Applied Soft Computing Journal 204 (2027) 116430

Contents lists available at ScienceDirect

Applied Soft Computing

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

Mamba-enhanced asymmetric contrastive graph-sequence learning for session-based recommendation

Weiyue Li[a,1], Ming Gao[a,1,*], Bowei Chen[b,1], Jingmin An[a,1], Hao Dong[a,1]

[a] School of Management Science and Engineering, Key Laboratory of Big Data Management Optimization and Decision of Liaoning Province, Dongbei University of Finance and Economics, Dalian, China

[b] Adam Smith Business School, University of Glasgow, Glasgow, United Kingdom

HIGHLIGHTS

- We propose Conan, an efficient asymmetric contrastive framework with novel session-level augmentation.
- As graph learning module, GamaFormer integrates self-attention with Mamba layers.
- As sequence learning module, Bi-Mason combines bidirectional Mamba with KAN.
- Conan outperforms 33 state-of-the-art baselines on four real-world datasets.

ARTICLE INFO

Keywords:
Session-based recommendation
Contrastive learning
Attention mechanism
State space models
Kolmogorov-Arnold Network

ABSTRACT

Session-based recommender systems aim to capture the latent preferences of the anonymous user and predict the next item based on the historical interactions within the current session. Most of existing works rely on attention-based modules to capture item dependencies derived from graph learning or sequence learning. However, these works suffer from the effectiveness and efficiency dilemma regarding the design of the overall framework and the modules, which hinders more precise recommendations on the larger scale datasets. To address these challenges, we propose Conan, an asymmetric contrastive framework that uses effective session-level augmentation and generates positive and negative samples with just a single forward propagation. Conan enhances the feature learning capacity of the graph learning module by leveraging the meaningful contrastive signals derived from sequence learning. For the specific module design, we introduce the Mamba block and the Fourier Kolmogorov-Arnold network to further enhance both effectiveness and efficiency. Extensive experiments conducted on four real-world datasets (Diginetica, Retailrocket, Dressipi, and KuaiRand) demonstrate the superiority of our approach over state-of-the-art methods, achieving average gains of 2.32% to 2.70% in Recall and 2.74% to 3.14% in two ranking metrics (MRR and NDCG). The ablation studies suggest the effectiveness of the overall framework and main components. The efficiency experiments demonstrate that our Conan achieves a well-balanced performance in terms of both time and space complexity.

1. Introduction

Session-based recommender systems (SBRSs) have become an important research topic due to their ability to provide personalized recommendations for anonymous users on online platforms. Unlike sequential recommender systems (SRSs), which often rely on long term user histories, SBRSs infer user preferences solely from interactions observed within the current session. This setting is particularly relevant in scenarios where user identities are unavailable or historical records are sparse. As a result, SBRSs have attracted growing attention from both academia and industry [1].

There are two main schemes to address the session-based recommendation task, including the graph-based methods and the sequence-based methods. Specifically, the graph-based methods first transform the raw session sequences into various session graphs, and then adopt attention-based and convolution-based methods to capture the non-sequential item dependencies. Moreover, the sequence-based methods

* Corresponding author.
Email addresses: gm@dufe.edu.cn; astroknt@qq.com (M. Gao).

[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.asoc.2026.116430
Received 19 February 2026; Received in revised form 25 August 2026; Accepted 8 September 2026
Available online 11 September 2026

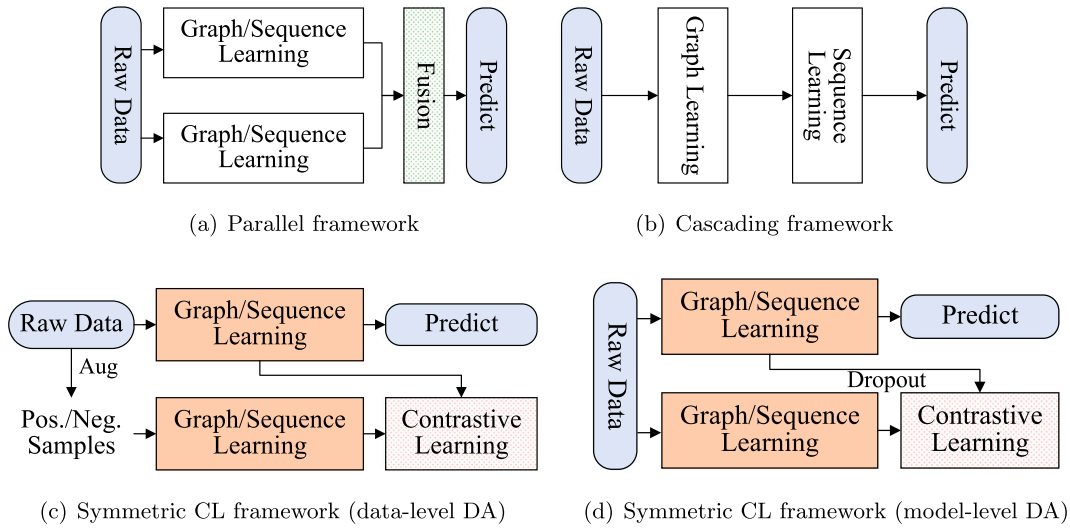


Fig. 1. Framework comparison. Red boxes indicate the same modules. DA and CL denote data augmentation and contrastive learning, respectively.

directly adopt the sequence modeling networks, such as recurrent neural networks (RNNs) and Transformers, to capture the sequential item dependencies. However, these works disregard the complementarity between non-sequential and sequential item dependencies, leading to a sub-optimal performance. Specifically, in terms of existing sequence-based SBRSs, user preferences are inherently more complex than simple sequential patterns in item transitions. In addition, existing graph-based SBRSs struggle to capture time-aware user preferences from sessions with complex structures, such as the ring structures. Therefore, there is an urge to integrate graph learning and sequence learning for more comprehensive capturing of item dependencies. While some attempts have been made to unify graph learning and sequence learning approaches, their suboptimal architectures often produce biased representations of user preferences and suffer from low efficiency. Specifically, as shown in Fig. 1 (a), the parallel frameworks adopt naive methods, such as concatenation and summation, to fuse the features derived from graph learning and sequence learning. However, these approaches fail to address potential feature conflicts between non-sequential and sequential item dependencies, leading to serious information loss and sub-optimal performance. Moreover, as shown in Fig. 1 (b), several approaches [2,3] adopt the cascading framework to integrate sequential information by applying sequence learning with positional encoding after graph learning. However, the cascading framework makes the sequential features overly dependent on graph features, resulting in a biased understanding of user preferences. Furthermore, although recent symmetric contrastive frameworks, as shown in Fig. 1 (c) and (d), have shown promising performance [4,5], their reliance on multiple forward propagations of an identical network for contrastive learning and recommendation inherently compromises training efficiency. In summary, it is still challenging for existing frameworks to balance effectiveness and efficiency.

As shown in Table 1, the effectiveness-efficiency dilemma also exists in the design of the specific modules, hindering the achievement of better performance on the larger scale datasets. In terms of efficiency, recurrent neural network-based modules are particularly costly to train, as their training process cannot be parallelized across time steps, limiting their training efficiency. Moreover, Transformer-based modules benefit from the global receptive field to capture long-term semantics. However, the quadratic computational complexity associated with self-attention limits their inference efficiency. General graph neural networks, including Gated Graph Neural Networks (GGNNs) and Graph Attention Networks (GATs), also suffer from comparable efficiency bottlenecks due to the RNN-based and attention-based message passing. In terms of effectiveness, the convolution-based modules benefit from the local receptive field and parameter sharing to capture local item dependencies efficiently. However, these modules fail to understand the global item dependencies due to the limited receptive field, leading to a sub-optimal performance. Furthermore, with the linear scalability in sequence length, recent Mamba is more efficient to train than recurrent neural networks (RNNs) and offers superior inference efficiency compared to Transformers. However, recent works also suggest that Mamba demonstrates limitations in some tasks [14,15], such as learning decision trees, retrieval tasks, and image classification. Particularly, MambaOut [15] suggests that Mamba struggles to learn refined features from limited context due to its lossy memory.

Table 1
Module comparison regarding effectiveness and efficiency.

Backbone	Representative work	Effectiveness	Efficiency
Recurrent neural network	GRU4Rec [6]	✓	✗
Transformer	SASRec [7]	✓	✗
Gated graph neural networks	SRGNN [8]	✓	✗
Graph attention networks	FAPAT [9]	✓	✗
Convolutional neural network	NextItNet [10]	✗	✓
Graph convolutional network	SPARE [11]	✗	✓
Light graph convolutional network	DCRec [12]	✗	✓
Mamba	Mamba4Rec [13]	✗	✓

To tackle these challenges, we introduce a framework and specific modules that achieve a balance between effectiveness and efficiency. In terms of framework design, we propose Conan, which leverages graph learning and sequence learning within an asymmetric contrastive framework to capture more comprehensive item dependencies. Compared to the popular symmetric contrastive frameworks that depend on a single network [4,5], Conan adopts different networks to effectively harmonize the representations captured by graph learning and sequence learning, where the diverse networks offer stronger feature learning capacity. Moreover, we employ session-level data augmentation within our contrastive learning (CL) framework to generate negative samples by shuffling session representations. This augmentation approach avoids disrupting the continuity of the original sequence and avoids tedious parameter tuning, which are significant limitations of existing data-augmentation methods [4,16]. Compared to existing contrastive frameworks, this framework requires only a single forward propagation to construct positive and negative samples rather than multiple forward propagations, significantly improving training efficiency. Furthermore, considering the recent success of graph-based recommender systems

[17,18], we emphasize the fundamental role of non-sequential item dependencies derived from graph learning. As for sequence learning, it contributes valuable contrastive signals that enhance the uniformity and alignment of representations. In terms of module design, the Mamba block and the Fourier Kolmogorov-Arnold network (KAN) are introduced to ensure the balance of effectiveness and efficiency. Specifically, we employ GaMaFormer for graph learning to capture the non-sequential item dependencies from session graphs, which interleaves self-attention layer with Mamba layer to leverage the strengths of both Transformer and Mamba network [19,20]. Moreover, Bi-Mason is proposed to highlight the sequential item dependencies derived from session sequences, which adopts the bidirectional Mamba with shared parameters as backbone. Inspired by the recent success of the KAN [21], we utilize the Fourier KAN to fuse bidirectional contextual features, employing 1D Fourier coefficients instead of B-spline coefficients to facilitate easier optimization and reduce the number of learnable parameters [22].

Extensive experiments on publicly available datasets demonstrate Conan's superior performance compared to 33 baseline models, with improvements ranging from 2.32% to 3.14% across three recommender system evaluation metrics. Ablation studies validate the effectiveness of the overall framework, the graph learning module, and the sequence learning module. Additionally, parameter sensitivity analyses are conducted to assess Conan's performance under different hyperparameter settings. Moreover, user group studies suggest that Conan can provide effective recommendations for users across different levels of activity. Furthermore, Conan's efficiency is compared with several representative baseline models, demonstrating a strong balance between effectiveness and efficiency. Lastly, we visualize the item embeddings of Conan and five baseline models to evaluate how session-level augmentation-based contrastive learning enhances representation learning. The visualization confirms that Conan produces a more optimal feature space distribution. This work makes four key contributions. First, we propose an asymmetric contrastive learning paradigm for jointly modeling graph structures and sequential patterns in session-based recommendation, enabling one-pass generation of positive and negative samples for more effective and efficient feature learning. Second, GaMaFormer combines self-attention, Mamba, and Stable MLPs to achieve efficient and expressive graph representation learning. Third, Bi-Mason employs bidirectional Mamba layers and Fourier KAN to capture contextual dependencies and improve sequence representation learning. Finally, extensive experiments on four benchmark datasets show that Conan consistently achieves state-of-the-art performance, and ablation studies further validate the contribution of each component.

2. Preliminaries

In this section, we first introduce the research objective and the problem definition of the session-based recommendation task. Subsequently, we briefly introduce the state space model and the selective mechanism of Mamba.

2.1. Research objective and problem definition

Our research objective is to predict the next item of interest for users based on their interactions within current sessions. Formally, let $V = \{v_1, \ldots, v_{|V|}\}$ denote the unique item set, where $|V|$ is the size of the item set. The current session of an anonymous user is defined as the sequential list of interacted items $s = \{v_1^s, \ldots, v_i^s, \ldots, v_l^s\}$, where $v_i^s \in V$ $(1 \le i \le |V|)$ denotes the item interacted with at the i-th time step, and l is the length of the session. Given the current session s, the goal of SBRSs is to predict the next item to be interacted with, v_{l+1}^s. Typically, at each time step, a top-K recommendation list is generated, consisting of items with the highest likelihood of being interacted with by the user.

In our SBRS, each item v_i is embedded as $e_i \in \mathbb{R}^d$ to represent its initial features, where d is the embedding dimension. Subsequently, the user preference representations h_{com}^{g} and $h_{com}^{seq} \in \mathbb{R}^d$ are learned by the graph learning and sequence learning modules, respectively. For graph learning, the original session sequence s is transformed into a directed session graph $G_s = (\mathcal{V}_s, \mathcal{E}_s)$, where $\mathcal{V}_s$ and $\mathcal{E}_s$ denote the item node set and edge set, respectively. The edge $(v_i, v_{i+1}) \in \mathcal{E}_s$ reflects the adjacent transition between two items. For sequence learning, the original session sequence s is transformed into a fixed-length session sequence s_{tr}, with a fixed-length threshold denoted as $L_{\max}$. If the sequence length exceeds $L_{\max}$, only the most recent $L_{\max}$ interactions are considered. If the sequence length is shorter than $L_{\max}$, padding items are added to the left until the sequence reaches the fixed length.

