, this frequency-based encoding does not fully capture the intricate dynamics of node interactions. For example, in the social relationship example in Fig. 1, User 1 had a business interaction with User 3 3 years previously. Conversely, User 2 engaged in a recent friendship interaction with User 3 only a month ago. The disparity in the timing and type of interaction suggests differing levels of User 3's influence on Users 1 and 2. Treating these interactions as equally significant is flawed. For a more efficient and adaptable approach to dynamic graph data, the integration of patch technology with transformer models has been introduced for dynamic link prediction [5]. However, these methods often focus on a single patch size during training, overlooking the richer narrative that multi-scale patching can provide.

To mitigate the identified issues, we propose a multi-scale transformers with a continuous time dynamic graph model (MTdyg) for link prediction. Initially, the proposed model augments the dynamic interplay between source and target nodes using a temporal attention-based encoding of historical common neighbours. This is followed by deploying a multi-scale patch technique that is adaptively selected based on the interaction frequency to divide and input the interaction sequences into the transformer.

Our research included empirical trials on six benchmark datasets that are available for the public domain. The data obtained from these trials suggest that the performance of the proposed MTdyg model consistently exceeds that of the established baseline approaches on most datasets.

* Corresponding author.
E-mail addresses: xulong@nudt.edu.cn (L. Xu), panzhiqiang@nudt.edu.cn (Z. Pan), chenhonghui@nudt.edu.cn (H. Chen).

https://doi.org/10.1016/j.knosys.2024.112245
Received 31 March 2024; Received in revised form 19 June 2024; Accepted 10 July 2024
Available online 15 July 2024

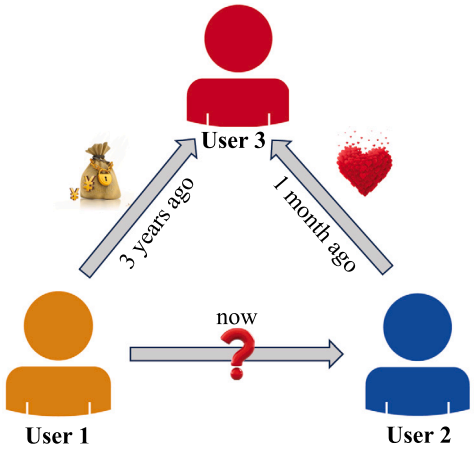


Fig. 1. A social example where the grey arrow represents an interaction between two users, the gold bag represents a business interaction, and the heart represents a friendship interaction.

In summary, the contributions of this article is summarised as follows:

- We propose a time-attention mechanism-based historical common neighbour coding method to explore how the historical common neighbour information of node pairs can be better used.
- We present a multi-scale patching technology based on frequency-adaptive selection that segments interactive sequences into transformer models to obtain sequence information more comprehensively and efficiently.
- We compared MTdyg's performance against state-of-the-art baselines on six public benchmark datasets, and the results demonstrated that MTdyg outperformed state-of-the-art baseline models on most datasets.

2. Related work

In this section, we briefly review three types of research related to our work: dynamic graph representation learning, Fourier transform and the patch technique.

2.1. Dynamic graph representation learning

To capitalise on temporal data, dynamic graph representation learning has become an increasingly prominent research domain [6]. It principally addresses three tasks: node, edge and subgraph predictions [7]. Our study is primarily concerned with edge prediction.

Existing dynamic graph representation learning methods can be categorised into two primary classes: discrete-time dynamic graph learning models and continuous time dynamic graph (CTDG) learning models [3]. Discrete-time techniques typically involve partitioning a dynamic graph into snapshots at predefined intervals. These are then integrated using graph and sequence encoding–decoding methods for subsequent use in downstream tasks. For example, Pareja et al. [8] leveraged recurrent neural networks (RNNs) to update GCN parameters over time, thereby capturing the dynamic characteristics of graph data. Similarly, Sijie Yan, Yuanjun Xiong and Dahua Lin [9] have modelled dynamic skeletal structures using a time-series representation of human joint positions, proposing a spatiotemporal GCN (ST-GCN) to grasp the evolving spatiotemporal relationships.

Methods for CTDG representation typically store the dynamic graph as a stream of update events, directly extracting information from each interaction as it occurs to learn the graph representation. For example, Da Xu et al. [10] devised a temporal encoding technique that employs self-attention mechanisms grounded in Bochner's theorem of harmonic analysis to effectively capture continuous temporal information. Emanuele Rossi et al. [11] utilised a memory module to capture and store time-varying information, adeptly handling the dynamic characteristics of graphs. In addition, Rakshit Trivedi et al. [12] have utilised an event-based approach to update node representations based on the type and timing of occurrences.

In addition, another study proposed a dynamic GNN model that leverages window-based subgraph sampling and attention mechanisms. Alomrani et al. [13] proposed an encoder that utilises time edge encoding and window-based subgraph sampling and is trained using a non-contrastive self-supervised learning framework. Cui et al. [14] extended the embedding propagation of GCN to dynamic methods, and they significantly improved the efficiency of dynamic graph models. Mei et al. [15] introduced the GCN_MA framework, which effectively combined GCN, RNN and multi-head attention methods.

Although these approaches effectively capture temporal information, most existing dynamic graph models have yet to effectively leverage the information from interactions between nodes. The intricate nature of graph structures, compounded by the need to factor in temporal aspects, poses a significant challenge when processing lengthy sequences of node interactions. Our research addresses this issue by introducing a transformer-based model that can extract long-term interactions and relationships between nodes.

2.2. Fourier transform

Fast Fourier transform (FFT) plays a pivotal role in signal processing and analysis because it is a potent tool for converting time and frequency domains. Numerous researchers have applied it in various fields, including image vision and deep learning. For instance, Xintian Mao et al. [16] developed a convolution-accelerated residual FFT approach that significantly enhances image-deblurring capabilities. Wenxian Wang et al. [17] executed a Fourier transform on images using a bilateral aggregate decoder, followed by the application of filters for target reconstruction. Keivan AlizadehVahid et al. [18] adapted the butterfly operation from traditional Fourier transforms for the architectural design of convolutional neural networks (CNNs), effectively easing the computational bottleneck presented by 1×1 convolutions in lightweight networks.

In the past few years, there has been a growing effort to employ Fourier transforms in analysing time-series data. Haixu Wu et al. [19] have used the FFT to detect distinct periodicities in sequential data, thereby identifying their cyclical patterns. Yuxing Tian, Yiyan Qi and Fan Guo [20] have translated graph interaction data into the frequency domain, thereby allowing a more in-depth investigation of interactive signals. Hao Wu et al. [21] have combined global Fourier transformations with a parallel local convolutional framework to detect time-dependent changes effectively.

In this study, selecting appropriate patch sizes is crucial for the application of multi-scale patches. We extracted frequencies from the data via Fourier transform to guide the selection of patch sizes at multiple scales. The experimental results corroborate the effectiveness of selecting patch sizes based on frequency-domain information.

2.3. Patch technique

Patch techniques have been extensively applied in image processing, significantly reducing the demand for computational resources. For instance, Alexey Dosovitskiy et al. [22] utilised a patch-based approach to convert images into sequential data for input into transformer models, achieving efficient image classification. Zhiyang Chen et al. [23] introduced adaptively adjustable deformable patches that, drawing on local context information, generate comprehensive features for each patch. Ugur Demir and Gozde Unal [24] employed a discriminator

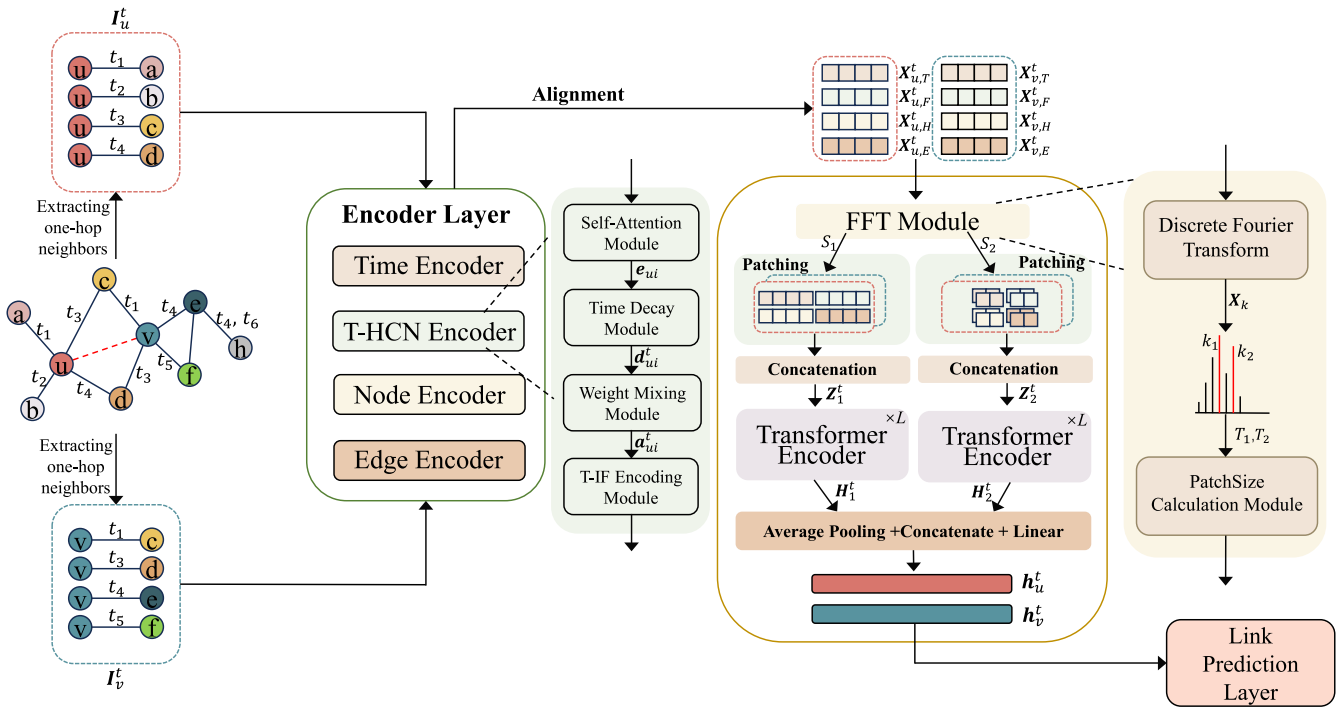


Fig. 2. Overview of MTdyg.

to assess the authenticity of each patch, thereby enhancing the local texture details of the generated images.

Patch technology has notably excelled in time series forecasting. Yuqi Nie et al. [5] have utilised patches to extract local semantic information from time-series data, thereby reducing the spatial and temporal complexity of the models. Keying Gong, Yujin Tang and Junwei Liang [25] have performed patch segmentation in both the time and frequency domains by employing a CNN+MLP architecture to obtain the final embeddings. Seunghan Lee, Taeyoung Park and Kibok Lee [26] have achieved commendable results in forecasting by relying solely on the information inherent to the patches themselves without the need for additional masking operations.

In the graph representation learning domain, researchers have also applied patch technology to expedite training efficiency [20]. However, they have overlooked the inherent flexibility of patch techniques, which allows for the consideration of both local details and broader global information. Therefore, this paper proposes a multi-scale patch transformer model that enhances the model's generalisability.

3. Problem definition

Given a sequence of continuously evolving dynamic graphs $\mathcal{G} = \{G_1, G_2, \ldots, G_T\}$, where each $G_t = (V_t, E_t)$ represents the graph at time step t. The V_t and E_t sets represent the nodes and edges, respectively, and can change over time. The continuous dynamic graph link prediction task predicts edge set $E_T + 1$ at the next time step $T + 1$. The input comprises a sequence of continuously evolving dynamic graphs $\mathcal{G} = \{G_1, G_2, \ldots, G_T\}$, where each $G_t = (V_t, E_t)$ represents the graph at time step t. Each graph G_t includes node feature matrix $\boldsymbol{X}_t \in \mathbb{R}^{|V_t| \times d}$ and an edge feature matrix $\boldsymbol{Y}_t \in \mathbb{R}^{|E_t| \times f}$, where d and f are the dimensions of the node and edge features, respectively. The node feature matrix $\boldsymbol{X}_t$ of graph G_t has the features of each node at time step t, with a dimension of $|V_t| \times d$. The edge feature matrix $\boldsymbol{Y}_t$ includes the features of each edge at time step t, with a dimension of $|E_t| \times f$. The output is the predicted edge set $\hat{E}_{T+1}$ at the future time step $T + 1$, which represents the edges that may exist at that time. The goal is to predict changes in the graph structure at time step $T+1$ based on the previous T time steps, particularly changes in edges. This involves predicting the emergence of new edges, disappearance of existing edges and changes in edge weights. These predictions benefit various applications, such as relationship prediction in social networks, optimisation paths in transportation networks and inferring relationships in knowledge graphs.