2.2. The state space model and the selective mechanism of mamba

The state space model (SSM) is a sequence modeling framework based on linear ordinary differential equations. Specifically, SSMs map the input sequence $x(t) \in \mathbb{R}^d$ to the output sequence $y(t) \in \mathbb{R}^d$ through hidden states $h(t) \in \mathbb{R}^{nd}$. The state dynamics are described by the first-order differential equations as follows:

$$h'(t) = Ah(t) + Bx(t), \tag{1}$$

$$y(t) = Ch(t), \tag{2}$$

where t denotes the time step, $A \in \mathbb{R}^{nd\times nd}$, $B \in \mathbb{R}^{nd\times d}$, and $C \in \mathbb{R}^{d\times nd}$ are learnable matrices. Specifically, the parameter A is the evolution matrix, which stores all previous historical information represented by a matrix of coefficients. It determines the degree of influence the previous hidden state has on the hidden state at the next time step, updating the spatial state accordingly. The projection matrix B determines how the input $x(t)$ affects the hidden state $h(t)$. Finally, the projection matrix C indicates how the hidden state $h(t)$ is transformed into the output $y(t)$.

Although SSM provides a unique approach to modeling sequences, it still faces three key challenges. First, to compute the output sequence $y(t)$ at time t, solving for $h(t)$ is analytically difficult. Second, real-world data is typically discrete rather than continuous. Third, the parameters in traditional SSMs remain invariant with respect to the input context and temporal dynamics, which fails to effectively handle selective tasks as equal attention is given to all input elements. To address these issues, the Mamba architecture employs the zero-order hold (ZOH) approach to model discrete sequences as an alternative to continuous functions. Formally,

$$\bar{A} = e^{A\Delta}, \tag{3}$$

$$\bar{B} = -A\Delta\left(e^{A\Delta} - I\right)\Delta B, \tag{4}$$

$$B = W_1 x_t + b_1, \tag{5}$$

$$C = W_2 x_t + b_2, \tag{6}$$

$$\Delta = \log\left\{1 + e^{W_3 x_t + b_3}\right\}, \tag{7}$$

where Δ is a scalar discretization parameter that converts the continuous coefficients A and B into $\bar{A}$ and $\bar{B}$. $x_t \in \mathbb{R}^d$ denotes the discreteAfter discretization, the model can be computed by a linear recurrence for inference:

$$y_t = Ch_t = C(\bar{A}h_{t-1} + \bar{B}x_t), \tag{8}$$

where the hidden state h_t can be seen as a fixed-size memory that stores all historical information. To have a sufficiently large memory of the past, the dimension of h_t is usually much larger than that of x_t. The recurrent mode is ideal for auto-regressive inference, since in that case parallelization is not possible.Moreover, the models can produce the output by executing a global convolution for training as follows:

$$y = x \odot \bar{K} = x \odot \left((C\bar{B},\ C\bar{A}\bar{B}, \ldots, C\bar{A}^L\bar{B})\right), \tag{9}$$

where $\odot$ is the Hadamard Product, L is the length of the input sequence, $\bar{K} \in \mathbb{R}^L$ denotes a structured convolutional kernel. Since A, B, and C

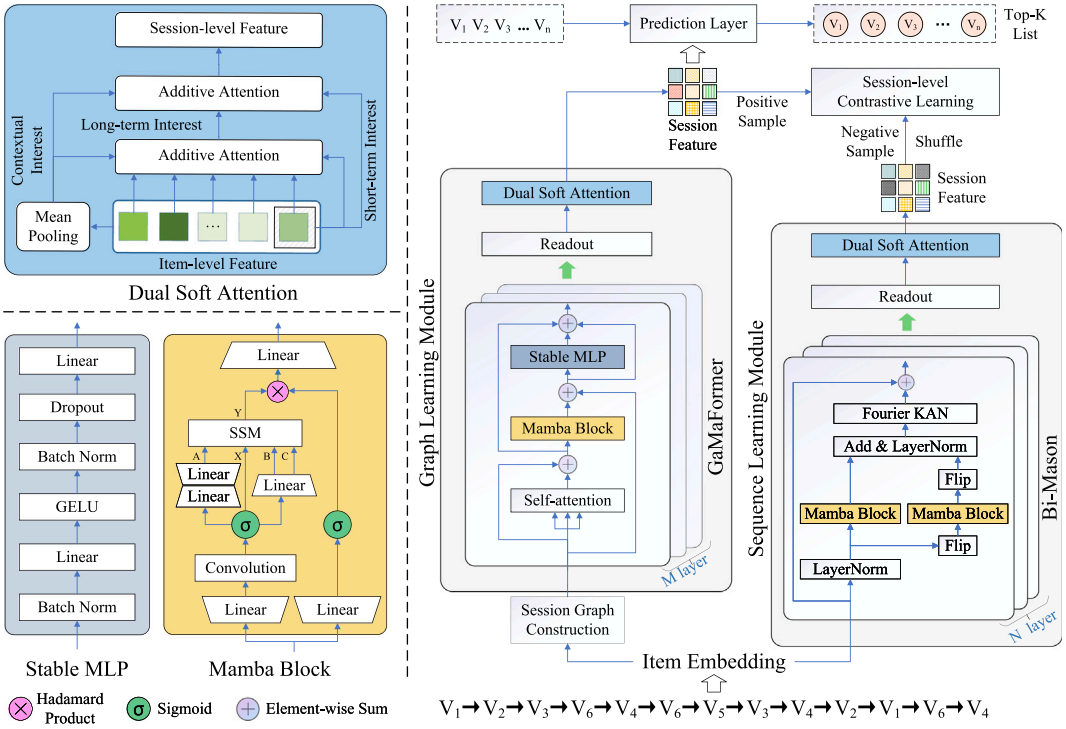


Fig. 2. Schematic view of conan: the right side illustrates the model's overall structure, featuring the graph learning module, sequence learning module, and prediction layer. Contrastive learning with session-level augmentation is included to enhance representation quality. The left side shows the architectural details of key components, such as the dual soft attention network, stable MLP, and mamba block.

are constants (if the inputs are constant) and the convolutional kernel $\bar{K}$ is static, the process of recomputing the inputs to states and states to outputs is avoided. Thus, the convolutional mode can fully exploit the parallelism of modern specialized hardware for training. For simplicity, we define this process as $y = \text{Mamba}(x)$.

3. Methodology

In this section, we provide an overview of three main parts of the proposed Conan, as shown in Fig. 2. Specifically, the graph learning module captures the non-sequential item transition pattern to learn the session-level user preference, which consists of the GaMaFormer and the dual soft attention mechanism. Moreover, the sequence learning module captures the sequential item transition pattern to learn the session-level user preference, which consists of the Bi-Mason and the dual soft attention mechanism. After that, the prediction layer calculates the score of each candidate item and provides the top-K recommendation list. Lastly, we introduce how to unify all the parts by the contrastive learning with the session-level data augmentation, which incorporates the sequential item transition information into the session-level user preference derived from the graph learning.

3.1. Graph learning module

Session graphs are constructed based on adjacent transitions between two items. The GaMaFormer is introduced to update the representations of item nodes, followed by the adoption of a dual attention mechanism to capture comprehensive user preferences by considering short-term interest, contextual interest, and long-term interest.

3.1.1. GaMaFormer

It is a powerful combination of a self-attention layer, a Mamba layer, and a stable MLP, each contributing unique strengths. The self-attention layer transforms the input item nodes into query, key, and value features. It computes long-range relationships by multiplying the similarity matrix, created by comparing the query and key features, with the value features. Specifically,

$$\alpha_{i,j} = \text{Softmax}\left\{\frac{\left(W_1^g e_i\right)^T \left(W_2^g e_j\right)}{\sqrt{d}}\right\}, \tag{10}$$

where $\alpha_{i,j}$ represents the attention weight between the target item node v_i and its neighboring item node v_j. The matrices W_1^g and $W_2^g \in \mathbb{R}^{d\times d}$ are trainable parameters.After calculating these weights, a gating network is used to update the representation of the target item node v_i as follows:

$$e_i' = \beta_i W_4^g e_i + \left(1-\beta_i\right) neigh_i, \tag{11}$$

$$neigh_i = W_3^g \sum_{j\in N(i)} \alpha_{i,j} e_j, \tag{12}$$

$$\beta_i = \sigma\left(W_\beta^T \left[W_4^g e_i \,||neigh_i|| \, W_4^g e_i \times neigh_i\right]\right), \tag{13}$$

where e_i' denotes the updated representation of the target item node v_i based on the self-attention layer, $[\cdot||\cdot||\cdot]$ denotes the concat operation among three tensors, σ denotes the Sigmoid activation function, W_3^g, $W_4^g \in \mathbb{R}^{d\times d}$, and $W_\beta \in \mathbb{R}^{3d}$ are the trainable parameters.

The self-attention layer lets each item focus on its neighboring items, helping to capture overall important information from the surrounding context. While self-attention is effective at finding relevant neighbors, it becomes less efficient as the number of connections grows, especially for items with many neighbors. To solve this, the Mamba layer is added to filter out unnecessary information and improve efficiency. That is

$$e_i'' = \text{Mamba}\left(e_i'\right), \tag{14}$$

where e_i'' denotes the updated representation of the target item node v_i based on the Mamba layer, which acts as a data-dependent node extractor, filtering relevant neighboring nodes at each recurrence step and focusing on the selected context.

Next, a stable MLP with GELU activation and residual connection are used to enhance the non-linear modeling capacity, helping the model handle changes in user behavior more effectively. Formally,

$$e_i''' = \mathrm{Drop}\left(\mathrm{BN}\left(\mathrm{GELU}\left(\mathrm{BN}\left(e_i' + e_i''\right) W_1^{smlp}\right)\right)\right) W_2^{smlp} + \left(e_i' + e_i''\right), \quad (15)$$

where e_i''' is the updated feature of the target item node v_i, $\mathrm{BN}(\cdot)$ refers to batch normalization, $\mathrm{Drop}(\cdot)$ is the dropout operation, and W_1^{smlp} and $W_2^{smlp} \in \mathbb{R}^{d\times d}$ are trainable parameters. For simplicity, this process is summarized as:

$$e_i''' = \mathrm{GaMaFormer}\left(e_i, E_{neigh}\right), \quad (16)$$

where E_{neigh} is the embedding matrix of the neighboring item nodes. The GaMaFormer layers are stacked to capture higher-order connectivity between item nodes.

Following the approach used in LightGCN [23], the comprehensive feature of the item node is learned by aggregating the representations across all layers. Thus

$$e_{i,(m)}''' = \mathrm{GaMaFormer}\left(e_i^{(m-1)}, E_{neigh}^{(m-1)}\right), \quad (17)$$

$$e_i^g = \frac{1}{M}\sum_{m=1}^{M} e_{i,(m)}''', \quad (18)$$

where e_i^g denotes the final representation of the target item node v_i, and M is the number of stacked GaMaFormer layers.

3.1.2. Dual soft attention network

After updating the item features, the dual soft attention network is introduced to learn the session-level user preference based on short-term, contextual, and long-term interests. Specifically, the short-term interest is defined as the last interacted item, i.e., $h_{st}^g = e_l^g$, while the contextual interest is calculated by averaging the representations of the items in the session, i.e., $h_{ct}^g = \frac{1}{l}\sum_{i=1}^{l} e_i^g$. The long-term interest h_l^g is learned using a soft attention mechanism that assigns weights to each item depending on the short-term and contextual interests. Formally,

$$h_l^g = \sum_{v_i \in S} \mu_i e_i^g, \quad (19)$$

$$\mu_i = \mathrm{Softmax}\left\{ W_\mu^T \sigma\left(\left[W_1^l e_i^g \middle\| W_2^l h_{ct}^g \middle\| W_3^l h_{st}^g\right]\right)\right\}, \quad (20)$$

where W_1^l, W_2^l, $W_3^l \in \mathbb{R}^{d\times d}$, and $W_\mu \in \mathbb{R}^{3d}$ are trainable parameters.Next, the short-term, contextual, and long-term interests are adaptively combined into the comprehensive preference using another soft attention mechanism. Thus

$$H_{pre} = \left[W_1^{com} h_{st}^g \middle\| W_2^{com} h_{ct}^g \middle\| W_3^{com} h_l^g\right]_{extra\_\dim}, \quad (21)$$

$$\eta = \mathrm{Softmax}\left\{H_{pre}\right\}, \quad (22)$$

$$h_{com}^g = \mathrm{SqueezeSum}\left(\eta H_{pre}\right), \quad (23)$$

where W_1^{com}, W_2^{com}, $W_3^{com} \in \mathbb{R}^{d\times d}$ are trainable parameters, $[\cdot||\cdot||\cdot]_{extra\_\dim}$ denotes the concatenation of three tensors along the extra preference dimension, and $\mathrm{SqueezeSum}(\cdot)$ denotes the sum along extra dimension and removes that dimension. Consequently, the session-level user preference representation is computed by the graph learning module.

3.2. Sequence learning module

Bi-Mason is designed to efficiently handle bidirectional context by using both a forward and reverse Mamba block with shared parameters. To enhance performance, the Fourier KAN is employed to seamlessly fuse contextual features from both directions. After updating the item features, a dual attention mechanism is applied to capture session-level user preferences through sequence learning.

Our Bi-Mason framework extends the standard Mamba block by transforming it from a unidirectional (left-to-right) structure to a bidirectional one. This is accomplished by applying the Mamba module twice: first to the original sequence, and then to the reversed sequence. The output from the reversed sequence is flipped along the sequence length dimension to preserve the session context. Finally, the Fourier KAN efficiently integrates the bidirectional contextual features. Formally,

$$E_s' = \mathrm{Mamba}\left(E_s\right) + \mathrm{Flip}\left(\mathrm{Mamba}\left(\mathrm{Flip}\left(E_s\right)_{ld}\right)\right)_{ld}, \quad (24)$$

$$E_s'' = \sum_{i=1}^{d}\sum_{k=1}^{g}\left(\cos\left(kE_s'\right)a_{ik} + \sin\left(kE_s'\right)b_{ik}\right), \quad (25)$$

where $\mathrm{Flip}(\cdot)_{ld}$ denotes flipping the input sequence along the length dimension, d represents the feature dimension, and a_{ik} and b_{ik} are trainable Fourier coefficients, and g refers to the grid size, which controls the number of terms (frequencies) used in the Fourier series expansion. Specifically, g determines the number of sine and cosine terms incorporated into the Fourier coefficients for each input dimension. For simplicity, we omit layer normalization and dropout operations in the equations above. The Fourier KAN excels at aggregating contextual features, delivering substantial performance improvements and enhancing training efficiency. Moreover, to prevent an increase in memory usage and avoid doubling the number of parameters, we share the projection weights between the forward and reverse Mamba blocks. Similarly, we stack multiple Bi-Mason layers to capture intricate semantic relationships among items. After stacking, a readout function based on mean-pooling is applied to aggregate contextual features from the different layers. Additionally, a dual soft attention network is employed to capture comprehensive user preferences through sequence learning. As a result, the session-level user preference, denoted as h_{com}^{seq}, is obtained from the sequence learning module.

3.3. Predicting layer

To predict the next item, the predicting layer provides the recommended score for each candidate item by multiplying the representations of user preference and item, that is

$$y_i = \frac{\exp\left\{\mathrm{sim}\left(\left(h_{com}^g\right), e_i\right)/\tau\right\}}{\sum_{v_j\in S}\exp\left\{\mathrm{sim}\left(\left(h_{com}^g\right), e_j\right)/\tau\right\}}, \quad (26)$$

where y_i denotes the score for item v_i, and τ denotes the temperature hyper-parameter. $\mathrm{sim}(\cdot)$ denotes the similarity measurement between the representation of user preference and the candidate item, such as cosine similarity adopted here.

Inspired by CORE [24], the robust distance measuring method is introduced to design the loss function as follows:

$$\mathscr{L}_{main} = -\log\left\{\frac{\exp\left\{\mathrm{sim}\left(h_{com}^g, e_+\right)/\tau\right\}}{\sum_{v_j\in S}\exp\left\{\mathrm{sim}\left(h_{com}^g, e_j\right)/\tau\right\}}\right\}, \quad (27)$$

where e_+ denotes the representation of the ground-truth next item v_+ for the session S.

3.4. Unifying graph learning and sequence learning with contrastive learning

The training process is further enhanced through an asymmetric contrastive learning objective. Contrastive learning encourages representations of different augmented views from the same session to be similar while separating representations from different sessions. In Conan, the graph learning module serves as the anchor branch, whereas the sequence learning module provides complementary supervision through representation alignment, thereby improving the quality of the learned

session-level user preferences. Existing contrastive learning methods for session-based recommendation predominantly rely on item-level augmentations. However, the limited length of user sessions often makes it difficult to preserve semantic consistency under such perturbations, leading to suboptimal positive pairs. To address this limitation, Conan adopts session-level contrastive learning as an auxiliary objective, enabling semantically consistent augmentation while further enhancing the quality of session-level preference representations.

Specifically, the proposed asymmetric contrastive learning framework constructs its supervisory signal within each session. The core idea is that the session-level representations derived from the graph learning module (h^g_{com}) and the sequence learning module (h^{seq}_{com}) both model the same session, capturing its user preference from different views. Therefore, they are naturally defined as a positive pair. To provide contrastive signal, a corrupted representation $\tilde{h}^g_{com}$ is generated by applying row-wise and column-wise shuffling operations to h^g_{com}, serving as a hard negative sample when paired with h^{seq}_{com}. We then employ the InfoNCE loss as our learning objective, which is defined as:

$$\mathscr{L}_{cl} = -\log\left\{\sigma\left(\mathrm{sim}\left(h^g_{com}, h^{seq}_{com}\right)\right)\right\} - \log\left\{\sigma\left(1-\mathrm{sim}\left(\tilde{h}^g_{com}, h^{seq}_{com}\right)\right)\right\}, \tag{28}$$

where h^g_{com} is the corrupted session-level user preference derived from the graph learning module.

Since the graph learning module and the sequence learning module capture user preferences from the same session, they can serve as ground truth for each other. The overall loss function of our Conan framework is defined as:

$$\mathscr{L} = \mathscr{L}_{main} + \lambda \mathscr{L}_{cl}, \tag{29}$$

where λ is a hyperparameter controlling the strength of the contrastive learning task.

The supervised recommendation loss $\mathscr{L}_{main}$ optimizes h^g_{com} with respect to the ground-truth next item, while the auxiliary contrastive loss $\mathscr{L}_{cl}$ reduces the discrepancy between h^g_{com} and h^{seq}_{com} of the same session. Following the alignment and uniformity perspectives of contrastive representation learning [25], the positive term improves cross-view alignment by reducing the distance between complementary representations of the same session, while the corrupted negative term prevents the representation space from collapsing to trivial solutions. Therefore, the sequence view acts as a directional regularizer that transfers temporal preference representations to the graph learning-derived preference representations.

3.5. Theoretical analysis

This section provides a theoretical analysis of the proposed Conan framework, including its time complexity, key module design, and asymmetric contrastive learning paradigm.

First, the learning procedure of Conan is summarized in Appendix A. The computational complexity of Conan is determined by four major components: GaMaFormer, Bi-Mason, the dual soft-attention networks, and the prediction layer. GaMaFormer has a complexity of $O(|E|_l(d + 5d^2))$, where $|E|_l$ denotes the number of edges in the session graph and d is the hidden dimension. The linear term corresponds to feature propagation, whereas the quadratic term is dominated by the self-attention, Mamba, and Stable MLP operations. Bi-Mason has a complexity of $O(2ld^2(1+g))$, where l denotes the session length and g is the Fourier grid size. The factor of 2 reflects bidirectional sequence modeling, while $(1+g)$ accounts for the computational cost of state-space modeling and Fourier transformation. The dual soft-attention networks contribute $O(2ld^2)$ by integrating graph- and sequence-based session representations. The prediction layer incurs a complexity of $O(|V|d)$, where $|V|$ denotes the number of candidate items. Therefore, the overall computational complexity of Conan is $O\big(|E|_l(d+5d^2) + 2ld^2(2+g) + |V|d\big)$. As the graph learning module and sequence learning module operate independently, these computations can be parallelized, by leveraging CUDA infrastructure for faster processing. Therefore, the overall time complexity of Conan is $O((|E|_l + |V|)d + (5|E|_l + (4+2\,g)l)d^2)$, where the first term $(|E|_l + |V|)d$ relates to linear operations, and the second term $(5|E|_l + (4+2\,g)l)d^2$ arises from the quadratic complexity of attention mechanisms and feature transformations.

Second, we compare the computational and memory characteristics of self-attention and selective state-space models to justify the proposed architecture. For a session of length l and hidden dimension d, self-attention requires $O(l^2d)$ time and $O(l^2)$ memory to construct the attention matrix through $QK^\top$, while the subsequent Softmax$(QK^\top)V$ operation also incurs $O(l^2d)$ time. During auto-regressive inference, a key–value cache of size $O(ld)$ must be maintained, causing memory consumption to grow linearly with the session length. In contrast, Mamba performs l selective state updates, each requiring $O(dn)$ operations for an n-dimensional hidden state, resulting in a total complexity of $O(ldn)$. Since n is fixed, the complexity reduces to $O(ld)$, achieving linear scalability with respect to the session length. Moreover, its memory requirement is limited to a fixed hidden state of size $O(dn)$, independent of l. These observations highlight a fundamental trade-off: self-attention provides global receptive fields but incurs quadratic computational and memory costs, whereas selective state-space models achieve linear efficiency by compressing sequential information into input-conditioned hidden states. Building on this advantage, Bi-Mamba introduces bidirectional state propagation to overcome the inherent limitation of standard Mamba's unidirectional information flow. Rather than replacing self-attention, GaMaFormer adopts a hybrid design in which self-attention captures fine-grained neighbor dependencies in session graphs, while Mamba efficiently filters redundant contextual information with superior scalability. The complementary strengths of these modules enable Conan to achieve an effective balance between recommendation accuracy and computational efficiency across sessions of varying lengths.

Third, unlike symmetric contrastive learning methods that align two equally treated augmented views generated by the same encoder, Conan adopts an asymmetric framework with distinct roles assigned to the graph and sequence branches. This design is motivated by the observation that architectural asymmetry stabilizes optimization and mitigates representation collapse in self-supervised learning, as demonstrated by the stop-gradient mechanism in SimSiam [26] and the target network in BYOL [27]. Rather than introducing asymmetry through weight sharing, momentum encoders, or gradient blocking, Conan naturally achieves it via two heterogeneous encoders with complementary inductive biases. Based on the alignment–uniformity perspective, the optimization dynamics of this asymmetric design can be characterized through gradient propagation. Let θ_g and θ_s denote the parameters of the graph and sequence learning modules, respectively. The gradients of the overall objective are

$$\nabla_{\theta_g}\mathscr{L} = \nabla_{\theta_g}\mathscr{L}_{main} + \lambda\nabla_{\theta_g}\mathscr{L}_{cl}, \tag{30}$$

$$\nabla_{\theta_s}\mathscr{L} = \lambda\nabla_{\theta_s}\mathscr{L}_{cl} \tag{31}$$

This asymmetric gradient flow brings two core advantages. First, the graph learning module is predominantly updated by the recommendation task loss, so its representation space is always anchored to the downstream prediction objective. The contrastive signal only provides mild alignment regularization and will not dominate the optimization direction, avoiding representation drift caused by the contrastive loss. Second, the sequence learning module is trained exclusively to align its representations toward the graph branch. This unidirectional alignment avoids the mutual pulling instability in symmetric frameworks, where both branches simultaneously adjust toward each other and may fall into degenerate solutions.

In addition, our asymmetric contrastive paradigm exhibits tangible efficiency advantages over mainstream symmetric contrastive frameworks. Specifically, the same encoder with parameters θ is applied to

two differently augmented views of the same input, requiring two separate forward propagations per training step. Although weight sharing keeps the parameter count unchanged at inference time, the duplicated forward passes effectively double the per-step training FLOPs of the encoder. By contrast, our asymmetric framework adopts a heterogeneous dual-branch architecture with total number of trainable parameters of $\theta_g + \theta_s$. Benefiting from the lightweight Mamba backbone, θ_s is smaller than θ_g. Accordingly, the overall parameter scale of our framework is lower than $2\theta_g$, which is required by symmetric contrastive learning paradigms to achieve comparable performance.

Finally, we derive a representation error bound for the asymmetric contrastive learning framework from an information-theoretic perspective. Formally, let h^g_{com} and h^{seq}_{com} denote the graph-view and sequence-view representations of the same session, respectively. Under our framework with InfoNCE loss $\mathcal{L}_{cl}$ and N negative samples, the mutual information between the two views satisfies:

$$I\left(h^g_{com}; h^{seq}_{com}\right) \geq \log N - \mathbb{E}\left[\mathcal{L}_{cl}\right]. \tag{32}$$

For the downstream recommendation task with task error ϵ_{task}, there exists a task-dependent constant C such that:

$$\epsilon_{task} \leq C \cdot \exp\left(-I\left(h^g_{com}; h^{seq}_{com}\right)\right). \tag{33}$$

The first inequality follows from the standard InfoNCE lower bound on mutual information. The second inequality is derived from information-theoretic generalization bounds, which link the mutual information of learned representations to the upper bound on downstream task error.

We further interpret the advantage of our asymmetric design via the alignment-uniformity framework. Unlike symmetric contrastive paradigms that jointly optimize alignment and uniformity through a single loss, our framework implicitly decouples the two objectives. Specifically, the graph encoder maintains representation uniformity under the main recommendation loss, while the sequence encoder optimizes alignment toward the stable graph branch via the contrastive loss. This decoupled design yields a more stable training landscape and contributes to a tighter representation error bound.

4. Experiments

In this section, comprehensive experiments are conducted to demonstrate the effectiveness and efficiency of our Conan, with the aim of answering the following questions: Compared with 33 competitive baseline models on four real-world datasets, how does our Conan perform on the task of session-based recommendation (RQ1)? How do the main parts improve the performance of the session-based recommendation (RQ2)? How does our Conan perform with different hyper-parameters (RQ3)? How does Conan perform in sessions of different lengths (RQ4)? What is the efficiency of different recommender systems (RQ5)? How does the contrastive learning with session-level augmentation help with model training (RQ6)?

4.1. Setup

The following presents the experimental setup, including the datasets, preprocessing procedures, evaluation metrics, baseline models, and implementation details.[2]

4.1.1. Datasets and preprocessing

The overall performance of our Conan and 33 competitive baseline models is evaluated on four public real-world datasets, including Diginetica, Retailrocket, Dressipi, and KuaiRand. Specifically, Diginetica is an SBRS benchmark dataset from CIKM Cup 2016.[3] The dataset contains user sessions extracted from an e-commerce search engine log. The transaction data of the dataset is adopted for our experiments. Retailrocket is a popular SBRS dataset from Kaggle Competition 2016.[4] It was collected from an e-commerce platform within 4.5 months. There are three types of user behaviors in Retailrocket, i.e., view, add to cart, and transaction. The view data of the dataset is adopted for our experiments. Dressipi is a dataset from RecSys Challenge 2022 focused on fashion recommendation.[5] Dressipi contains training data of 1 million sessions within a month, which is collected from an e-commerce platform. KuaiRand is a dataset collected from the recommendation logs of the video-sharing mobile app named Kuaishou.[6] In this study, we utilize the version known as KuaiRand-Pure due to the constraints of the experimental conditions. Notably, the Dressipi dataset is over four times larger than the other three datasets, featuring more interactions and sessions.