4. Formulation of MTdyg

We propose the MTdyg model, and an overview is provided in. Fig. 2. To simplify the task model, we initially construct a one-hop interaction sequence $\boldsymbol{I}_u^t$ and $\boldsymbol{I}_v^t$ for N source and target nodes in chronological order, where u denotes the source node, v is the target node and t is the interaction time. Subsequently, we encode the time, node, edge and historical common neighbour frequency for each interaction sequence to obtain sequence embeddings. Next, we set two patch size values based on the sequence frequency information obtained via FFT and divide each sequence encoding into multiple patches accordingly. Then, patches of varying sizes are input to different transformer modules. Finally, the output features are concatenated and applied to a linear layer for downstream tasks in link prediction.

4.1. Encoder layer

In the following subsections, we focus exclusively on source node analysis. The target nodes are processed using a procedure identical to that of the source nodes, resulting in the corresponding encodings, $\boldsymbol{X}_{v,T}^t$, $\boldsymbol{X}_{v,H}^t$, $\boldsymbol{X}_{v,E}^t$ and $\boldsymbol{X}_{v,F}^t$.

4.1.1. Time encoder

The purpose of the time encoder is to map interaction time intervals to the vector space $\boldsymbol{X}_{u,T}^t \in \mathbb{R}^{N \times d_T}$, with d_T indicates the dimensionality of the time embeddings. Compared to the DyGFormer approach, the time interval $\Delta t_n = t - t_n$ is encoded as a vector $\sqrt{\frac{1}{d_T}}[\cos(\Delta t_n \omega_1), \sin(\Delta t_n \omega_1), \ldots, \cos(\Delta t_n \omega_{d_T}), \sin(\Delta t_n \omega_{d_T})]$, thereby producing cyclic time encoding, where $\omega_1, \cdots, \omega_{d_T}$ are trainable parameters.

4.1.2. Node/edge encoder

In the graph data domain, both nodes and edges are characterised by distinct attributes. To encode the interaction characteristics, embeddings for adjacent nodes and edges are extracted directly from the interaction sequence $\boldsymbol{I}_u^t$. Nodes are then encoded into the matrix $\boldsymbol{X}_{u,H}^t \in \mathbb{R}^{N\times d_H}$, where d_H is the dimensionality assigned to node embeddings. The edges are encoded into $\boldsymbol{X}_{u,E}^t \in \mathbb{R}^{N\times d_E}$, with d_E denotes the corresponding dimensionality for edge embeddings.

4.1.3. Temporal attention-based historical common neighbours(T-HCN) encoder

Most current approach encode node interaction sequences independently without considering their interdependencies. Yu et al. [4] recently developed a method for encoding neighbour co-occurrences based on the premise that nodes with a greater number of historical common neighbours have a higher propensity for future interactions. Nonetheless, this approach overlooks the substantial variation in the duration and type of interactions across different neighbour categories. Adhering to this co-occurrence encoding framework implicitly assigns equal significance to all interactions, a notion that diverges from the intricacies of real-life interaction dynamics.

To address this issue, we propose a historical common neighbour encoding method based on a temporal attention mechanism. This method not only considers the frequency of common historical neighbours but also quantitatively evaluates their significance. It can distinguish the influence of different times and types of interactions on future interactions between two nodes. Specifically, for interaction sequences $\boldsymbol{I}_u^t$ and $\boldsymbol{I}_v^t$, we calculate the number of interactions for each common neighbour and the interactions between them. Calculating this interaction frequency is not simply incremented by 1 for each occurrence; instead, the significance of the interaction is weighted before being added to the total frequency. For example, assuming that the historical interaction sequences for u and v are $\{a, b, v, a, c, a, b\}$ and $\{a, b, u, b, c, b, c\}$ respectively, we can determine the interaction counts between a, b, c and u, v as [3, 1], [2, 3] and [1, 2], respectively. The historical interaction count between u and v is denoted by [1, 1]. Therefore, the node interaction counts for u and v can be represented as follows:

$$\boldsymbol{F}_u^t = [[3,1],[2,3],[1,1],[3,1],[1,2],[3,1],[2,3]]^T$$

$$\boldsymbol{F}_v^t = [[3,1],[2,3],[1,1],[2,3],[1,2],[2,3],[1,2]]^T$$

The encoding of historical common neighbours necessitates using a temporal attention mechanism to apply weights to the interaction frequencies of nodes u and v, thereby determining the significance of varying interactions. The representation of historical common neighbours is delineated as follows:

$$\boldsymbol{X}_{u,F}^t = f(TA(\boldsymbol{F}_u^t[:,0])) + f(TA(\boldsymbol{F}_u^t[:,1])) \in \mathbb{R}^{N\times d_F} \tag{1}$$

where d_F is the embedded dimension of the historical common neighbour, and $TA(\cdot)$ is the time-attention function. $f(\cdot)$ has an input dimension of 1 and output dimension of d_F. This research uses a two-layer MLP whose activation function is RELU to implement $f(\cdot)$[27].

Next, we explain the temporal attention function $TA(\cdot)$, utilising node u as an illustrative case. For each historical common neighbour i of node u, the process begins by calculating the self-attention score $e_u i$.

$$e_{ui} = Softmax\left(\frac{(\boldsymbol{W}_q \boldsymbol{F}_{ui}) \cdot (\boldsymbol{W}_k \boldsymbol{F}_{ui})^T}{\sqrt{d_k}}\right) \tag{2}$$

where $\boldsymbol{W}_q$ and $\boldsymbol{W}_k$ denote the trainable weight matrices, with $\boldsymbol{F}_{ui}$ representing the embedding of link attributes between nodes u and i, which is obtained from the edge attributes in the original dataset using an MLP. This allows the capture of weighted information for distinct interaction types. We then determine the time decay function $\boldsymbol{d}_{ui}^t$ for every interaction.

$$\boldsymbol{d}_{ui}^t = \frac{\exp(-\alpha \cdot \Delta \boldsymbol{t}_{ui})}{\sum_j \exp(-\alpha \cdot \Delta \boldsymbol{t}_{uj})} \tag{3}$$

where α is the decay rate. The attention score and the time decay factor are combined to obtain the time-attention weight $\boldsymbol{a}_{ui}^t$.

$$\boldsymbol{a}_{ui}^t = \frac{\exp(e_{ui} \cdot \boldsymbol{d}_{ui}^t)}{\sum_k \exp(e_{uk} \cdot \boldsymbol{d}_{uk}^t)} \tag{4}$$

Finally, we use the established temporal attention weights to weight the encoding of the frequency of node interactions, which is referred to as the temporal attention function $TA(\cdot)$ described in this text.

$$TA(\boldsymbol{p}) = \boldsymbol{a}_{ui}^t \cdot \boldsymbol{p} \tag{5}$$

4.2. Multi-scale transformer module based on adaptive patches

4.2.1. Setting patch size values based on fourier transform

Fourier transform exposes the inherent periodicity and spectral makeup of a signal, which is essential for understanding the underlying patterns of interaction sequences. Specifically, we apply discrete Fourier transform into the encoded vectors of node pairs to identify the primary frequency constituents. This step is crucial because it allows us to detect the prominent periodic behaviours within the sequences, which indicate recurring interaction patterns between nodes. The Fourier transform is mathematically expressed as follows:

$$\boldsymbol{X}_k = \sum_{n=0}^{L-1} \boldsymbol{x}_l \cdot e^{-j(2\pi/N)kl} \tag{6}$$

where $\boldsymbol{X}_k$ is the coefficient in the frequency domain, k is the frequency index, l is the time index, $\boldsymbol{x}_l$ is the signal amplitude in the time domain (i.e. the value at time index l, and j is the imaginary number unit. By examining the magnitude $|\boldsymbol{X}_k|$, we can determine the main periodic components of the sequence.

From the transformed data, we identify the T_1 and T_2 periods corresponding to the two most significant frequency components as determined by the peaks in the frequency spectrum. These periods are calculated as follows

$$T_1 = \frac{Len}{k_1},\ T_2 = \frac{Len}{k_2} \tag{7}$$

where k_1 and k_2 are the indices of the first and second peaks, respectively. Len is the sequence length. These periods reflect the most prominent cycles in the interaction sequences. Subsequently, we compute the multi-scale patch sizes S_1 and S_2 using the T_1 and T_2 periods as follows:

$$S_1 = \lceil \beta \cdot T_1 \rceil, S_2 = \lceil \beta \cdot T_2 \rceil \tag{8}$$

β is a hyperparameter, and $\lceil \cdot \rceil$ denotes the ceiling function that ensures that the patch size is an integer.

By leveraging the Fourier transform to determine these patch sizes, our model can more effectively capture both short- and long-term dependencies within the interaction sequences, which are crucial for accurate link prediction.

4.2.2. Patching operation

To obtain more comprehensive long-term sequential data, the encoded sequence is dissected into several patches. In the antecedent section, we determined two patch dimensions, S_1 and S_2. The encoded vector $\boldsymbol{X}_{u,T}^t \in \mathbb{R}^{N\times d_T}$ is initially subjected to patching, partitioning it into $a_{u,n}^t = \left\lceil \frac{N}{S_n} \right\rceil$ patches, where n indicates either 1 or 2. If N is not divisible by S_n, padding is performed on $\boldsymbol{X}_{u,T}^t$. Consequently, this yields encoded patches $\boldsymbol{P}_{u,T,n}^t \in \mathbb{R}^{a_{u,n}^t \times d_T \cdot S_n}$. In a similar vein, we obtain $\boldsymbol{P}_{u,H,n}^t \in \mathbb{R}^{a_{u,n}^t \times d_H \cdot S_n}$, $\boldsymbol{P}_{u,E,n}^t \in \mathbb{R}^{a_{u,n}^t \times d_E \cdot S_n}$, $\boldsymbol{P}_{u,F,n}^t \in \mathbb{R}^{a_{u,n}^t \times d_F \cdot S_n}$, $\boldsymbol{P}_{v,T,n}^t \in \mathbb{R}^{a_{v,n}^t \times d_T \cdot S_n}$, $\boldsymbol{P}_{v,H,n}^t \in \mathbb{R}^{a_{v,n}^t \times d_H \cdot S_n}$, $\boldsymbol{P}_{v,E,n}^t \in \mathbb{R}^{a_{v,n}^t \times d_E \cdot S_n}$ and $\boldsymbol{P}_{v,F,n}^t \in \mathbb{R}^{a_{v,n}^t \times d_F \cdot S_n}$.

4.2.3. Transformer encoder

Initially, the four acquired patched encodings are concatenated with adaptable weight matrices $\boldsymbol{W}_{*,n} \in \mathbb{R}^{d*\cdot S_n \times d}$ and corresponding bias vectors $\boldsymbol{b}_{*,n} \in \mathbb{R}^d$. This alignment to a d-dimensional framework facilitates the synthesis of the dimensionally uniform encodings $\boldsymbol{Z}^t_{u,*,n} \in \mathbb{R}^{a^t_{u,n} \times d}$ and $\boldsymbol{Z}^t_{v,*,n} \in \mathbb{R}^{a^t_{v,n} \times d}$.

$$\boldsymbol{Z}^t_{u,*,n} = \boldsymbol{P}^t_{u,*,n}\boldsymbol{W}_{*,\,n} + \boldsymbol{b}_{*,\,n},\ \boldsymbol{Z}^t_{v,*,n} = \boldsymbol{P}^t_{v,*,n}\boldsymbol{W}_{*,\,n} + \boldsymbol{b}_{*,\,n} \tag{9}$$

where $*$ denotes any of T, H, E or F. S_n denotes the size of the patch. $a^t_{u,n}$ and $a^t_{v,n}$ indicate the number of patches.

Subsequently, the patched encodings corresponding to u and v are concatenated individually, yielding $\boldsymbol{Z}^t_{u,n}$ and $\boldsymbol{Z}^t_{v,n}$, which are represented as $\boldsymbol{Z}^t_{u,n} = \boldsymbol{Z}^t_{u,T,n} \| \boldsymbol{Z}^t_{u,H,n} \| \boldsymbol{Z}^t_{u,E,n} \| \boldsymbol{Z}^t_{u,F,n} \in \mathbb{R}^{a^t_{u,n} \times 4d}$ and $\boldsymbol{Z}^t_{v,n} = \boldsymbol{Z}^t_{v,T,n} \| \boldsymbol{Z}^t_{v,H,n} \| \boldsymbol{Z}^t_{v,E,n} \| \boldsymbol{Z}^t_{v,F,n} \in \mathbb{R}^{a^t_{v,n} \times 4d}$, respectively. Before their introduction to the transformer module, $\boldsymbol{Z}^t_{u,n}$ and $\boldsymbol{Z}^t_{v,n}$ undergo an additional concatenation to facilitate the learning of the intrinsic associative dynamics between u and v, culminating in $\boldsymbol{Z}^t_n = \boldsymbol{Z}^t_{u,n} \| \boldsymbol{Z}^t_{v,n} \in \mathbb{R}^{(a^t_{u,n}+a^t_{v,n}) \times 4d}$

We now introduce the structural design of our transformer module. This configuration employs L layers, each featuring a multi-head self-attention block and a subsequent feedforward neural network block. Layer normalisation [28] is implemented at the inception of each block, and the Gaussian error linear unit (GELU) is adopted as the activation function [29]. Following each block, we incorporate a residual connection to ensure the continuity and stability of the learning process [30]. The internal computational methodology of the proposed module is explicated subsequently:

$$Attention(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = softmax(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}})\boldsymbol{V} \tag{10}$$

$$MultiH(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = [hd_1 \| \ldots \| hd_h]\boldsymbol{W}^O \tag{11}$$

$$hd_i = Attention(\boldsymbol{Q}\boldsymbol{W}^Q_i, \boldsymbol{K}\boldsymbol{W}^K_i, \boldsymbol{V}\boldsymbol{W}^V_i) \tag{12}$$

$$FFN(\boldsymbol{x}) = GELU(\boldsymbol{x}\boldsymbol{W}_1 + \boldsymbol{b}_1)\boldsymbol{W}_2 + \boldsymbol{b}_2 \tag{13}$$

$$Z = LN(\boldsymbol{Z}^{t,l-1}_n) \tag{14}$$

$$\boldsymbol{x}^{t,l}_{i,n} = Attention(Z\boldsymbol{W}^{Q,l}_{i,n}, Z\boldsymbol{W}^{K,l}_{i,n}, Z\boldsymbol{W}^{V,l}_{i,n}) \tag{15}$$

$$\begin{aligned}\boldsymbol{x}^{t,l}_n &= MultiH(Z\boldsymbol{W}^{Q,l}_{i,n}, Z\boldsymbol{W}^{K,l}_{i,n}, Z\boldsymbol{W}^{V,l}_{i,n}) + \boldsymbol{Z}^{t,l-1}_n \\ &= Concat(\boldsymbol{x}^{t,l}_{1,n}, \ldots, \boldsymbol{x}^{t,l}_{h,n})\boldsymbol{W}^{O,l}_n + \boldsymbol{Z}^{t,l-1}_n\end{aligned} \tag{16}$$

$$\begin{aligned}\boldsymbol{Z}^{t,l}_n &= FFN(\boldsymbol{x}^{t,l}_n) + \boldsymbol{x}^{t,l}_n \\ &= GELU(\boldsymbol{x}^{t,l}_n\boldsymbol{W}^l_1 + \boldsymbol{b}^l_1)\boldsymbol{W}^l_2 + \boldsymbol{b}^l_2 + \boldsymbol{x}^{t,l}_n\end{aligned} \tag{17}$$

where $\boldsymbol{W}^{Q,l}_{i,n}$, $\boldsymbol{W}^{K,l}_{i,n} \in \mathbb{R}^{4d \times d_k}$, $\boldsymbol{W}^{V,l}_{i,n} \in \mathbb{R}^{4d \times d_v}$, $\boldsymbol{W}^{O,l}_n \in \mathbb{R}^{h \cdot d_v \times 4d}$, $\boldsymbol{W}^l_1 \in \mathbb{R}^{4d \times 16d}$, $\boldsymbol{W}^l_2 \in \mathbb{R}^{16d \times 4d}$, $\boldsymbol{b}^l_1 \in \mathbb{R}^{16d}$, $\boldsymbol{b}^l_2 \in \mathbb{R}^{4d}$ for the first l layer can be trained in the module parameters. We set $d_k = d_v = 4d/h$, where h is the number of attention heads. The first layer of the input is $\boldsymbol{Z}^{t,0}_n = \boldsymbol{Z}^t_n \in \mathbb{R}^{(a^t_{u,n}+a^t_{v,n}) \times 4d}$, The final output is $\boldsymbol{H}^t_n = \boldsymbol{Z}^{t,L}_n \in \mathbb{R}^{(a^t_{u,n}+a^t_{v,n}) \times 4d}$.

4.2.4. Output layer

In the proposed model, we compute the average of the embedding representation $\boldsymbol{H}^t_n$ in the output layer. Subsequently, we concatenate the features from the two patches and input them into a linear layer that yields node embeddings for nodes u and v at timestamp t.

$$\boldsymbol{h}_{u,i} = Avg(\boldsymbol{H}^t_i[:\, \boldsymbol{a}^t_{u,i}, :]) \tag{18}$$

$$\boldsymbol{h}_{v,i} = Avg(\boldsymbol{H}^t_i[\boldsymbol{a}^t_{u,i} :\, \boldsymbol{a}^t_{u,i} + \boldsymbol{a}^t_{v,i}, :]) \tag{19}$$

$$\boldsymbol{h}^t_u = Linear(\boldsymbol{h}_{u,1} \| \boldsymbol{h}_{u,2}), \boldsymbol{h}^t_v = Linear(\boldsymbol{h}_{v,1} \| \boldsymbol{h}_{v,2}) \tag{20}$$

$\boldsymbol{h}^t_u$, $\boldsymbol{h}^t_u \in \mathbb{R}^{d_{out}}$, d_{out} is the dimension of the output node.

4.3. Link prediction layer

The concatenated representations of node pairs are processed using two multilayer perceptions coupled with a rectified linear unit activation function. Ultimately, we employ a softmax function to output the predicted probability distribution $\hat{y}$.

$$\hat{y} = Softmax(MLP(RELU(MLP(\boldsymbol{h}^t_u \| \boldsymbol{h}^t_v)))) \tag{21}$$

4.4. Loss function

Finally, for the loss function of the link prediction task, we use the commonly used binary cross-entropy loss function, which is defined as follows:

$$\mathcal{L} = -\frac{1}{M}\sum_{i=1}^{I}(y_i * log\hat{y}_i + (1 - y_i) * log(1 - \hat{y}_i)) \tag{22}$$

where M is the total number of samples, y_i is the actual label of the i th sample and $\hat{y}_i$ represents the predicted value.

5. Experiments

5.1. Research questions

To evaluate the performance of the MTdyg model, we addressed the following three research questions:

(RQ1) Can the proposed MTdyg model outperform competitive baseline models in the continuous dynamic graph link prediction domain?

(RQ2) Do the two innovative modules in the MTdyg model significantly contribute to its performance?

(RQ3) How does the performance of the MTdyg model fluctuate for different time decay rates α and hyperparameter values β?

5.2. Datasets

In our work, we use eight real-world public datasets: Wikipedia, Reddit, MOOC, Enron, UCI, US Legis., UN Trade, Contact.

- **Wikipedia** is a dichotomous interaction graph that contains the interactions between users and Wikipedia pages over a month.
- **Reddit** is a binary dataset that includes all posts made by users within a month.
- **MOOC** is a two-way interactive network of online resources containing data on student access to courses over a year and a half.
- **Enron** is an email communications dataset containing over three years of email communications data from Enron Energy employees.
- **UCI** is an online communication network that contains 196 days of online communication data.
- **US Legis.** is a legislation-related network sponsored by the U.S. Senate that tracks the social activities of members of the Senate.
- **UN Trade** is a dynamic trade network that has preserved more than 30 years of trade interactions between countries.
- **Contact** is a dataset that introduces the distance changes between individuals within a month, where the weights on the edges represent the proximity of the distances online.

The detailed statistical characteristics of the datasets are delineated in Table 1. For enhanced efficiency in model training, we divided each dataset sequentially according to time, allocating 70% for the training set, 15% for the validation set, and 15% for the test set.

Table 1
Statistics of the datasets.

Datasets	Nodes	Links	N/L Feat	Duration	Time Granularity
Wiki	9,227	157,474	_/172	1 month	Unix timestamps
Reddit	10,984	672,447	_/172	1 month	Unix timestamps
MOOC	7,144	411,749	_/4	17 months	Unix timestamps
Enron	184	125,235	_/_	3 years	Unix timestamps
UCI	1,899	59,835	_/_	196 days	Unix timestamps
US Legis.	225	60,396	_/1	12 congresses	congresses
UN Trade	255	507,497	_/1	32 years	years
Contact	692	2,426,279	_/1	1 month	5 min

5.3. Model summary

We compared MTdyg with nine popular baselines in the field of continuous dynamic graph link prediction, including JODIE [31], DyRep [12], TGAT [10], TGN [11], CAWN [32], TCL [33], GraphMixer [34] and DyGFormer [4].

- **JODIE:** A dynamic user–item interaction embedding model that utilises a unique RNN architecture to handle time-series data, effectively addressing the cold start problem.
- **DyRep:** Introduces a temporal attention aggregation module to extract structural information from dynamic graphs as they evolve over time.
- **TGAT:** Employs a time-aware attention mechanism to learn the temporal influence of nodes, thereby capturing the dynamics and trends in graphs with greater precision.
- **TGN:** Combines memory mechanisms with graph neural networks to effectively capture the structural information of graphs that changes over time.
- **CAWN:** Uses temporal random walks to extract temporal graph information for representing dynamic networks.
- **TCL:** Applies a graph transformer that integrates topological and temporal information for node representation, with a cross-attention mechanism to simulate dependencies between interacting nodes.
- **GraphMixer:** Leverages an MLP-Mixer encoder to learn temporal link features and uses a neighbour mean strategy to aggregate node features.
- **DyGFormer:** Utilises historical neighbour information and employs patch techniques in sequence processing.

5.4. Experimental setup and evaluation metrics

To assess the performance of our models, we employed average precision (AP) and area under receiver operating characteristic curve (AUC-ROC) as evaluation metrics. We conducted experiments on the link prediction task under two distinct experimental settings: the transductive setting, which primarily predicts future links between nodes observed during the training process, and the inductive setting, which predicts future links between nodes not seen during training. To comprehensively compare the performance across different models, we evaluated the models using three negative sampling strategies: random, historical and inductive [35].

Our experimental protocol for all models involved 100 iterations, with an early stopping patience mechanism after 20 epochs to prevent overfitting. The models' performance was evaluated on the validation set, and the model with the highest efficacy was advanced to the final testing phase. The optimisation was conducted using the Adam algorithm with the learning rate and batch size preset to 1e-3 and 200, respectively. To ensure robustness and reduce bias, each model was subjected to five runs across different datasets using distinct random seeds, and the mean performance was computed and reported. The hyperparameters within each baseline model adhered to the configurations specified in the respective original studies. For our MTdyg model, we determined the temporal dimension d_T, historical common neighbour embedding dimension d_F and alignment dimension d at a value of 50, and the output node feature dimension was set to 172. The transformer module was configured with two layers L and two attention heads h. The decay rate α and hyperparameter β are reserved for a dedicated sensitivity analysis section. Experiments were executed on an NVIDIA 3090Ti 24 GB GPU.

6. Results and discussion

6.1. Overall performance

The performance evaluations of the proposed MTdyg model, within the context of continuous dynamic graphs for both transductive and inductive tasks, are systematically listed in Tables 2 and 3 compared with the established baselines. To elucidate the merits of the proposed MTdyg model, the results obtained using three different negative sampling strategies are presented. As indicated in the tables, the MTdyg model demonstrated superior performance in most tested scenarios.

Particular attention is given to the DyGFormer model, which proposes a basic scheme for encoding neighbour contributions and employs a technique to partition interaction sequences into patches to capture long-term dependencies. The experimental results suggest that outstrips previous models in several dimensions, confirming the utility of the proposed components. Notwithstanding, it is evident that the model's performance significantly falters when faced with more rigorous sampling strategies. This highlights the procedural limitations of the DyGFormer approach, where neighbour co-occurrence encoding is based solely on a simplistic counting method, and patch information is confined to a single scale. In stark contrast, the proposed approach not only accounts for the differential significance of historical neighbour interactions but also integrates information from patches across multiple scales and delves into the optimal selection of these scales, substantially improving model performance.

Subsequently, we compared the training times of the models (Fig. 3). Our experiments were conducted on the WIKI and MOOC datasets. In this figure, the x-axis denotes the training time, the y-axis represents the AP, bubble size indicates the number of model parameters and bubble colour signifies the overall ranking of the models. The results demonstrate that although the JODIE model has few parameters and trains quickly, it exhibits suboptimal performance. In contrast, the CAWN model's extensive parameters resulted in significantly slower training. The proposed MTdyg model, with a moderate number of parameters, achieves optimal performance in a comparatively shorter training time.