Table 2
Statistics of diginetica, retailrocket, dressipi, and KuaiRand.

Dataset	# Sessions	# Items	# Interactions	Avg. session length	Avg. action per item
Diginetica	204,062	42,172	989,204	4.85	23.46
Retailrocket	328,904	58,000	1,413,976	4.30	24.38
Dressipi	691,383	19,343	4,428,574	6.41	228.96
KuaiRand	25,598	7116	1,139,347	44.51	160.13

Following related works [24,28,29], the sessions longer than 1 and the items appearing more than 4 times are retained in all the datasets. Table 2 shows the statistics for the four datasets. For fair comparison, the data augment method [30] is adopted which generates the sessions and corresponding labels by splitting the input session. For example, for an input session $S = \{v_1, \ldots, v_{|S|}\}$, the generated sessions and the corresponding labels are $(\{v_1\}, v_2), (\{v_1, v_2\}, v_3), \ldots, (\{v_1, \ldots, v_{|S|-1}\}, v_{|S|})$. Moreover, leave-one-out strategy is adopted to split datasets. Specifically, we preserve the last and the second last interactions in each session as the testing and validation data, while the rest is taken as the training data.

4.1.2. Evaluation metrics

To evaluate the performance of the baseline models and our Conan, we adopt three metrics, i.e., **Recall@N**, **Mean Reciprocal Rank** (MRR@N), and **Normalized Discounted Cumulative Gain** (NDCG@N). For more detailed comparison, the N is set to 5, 10, 15, and 20 in the experiments. Specifically, Recall@N is the metric that assesses the proportion of test cases in which the correct items are recommended within the top-N list. In addition, MRR@N is a ranking metric which measures the average of the reciprocal ranks of the top-ranked relevant item for a set of recommendations. Moreover, NDCG@N is another ranking metric which takes into account both the relevance and the position of the recommended items. Furthermore, to avoid the potential biases, the rankings of all the items were calculated during the evaluation.

In terms of efficiency, we adopt five metrics, i.e., GPU memory, training time, inference time, wall time per forward pass, and number of epochs for training. Specifically, GPU memory evaluates the memory size required for model training and inference. Moreover, the training time and inference time quantify the time cost associated with model training and inference for each epoch, respectively. Additionally, wall time per forward pass is defined as the wall-clock time required for a single forward pass on one batch. Furthermore, the number of training epochs records the total number of epochs each model completes during full training.

[2] The source code of Conan is publicly available at https://github.com/usernameAI/Conan.

[3] http://cikm2016.cs.iupui.edu/cikm-cup.

[4] https://www.kaggle.com/retailrocket/ecommerce-dataset.

[5] http://www.recsyschallenge.com/2022/#participation.

[6] https://kuairand.com/.

Table 3
Benchmarked baseline models.

Group	Method
Traditional methods	POP, Item-KNN [34], CORE [24]
Graph learning methods	SRGNN [8], GCSAN [35], TAGNN [36], LESSR [37], COTREC [38]
Sequence learning methods	**RNN-based**: GRU4Rec [6], NARM [39], GLINT-RU [40] **Time-unaware**: STAMP [41], NextItNet [10] **Transformer-based**: SASRec [7], SR-PLR [42], TiSASRec [43], STAR [44], AC-TSR [45], MiaSRec [46], CL4SRec [4], DuoRec [5], AdaMCT [47] **Fourier transform-based**: FEARec [48], SLIME4Rec [49] **Mamba-based**: Mamba4Rec [13], RecMamba [50], EchoMamba4Rec [51], MLSA4Rec [52], SIGMA [53]
Graph-sequence learning methods	SGNN-HN [2], GCEGNN [54], CM-GNN [55], GSAU [56]

4.1.3. Implementation details

Our proposed Conan and all the baseline models are implemented based on the popular recommendation framework RecBole [31] and its extension RecBole-GNN [32] for easy development and reproduction. Following related works [8,11], the embedding dimension and batch size are set to 100. The initial learning rate is set to 0.001 and will decay by 10% after every 3 epochs. The Adam optimizer is adopted to train parameters. The maximum length of session is set to 20 for all the baseline models and our Conan. The SSM state expansion factor, the local convolution width, and the block expansion factor are respectively set to 50, 4, and 2 for all models that include Mamba blocks. The temperature parameter is set to 0.07 [33]. To alleviate over-fitting problem, the dropout strategy with 20% ratio is applied to our model. The stacking layer numbers of GaMaFormer and Bi-Mason are searched over {1, 2, 3, 4, 5}. The loss weight λ is searched over {0.0001, 0.001, 0.01, 0.1, 1}. Specifically, the optimal value of λ is 0.01 for the Diginetica dataset, and 0.1 for the Retailrocket, Dressipi, and KuaiRand datasets. The grid size g of Fourier KAN is searched over {1, 2, 4, 8}. For all other parameters, the baseline models follow the optimal configurations reported in their respective references. Adjustments were made when original parameter settings led to issues such as gradient explosion or out-of-memory errors during our experiments. To ensure robustness and reduce variance, all models were evaluated over 5 independent runs with different random seeds. The best results of all the models were recorded. Additionally, the statistically significant results ($p < 0.05$) were confirmed by a paired t-test against the best baseline model on each dataset, ensuring that the observed improvement is not attributable to random chance. To further demonstrate the stability and reliability of experimental results, we provide the standard deviations of all performance metrics across 5 independent runs for our Conan model and the strongest baseline on each dataset. Detailed results are organized in Table D.18 in Appendix D.

4.2. Baseline models

Table 3 presents the baseline models, categorized into four groups, including 1) two traditional methods, 2) five graph learning-based methods, 3) twenty sequence learning-based methods, including two RNN-based methods, and 4) three graph-sequence learning-based methods. More details about each baseline method can be found in Appendix B.

Recently, LLM-based recommender systems [57,58] have garnered growing attention for their promising performance in semantic understanding and cross-domain generalization, while parameter-efficient fine-tuning (PEFT) has yielded notable advances in reducing the adaptation cost of large language models for downstream recommendation tasks. However, LLM-based methods are excluded from our primary baselines for three key reasons.

First, regarding input modality adaptation, our work focuses on anonymous ID-only session recommendation with purely implicit feedback, where each session consists solely of item interaction sequences without auxiliary textual descriptions. In contrast, the performance advantages of LLM-based methods are heavily contingent on rich textual item content and pre-trained external knowledge [59,60]. When such semantic information is absent or incomplete, their recommendation quality degrades noticeably, and PEFT alone cannot compensate for this inherent information gap.

Second, in terms of deployment costs, even with PEFT strategies such as LoRA [61], fine-tuned LLM-based methods still require loading the full billion-parameter backbone during inference (e.g., Qwen-7B-Chat in LLM4SBRT [62]), incurring persistently high GPU memory overhead and long inference latency. For real-time recommendation services, this translates into substantial infrastructure and operational costs. In comparison, our Conan model maintains a lightweight parameter scale, runs on a single consumer-grade GPU, and delivers significantly lower inference latency.

Third, for scenario-specific suitability, our method is well suited to anonymous session recommendation scenarios with short interaction sequences. In such settings, persistent user profiles are unavailable and interaction sequences are sparse, which limits LLMs' ability to leverage user-level context and pre-trained knowledge. Conan models local item transition patterns and structural co-occurrence relations, capturing user intent more effectively from very short sessions.

4.3. Overall performance

To address RQ1, we conduct an empirical evaluation of overall performance on the session-based recommendation task across four real-world datasets in e-commerce: Diginetica, Retailrocket, Dressipi, and KuaiRand. To enhance readability and facilitate cross-dataset consistency assessments, the average results of our proposed Conan and the baseline models in terms of Recall, MRR, and NDCG with various lengths of top-N recommendation lists, are reported in Table 4. More details on each dataset are shown in Appendix C. We have the following observations.

On average, our Conan outperforms all the baseline models in terms of all the metrics on the four datasets, demonstrating its promising generalization ability. It can be explained by three facts. 1) The session-level augmentation-based contrastive learning method enhances the representation quality of session-level user preferences derived from graph learning through sequence learning, which alleviates the feature overwhelming caused by low-effective item-level feature fusion methods. 2) Our GaMaFormer learns representative item features on vanilla session graphs, where its message passing method benefits from both Mamba block and self-attention mechanism. 3) Our Bi-Mason provides meaningful self-supervised training signals by capturing consistent bi-directional sequential item transition patterns. Moreover, as shown in the last row of Table 4, the improvements of Conan over the best baseline model are 2.32% to 2.70% in terms of Recall@N with N = 5, 10, 15, and 20, while the respective improvements in terms of MRR@N and NDCG@N with N = 5, 10, 15, and 20 are 2.74% to 2.86% and 2.97% to 3.14%. It shows that our Conan contributes more to ranking the target item at a high position than hitting it in the recommendation list.

In most cases, it is found that the traditional methods achieve worse performance than the neural network-based methods, as they are too simple to capture the complex item transition patterns. In addition, with the increasing amount of interaction data, the inference efficiency of these methods will continue to decrease, which restricts large-scale real-world deployment. Graph learning-based methods and sequence learning-based methods generally outperform traditional approaches, indicating the necessity of session modeling. Specifically, graph learning-based methods first transform session sequences into various session graphs to retain complex item dependencies, such as the vanilla session graphs [8] and the edge-order preserving multigraphs [37]. After that, graph neural networks, such as gated graph neural networks, are adopted to update item node features. Subsequently,

Table 4
Average performance of the examined models across the used datasets. For each column, the best performance and the second best performance methods are denoted in **bold** and underlined fonts respectively. * indicates that the improvement over the strongest baseline is statistically significant ($p < 0.05$) based on a paired t-test.

Model	Category	@5			@10			@15			@20		
		Recall	MRR	NDCG	Recall	MRR	NDCG	Recall	MRR	NDCG	Recall	MRR	NDCG
Pop	Traditional Method	1.26	0.69	0.82	2.00	0.78	1.06	2.71	0.84	1.25	3.29	0.87	1.38
Item-KNN		12.05	7.12	8.24	17.68	7.86	10.05	21.55	8.17	11.08	24.47	8.33	11.77
CORE		27.35	18.06	20.38	34.67	19.04	22.74	39.36	19.41	23.98	42.75	19.60	24.78
SRGNN	Graph Learning Method	27.75	18.59	20.87	35.06	19.57	23.23	39.60	19.92	24.43	43.00	20.11	25.23
GCSAN		28.37	19.15	21.44	35.59	20.11	23.78	40.14	20.47	24.98	43.52	20.66	25.78
TAGNN		29.00	19.34	21.74	36.40	20.32	24.13	41.01	20.75	25.35	44.47	20.88	26.17
LESSR		28.07	18.79	21.10	35.42	19.77	23.47	39.99	20.13	24.68	43.43	20.32	25.50
COTREC		22.08	17.82	18.89	24.71	18.16	19.71	26.41	18.29	20.14	27.77	18.36	20.45
GRU4Rec	Sequence Learning	27.54	18.44	20.70	34.99	19.44	23.11	39.73	19.81	24.37	43.23	20.01	25.19
NARM	Method	27.72	18.55	20.82	35.17	19.54	23.23	39.93	19.91	24.49	43.43	20.11	25.32
GLINT-RU		28.62	19.40	21.69	35.99	20.38	24.07	40.64	20.75	25.30	44.14	20.95	26.13
STAMP		25.44	17.55	19.50	32.29	18.46	21.72	36.79	18.81	22.91	40.16	19.00	23.70
NextItNet		22.75	14.93	16.87	29.64	15.85	19.09	34.14	16.20	20.29	37.49	16.39	21.08
SASRec		27.50	18.73	20.91	35.06	19.73	23.34	39.93	20.12	24.63	43.60	20.32	25.50
SR-PLR		26.13	17.89	19.94	33.13	18.82	22.19	37.64	19.17	23.39	41.06	19.36	24.19
TiSASRec		28.46	19.12	21.44	35.84	20.10	23.83	40.56	20.48	25.08	44.12	20.68	25.92
STAR		25.66	18.33	20.15	31.48	19.10	22.03	35.21	19.39	23.02	38.01	19.55	23.67
AC-TSR		26.61	18.31	20.38	33.85	19.27	22.71	38.59	19.65	23.96	37.33	19.85	24.80
MiaSRec		25.55	17.52	19.52	31.56	18.32	21.46	35.35	18.61	22.46	38.24	18.78	23.14
CL4SRec		25.29	17.30	19.28	32.12	18.21	21.49	36.58	18.56	22.67	39.95	18.75	23.47
DuoRec		26.91	18.46	20.56	34.14	19.42	22.89	38.85	19.79	24.14	42.39	19.99	24.97
FEARec		26.94	18.48	20.58	34.05	19.42	22.87	38.58	19.78	24.07	41.97	19.97	24.87
SLIME4Rec		28.59	19.24	21.57	36.18	20.25	24.01	40.97	20.63	25.29	44.58	20.83	26.13
AdaMCT		27.64	18.95	21.10	34.92	19.91	23.45	39.61	20.28	24.70	43.10	20.48	25.52
Mamba4Rec		28.41	19.09	21.40	35.88	20.08	23.81	40.61	20.45	25.06	44.17	20.65	25.90
RecMamba		28.87	19.53	21.85	36.18	20.51	24.22	40.78	20.87	25.44	44.21	21.06	26.24
EchoMamba4Rec		28.61	19.54	21.79	35.64	20.47	24.06	40.12	20.82	25.25	43.35	21.01	26.01
MLSA4Rec		28.16	19.05	21.32	35.29	20.01	23.62	39.75	20.36	24.80	43.04	20.54	25.58
SIGMA		27.94	18.78	21.06	35.35	19.77	23.45	40.07	20.14	24.70	43.54	20.33	25.52
GCEGNN	Graph-Sequence	29.06	19.30	21.73	36.61	20.30	24.16	41.28	20.67	25.40	44.75	20.86	26.22
SGNN-HN	Learning Method	28.78	18.89	21.35	36.43	19.91	23.83	41.14	20.28	25.07	44.62	20.48	25.89
CM-GNN		28.53	18.86	21.27	36.05	19.86	23.70	40.76	20.23	24.94	44.31	20.43	25.78
GSAU		25.88	17.46	19.55	33.09	18.42	21.88	37.76	18.79	23.11	41.27	18.99	23.94
Conan (ours)		**29.85***	**20.07***	**22.50***	**37.50***	**21.09***	**24.98***	**42.25***	**21.47***	**26.23***	**45.78***	**21.66***	**27.07***
Improv.		2.70%	2.74%	2.97%	2.43%	2.84%	3.14%	2.35%	2.86%	3.14%	2.32%	2.85%	3.14%