6.2. Ablation studies

We conducted ablation experiments on six datasets to ascertain the practicality of the two modules proposed in our MTdyg model: the historical common neighbours encoding method based on time-aware attention mechanisms and the multi-scale patch technique. Specifically, we compared the performance of MTdyg against three of its variants—MTdyg without the time-aware historical common neighbour encoding method $\text{MTdyg}_{w/o[T-HCN]}$, without the multi-scale patch technique $\text{MTdyg}_{w/o[MPT]}$ and without both components $\text{MTdyg}_{w/o[ALL]}$. In addition, to validate the effectiveness of the proposed frequency-adaptive patch size selection method, we substitute it with a random patch size selection approach, which is denoted as MTdyg_{random}. The outcomes of MTdyg and its four variants are illustrated in Fig. 4. As can be seen, MTdyg consistently outperforms $\text{MTdyg}_{w/o[T-HCN]}$, $\text{MTdyg}_{w/o[MPT]}$ and $\text{MTdyg}_{w/o[ALL]}$ across all six datasets, demonstrating that both time-aware attention-based historical common neighbour encoding and the multi-scale patch technique enhance the model's link prediction performance. Furthermore, across all datasets, removing the time-aware historical common neighbour encoding method

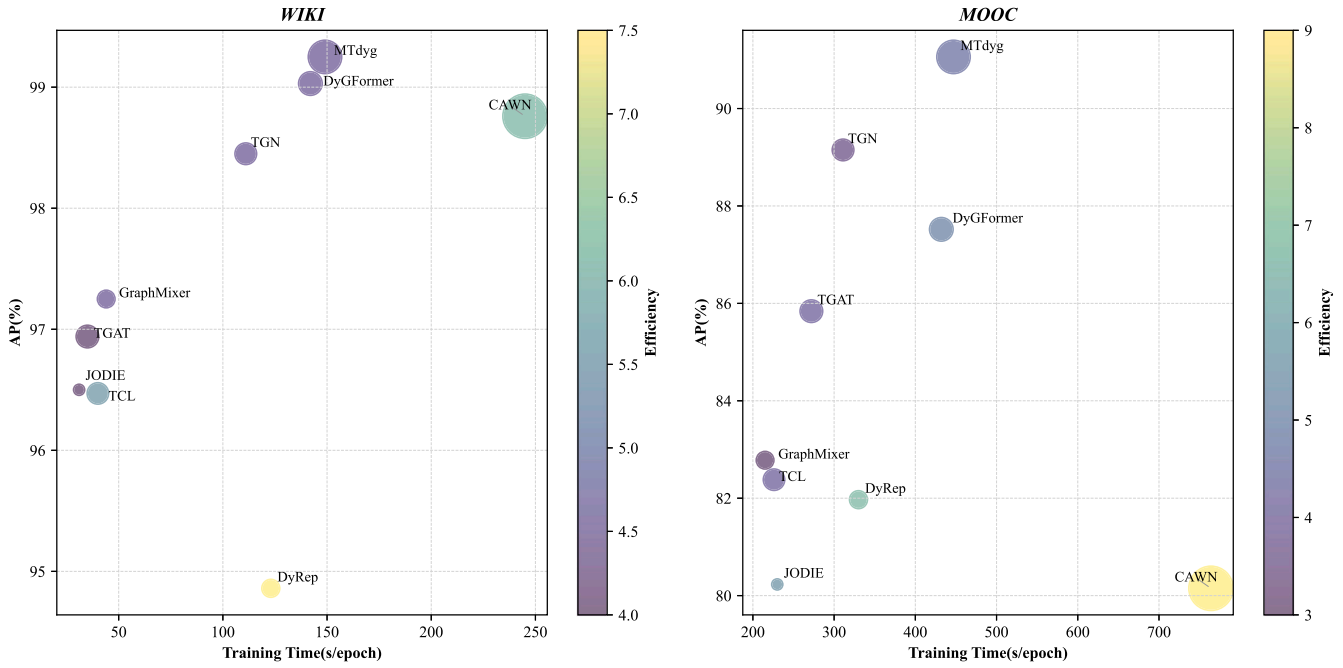


Fig. 3. Comparison of model training efficiency.

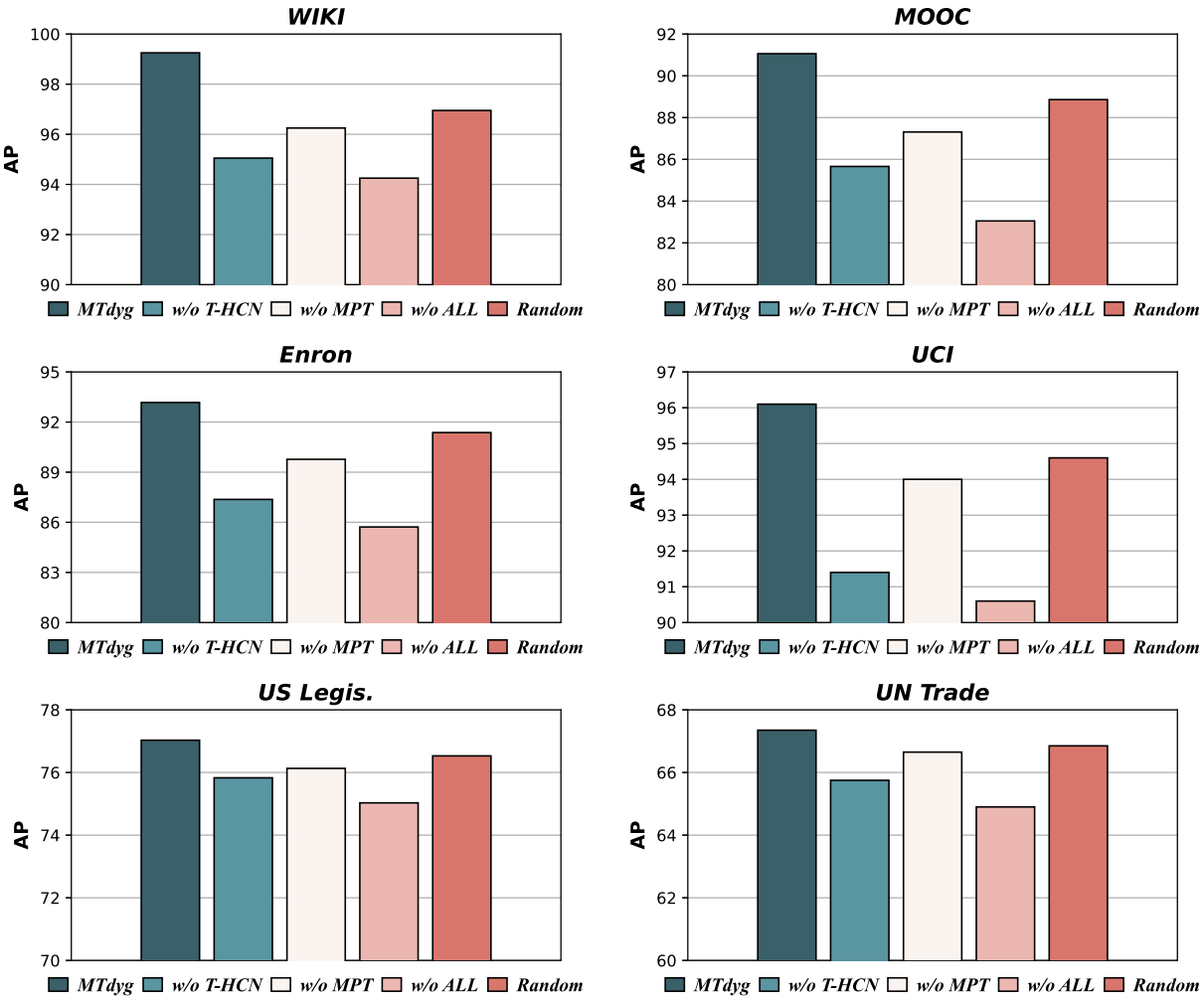


Fig. 4. Ablation study of MTdyg under random negative sampling setting, where w/o NS and w/o FE represent our MTdyg without Time-aware Historical Common Neighbours encoding module and Multi-Scale Patch Technique.

Table 2
AP for transductive link prediction with random, historical and inductive negative sampling strategies.

NSS	Datasets	Wikipedia	Reddit	MOOC	Enron	UCI	US Legis.	UN Trade	Contact
rnd	JODIE	96.50 ± 0.14	98.31 ± 0.14	80.23 ± 2.44	84.77 ± 0.30	89.43 ± 1.09	75.05 ± 1.52	64.94 ± 0.31	95.31 ± 1.33
	DyRep	94.86 ± 0.06	98.22 ± 0.04	81.97 ± 0.49	82.38 ± 3.36	65.14 ± 2.30	75.34 ± 0.39	63.21 ± 0.93	95.98 ± 0.15
	TGAT	96.94 ± 0.06	98.52 ± 0.02	85.84 ± 0.15	71.12 ± 0.97	79.63 ± 0.70	68.52 ± 3.16	61.47 ± 0.18	96.28 ± 0.09
	TGN	98.45 ± 0.06	98.63 ± 0.06	89.15 ± 1.60	86.53 ± 1.11	92.34 ± 1.04	75.99 ± 0.58	65.03 ± 1.37	96.89 ± 0.56
	CAWN	98.76 ± 0.03	99.11 ± 0.01	80.15 ± 0.25	89.56 ± 0.09	95.18 ± 0.06	70.58 ± 0.48	65.39 ± 0.12	90.26 ± 0.28
	TCL	96.47 ± 0.16	97.53 ± 0.02	82.38 ± 0.24	79.70 ± 0.71	89.57 ± 1.63	69.59 ± 0.48	62.21 ± 0.03	92.44 ± 0.12
	GraphMixer	97.25 ± 0.03	97.31 ± 0.01	82.78 ± 0.15	82.25 ± 0.16	93.25 ± 0.57	70.74 ± 1.02	62.61 ± 0.27	91.92 ± 0.03
	DyGFormer	99.03 ± 0.02	99.22 ± 0.01	87.52 ± 0.49	92.47 ± 0.12	95.79 ± 0.17	71.11 ± 0.59	66.46 ± 1.29	98.29 ± 0.01
	MTdyg	**99.25 ± 0.02**	**99.34 ± 0.03**	**91.06 ± 0.08**	**93.17 ± 0.50**	**96.10 ± 0.02**	**77.03 ± 1.30**	**67.35 ± 0.10**	**99.04 ± 0.05**
hist	JODIE	83.01 ± 0.66	80.03 ± 0.36	78.94 ± 1.25	69.85 ± 2.70	75.24 ± 5.80	51.71 ± 5.76	61.39 ± 1.83	95.31 ± 2.13
	DyRep	79.93 ± 0.56	79.83 ± 0.31	75.60 ± 1.12	71.19 ± 2.76	55.10 ± 3.14	86.88 ± 2.25	59.19 ± 1.07	96.39 ± 0.20
	TGAT	87.38 ± 0.22	79.55 ± 0.20	82.19 ± 0.62	64.07 ± 1.05	68.27 ± 1.37	62.14 ± 6.60	55.74 ± 0.91	96.05 ± 0.52
	TGN	86.86 ± 0.33	81.22 ± 0.61	87.06 ± 1.93	73.91 ± 1.76	80.43 ± 2.12	74.00 ± 7.57	58.44 ± 5.51	93.05 ± 2.35
	CAWN	71.21 ± 1.67	80.82 ± 0.45	74.05 ± 0.95	64.73 ± 0.36	65.30 ± 0.43	68.82 ± 8.23	55.71 ± 0.38	84.16 ± 0.49
	TCL	89.05 ± 0.39	77.14 ± 0.16	77.06 ± 0.41	70.66 ± 0.39	80.25 ± 2.74	80.53 ± 3.95	55.90 ± 1.17	93.86 ± 0.21
	GraphMixer	**90.90 ± 0.10**	78.44 ± 0.18	77.77 ± 0.92	**77.98 ± 0.92**	84.11 ± 1.35	81.65 ± 1.02	57.05 ± 1.22	93.36 ± 0.41
	DyGFormer	82.23 ± 2.54	81.57 ± 0.67	85.85 ± 0.66	75.63 ± 0.73	82.17 ± 0.82	85.30 ± 3.88	64.41 ± 1.40	97.57 ± 0.06
	MTdyg	89.50 ± 0.51	**82.19 ± 0.54**	**87.89 ± 0.38**	76.68 ± 1.24	**85.59 ± 0.59**	**87.02 ± 0.28**	**69.97 ± 0.25**	**97.92 ± 0.13**
ind	JODIE	75.65 ± 0.79	86.98 ± 0.16	65.23 ± 2.19	68.96 ± 0.98	65.99 ± 1.40	50.27 ± 5.13	60.42 ± 1.48	93.43 ± 1.78
	DyRep	70.21 ± 1.58	86.30 ± 0.26	61.66 ± 0.95	67.79 ± 1.53	54.79 ± 1.76	83.44 ± 1.16	60.19 ± 1.24	94.18 ± 0.10
	TGAT	87.00 ± 0.16	89.59 ± 0.24	75.95 ± 0.64	63.94 ± 1.36	68.67 ± 0.84	61.91 ± 5.82	60.61 ± 1.24	94.35 ± 0.48
	TGN	85.62 ± 0.44	88.10 ± 0.24	77.50 ± 2.91	70.89 ± 2.72	70.94 ± 0.71	67.57 ± 6.47	61.04 ± 6.01	90.18 ± 3.28
	CAWN	74.06 ± 2.62	91.67 ± 0.24	73.51 ± 0.94	75.15 ± 0.58	64.61 ± 0.48	65.81 ± 8.52	62.54 ± 0.67	89.31 ± 0.27
	TCL	86.76 ± 0.72	87.45 ± 0.29	74.65 ± 0.54	71.29 ± 0.32	76.01 ± 1.11	78.15 ± 3.34	61.06 ± 1.74	91.35 ± 0.21
	GraphMixer	**88.59 ± 0.17**	85.26 ± 0.11	74.27 ± 0.92	75.01 ± 0.79	**80.10 ± 0.51**	79.63 ± 0.84	60.15 ± 1.29	90.87 ± 0.35
	DyGFormer	78.29 ± 5.38	91.11 ± 0.40	81.24 ± 0.69	77.41 ± 0.89	72.25 ± 1.71	81.25 ± 3.62	55.79 ± 1.02	94.75 ± 0.28
	MTdyg	79.84 ± 3.36	**92.35 ± 0.61**	**82.65 ± 0.48**	**78.17 ± 0.33**	77.70 ± 0.97	**84.51 ± 1.87**	**68.21 ± 0.47**	**95.86 ± 0.79**

Table 3
AP for inductive link prediction with random, historical and inductive negative sampling strategies.

NSS	Datasets	Wikipedia	Reddit	MOOC	Enron	UCI	US Legis.	UN Trade	Contact
rnd	JODIE	94.82 ± 0.20	96.50 ± 0.13	79.63 ± 1.92	80.72 ± 1.39	79.86 ± 1.48	54.93 ± 2.29	59.65 ± 0.77	94.34 ± 1.45
	DyRep	92.43 ± 0.37	96.09 ± 0.11	81.07 ± 0.44	74.55 ± 3.95	57.48 ± 1.87	57.28 ± 0.71	57.02 ± 0.69	92.18 ± 0.41
	TGAT	96.22 ± 0.07	97.09 ± 0.04	85.50 ± 0.19	67.05 ± 1.51	79.54 ± 0.48	51.00 ± 3.11	61.03 ± 0.18	95.87 ± 0.11
	TGN	97.83 ± 0.04	97.50 ± 0.07	89.04 ± 1.17	77.94 ± 1.02	88.12 ± 2.05	58.63 ± 0.37	58.31 ± 3.15	93.82 ± 0.99
	CAWN	98.24 ± 0.03	98.62 ± 0.01	81.42 ± 0.24	86.35 ± 0.51	92.73 ± 0.06	53.17 ± 1.20	65.24 ± 0.21	89.55 ± 0.30
	TCL	96.22 ± 0.17	94.09 ± 0.07	80.60 ± 0.22	76.14 ± 0.79	87.36 ± 2.03	52.59 ± 0.97	62.21 ± 0.12	91.11 ± 0.12
	GraphMixer	96.65 ± 0.02	95.26 ± 0.02	81.41 ± 0.21	75.88 ± 0.48	91.19 ± 0.42	50.71 ± 0.76	62.17 ± 0.31	90.59 ± 0.05
	DyGFormer	98.59 ± 0.03	98.84 ± 0.02	86.96 ± 0.43	89.76 ± 0.34	94.54 ± 0.12	54.28 ± 2.87	64.55 ± 0.62	98.03 ±0.02
	MTdyg	**98.94 ± 0.04**	**99.15 ± 0.01**	**90.94 ± 0.15**	**90.29 ± 0.06**	**94.72 ± 0.01**	**59.64 ± 0.69**	**65.88 ± 0.30**	**98.96 ± 0.06**
hist	JODIE	68.69 ± 0.39	62.34 ± 0.54	63.22 ± 1.55	65.86 ± 3.71	63.11 ± 2.27	52.94 ± 2.11	55.46 ± 1.19	90.42 ± 2.34
	DyRep	62.18 ± 1.27	61.60 ± 0.72	62.93 ± 1.24	62.08 ± 2.27	52.47 ± 2.06	62.10 ± 1.41	55.49 ± 0.84	89.22 ± 0.66
	TGAT	84.17 ± 0.22	63.47 ± 0.36	76.73 ± 0.29	61.40 ± 1.31	70.52 ± 0.93	51.83 ± 3.95	55.28 ± 0.71	**94.15 ± 0.45**
	TGN	81.76 ± 0.32	64.85 ± 0.85	77.07 ± 3.41	62.91 ± 1.16	70.78 ± 0.78	61.18 ± 1.10	52.80 ± 3.19	88.13 ± 1.50
	CAWN	67.27 ± 1.63	63.67 ± 0.41	74.68 ± 0.68	60.70 ± 0.36	64.54 ± 0.47	55.56 ± 1.71	55.00 ± 0.38	74.20 ± 0.80
	TCL	82.20 ± 2.18	60.83 ± 0.25	74.27 ± 0.53	67.11 ± 0.62	76.71 ± 1.00	53.87 ± 1.41	55.76 ± 1.03	90.44 ± 0.17
	GraphMixer	**87.60 ± 0.30**	64.50 ± 0.26	74.00 ± 0.97	**72.37 ± 1.37**	**81.66 ± 0.49**	52.03 ± 1.02	54.94 ± 0.97	89.91 ± 0.36
	DyGFormer	71.42 ± 4.43	65.37 ± 0.60	80.82 ± 0.30	67.07 ± 0.62	72.13 ± 1.87	56.31 ± 3.46	53.20 ± 1.07	93.56 ± 0.52
	MTdyg	78.93 ± 2.07	**66.19 ± 0.43**	**81.84 ± 0.37**	68.26 ± 0.30	76.98 ± 0.05	**62.75 ± 1.12**	**61.32 ± 1.40**	93.76 ± 0.41
ind	JODIE	68.70 ± 0.39	62.32 ± 0.54	63.22 ± 1.55	65.86 ± 3.71	63.16 ± 2.27	52.94 ± 2.11	55.43 ± 1.20	90.43 ± 2.33
	DyRep	62.19 ± 1.28	61.58 ± 0.72	62.92 ± 1.24	62.08 ± 2.27	52.47 ± 2.09	62.10 ± 1.41	55.42 ± 0.87	89.22 ± 0.65
	TGAT	84.17 ± 0.22	63.40 ± 0.36	76.72 ± 0.30	61.40 ± 1.30	70.49 ± 0.93	51.83 ± 3.95	55.58 ± 0.68	**94.14 ± 0.45**
	TGN	81.77 ± 0.32	64.84 ± 0.84	77.07 ± 3.40	62.90 ± 1.16	70.73 ± 0.79	61.18 ± 1.10	52.80 ± 3.24	88.12 ± 1.50
	CAWN	67.24 ± 1.63	63.65 ± 0.41	74.69 ± 0.68	60.72 ± 0.36	64.54 ± 0.47	55.56 ± 1.71	54.97 ± 0.38	74.19 ± 0.81
	TCL	82.20 ± 2.18	60.81 ± 0.26	74.28 ± 0.53	67.11 ± 0.62	76.65 ± 0.99	53.87 ± 1.41	55.66 ± 0.98	90.43 ± 0.17
	GraphMixer	**87.60 ± 0.29**	64.49 ± 0.25	73.99 ± 0.97	**72.37 ± 1.38**	**81.64 ± 0.49**	52.03 ± 1.02	54.88 ± 1.01	89.91 ± 0.36
	DyGFormer	71.42 ± 4.43	65.35 ±0.60	80.82 ± 0.30	67.07 ± 0.62	72.13 ± 1.86	56.31 ± 3.46	52.56 ± 1.70	93.55 ± 0.52
	MTdyg	78.93 ± 2.07	**66.19 ± 0.43**	**80.86 ± 0.58**	68.26 ± 0.30	76.97 ± 0.06	**62.75 ± 1.12**	**60.98 ± 1.87**	93.74 ± 0.41

resulted in a more substantial performance reduction than removing the multi-scale patch technique. This shows that effectively utilising historical common neighbour interaction information in dynamic graphs significantly enhances the model's efficacy.

Subsequently, a comparative analysis between MTdyg and $\text{MTdyg}_{w/o[T-HCN]}$ reveals a significantly steeper decline in performance on datasets such as WIKI, MOOC, Enron and UCI relative to UN Trade and US Legislative datasets following the omission of the time-attention-based historical common neighbour encoding. This discrepancy is likely due to the greater frequency and complexity of interactions in the WIKI, MOOC, Enron and UCI datasets, which exhibit a higher prevalence of historical common neighbour links within their interaction networks. Therefore, the encoding module extracts more substantial and impactful information from these datasets, which enhances its effectiveness. In contrast, the UN Trade and US Legislative datasets, which have more consistent, less frequent interactions and fewer repetitive links, demonstrate a more muted benefit from the proposed module.

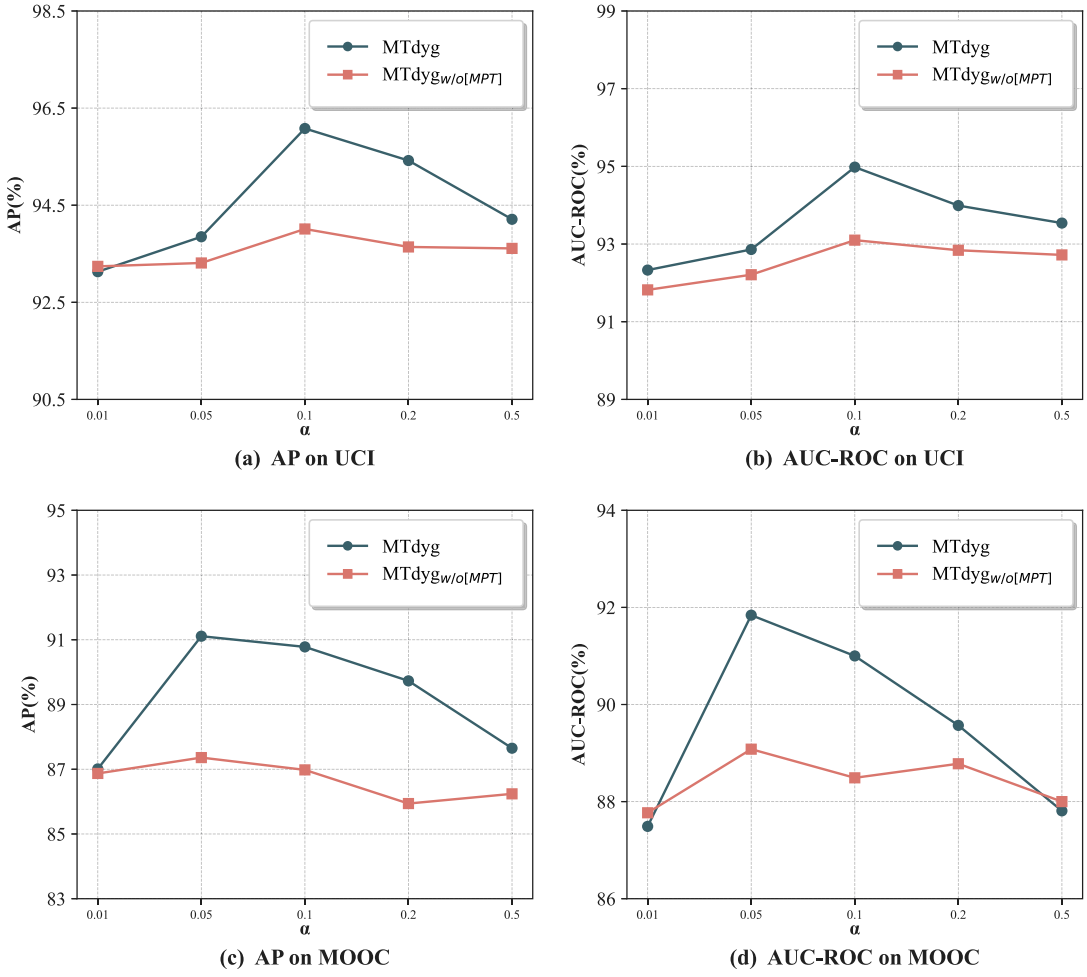


Fig. 5. Model performance with different α in the T-HCN.

In addition, the comparison between MTdyg$_{random}$ and MTdyg$_{w/o[MPT]}$ indicates the effectiveness of the random patch size selection method in capturing multi-scale information. However, the frequency-adaptive method for selecting patch sizes surpasses the random approach because MTdyg significantly outperforms MTdyg$_{random}$. This superior performance is attributable to MTdyg's use of the Fourier transform to identify periodic patterns in frequency-domain data to generate patch sizes, which is inherently more efficient than random selection because it integrates more contextual data.

6.3. Hyperparameter study

In this section, we describe experiments conducted to investigate the influence of the pivotal hyperparameters on the performance of the MTdyg model. This subsection includes the decay factor α implemented in the historical common neighbour encoding strategy based on temporal attention and the balancing parameter β introduced in the multi-scale patch technique. The impact of sampling neighbours on the MTdyg model was also evaluated. All experiments were executed in a transductive setting employing AP and AUC-ROC as experimental benchmarks.

6.3.1. Impact of the decay factor α

To ascertain the influence of the decay factor α on the efficiency of our models, we adjusted the value of α in both MTdyg and the pared-down MTdyg$_{w/o[MPT]}$, selecting from a set of prospective values: {0.01, 0.05, 0.1, 0.2, 0.5}. The subsequent effects on the metrics of AP and AUC-ROC were monitored. The outcomes of these empirical trials are presented in the charts provided. Fig. 5 reveals that on the test datasets, MTdyg frequently outperforms MTdyg$_{w/o[MPT]}$ across the spectrum of α values, validating the efficacy of our adaptive multi-scale module in capturing and leveraging multi-scale information to boost performance.