some follow-up neural networks are introduced to learn the session-level representations, such as self-attention blocks [35] and target-aware attention mechanism [36]. By contrast, sequence learning-based methods directly adopt sequential neural networks, like RNNs [6] and Transformers [7], to capture contextual features from session sequences. Particularly, inspired by Mamba's success [19], new Mamba-based methods are emerging, like MLSA4Rec [52], which integrates Mamba blocks with LightSANs. Moreover, EchoMamba4Rec [51] introduces bi-directional feature learning and gated linear units for fusion. Despite certain successes, these single view-based methods face challenges in fully capturing users' comprehensive preferences. Additionally, both the learning method and the feature fusion method can be further optimized to improve their efficiency and effectiveness.

Graph-sequence learning-based methods simultaneously benefit from various data structures, where session graphs and session sequences contain the spatial and temporal item transition information respectively. Specifically, GCEGNN [54] and SGNN-HN [2] first employ graph learning to update item representations, and subsequently adopt sequence learning to capture the session-level sequential behavioral patterns. Specifically, for graph learning, GCEGNN [54] introduces the global session graph to consider cross-session item transitions, while SGNN-HN [2] adopts the session star graph to facilitate the long-distance message passing. Although GCEGNN [54] achieves the best performance among all the baseline models, its memory-inefficient global session graph restricts large-scale deployment. Additionally, CM-GNN [55] introduces the session-level augmentation-based contrastive learning task built upon GCEGNN [54]. However, CM-GNN [55] achieves sub-optimal performance. This indicates that the stacking architecture fails to decouple the features respectively derived from graph learning and sequence learning, resulting in the capture of biased user preferences. Furthermore, GSAU [56] adopts a parallel framework to organize graph learning and sequence learning, enabling separate preference modeling. However, its suboptimal backbone prevents it from achieving promising performance.

4.4. Ablation study

To address RQ2, we conduct four groups of ablation studies comparing our Conan with its variants. This analysis focuses on the effectiveness of the graph-sequence contrastive learning framework, as well as each component of the graph and sequence learning modules. The ablation studies are based on Recall@N, MRR@N, and NDCG@N on the Diginetica, RetailRocket, Dressipi, and KuaiRand datasets, where the N is set to 5, 10, 15, and 20.

4.4.1. The effects of graph-sequence contrastive learning framework

The first group of ablation studies aims to demonstrate the effectiveness of the graph-sequence asymmetric contrastive learning framework. Specifically, we produce eleven variants for comparison, including: (1) w/o GL, which only retains the sequence learning module. (2) w/o SL, which only retains the graph learning module. (3) w/o CL, which removes the contrastive learning and sums the features derived from graph learning and sequence learning. For fair comparison, it only utilizes the features derived from graph learning for final prediction, without incorporating sequence learning features. (4) w/o CL (sum), which removes

the contrastive learning module and fuses graph and sequence learning features via element-wise summation for training. For prediction, it sums the features derived from graph learning and sequence learning. (5) w/o CL (concat), which removes the contrastive learning module and fuses graph and sequence learning features via concatenation followed by a linear projection layer for training. For prediction, it sums the features derived from graph learning and sequence learning. (6) SL with Data-level CL, which adopts the data-level contrastive learning [4] to enhance w/o GL. (7) SL with Model-level CL, which adopts the model-level contrastive learning [5] to enhance w/o GL. (8) GL with Feature-level CL, which adopts the feature-level contrastive learning [63] to enhance w/o SL. (9) Mirroring Conan, which enhances the representation of session-level user preferences derived from sequence learning via graph learning. (10) Dual GL with SDA, which employs two identical GaMaFormer encoders as the main branch and the auxiliary contrastive branch, respectively, and uses the session-level shuffling augmentation for view construction. (11) Dual SL with SDA, which adopts two identical Bi-Mason encoders as the main and auxiliary branches, and uses the same session-level shuffling augmentation for view construction. For fair comparison, we keep the best configurations as the same as CL4SRec [4], DuoRec [5] and XSimGCL [63] when applying the data-level, the model-level, and the feature-level augmentations, respectively. Specifically, we choose the item cropping as the data-level augmentation operator. Moreover, the cropping proportion and the loss weight of contrastive learning task are set to 0.2 and 0.1, respectively. For model-level augmentation, we set the dropout ratio and the loss weight of contrastive learning task to 0.2 and 0.1, respectively. For feature-level augmentation of the variant called GL with Feature-level CL, the weight of the uniform noise and the loss weight of contrastive learning task are set to 0.2 and 0.2, respectively. The results are shown in Table 5.

As shown in Table 5, it is observed that our proposed Conan outperforms seven variant models in all cases on four datasets, demonstrating the effectiveness of the graph-sequence contrastive learning framework. Specifically, removing each part of our framework consistently results in a performance drop, indicating the essential roles of the graph learning, the sequence learning, and the contrastive learning. Particularly, we observe that Conan consistently outperforms w/o CL and its two variants, suggesting that the contrastive self-supervised signal is essential for our model to improve the quality of representations. One of the reasons is that the contrastive learning effectively unifies graph learning and sequence learning, alleviating the hidden conflicts between sequential and non-sequential item dependencies. Furthermore, w/o GL outperforms SL with Data-level CL and SL with Model-level CL, showing that it is challenging for these augmentations to cope with the session-based recommendation task due to the limited contextual information. Similar trend is found in the comparison between w/o SL and GL with Feature-level CL, implying that the noise-based embedding augmentation is still unsuitable for session-based recommendation. On the one hand, directly applying invasive augmentation [4] will destroy the semantic information of the original data, which is ineffective and even detrimental in some cases. On the other hand, other augmentations [5,63] heavily rely on parameter tuning to fit various datasets, which restricts their generalization ability. In addition, comparing Conan to Mirroring Conan, we observe that Mirroring Conan does not achieve satisfactory performance. The key reason is that the feature learning capacity of the hybrid network in the graph learning module is stronger than that of the Mamba-oriented network in the sequence learning module, enabling it to effectively handle the rich contextual information present in session graphs. Finally, experimental results demonstrate that our asymmetric heterogeneous framework consistently achieves better recommendation performance than both symmetric variants on all four datasets and evaluation metrics. Specifically, our approach outperforms Dual GL with SDA and Dual SL with SDA by harnessing the complementary inductive biases from graph and sequence branches for more comprehensive session representation learning.

To validate the effectiveness of the asymmetric contrastive learning method, we conduct experiments on two representative classical backbones, SRGNN and SASRec, and evaluate their performance with the asymmetric contrastive learning method. We test four dual-branch configurations, including two homogeneous settings (SRGNN + SRGNN, SASRec + SASRec) and two heterogeneous settings (SRGNN as the main branch with SASRec as the auxiliary branch, SASRec as the main branch with SRGNN as the auxiliary branch). The vanilla single-branch SRGNN and SASRec are included as baselines, and our Conan model is presented as a reference for performance comparison. The full results on four benchmark datasets are reported in Table 6.

We find that the models with the asymmetric contrastive learning achieve consistent performance improvements over their vanilla models in most cases. For example, in the Diginetica dataset, the homogeneous SRGNN dual-branch setting yields a 3.86% gain in Recall@20 compared with vanilla SRGNN, and the homogeneous SASRec dual-branch setting improves Recall@20 by 1.71% over vanilla SASRec. The consistent improvements across different backbones and datasets demonstrate that the asymmetric contrastive learning is a general and effective method that can be readily adapted to various backbones.

In summary, the components of the Conan framework exhibit a synergistic relationship derived from two design principles. Graph learning captures non-sequential item co-occurrence patterns through message passing on session graphs, while sequence learning captures temporal dependencies via bidirectional sequential modeling. These two views provide complementary information about user preferences. Specifically, the graph view reveals structural patterns, while the sequence view uncovers temporal patterns. Furthermore, directly fusing graph and sequence features, such as simple summation, can lead to feature conflicts, as graph and sequence representations are distributed across separate representation spaces. The asymmetric contrastive learning framework effectively unifies the two modules by aligning sequence representations with graph representations during training, which alleviates the underlying conflicts between sequential and non-sequential item dependencies.

4.4.2. *The effects of graph learning module*

The second group of ablation studies aims to demonstrate the effectiveness of the message passing method in our GaMaFormer. Specifically, we produce three variants for comparison, including: (1) w/o SAN, which combines the basic Mamba block with the stable MLP for message passing; (2) w/o Mamba, which combines the self-attention layer with the stable MLP for message passing; and (3) w/o stable MLP, which removes only the stable MLP for message passing. The results are shown in Table 7.

From Table 7, we observe that three variant models show worse performance than Conan, which demonstrates the effectiveness of each component in GaMaFormer. Specifically, the fundamental novelty of GaMaFormer is its hybrid Transformer-Mamba architecture for graph learning. Although Transformer outperforms RNNs in terms of training efficiency and has a global receptive field compared to convolutional neural networks (CNNs), the quadratic bottleneck limits its inference efficiency. Recent Mamba demonstrates promising results in terms of both efficiency and performance, while still lagging behind the performance of comparably sized Transformer-based models. Taking advantage of both Transformer and Mamba, GaMaFormer combines both of them to improve the efficiency and context learning capability. Comparing w/o SAN and w/o Mamba on the four datasets, we can observe that w/o SAN outperforms w/o Mamba on Diginetica, Dressipi, and KuaiRand, while w/o Mamba outperforms w/o SAN on Retailrocket. This suggests that the self-attention layer and the Mamba layer have advantages in handling session sequences of different lengths. Specifically, Retailrocket is the sparsest dataset with the shortest average session length among the four datasets. The self-attention mechanism memorizes all historical interactions to learn session-level user preferences, demonstrating

Table 5
Effects of graph-sequence contrastive learning framework. For the experimental results on each dataset, the **bold**-faced number is the best score.