Regarding the UCI dataset, we found that both models' performance improved as α increased, climaxing at an α of 0.1, after which performance gradually diminished. This indicates that a lower α did not sufficiently encapsulate the diminishing nature of historical interactions over time. Conversely, a higher α could disproportionately highlight recent interactions, disregarding earlier but potentially crucial interactions. An optimal, moderate α appears to strike a balance between these extremes, using historical interactions to refine predictions of future engagements.

This pattern persisted in the MOOC dataset, where a similar trend was detected. The subsequent analysis suggested that the ideal α value might shift depending on the temporal window size. For shorter windows, higher (α might be necessary to underscore recent interactions given the diminished relevance of older interactions to future outcomes. In contrast, a lower α could be more beneficial in longer windows because it allows the model to consider older neighbour interactions that may still hold significance for future interaction predictions.

6.3.2. Impact of the parameter β

To elucidate the influence of the hyperparameter β on our model's link prediction performance, we adjusted β within the MTdyg framework and its variant, MTdyg$_{w/o[T-HCN]}$, which lacks the time-aware

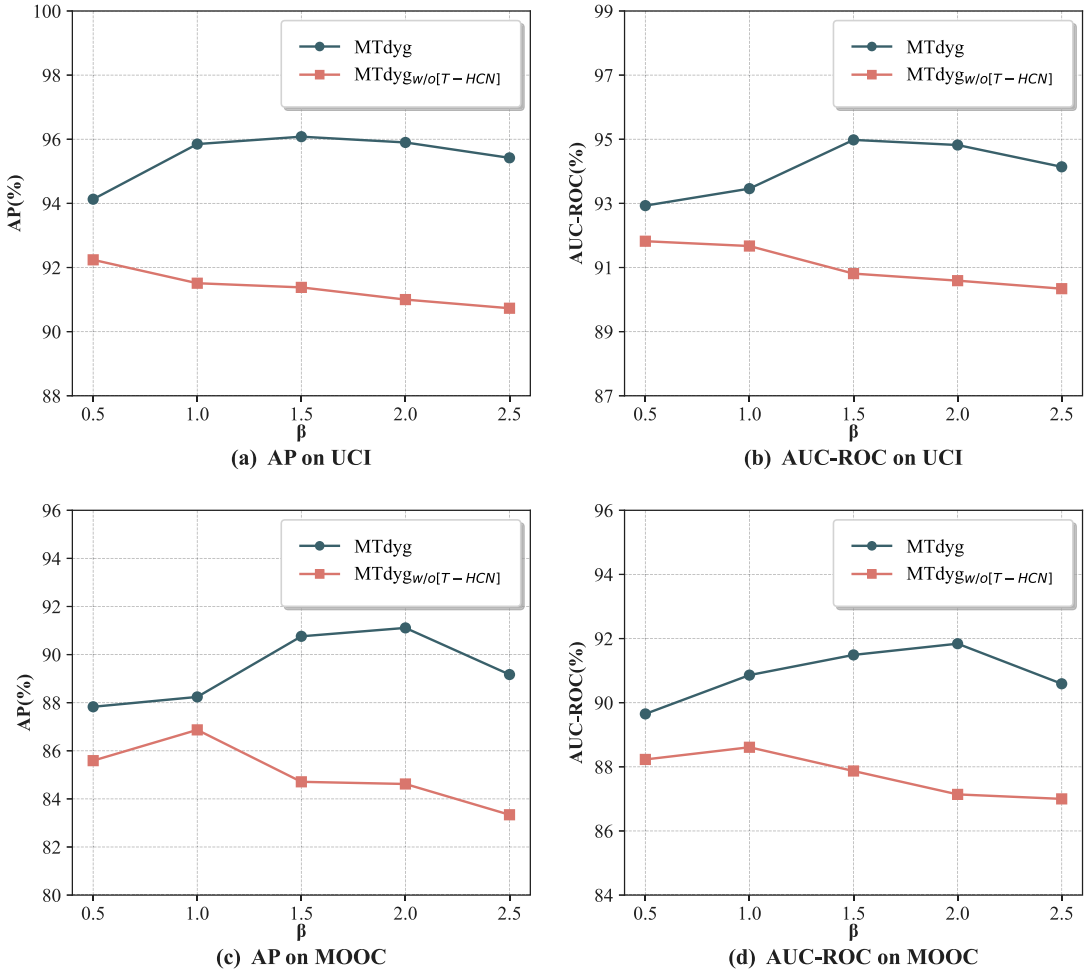


Fig. 6. Model performance with different β in the MPT.

historical common neighbour encoding module based on the attention mechanism. We selected from a set of potential values: {0.5, 1.0, 1.5, 2.0, 2.5}, to observe how the hyperparameter β affects link prediction performance both with and without the time-aware attention mechanism. The results obtained on the UCI and MOOC datasets are illustrated in Fig. 6. As shown in the figure, MTdyg outperforms its variant, MTdyg$_{w/o[T-HCN]}$, across most values of β, which underscores the efficacy of the time-aware attention-based historical common neighbour encoding in extracting crucial information about the varying significance of historical interactions between node pairs, thereby enhancing performance.

Moreover, on the UCI dataset, the performance of MTdyg in terms of AP and AUC-ROC initially increased with β before declining, which can be attributed to the model overfitting the training data when β is set too low, resulting in an excessively small patch size. Therefore, a medium value of $\beta = 1.5$ yields the optimal performance. In contrast, the performance of MTdyg$_{w/o[T-HCN]}$ on both metrics consistently declined as β increases. The divergent trend between MTdyg and MTdyg$_{w/o[T-HCN]}$ may stem from the time-aware attention-based encoding in MTdyg, which sufficiently learns precise interaction feature representations, thus obviating the need for more local features to capture finer-grained patterns. For MTdyg$_{w/o[T-HCN]}$, more local feature information is required. A larger patch size can provide broader contextual information; however, it can also lead to underfitting and subsequent performance degradation in models where information acquisition is insufficient.

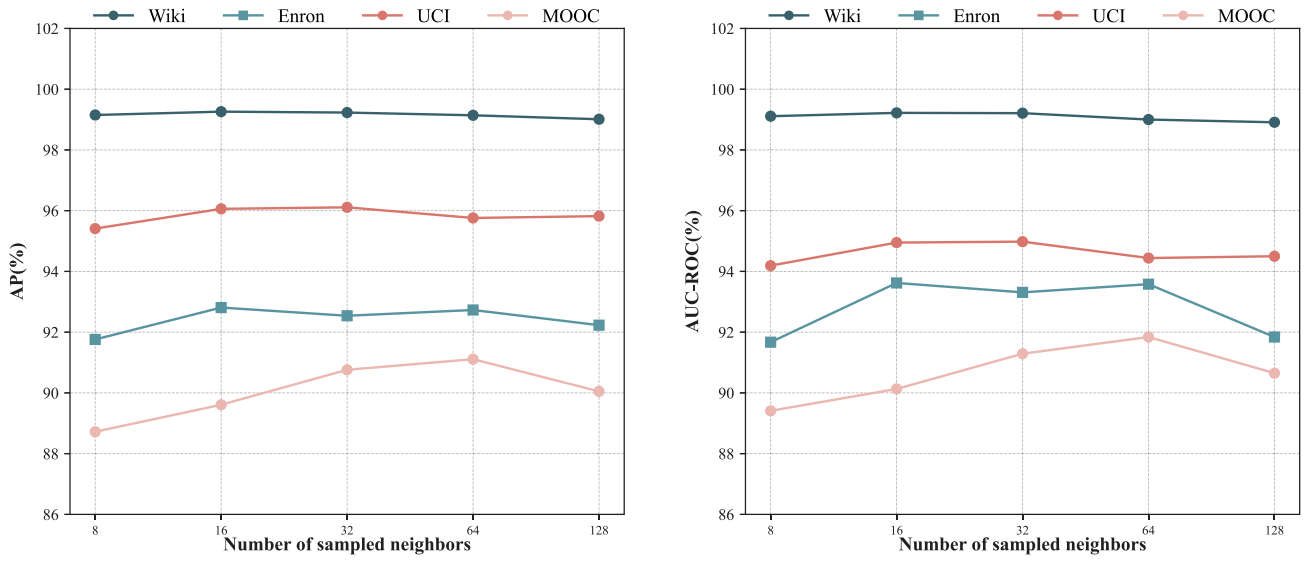
6.3.3. *Impact of the number of neighbour samples L*

To assess the impact of the number of sampled neighbours L on the MTdyg model's efficacy, we adjusted (L) within the proposed MTdyg, choosing from {8, 16, 32, 64, 128} as potential values. The effects on model performance are illustrated in Fig. 7. Our preliminary analysis indicates that a small number of sampled neighbours generally yield substandard performance, presumably due to the failure to encapsulate sufficient local graph structural information. In addition, the neighbour sampling count corresponding to the peak performance differs across different datasets.

In detail, limited sampling constrains the model's capabilities because it likely does not encompass ample neighbourhood information. An increment in sampled neighbours to a moderate quantity markedly enhances the model's efficacy, suggesting that an expanded neighbour base aids in a more accurate representation of node interactions and graph structure intricacies. A further increase in sampled neighbours leads to a performance plateau or a minor decrease attributable to the superfluous information, which computationally burdens the model without commensurate performance improvements. In particular, for datasets with intricate and frequent interactions, a more substantial neighbour sampling is required for the best model performance.

7. Conclusion

This research presents a new model for continuous dynamic graph link prediction called the MTdyg model. This model improves the efficiency of leveraging historical interaction data by employing a time-aware encoding mechanism for common historical neighbours and a



(a) AP on Datasets (b) AUC-ROC on Datasets

Fig. 7. Performance of MTdyg models with different sampled neighbour numbers.

multi-scale patch approach. The essential feature of the MTdyg model is its proficiency in capturing enduring interaction patterns among nodes and its systematic exploitation of multi-scale information, which has led to superior link prediction performance on various public datasets. Future research directions may include refining the computational efficiency of the model and augmenting it to manage sophisticated dynamic graph data, such as accommodating incremental updates and executing real-time dynamic graph processing.

CRediT authorship contribution statement

Long Xu: Writing – review & editing, Writing – original draft, Visualization, Software, Methodology, Conceptualization. **Zhiqiang Pan:** Software, Methodology, Conceptualization. **Honghui Chen:** Supervision, Resources, Investigation, Formal analysis, Data curation. **Shen Wang:** Visualization, Software, Funding acquisition, Formal analysis.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data used in this study are publicly available.

Appendix A. Configurations of baselines

All of our baseline models were run on an NVIDIA 3090Ti 24 GB GPU, utilising Python version 3.8.10 and PyTorch version 1.9.0+cuda 11.1. We set the temporal encoding dimension to 100 and the output encoding dimension to 172 for all models. For JODIE, DyRep, TGAT, and TGN, the node memory dimension was set to 172. Models using multi-head attention mechanism had the number of heads set to 2. In DyRep and TGN, the number of graph convolution layers was set to 1, while in TGAT it was set to 2. The positional encoding in CAWN and the depth encoding in TCL were both set to 172. The number of Transformer encoder layers in the baselines was set to 2. In GraphMixer, the number of MLP-Mixer encoder layers was set to 2, and the time lag was set to 2000. In DyGFormer, both the common neighbour encoding dimension and the encoding alignment dimension were set to 50. We ensured that all baseline model hyperparameters found in the original text were consistently maintained.

Appendix B. The pseudo-code of MTdyg

Following the suggestion of the reviewers, we briefly describe the procedure of MTdyg in Algorithm 1.

Algorithm 1 Training Pipeline for MTdyg

Require: Graph G, Node pair (u, v), Timestamp t, Neighbor sample number L
Ensure: Probability of interaction $\hat{y}$
Initialization: Initialize model parameters Θ
Input: Graph G, Node pair (u, v), Timestamp t, Neighbor sample number L
for each epoch **do**
  for each batch (u, v, t) **do**
    Construct $\boldsymbol{I}_u^t$ and $\boldsymbol{I}_v^t$ for node u and v
    $\boldsymbol{X}_{u,T}^t = \text{TimeEncoder}(\boldsymbol{I}_u^t)$, $\boldsymbol{X}_{v,T}^t = \text{TimeEncoder}(\boldsymbol{I}_v^t)$
    $\boldsymbol{X}_{u,H}^t = \text{NodeEncoder}(\boldsymbol{I}_u^t)$, $\boldsymbol{X}_{v,H}^t = \text{NodeEncoder}(\boldsymbol{I}_v^t)$
    $\boldsymbol{X}_{u,E}^t = \text{EdgeEncoder}(\boldsymbol{I}_u^t)$, $\boldsymbol{X}_{v,E}^t = \text{EdgeEncoder}(\boldsymbol{I}_v^t)$
    $\boldsymbol{X}_{u,F}^t = \text{HCNEncoder}(\boldsymbol{I}_u^t)$, $\boldsymbol{X}_{v,F}^t = \text{HCNEncoder}(\boldsymbol{I}_v^t)$
    $T_1, T_2 = \text{FourierTransform}(\boldsymbol{X}_{u,T}^t, \boldsymbol{X}_{v,T}^t)$
    $S_1 = \lceil \beta \cdot T_1 \rceil$, $S_2 = \lceil \beta \cdot T_2 \rceil$
    $\boldsymbol{P}_{u,T,1}^t = \text{Patch}(\boldsymbol{X}_{u,T}^t, S_1)$, $\boldsymbol{P}_{u,T,2}^t = \text{Patch}(\boldsymbol{X}_{u,T}^t, S_2)$
    $\boldsymbol{P}_{v,T,1}^t = \text{Patch}(\boldsymbol{X}_{v,T}^t, S_1)$, $\boldsymbol{P}_{v,T,2}^t = \text{Patch}(\boldsymbol{X}_{v,T}^t, S_2)$
    // Similarly patch $\boldsymbol{X}_{u,H}^t$, $\boldsymbol{X}_{u,E}^t$, $\boldsymbol{X}_{u,F}^t$, $\boldsymbol{X}_{v,H}^t$, $\boldsymbol{X}_{v,E}^t$, $\boldsymbol{X}_{v,F}^t$
    $\boldsymbol{Z}_n^t = \text{TransformerEncoder}(\boldsymbol{P}_{u,T,n}^t, \boldsymbol{P}_{v,T,n}^t, \ldots)$
    $\boldsymbol{h}_u^t = \text{Avg}(\boldsymbol{Z}_n^t[: a_{u,n}^t, :])$, $\boldsymbol{h}_v^t = \text{Avg}(\boldsymbol{Z}_n^t[a_{u,n}^t : a_{u,n}^t + a_{v,n}^t, :])$
    $\boldsymbol{h}_u^t = \text{Linear}(\boldsymbol{h}_{u,1} \parallel \boldsymbol{h}_{u,2})$, $\boldsymbol{h}_v^t = \text{Linear}(\boldsymbol{h}_{v,1} \parallel \boldsymbol{h}_{v,2})$
    $\hat{y} = \text{Softmax}(\text{MLP}(\text{RELU}(\text{MLP}(\boldsymbol{h}_u^t \parallel \boldsymbol{h}_v^t))))$
    $\mathcal{L} = \text{BinaryCrossEntropy}(\hat{y}, y)$
    Update parameters Θ using gradients
  end for
end for
Return: $\hat{y}$

Table 4
AUC-ROC for transductive link prediction with random, historical and inductive negative sampling strategies.

NSS	Datasets	Wikipedia	Reddit	MOOC	Enron	UCI	US Legis.	UN Trade	Contact
rnd	JODIE	96.33 ± 0.07	98.31 ± 0.05	83.81 ± 2.09	87.96 ± 0.52	90.44 ± 0.49	82.85 ± 1.07	69.62 ± 0.44	96.66 ± 0.89
	DyRep	94.37 ± 0.09	98.17 ± 0.05	85.03 ± 0.58	84.89 ± 3.00	68.77 ± 2.34	82.28 ± 0.32	67.44 ± 0.83	96.48 ± 0.14
	TGAT	96.67 ± 0.07	98.47 ± 0.02	87.11 ± 0.19	68.89 ± 1.10	78.53 ± 0.74	75.84 ± 1.99	64.01 ± 0.12	96.95 ± 0.08
	TGN	98.37 ± 0.07	98.60 ± 0.06	91.21 ± 1.15	88.32 ± 0.99	92.03 ± 1.13	**83.34 ± 0.43**	69.10 ± 1.67	97.54 ± 0.35
	CAWN	98.54 ± 0.04	99.01 ± 0.01	80.38 ± 0.26	90.45 ± 0.14	93.87 ± 0.08	77.16 ± 0.39	68.54 ± 0.18	89.99 ± 0.34
	TCL	95.84 ± 0.18	97.42 ± 0.02	83.12 ± 0.18	75.74 ± 0.72	87.82 ± 1.36	76.27 ± 0.63	64.72 ± 0.05	94.15 ± 0.09
	GraphMixer	96.92 ± 0.03	97.17 ± 0.02	84.01 ± 0.17	84.38 ± 0.21	91.81 ± 0.67	76.96 ± 0.79	65.52 ± 0.51	93.94 ± 0.02
	DyGFormer	98.91 ± 0.02	99.15 ± 0.01	87.91 ± 0.58	93.33 ± 0.13	94.49 ± 0.26	77.90 ± 0.58	70.20 ± 1.44	98.53 ± 0.01
	MTdyg	**99.14 ± 0.02**	**99.31 ± 0.03**	**91.95 ± 0.16**	**94.04 ± 0.48**	**95.12 ± 0.04**	83.26 ± 1.37	**71.19 ± 0.28**	**99.26 ± 0.05**
hist	JODIE	80.77 ± 0.73	80.52 ± 0.32	82.75 ± 0.83	75.39 ± 2.37	78.64 ± 3.50	67.47 ± 6.40	68.92 ± 1.40	96.35 ± 0.92
	DyRep	77.74 ± 0.33	80.15 ± 0.18	81.06 ± 0.94	74.69 ± 3.55	57.91 ± 3.12	91.44 ± 1.18	64.36 ± 1.40	96.00 ± 0.23
	TGAT	82.87 ± 0.22	79.33 ± 0.16	80.81 ± 0.67	61.85 ± 1.43	58.89 ± 1.57	73.47 ± 5.25	60.37 ± 0.68	95.39 ± 0.43
	TGN	82.74 ± 0.32	81.11 ± 0.19	**88.00 ± 1.80**	77.09 ± 2.22	77.25 ± 2.68	83.53 ± 4.53	63.93 ± 5.41	93.76 ± 1.29
	CAWN	67.84 ± 0.64	80.27 ± 0.30	71.57 ± 1.07	65.10 ± 0.34	57.86 ± 0.15	78.62 ± 7.46	63.09 ± 0.74	83.06 ± 0.32
	TCL	85.76 ± 0.46	76.49 ± 0.16	72.09 ± 0.56	67.95 ± 0.88	72.25 ± 3.46	83.97 ± 3.71	61.43 ± 1.04	93.34 ± 0.19
	GraphMixer	**87.68 ± 0.17**	77.80 ± 0.12	76.68 ± 1.40	75.27 ± 1.14	77.54 ± 2.02	85.17 ± 0.70	63.20 ± 1.54	93.14 ± 0.34
	DyGFormer	78.80 ± 1.95	80.54 ± 0.29	87.04 ± 0.35	76.55 ± 0.52	76.97 ± 0.24	90.77 ± 1.96	73.86 ± 1.13	97.17 ± 0.05
	MTdyg	85.73 ± 0.50	**82.16 ± 0.41**	87.94 ± 0.34	**77.62 ± 1.17**	**79.58 ± 0.65**	**92.27 ± 0.26**	**74.09 ± 1.76**	**97.53 ± 0.11**
ind	JODIE	70.96 ± 0.78	83.51 ± 0.15	66.63 ± 2.30	70.92 ± 1.05	64.14 ± 1.26	59.05 ± 5.52	66.82 ± 1.27	94.47 ± 1.08
	DyRep	67.36 ± 0.96	82.90 ± 0.31	63.26 ± 1.01	68.73 ± 1.34	54.25 ± 2.01	89.44 ± 0.71	65.60 ± 1.28	94.23 ± 0.18
	TGAT	81.93 ± 0.22	87.13 ± 0.20	73.18 ± 0.33	60.45 ± 2.12	60.80 ± 1.01	71.62 ± 5.42	66.13 ± 0.78	94.10 ± 0.41
	TGN	80.97 ± 0.31	84.56 ± 0.24	77.44 ± 2.86	71.34 ± 2.46	64.11 ± 1.04	78.12 ± 4.46	66.37 ± 5.39	91.64 ± 1.72
	CAWN	70.95 ± 0.95	**88.04 ± 0.29**	70.32 ± 1.43	75.17 ± 0.50	58.06 ± 0.26	76.45 ± 7.02	71.73 ± 0.74	87.68 ± 0.24
	TCL	82.19 ± 0.48	84.67 ± 0.29	70.36 ± 0.37	67.64 ± 0.86	70.05 ± 1.86	82.54 ± 3.91	**67.80 ± 1.21**	91.23 ± 0.19
	GraphMixer	**84.28 ± 0.30**	82.21 ± 0.13	72.45 ± 0.72	71.53 ± 0.85	**74.59 ± 0.74**	84.22 ± 0.91	66.53 ± 1.22	90.96 ± 0.27
	DyGFormer	75.09 ± 3.70	86.23 ± 0.51	80.76 ± 0.76	74.07 ± 0.64	65.96 ± 1.18	87.96 ± 1.80	62.56 ± 1.51	95.01 ± 0.15
	MTdyg	77.14 ± 2.21	88.48 ± 0.64	**82.22 ± 0.49**	**74.91 ± 0.50**	69.88 ± 0.72	**90.31 ± 1.28**	67.47 ± 0.44	**95.98 ± 1.17**

Table 5
AUC-ROC for inductive link prediction with random, historical and inductive negative sampling strategies.

NSS	Datasets	Wikipedia	Reddit	MOOC	Enron	UCI	US Legis.	UN Trade	Contact
rnd	JODIE	94.33 ± 0.27	96.52 ± 0.13	83.16 ± 1.30	81.96 ± 1.34	78.80 ± 0.94	58.12 ± 2.35	62.28 ± 0.50	95.37 ± 0.92
	DyRep	91.49 ± 0.45	96.05 ± 0.12	84.03 ± 0.49	76.34 ± 4.20	58.08 ± 1.81	61.07 ± 0.56	58.82 ± 0.98	91.89 ± 0.38
	TGAT	95.90 ± 0.09	96.98 ± 0.04	86.84 ± 0.17	64.63 ± 1.74	77.64 ± 0.38	48.27 ± 3.50	62.72 ± 0.12	96.53 ± 0.10
	TGN	97.72 ± 0.03	97.39 ± 0.07	91.24 ± 0.99	78.83 ± 1.11	86.68 ± 2.29	**62.38 ± 0.48**	59.99 ± 3.50	94.84 ± 0.75
	CAWN	98.03 ± 0.04	98.42 ± 0.02	81.86 ± 0.25	87.02 ± 0.50	90.40 ± 0.11	51.49 ± 1.13	67.05 ± 0.21	89.07 ± 0.34
	TCL	95.57 ± 0.20	93.80 ± 0.07	81.43 ± 0.19	72.33 ± 0.99	84.49 ± 1.82	50.43 ± 1.48	63.76 ± 0.07	93.05 ± 0.09
	GraphMixer	96.30 ± 0.04	94.97 ± 0.05	82.77 ± 0.24	76.51 ± 0.71	89.30 ± 0.57	47.20 ± 0.89	63.48 ± 0.37	92.83 ± 0.05
	DyGFormer	98.48 ± 0.03	98.71 ± 0.01	87.62 ± 0.51	90.69 ± 0.26	92.63 ± 0.13	53.21 ± 3.04	67.25 ± 1.05	98.30 ± 0.02

(*continued on next page*)

Appendix C. Complexity analysis of MTdyg model

To analyse the complexity of the MTdyg model, we break down its components and evaluate the computational complexity of each part separately. The time encoder maps time intervals to a vector space, with a computational complexity of $\mathcal{O}(N \cdot d_T)$, where N is the number of time intervals, and d_T is the dimension of each interval. For the node and edge encodings, assuming there are N nodes with node dimension d_H and edge dimension d_E, the computational complexity for node encoding is $\mathcal{O}(N \cdot d_H)$ and for edge encoding is $\mathcal{O}(N \cdot d_E)$. The time-based historical common neighbour (T-HCN) encoder, which calculates the frequency of common neighbours and applies a time attention mechanism, has a complexity of $\mathcal{O}(N \cdot k \cdot d_F)$, where k is the number of neighbours per node and d_F is the attention dimension.

The multi-scale transformer module includes setting patch size values based on the Fourier transform, which has a complexity of $\mathcal{O}(N \log N)$. The patching operation, which segments the encoded sequence into multiple patches, has a complexity of $\mathcal{O}(N \cdot d)$, assuming the patch size is S, sequence length is N, and each patch dimension is d. The Transformer encoder itself, which includes both the multi-head attention mechanism and the feedforward neural network, has a complexity of $\mathcal{O}(L \cdot (N^2 \cdot d + N \cdot d^2))$ for the multi-head attention and $\mathcal{O}(L \cdot N \cdot d^2)$ for the feedforward network, where L is the number of layers, h is the number of attention heads, and each attention head dimension is d/h.

The output layer, which computes the average and concatenates the features before passing them into a linear layer, has a complexity of $\mathcal{O}(N \cdot d)$. For the link prediction layer, the representations of node pairs are passed into two multilayer perceptrons (MLPs) with ReLU activation functions, yielding a complexity of $\mathcal{O}(P \cdot d_{out} \cdot C)$, where d_{out} is the input dimension of the MLP, C is the output dimension, and P is the number of node pairs. The binary cross-entropy loss function used for training has a complexity of $\mathcal{O}(M)$, where M is the number of samples.