Models	Graph Learning	Sequence Learning	Contrastive Learning	@5			@10			@15			@20		
				Recall	MRR	NDCG	Recall	MRR	NDCG	Recall	MRR	NDCG	Recall	MRR	NDCG
Diginetica															
Conan (ours)	✓ (main)	✓	Feature-level	**32.13**	**19.00**	**22.26**	**43.53**	**20.52**	**25.94**	**50.64**	**21.08**	**27.82**	**55.73**	**21.37**	**29.02**
w/o GL	✗	✓	Feature-level	25.97	14.86	17.61	36.67	16.28	21.06	43.55	16.82	22.88	48.55	17.10	24.06
w/o SL	✓	✗	Feature-level	31.99	18.90	22.14	43.45	20.43	25.85	50.60	20.99	27.74	55.68	21.28	28.94
w/o CL	✓	✓	w/o	27.05	15.79	18.58	37.58	17.19	21.97	44.35	17.72	23.76	49.27	18.00	24.93
w/o CL (concat)	✓	✓	w/o	31.15	18.74	21.81	41.91	20.17	25.29	48.53	20.69	27.04	53.32	20.96	28.17
w/o CL (sum)	✓	✓	w/o	30.91	18.55	21.61	41.65	19.98	25.09	48.38	20.52	26.87	53.24	20.79	28.02
SL wtih Data-level CL	✗	✓	Data-level	20.85	13.96	15.68	25.68	14.60	17.23	28.60	14.83	18.01	30.82	14.95	18.53
SL wtih Model-level CL	✗	✓	Model-level	14.74	7.94	9.62	22.70	8.99	12.18	28.32	9.43	13.66	32.73	9.68	14.71
GL wtih Feature-level CL	✓	✗	Feature-level	31.17	18.50	21.64	42.11	19.96	25.18	48.95	20.50	26.99	53.83	20.77	28.14
Mirroring Conan	✓	✓ (main)	Feature-level	31.18	18.77	21.84	41.78	20.18	25.27	48.44	20.7	27.03	53.27	20.98	28.17
Dual GL with SDA	✓ (dual)	✗	Feature-level	30.22	16.97	20.25	42.63	18.62	24.26	50.35	19.23	26.30	55.80	19.54	27.59
Dual SL with SDA	✗	✓ (dual)	Feature-level	29.34	16.89	19.97	40.92	18.43	23.71	48.20	19.00	25.64	53.43	19.30	26.87
Retailrocket															
Conan (ours)	✓ (main)	✓	Feature-level	**57.32**	**43.48**	**46.95**	**63.99**	**44.38**	**49.12**	**67.32**	**44.64**	**50.00**	**69.46**	**44.76**	**50.51**
w/o GL	✗	✓	Feature-level	56.92	**43.48**	46.86	63.28	44.35	48.93	66.48	44.60	49.77	68.57	44.72	50.27
w/o SL	✓	✗	Feature-level	57.17	43.35	46.81	63.84	44.25	48.98	67.24	44.52	49.88	69.43	44.64	50.40
w/o CL	✓	✓	w/o	53.90	41.12	44.33	60.34	41.99	46.42	63.60	42.25	47.28	65.85	42.38	47.82
w/o CL (concat)	✓	✓	w/o	56.29	43.32	46.58	61.83	44.06	48.38	64.56	44.28	49.10	66.32	44.38	49.52
w/o CL (sum)	✓	✓	w/o	56.36	43.16	46.47	62.36	43.97	48.43	65.36	44.21	49.22	67.35	44.32	49.70
SL wtih Data-level CL	✗	✓	Data-level	46.96	39.64	41.52	48.27	39.82	41.94	48.92	39.87	42.12	49.39	39.90	42.23
SL wtih Model-level CL	✗	✓	Model-level	37.82	33.37	34.49	40.07	33.67	35.22	41.40	33.78	35.57	42.31	33.83	35.79
GL wtih Feature-level CL	✓	✗	Feature-level	56.68	43.49	46.80	62.85	44.32	48.80	65.97	44.57	49.63	67.96	44.68	50.10
Mirroring Conan	✓	✓ (main)	Feature-level	56.41	43.23	46.54	62.21	44.02	48.43	65.07	44.25	49.19	66.94	44.35	49.63
Dual GL with SDA	✓ (dual)	✗	Feature-level	55.06	40.90	44.44	62.54	41.91	46.88	66.43	42.22	47.90	68.95	42.36	48.50
Dual SL with SDA	✗	✓ (dual)	Feature-level	54.37	40.74	44.16	61.14	41.65	46.36	64.79	41.94	47.32	67.07	42.07	47.86
Dressipi															
Conan (ours)	✓ (main)	✓	Feature-level	**23.77**	**14.14**	**16.52**	**32.19**	**15.27**	**19.25**	**37.39**	**15.67**	**20.62**	**41.15**	**15.89**	**21.51**
w/o GL	✗	✓	Feature-level	22.53	13.31	15.59	30.71	14.40	18.24	35.85	14.81	19.60	39.57	15.02	20.48
w/o SL	✓	✗	Feature-level	23.67	14.07	16.45	32.11	15.19	19.17	37.35	15.60	20.56	41.10	15.82	21.44
w/o CL	✓	✓	w/o	7.14	3.66	4.52	10.88	4.15	5.72	13.68	4.37	6.46	15.98	4.50	7.01
w/o CL (concat)	✓	✓	w/o	22.30	13.02	15.32	30.51	14.11	17.97	35.62	14.52	19.32	39.38	14.73	20.21
w/o CL (sum)	✓	✓	w/o	22.40	13.09	15.39	30.60	14.18	18.04	35.76	14.59	19.41	39.51	14.80	20.29
SL wtih Data-level CL	✗	✓	Data-level	12.17	7.54	8.69	16.63	8.14	10.13	19.59	8.37	10.91	21.86	8.50	11.44
SL wtih Model-level CL	✗	✓	Model-level	9.14	4.91	5.95	13.79	5.53	7.45	17.09	5.79	8.32	19.74	5.93	8.95
GL wtih Feature-level CL	✓	✗	Feature-level	23.64	14.09	16.46	31.87	15.19	19.12	36.92	15.59	20.45	40.69	15.80	21.34
Mirroring Conan	✓	✓ (main)	Feature-level	23.22	14.02	16.30	31.20	15.08	18.88	36.15	15.47	20.19	39.78	15.68	21.05
Dual GL with SDA	✓ (dual)	✗	Feature-level	22.03	12.93	15.18	30.06	14.00	17.78	35.15	14.40	19.12	38.91	14.61	20.01
Dual SL with SDA	✗	✓ (dual)	Feature-level	21.46	12.63	14.82	29.37	13.69	17.38	34.41	14.08	18.71	38.08	14.29	19.57
KuaiRand															
Conan (ours)	✓ (main)	✓	Feature-level	**6.73**	**3.98**	**4.66**	**10.80**	**4.51**	**5.96**	**14.05**	**4.77**	**6.82**	**17.09**	**4.94**	**7.54**
w/o GL	✗	✓	Feature-level	6.61	3.89	4.56	10.48	4.40	5.80	13.89	4.67	6.70	16.83	4.83	7.40
w/o SL	✓	✗	Feature-level	6.51	3.81	4.47	10.37	4.31	5.71	13.64	4.57	6.57	16.65	4.74	7.28
w/o CL	✓	✓	w/o	2.05	1.10	1.33	3.45	1.28	1.78	4.79	1.38	2.13	5.88	1.45	2.39
w/o CL (concat)	✓	✓	w/o	6.56	3.85	4.51	10.47	4.36	5.77	13.54	4.60	6.58	16.40	4.76	7.25
w/o CL (sum)	✓	✓	w/o	6.31	3.77	4.40	10.13	4.27	5.62	13.48	4.54	6.51	16.17	4.69	7.14
SL wtih Data-level CL	✗	✓	Data-level	3.49	1.77	2.19	5.89	2.08	2.96	8.09	2.25	3.54	9.98	2.36	3.98
SL wtih Model-level CL	✗	✓	Model-level	3.34	1.92	2.27	5.54	2.21	2.97	7.42	2.35	3.47	9.18	2.45	3.89
GL wtih Feature-level CL	✓	✗	Feature-level	6.37	3.81	4.44	10.17	4.30	5.65	13.23	4.54	6.46	16.08	4.71	7.14
Mirroring Conan	✓	✓ (main)	Feature-level	6.34	3.88	4.49	10.25	4.39	5.74	13.47	4.64	6.59	16.48	4.81	7.30
Dual GL with SDA	✓ (dual)	✗	Feature-level	6.37	3.79	4.42	10.37	4.32	5.71	13.90	4.60	6.64	16.95	4.77	7.37
Dual SL with SDA	✗	✓ (dual)	Feature-level	6.37	3.75	4.39	9.93	4.21	5.53	13.12	4.46	6.38	15.86	4.62	7.02

Table 6
Effects of asymmetric contrastive learning method. For the experimental results for each dataset, the **bold**-faced number is the best score.

Models	@5			@10			@15			@20		
	Recall	MRR	NDCG	Recall	MRR	NDCG	Recall	MRR	NDCG	Recall	MRR	NDCG
Diginetica												
Conan (ours)	**32.13**	**19.00**	**22.26**	**43.53**	**20.52**	**25.94**	**50.64**	**21.08**	**27.82**	**55.73**	**21.37**	**29.02**
SRGNN (main) + SASRec	30.45	17.01	20.33	42.76	18.64	24.31	50.20	19.23	26.27	55.67	19.54	27.57
Homogeneous SRGNN	30.12	16.92	20.18	42.25	18.52	24.10	49.59	19.10	26.04	54.96	19.40	27.30
SRGNN + SASRec (main)	29.64	16.71	19.91	41.68	18.31	23.80	49.12	18.90	25.76	54.28	19.19	26.98
Homogeneous SASRec	29.56	16.68	19.86	41.33	18.24	23.67	48.72	18.83	25.62	53.94	19.12	26.86
SRGNN	27.85	16.23	19.10	38.91	17.70	22.67	45.93	18.25	24.53	51.10	18.54	25.75
SASRec	28.76	16.91	19.85	39.93	18.40	23.45	47.04	18.96	25.33	52.23	19.25	26.56
Retailrocket												
Conan (ours)	**57.32**	**43.48**	**46.95**	**63.99**	**44.38**	**49.12**	**67.32**	**44.64**	**50.00**	**69.46**	**44.76**	**50.51**
SRGNN (main) + SASRec	55.32	40.99	44.58	62.57	41.97	46.93	66.43	42.27	47.95	68.90	42.41	48.54
Homogeneous SRGNN	55.12	40.96	44.51	62.25	41.93	46.83	65.96	42.22	47.81	68.35	42.36	48.38
SRGNN + SASRec (main)	56.43	43.31	46.60	62.37	44.11	48.53	65.41	44.35	49.34	67.36	44.46	49.80
Homogeneous SASRec	54.45	40.76	44.19	61.32	41.69	46.43	64.91	41.97	47.38	67.28	42.11	47.94
SRGNN	54.65	40.90	44.35	61.56	41.84	46.60	65.05	42.11	47.52	67.40	42.25	48.07
SASRec	53.18	41.11	44.12	60.45	42.08	46.48	64.37	42.40	47.52	67.02	42.54	48.15
Dressipi												
Conan (ours)	**23.77**	**14.14**	**16.52**	**32.19**	**15.27**	**19.25**	**37.39**	**15.67**	**20.62**	**41.15**	**15.89**	**21.51**
SRGNN (main) + SASRec	22.02	12.92	15.18	30.03	13.99	17.76	35.06	14.39	19.10	38.80	14.60	19.98
Homogeneous SRGNN	22.29	13.00	15.30	30.40	14.09	17.92	35.36	14.48	19.24	39.15	14.69	20.13
SRGNN + SASRec (main)	21.85	12.89	15.11	29.78	13.95	17.67	34.72	14.34	18.98	38.40	14.55	19.85
Homogeneous SASRec	21.82	12.88	15.09	29.70	13.93	17.64	34.68	14.32	18.96	38.32	14.53	19.82
SRGNN	22.19	13.33	15.52	29.93	14.36	18.02	34.81	14.74	19.31	38.34	14.94	20.15
SASRec	21.58	12.96	15.09	29.41	14.00	17.62	34.47	14.40	18.96	38.21	14.61	19.84
KuaiRand												
Conan (ours)	**6.73**	**3.98**	**4.66**	**10.80**	**4.51**	**5.96**	**14.28**	**4.77**	**6.82**	**17.33**	**4.94**	**7.54**
SRGNN (main) + SASRec	6.52	3.86	4.52	10.47	4.38	5.78	13.96	4.65	6.71	17.07	4.83	7.44
Homogeneous SRGNN	6.56	3.86	4.52	10.66	4.40	5.84	14.05	4.68	6.79	17.09	4.85	7.51
SRGNN + SASRec (main)	6.65	3.95	4.61	10.48	4.46	5.85	13.89	4.72	6.75	16.79	4.89	7.43
Homogeneous SASRec	6.50	3.90	4.54	10.59	4.43	5.85	13.90	4.69	6.72	16.80	4.86	7.41
SRGNN	6.32	3.90	4.49	9.84	4.36	5.62	12.61	4.57	6.35	15.14	4.72	6.95
SASRec	6.49	3.93	4.56	10.43	4.44	5.82	13.85	4.71	6.72	16.92	4.88	7.45

Table 7
Effects of graph learning module. For the experimental results on each dataset, the **bold**-faced number is the best score.

Models	Self-attention	Mamba	Stable	@5			@10			@15			@20		
	Layer	Layer	MLP	Recall	MRR	NDCG	Recall	MRR	NDCG	Recall	MRR	NDCG	Recall	MRR	NDCG
Diginetica															
Conan (ours)	✓	✓	✓	**32.13**	**19.00**	**22.26**	**43.53**	**20.52**	**25.94**	**50.64**	**21.08**	**27.82**	**55.73**	**21.37**	**29.02**
w/o SAN	✗	✓	✓	31.86	18.89	22.10	43.31	20.42	25.81	50.32	20.97	27.67	55.50	21.26	28.89
w/o Mamba	✓	✗	✓	31.66	18.69	21.90	43.05	20.21	25.58	50.18	20.77	27.47	55.37	21.06	28.70
w/o stable MLP	✓	✓	✗	31.87	18.90	22.12	43.48	20.45	25.87	50.57	21.01	27.74	55.59	21.29	28.93
Retailrocket															
Conan (ours)	✓	✓	✓	**57.32**	**43.48**	**46.95**	**63.99**	**44.38**	**49.12**	**67.32**	**44.64**	**50.00**	**69.46**	**44.76**	**50.51**
w/o SAN	✗	✓	✓	56.87	43.27	46.68	63.38	44.15	48.80	66.72	44.42	49.68	68.84	44.53	50.18
w/o Mamba	✓	✗	✓	57.04	43.30	46.74	63.74	44.20	48.92	67.10	44.47	49.81	69.23	44.59	50.31
w/o stable MLP	✓	✓	✗	57.16	43.34	46.80	63.85	44.25	48.98	67.22	44.51	49.87	69.43	44.64	50.39
Dressipi															
Conan (ours)	✓	✓	✓	**23.77**	**14.14**	**16.52**	**32.19**	**15.27**	**19.25**	**37.39**	**15.67**	**20.62**	**41.15**	**15.89**	**21.51**
w/o SAN	✗	✓	✓	23.56	14.05	16.40	31.91	15.16	19.10	37.10	15.57	20.47	40.86	15.78	21.36
w/o Mamba	✓	✗	✓	23.43	13.94	16.29	31.71	15.05	18.97	36.85	15.45	20.33	40.61	15.66	21.22
w/o stable MLP	✓	✓	✗	23.75	14.12	16.50	32.10	15.24	19.20	37.27	15.64	20.57	41.04	15.85	21.46
KuaiRand															
Conan (ours)	✓	✓	✓	**6.73**	**3.98**	**4.66**	**10.80**	**4.51**	**5.96**	**14.05**	**4.77**	**6.82**	**17.09**	**4.94**	**7.54**
w/o SAN	✗	✓	✓	6.59	3.92	4.58	10.54	4.43	5.84	13.94	4.70	6.74	16.93	4.87	7.44
w/o Mamba	✓	✗	✓	6.39	3.78	4.43	10.49	4.32	5.74	13.79	4.58	6.61	16.77	4.74	7.31
w/o MLP	✓	✓	✗	6.34	3.79	4.42	10.43	4.33	5.73	13.89	4.60	6.64	16.87	4.76	7.34

distinct advantages in processing short session sequences. In addition, the Mamba block introduces the unique selection mechanism to filter out irrelevant information and compress the previous context into fixed-size hidden state, which performs better than self-attention for the long sequence feature learning. Moreover, we propose a dual layer norm enhanced MLP $stableMLP$ to capture deeper semantic features and improve training stability. To further improve its performance, we employ the GaMaFormer for graph learning instead of sequence learning, as session graphs contain more contextual information than session sequences.

4.4.3. The effects of sequence learning module

The third group of ablation studies aims to demonstrate the effectiveness of each component in our Bi-Mason. Specifically, we produce

Table 8
Effects of sequence learning module. For the experimental results on each dataset, the **bold**-faced number is the best score.

Models	Shared Parameter	Feature fusion method	Bidirectional Mechanism	@5			@10			@15			@20		
				Recall	MRR	NDCG	Recall	MRR	NDCG	Recall	MRR	NDCG	Recall	MRR	NDCG
Diginetica															
Conan (ours)	✓	Fourier KAN	✓	**32.13**	**19.00**	**22.26**	**43.53**	**20.52**	**25.94**	**50.64**	**21.08**	**27.82**	**55.73**	**21.37**	**29.02**
w/o Shared Parameter	✗	Fourier KAN	✓	31.96	18.96	22.18	43.29	20.46	25.84	50.34	21.02	27.70	55.47	21.31	28.91
MLP fusion	✓	MLP	✓	31.99	18.89	22.14	43.31	20.40	25.79	50.52	20.97	27.70	55.65	21.25	28.91
Gating fusion	✓	Gating network	✓	31.11	18.63	21.72	41.66	20.03	25.13	48.22	20.55	26.87	52.86	20.81	27.96
w/o Bidirectional Mechanism	✓	Fourier KAN	✗	31.89	18.88	22.11	43.39	20.41	25.82	50.52	20.98	27.71	55.51	21.26	28.89
Retailrocket															
Conan (ours)	✓	Fourier KAN	✓	**57.32**	**43.48**	**46.95**	**63.99**	**44.38**	**49.12**	**67.32**	**44.64**	**50.00**	**69.46**	**44.76**	**50.51**
w/o Shared Parameter	✗	Fourier KAN	✓	57.20	43.42	46.87	63.83	44.31	49.03	67.15	44.58	49.91	69.37	44.70	50.43
MLP fusion	✓	MLP	✓	57.13	43.39	46.84	63.81	44.29	49.01	67.22	44.56	49.91	69.42	44.69	50.43
Gating fusion	✓	Gating network	✓	56.83	43.44	46.80	62.98	44.28	48.80	66.08	44.52	49.63	68.09	44.64	50.10
w/o Bidirectional Mechanism	✓	Fourier KAN	✗	57.04	43.34	46.78	63.61	44.23	48.91	66.92	44.49	49.79	69.06	44.61	50.29
Dressipi															
Conan (ours)	✓	Fourier KAN	✓	**23.77**	**14.14**	**16.52**	**32.19**	**15.27**	**19.25**	**37.39**	**15.67**	**20.62**	**41.15**	**15.89**	**21.51**
w/o Shared Parameter	✗	Fourier KAN	✓	23.68	14.08	16.46	32.12	15.20	19.18	37.35	15.62	20.57	41.14	15.83	21.46
MLP fusion	✓	MLP	✓	23.70	14.09	16.47	32.05	15.21	19.17	37.16	15.61	20.52	40.89	15.82	21.40
Gating fusion	✓	Gating network	✓	22.29	13.03	15.32	30.51	14.13	17.98	35.64	14.53	19.34	39.38	14.74	20.22
w/o Bidirectional Mechanism	✓	Fourier KAN	✗	23.58	14.05	16.41	31.95	15.17	19.12	37.06	15.57	20.47	40.79	15.78	21.35
KuaiRand															
Conan (ours)	✓	Fourier KAN	✓	**6.73**	**3.98**	**4.66**	**10.80**	**4.51**	**5.96**	14.05	**4.77**	**6.82**	**17.09**	**4.94**	**7.54**
w/o Shared Parameter	✗	Fourier KAN	✓	6.38	3.84	4.46	10.41	4.36	5.75	13.81	4.63	6.65	16.84	4.80	7.36
MLP fusion	✓	MLP	✓	6.42	3.85	4.48	10.44	4.37	5.77	13.86	4.64	6.67	16.78	4.80	7.36
Gating fusion	✓	Gating network	✓	6.51	3.89	4.53	10.61	4.42	5.84	14.07	4.69	6.75	16.98	4.85	7.44
w/o Bidirectional Mechanism	✓	Fourier KAN	✗	6.52	3.89	4.53	10.51	4.41	5.81	13.86	4.67	6.69	16.89	4.84	7.41

four variants for comparison, including: (1) w/o Shared Parameters, which removes the parameter sharing between the forward Mamba layer and the reverse Mamba layer; (2) MLP fusion, which adopts the vanilla feed-forward network with GELU activation to fuse the bidirectional contextual features; (3) Gating fusion, which adopts the vanilla gating network to perform adaptive fusion of bidirectional contextual features; and (4) w/o Bidirectional Mechanism, which adopts basic Mamba block and the vanilla feed-forward network with GELU activation for sequence learning. The results are shown in Table 8.

From Table 8, we can observe that altering each part of our Bi-Mason leads to a performance decrease and the complete Conan achieves peak performance, demonstrating the effectiveness of the parameter-sharing mechanism, the Fourier KAN-based feature fusion method, and the bidirectional mechanism. Specifically, comparing Conan to w/o Bidirectional Mechanism, we can observe that Conan achieves better performance in most cases, indicating that it is challenging to capture user preferences by simply stacking the basic Mamba blocks. Left-to-right causal modeling falls short in learning optimal representations of session-level user preferences, as each item only captures information from preceding interactions. This approach introduces a rigid order assumption, which may not accurately reflect user behaviors in real-world scenarios. Moreover, the parameter-sharing mechanism is crucial for consistent feature learning.

Furthermore, it is found that Fourier KAN plays a vital role in efficient and effective feature fusion. By adopting 1D Fourier coefficients instead of B-spline coefficients, Fourier KAN offers easier optimization due to the denser nature of Fourier coefficients, which operate on a global scale, in contrast to the local nature of splines. Furthermore, the use of Fourier coefficients benefits from periodicity, making the functions more numerically bounded and helping to avoid issues related to going outside the grid.

4.4.4. The effects of preference learning module

The dual soft attention network is adopted in both the graph learning module and the sequence learning module for session-level preference modeling. The fourth group of ablation studies aims to demonstrate the effectiveness of our dual soft attention network compared to vanilla methods, such as vanilla soft attention network and mean-pooling. Specifically, we produce two variants for comparison, including: (1) Vanilla soft attn, which retains the item-level soft attention mechanism to learn long-term interests based on short-term and contextual references. However, it replaces the second preference-level attention fusion layer with a simple concatenation followed by a linear projection; and (2) Mean-pooling, which computes the average of all updated item representations within a session as the final session-level user preference.

The experimental results on four benchmark datasets are presented in Table 9. It can be observed that the full Conan model equipped with the dual soft attention network consistently outperforms both baseline variants across all evaluation metrics on all datasets. In particular, the performance gap is more prominent on the Diginetica and KuaiRand datasets. For example, on the KuaiRand dataset, our full model achieves 16.2% relative improvement on Recall@5 compared with the mean-pooling method, and also outperforms the vanilla soft attention variant by 11.6% on Recall@5.

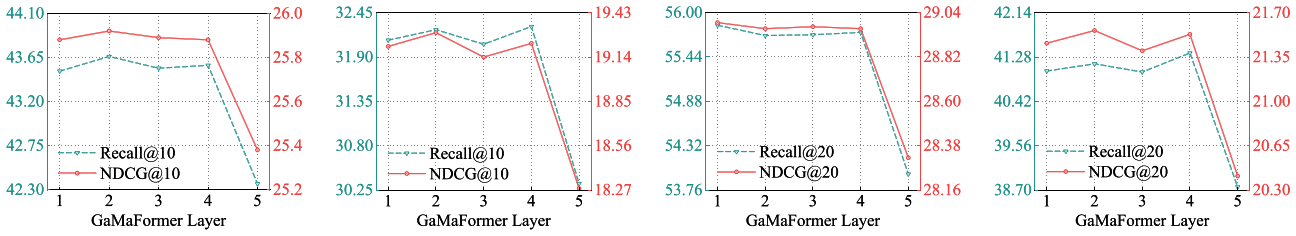
The consistent performance gains verify the rationality of our dual attention design. On one hand, unlike the mean-pooling strategy that treats all interacted items equally, the item-level soft attention can dynamically assign differentiated weights to each item according to its relevance to the user's current interest state, which effectively amplifies the influence of core interactive items and suppresses the noise caused by accidental clicks. On the other hand, compared with the fixed fusion paradigm of vanilla soft attention, the additional preference-level attention layer enables the model to adaptively adjust the contribution ratios of short-term, contextual, and long-term interests according to specific session characteristics. Specifically, for sessions with clear sequential intent, the model automatically emphasizes short-term interest. Additionally, for sessions with scattered interaction behaviors, it relies more on contextual and long-term interests. In summary, this two-layer adaptive weighting mechanism endows the model with finer-grained preference modeling capability.

4.5. Parameter sensitivity

To address RQ3, we examine the impact of the key hyper-parameters on the performance of Conan, including the number of stacking layers

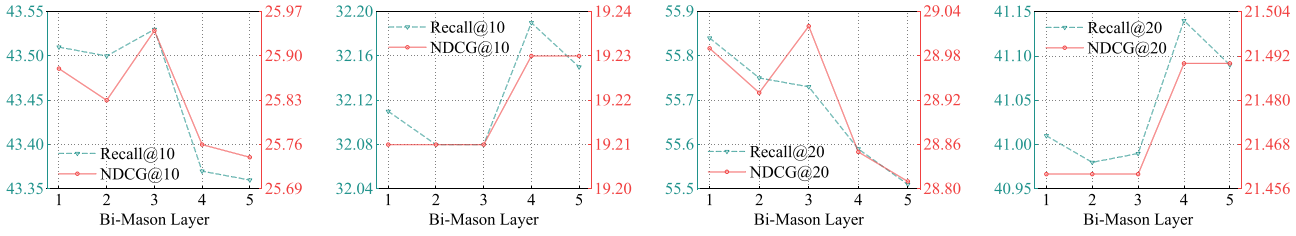
Table 9
Effects of preference learning module. For the experimental results on each dataset, the **bold**-faced number is the best score.

Models	@5			@10			@15			@20		
	Recall	MRR	NDCG	Recall	MRR	NDCG	Recall	MRR	NDCG	Recall	MRR	NDCG
Diginetica												
Conan (ours)	**32.13**	**19.00**	**22.26**	**43.53**	**20.52**	**25.94**	**50.64**	**21.08**	**27.82**	**55.73**	**21.37**	**29.02**
Vanilla soft attn	30.48	17.31	20.57	42.70	18.93	24.51	50.39	19.54	26.54	55.71	19.84	27.80
Mean-pooling	30.41	16.97	20.30	42.80	18.62	24.30	50.28	19.21	26.28	55.67	19.51	27.55
Retailrocket												
Conan (ours)	**57.32**	**43.48**	**46.95**	**63.99**	**44.38**	**49.12**	**67.32**	**44.64**	**50.00**	**69.46**	**44.76**	**50.51**
Vanilla soft attn	57.05	43.25	46.71	63.71	44.15	48.87	67.00	44.41	49.74	69.20	44.53	50.26
Mean-pooling	57.04	43.24	46.70	63.75	44.14	48.88	67.15	44.41	49.78	69.38	44.54	50.31
Dressipi												
Conan (ours)	**23.77**	**14.14**	**16.52**	**32.19**	**15.27**	**19.25**	**37.39**	**15.67**	**20.62**	**41.15**	**15.89**	**21.51**
Vanilla soft attn	22.39	13.09	15.39	30.60	14.18	18.04	35.72	14.59	19.40	39.51	14.80	20.29
Mean-pooling	22.42	13.10	15.41	30.61	14.19	18.06	35.71	14.59	19.41	39.48	14.81	20.29
KuaiRand												
Conan (ours)	6.73	3.98	4.66	10.80	4.51	5.96	14.05	4.77	6.82	17.09	4.94	7.54
Vanilla soft attn	6.03	3.60	4.20	9.49	4.06	5.31	12.39	4.29	6.08	14.90	4.43	6.67
Mean-pooling	5.79	3.61	4.14	8.92	4.02	5.15	11.29	4.20	5.77	13.49	4.33	6.29



(a) Metrics@10 on Digi. (b) Metrics@10 on Dres. (c) Metrics@20 on Digi. (d) Metrics@20 on Dres.