Considering all parts, the overall complexity of the MTdyg model can be written as $\mathcal{O}(N \cdot d_T) + \mathcal{O}(N \cdot d_H) + \mathcal{O}(N \cdot d_E) + \mathcal{O}(N \cdot k \cdot d_F) + \mathcal{O}(N \log N) + \mathcal{O}(N \cdot d) + \mathcal{O}(L \cdot (N^2 \cdot d + N \cdot d^2)) + \mathcal{O}(L \cdot N \cdot d^2) + \mathcal{O}(N \cdot d) + \mathcal{O}(P \cdot d_{out} \cdot C) + \mathcal{O}(M)$. Given the dominant term $\mathcal{O}(L \cdot (N^2 \cdot d + N \cdot d^2))$, the overall complexity is approximated as $\mathcal{O}(L \cdot (N^2 \cdot d + N \cdot d^2))$, which is proportional to the square of the input sequence length N and the encoding dimension d, and increases linearly with the number of Transformer layers L.

Appendix D. The experimental results on AUC-ROC

See Tables 4 and 5.

Table 5 (*continued*).

NSS	Datasets	Wikipedia	Reddit	MOOC	Enron	UCI	US Legis.	UN Trade	Contact
	MTdyg	**98.91 ± 0.04**	**99.07 ± 0.01**	**91.54 ± 0.17**	**91.12 ± 0.06**	**92.89 ± 0.01**	58.36 ± 1.45	**67.93 ± 0.69**	**99.27 ± 0.04**
hist	JODIE	61.86 ± 0.53	61.69 ± 0.39	64.48 ± 1.64	65.32 ± 3.57	60.24 ± 1.94	58.12 ± 2.94	58.73 ± 1.19	90.80 ± 1.18
	DyRep	57.54 ± 1.09	60.45 ± 0.37	64.23 ± 1.29	61.50 ± 2.50	51.25 ± 2.37	**67.94 ± 0.98**	57.90 ± 1.33	88.88 ± 0.68
	TGAT	78.38 ± 0.20	64.43 ± 0.27	74.08 ± 0.27	57.84 ± 2.18	62.32 ± 1.18	49.99 ± 4.88	59.74 ± 0.59	93.76 ± 0.41
	TGN	75.75 ± 0.29	64.55 ± 0.50	77.69 ± 3.55	62.68 ± 1.09	62.69 ± 0.90	64.87 ± 1.65	55.61 ± 3.54	88.84 ± 1.39
	CAWN	62.04 ± 0.65	64.94 ± 0.21	71.68 ± 0.94	62.25 ± 0.40	56.39 ± 0.10	54.41 ± 1.31	60.95 ± 0.80	74.79 ± 0.37
	TCL	79.79 ± 0.96	61.43 ± 0.26	69.82 ± 0.32	64.06 ± 1.02	70.46 ± 1.94	52.12 ± 2.13	61.12 ± 0.97	90.37 ± 0.16
	GraphMixer	**82.87 ± 0.21**	64.27 ± 0.13	72.53 ± 0.84	**68.20 ± 1.62**	**75.98 ± 0.84**	49.28 ± 0.86	59.88 ± 1.17	90.04 ± 0.29
	DyGFormer	68.33 ± 2.82	64.81 ± 0.25	81.77 ± 0.63	65.78 ± 0.42	65.55 ± 1.01	56.57 ± 3.22	58.46 ± 1.65	94.14 ± 0.26
	MTdyg	73.46 ± 4.36	**65.35 ± 0.41**	**82.55 ± 0.87**	68.00 ± 0.26	70.76 ± 2.24	63.67 ± 1.81	**61.42 ± 1.43**	**94.82 ± 0.62**
ind	JODIE	61.87 ± 0.53	61.69 ± 0.39	64.48 ± 1.64	65.32 ± 3.57	60.27 ± 1.94	58.12 ± 2.94	58.71 ± 1.20	90.80 ± 1.18
	DyRep	57.54 ± 1.09	60.44 ± 0.37	64.22 ± 1.29	61.50 ± 2.50	51.26 ± 2.40	**67.94 ± 0.98**	57.87 ± 1.36	88.87 ± 0.67
	TGAT	78.38 ± 0.20	64.39 ± 0.27	74.07 ± 0.27	57.83 ± 2.18	62.29 ± 1.17	49.99 ± 4.88	59.98 ± 0.59	93.76 ± 0.40
	TGN	75.76 ± 0.29	64.55 ± 0.50	77.68 ± 3.55	62.68 ± 1.09	62.66 ± 0.91	64.87 ± 1.65	55.62 ± 3.59	88.85 ± 1.39
	CAWN	62.02 ± 0.65	64.91 ± 0.21	71.69 ± 0.94	62.27 ± 0.40	56.39 ± 0.11	54.41 ± 1.31	60.88 ± 0.79	74.79 ± 0.38
	TCL	79.79 ± 0.96	61.36 ± 0.26	69.83 ± 0.32	64.05 ± 1.02	70.42 ± 1.93	52.12 ± 2.13	61.01 ± 0.93	90.37 ± 0.16
	GraphMixer	**82.88 ± 0.21**	64.27 ± 0.13	72.52 ± 0.84	68.19 ± 1.63	**75.97 ± 0.85**	49.28 ± 0.86	59.71 ± 1.17	90.04 ± 0.29
	DyGFormer	68.33 ± 2.82	64.80 ± 0.25	80.77 ± 0.63	65.79 ± 0.42	65.58 ± 1.00	56.57 ± 3.22	57.28 ± 3.06	94.14 ± 0.26
	MTdyg	73.46 ± 4.36	**65.35 ± 0.41**	**82.54 ± 0.87**	68.01 ± 0.26	70.74 ± 2.23	63.67 ± 1.81	**60.94 ± 2.37**	**94.82 ± 0.62**

References

[1] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: 5th International Conference on Learning Representations, ICLR, 2017.

[2] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph attention networks, in: 6th International Conference on Learning Representations, ICLR, 2018.

[3] Y. Zhu, F. Lyu, C. Hu, X. Chen, X. Liu, Encoder-decoder architecture for supervised dynamic graph learning: A survey, CoRR abs/2203.10480, 2022.

[4] L. Yu, L. Sun, B. Du, W. Lv, Towards better dynamic graph learning: New architecture and unified library, in: A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, S. Levine (Eds.), Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems, NeurIPS, 2023.

[5] Y. Nie, N.H. Nguyen, P. Sinthong, J. Kalagnanam, A time series is worth 64 words: Long-term forecasting with transformers, in: The Eleventh International Conference on Learning Representations, ICLR, 2023.

[6] J. Skarding, B. Gabrys, K. Musial, Foundations and modeling of dynamic networks using dynamic graph neural networks: A survey, IEEE Access 9 (2021) 79143–79168.

[7] Y. Han, G. Huang, S. Song, L. Yang, H. Wang, Y. Wang, Dynamic neural networks: A survey, IEEE Trans. Pattern Anal. Mach. Intell. 44 (11) (2022) 7436–7456.

[8] A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, T.B. Schardl, C.E. Leiserson, EvolveGCN: Evolving graph convolutional networks for dynamic graphs, in: The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI, AAAI Press, 2020, pp. 5363–5370.

[9] S. Yan, Y. Xiong, D. Lin, Spatial temporal graph convolutional networks for skeleton-based action recognition, in: S.A. McIlraith, K.Q. Weinberger (Eds.), The 8th AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI, AAAI Press, 2018, pp. 7444–7452.

[10] D. Xu, C. Ruan, E. Körpeoglu, S. Kumar, K. Achan, Inductive representation learning on temporal graphs, in: 8th International Conference on Learning Representations, ICLR, 2020.

[11] E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, F. Monti, M.M. Bronstein, Temporal graph networks for deep learning on dynamic graphs, CoRR abs/2006.10637, 2020.

[12] R. Trivedi, M. Farajtabar, P. Biswal, H. Zha, DyRep: Learning representations over dynamic graphs, in: 7th International Conference on Learning Representations, ICLR, 2019.

[13] M.A. Alomrani, M. Biparva, Y. Zhang, M. Coates, DyG2Vec: Efficient representation learning for dynamic graphs, 2024, arXiv:2210.16906.

[14] Z. Cui, Z. Li, S. Wu, X. Zhang, Q. Liu, L. Wang, M. Ai, DyGCN: Efficient dynamic graph embedding with graph convolutional network, IEEE Trans. Neural Networks Learn. Syst. 35 (4) (2024) 4635–4646.

[15] P. Mei, Y. hong Zhao, Dynamic network link prediction with node representation learning from graph convolutional networks, Sci. Rep. 14 (2024).

[16] X. Mao, Y. Liu, W. Shen, Q. Li, Y. Wang, Deep residual Fourier transformation for single image deblurring, CoRR abs/2111.11745, 2021.

[17] W. Wang, J. Wang, C. Chen, J. Jiao, L. Sun, Y. Cai, S. Song, J. Li, FreMAE: Fourier transform meets masked autoencoders for medical image segmentation, CoRR abs/2304.10864, 2023.

[18] K. Alizadeh-Vahid, A. Prabhu, A. Farhadi, M. Rastegari, Butterfly transform: An efficient FFT based neural architecture design, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, Computer Vision Foundation / IEEE, 2020, pp. 12021–12030.

[19] H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, M. Long, TimesNet: Temporal 2D-variation modeling for general time series analysis, in: The Eleventh International Conference on Learning Representations, ICLR, 2023.

[20] Y. Tian, Y. Qi, F. Guo, FreeDyG: Frequency enhanced continuous-time dynamic graph model for link prediction, in: The Twelfth International Conference on Learning Representations, 2024.

[21] H. Wu, S. Wang, Y. Liang, Z. Zhou, W. Huang, W. Xiong, K. Wang, Earthfarseer: Versatile spatio-temporal dynamical systems modeling in one model, CoRR abs/2312.08403, 2023.

[22] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, N. Houlsby, An image is worth 16x16 words: Transformers for image recognition at scale, in: 9th International Conference on Learning Representations, ICLR, 2021.

[23] Z. Chen, Y. Zhu, C. Zhao, G. Hu, W. Zeng, J. Wang, M. Tang, DPT: deformable patch-based transformer for visual recognition, in: H.T. Shen, Y. Zhuang, J.R. Smith, Y. Yang, P. César, F. Metze, B. Prabhakaran (Eds.), MM: ACM Multimedia Conference, ACM, 2021, pp. 2899–2907.

[24] U. Demir, G.B. Ünal, Patch-based image inpainting with generative adversarial networks, CoRR abs/1803.07422, 2018.

[25] Z. Gong, Y. Tang, J. Liang, PatchMixer: A patch-mixing architecture for long-term time series forecasting, CoRR abs/2310.00655, 2023.

[26] S. Lee, T. Park, K. Lee, Learning to embed time series patches independently, CoRR, abs/2312.16427, 2023, http://dx.doi.org/10.48550/ARXIV.2312.16427.

[27] V. Nair, G.E. Hinton, Rectified linear units improve restricted Boltzmann machines, in: J. Fürnkranz, T. Joachims (Eds.), Proceedings of the 27th International Conference on Machine Learning, ICML, Omni Press, 2010, pp. 807–814.

[28] L.J. Ba, J.R. Kiros, G.E. Hinton, Layer normalization, CoRR abs/1607.06450, 2016.

[29] D. Hendrycks, K. Gimpel, Bridging nonlinearities and stochastic regularizers with Gaussian error linear units, CoRR abs/1606.08415, 2016.

[30] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: IEEE Conference on Computer Vision and Pattern Recognition, CVPR, IEEE Computer Society, 2016, pp. 770–778.

[31] S. Kumar, X. Zhang, J. Leskovec, Predicting dynamic embedding trajectory in temporal interaction networks, in: A. Teredesai, V. Kumar, Y. Li, R. Rosales, E. Terzi, G. Karypis (Eds.), Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD, ACM, 2019, pp. 1269–1278.

[32] Y. Wang, Y. Chang, Y. Liu, J. Leskovec, P. Li, Inductive representation learning in temporal networks via causal anonymous walks, in: 9th International Conference on Learning Representations, ICLR, 2021.

[33] L. Wang, X. Chang, S. Li, Y. Chu, H. Li, W. Zhang, X. He, L. Song, J. Zhou, H. Yang, TCL: transformer-based dynamic graph modelling via contrastive learning, CoRR abs/2105.07944, 2021.

[34] W. Cong, S. Zhang, J. Kang, B. Yuan, H. Wu, X. Zhou, H. Tong, M. Mahdavi, Do we really need complicated model architectures for temporal networks? in: The Eleventh International Conference on Learning Representations, ICLR, 2023.

[35] F. Poursafaei, S. Huang, K. Pelrine, R. Rabbany, Towards better evaluation for dynamic link prediction, in: S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, A. Oh (Eds.), Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems, NeurIPS, 2022.