Fig. 3. Performance of conan when varying GaMaFormer layer number on diginetica (Digi.) and dressipi (Dres.).



(a) Metrics@10 on Digi. (b) Metrics@10 on Dres. (c) Metrics@20 on Digi. (d) Metrics@20 on Dres.

Fig. 4. Performance of conan when varying bi-mason layer number on diginetica (Digi.) and dressipi (Dres.).

and the grid size. Moreover, the loss weight that controls the influence of contrastive learning task is also taken into consideration. Due to the limited space, we only report the results of Recall@10, Recall@20, MRR@10, and MRR@20 for the Diginetica and Dressipi datasets. Similar trends can be observed across other metrics and datasets.

4.5.1. The impact of different numbers of stacking layers

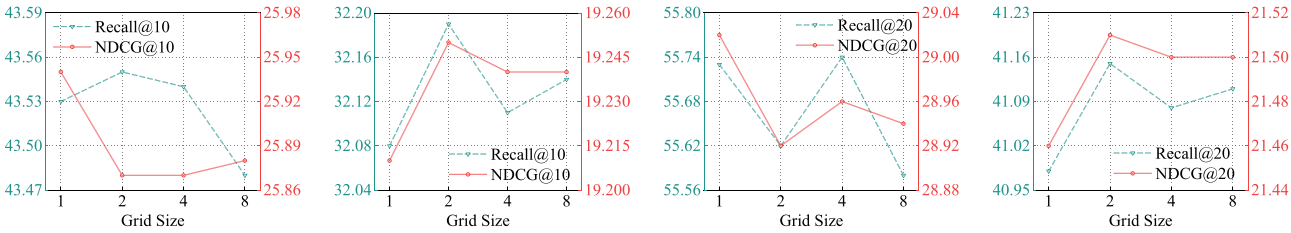
As shown in Figs. 3 and 4, to investigate the influence of the number of stacking layers on the model performance, we tune it to $\{1, 2, 3, 4, 5\}$ for GaMaFormer and Bi-Mason, respectively. For GaMaFormer, we observe that our Conan achieves optimal performance with varying parameters. Specifically, stacking additional layers with GaMaFormer does not enhance performance on Diginetica, while the peak performance on Dressipi is obtained by stacking 2 layers. We attribute the reason to over-smoothing problem which causes homogenization of the information. In other words, it leads to the representations of different items to finally contain the similar information. Moreover, Conan benefits from stacking more layers with Bi-Mason, especially on Diginetica and Dressipi. This is because the complex semantic features derived from deep layers are crucial for delivering self-supervised training signals.

4.5.2. The impact of different grid sizes

As shown in Fig. 5, we evaluate the impact of the grid size from $\{1, 2, 4, 8\}$. It is observed that the best grid size is set to 1 on Diginetica and Retailrocket, while the optimal grid size is set to 2 on Dressipi. The grid size dictates the number of sine and cosine terms included in the Fourier coefficients for each input dimension, which affects the difficulty of training. A larger grid size is essential for longer session sequences on Dressipi to capture complex patterns, while a smaller grid size is sufficient for shorter session sequences on Diginetica.

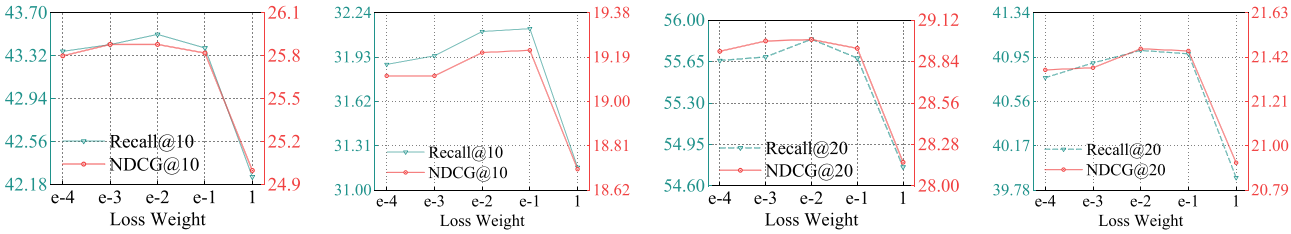
4.5.3. The impact of different loss weights

In Conan, the hyper-parameter is adopted to scale the contrastive loss for self-supervised task. As shown in Fig. 6, to demonstrate the



(a) Metrics@10 on Digi. (b) Metrics@10 on Dres. (c) Metrics@20 on Digi. (d) Metrics@20 on Dres.

Fig. 5. Performance for conan with varying grid sizes on diginetica (Digi.) and dressipi (Dres.).



(a) Metrics@10 on Digi. (b) Metrics@10 on Dres. (c) Metrics@20 on Digi. (d) Metrics@20 on Dres.

Fig. 6. Performance of conan with varying loss weights on diginetica (Digi.) and dressipi (Dres.).

Table 10
Statistics of the three sub-datasets.

Dataset	# Sessions	# Items	# Interactions	Avg. session length	Avg. action per item
Cold-start group	144,939	40,782	444,678	3.07	10.90
Common group	54,984	41,011	462,360	8.41	11.27
Active group	4141	23,071	82,166	19.85	3.56

contribution of each task, we compare the experimental results by tuning the values from $\{0.0001, 0.001, 0.01, 0.1, 1\}$. We observe that Conan consistently achieves the peak performance when the loss weight is set to 0.01 in most cases, while smaller or larger loss weight does not show a tendency toward better performance. The reason is that the incorporated self-supervision signals are insufficient to improve the representations of session-level user preferences when the loss weight is small. By contrast, the large loss weight will force Conan to consider excessive sequential information, thereby preventing non-sequential signal modeling in graph learning module.

4.6. User group study

The user group study addresses RQ4, demonstrating that Conan can deliver effective recommendations for users with varying levels of activity. Specifically, we categorize the users in the Diginetica dataset into three groups based on their activity levels: cold-start users (session length of 5 or fewer), common users (session length of 5 to 15), and active users (session length greater than 15). The dataset statistics are presented in Table 10, which indicate that 71.03% of the users are classified as cold-start users, 26.94% as common users, and 2.03% as active users. This distribution reveals that most real-world sessions are short, making it challenging to capture user behavioral patterns due to limited contextual information. Subsequently, we explore the performance of our model, Conan and seven representative baseline models, i.e., TAGNN [36], SASRec [7], SGNN-HN [2], NARM [39], CORE [24], GCEGNN [54], and CM-GNN [55]. Consistent with the settings of the overall experiment, all models in this user group study are evaluated over 5 independent runs with different random seeds. We perform paired t-tests against the strongest baseline within each user group.

Table 11 shows that Conan outperforms the seven baseline models on all the three sub-datasets. In particular, the performance gains on the cold-start user group also reach statistical significance. These results highlight the effectiveness of Conan in session-based recommendation tasks, even with limited session lengths. Specifically, Conan benefits from the graph learning to extract representative non-sequential item dependencies. Moreover, the sequence learning facilitates Conan in capturing fine-grained temporal item transition patterns. Furthermore, it is found that the graph-sequence learning-based methods achieve better performance than the single learning-based methods in most cases. Specifically, SGNN-HN [2], GCEGNN [54], and CM-GNN [55] consistently outperform TAGNN [36], NARM [39], and SASRec [7]. We also note that the performance of all the models deteriorates as the session length increases. This decline can be attributed to the presence of a greater number of noisy clicks and multiple user intents in longer sessions. Therefore, it is challenging for these models to precisely predict user behaviors under such complex circumstances.

4.7. Model efficiency

The model efficiency experiment addresses RQ5 by comparing the efficiency between our Conan model and four representative baseline models, i.e., MLSA4Rec [52], TAGNN [36], GCEGNN [54], and CM-GNN [55]. We train these models on a server equipped with an Intel(R) Xeon(R) Platinum 8375 C CPU (RAM: 755.51 GB) and a single NVIDIA GeForce RTX 3090 GPU. Moreover, for these models, we maintain the configuration where the embedding dimension and the batch size are set to 100, with the maximum session length fixed at 20. For each epoch, we evaluate and record the computational costs, including GPU memory consumption, training duration, inference time, wall time per forward pass, and number of epochs for training as shown in Table 12.

We can observe that our Conan optimally balances space complexity and time complexity, while achieving promising performance. Specifically, compared to the representative graph-sequence learning-based baseline model GCEGNN [54], our Conan reduces GPU memory consumption by about 94.47% training duration by 52.08%, and inference time by 43.77% on average. The similar trends can be observed in the comparison with another representative graph-sequence learning-based baseline model CM-GNN [55], where our Conan reduces GPU

Table 11
Model performance on the three sub-datasets. For each column, the best performance and the second best performance methods are denoted in **bold** and underlined fonts respectively. * indicates that the improvement over the strongest baseline is statistically significant ($p < 0.05$) based on a paired t-test.

Model	Cold-start group						Common group						Active group					
	@10			@20			@10			@20			@10			@20		
	Recall	MRR	NDCG	Recall	MRR	NDCG	Recall	MRR	NDCG	Recall	MRR	NDCG	Recall	MRR	NDCG	Recall	MRR	NDCG
Conan (ours)	**40.82***	**20.36***	**25.20***	**50.34***	**21.02***	**27.61***	**38.41***	**17.37***	**22.30***	**50.65***	**18.21***	**25.39***	**30.29***	**13.85***	**17.71***	**39.11***	**14.46***	**19.94***
TAGNN	36.14	18.11	22.38	44.58	18.70	24.51	34.21	15.23	19.67	45.40	16.00	22.50	24.35	11.52	14.53	32.00	12.06	16.47
SASRec	36.51	18.30	22.58	46.14	18.96	25.01	33.60	14.98	19.33	45.70	15.82	22.39	27.13	12.63	16.03	36.14	13.24	18.30
CORE	39.16	18.85	23.65	49.48	19.57	26.26	35.88	16.47	21.02	47.04	17.24	23.85	29.23	12.93	16.73	38.09	13.54	18.98
NARM	33.40	15.91	20.03	42.32	16.53	21.41	33.07	14.42	18.78	44.50	15.22	21.67	25.07	11.15	14.42	32.42	11.65	16.27
GCEGNN	38.86	19.36	23.97	48.51	20.03	26.41	37.04	16.26	21.14	48.94	17.09	24.14	29.03	13.73	17.33	35.92	14.21	19.08
SGNN-HN	38.80	19.14	23.78	48.24	19.79	26.17	35.48	15.74	20.36	47.14	16.54	23.31	29.11	13.33	17.03	36.79	13.86	18.97
CM-GNN	38.10	18.93	23.47	47.17	19.56	25.76	35.62	15.61	20.30	47.31	16.41	23.25	28.55	13.12	16.76	36.50	13.67	18.77

Table 12
Efficiency performance across the four datasets.

Dataset	Method	GPU memory (MB)	Training time (s/epoch)	Inference time (s/epoch)	Wall time per forward pass (s/batch)	# epochs for training
Diginetica	MLSA4Rec	820	289.99	55.46	0.17	25
	TAGNN	7456	586.11	99.04	0.34	9
	GCEGNN	13,286	958.99	107.86	0.52	22
	CMGNN	13,288	1323.16	111.21	0.70	12
	Conan (Ours)	858	391.76	63.75	0.22	18
Retailrocket	MLSA4Rec	988	460.41	58.65	0.16	20
	TAGNN	10,124	1155.33	97.45	0.38	9
	GCEGNN	13,286	960.46	108.76	0.32	22
	CMGNN	12,304	2075.36	120.65	0.67	12
	Conan (Ours)	1020	614.98	66.01	0.21	17
Dressipi	MLSA4Rec	584	1537.98	233.28	0.26	32
	TAGNN	3604	1477.09	338.51	0.26	11
	GCEGNN	13,272	4961.69	490.18	0.79	23
	CMGNN	15,480	8201.25	549.19	1.27	34
	Conan (Ours)	598	2058.37	274.36	0.34	18
KuaiRand	MLSA4Rec	450	592.31	10.97	2.36	25
	TAGNN	1558	312.48	11.81	1.27	14
	GCEGNN	13,268	1704.95	25.78	6.76	34
	CMGNN	12,374	2616.58	26.01	10.32	14
	Conan (Ours)	462	772.81	12.67	3.07	19

memory consumption by about 94.41% training duration by 71.53%, and inference time by 47.32% on average. Notably, GCEGNN and CM-GNN adopt a global session graph design, with their memory footprint scaling with the total number of items in the dataset. In contrast, our Conan eliminates this architectural limitation by replacing the global session graph with the lightweight session graph. Moreover, we find that our Conan achieves superior performance while maintaining a comparable computational cost to the representative graph learning-based baseline model TAGNN [36], particularly regarding memory cost and inference time. In addition, although MLSA4Rec [52] requires less training and inference time, our Conan model achieves comparable GPU memory consumption while delivering better performance. To further elaborate the efficiency advantage of the asymmetric contrastive paradigm, we provide parameter counts of the two core modules under the default experimental configuration (embedding dimension = 100). In terms of parameter scale, the graph learning module contains 176,400 parameters, while the sequence learning module contains 148,800 parameters. This verifies that the sequence learning module serving as the contrastive view has a smaller parameter scale than the main graph learning module.

The overall efficiency advantage of Conan stems from the joint optimization of the asymmetric contrastive framework and the lightweight Mamba backbone. On one hand, the proposed asymmetric contrastive learning paradigm reduces redundant computational overhead at the framework level. Conventional symmetric contrastive learning frameworks typically require two independent forward propagations of the same encoder over two augmented views in each training step. By contrast, our heterogeneous dual-branch design performs only one forward pass for each branch, and the sequence learning branch serving as the contrastive view has a much smaller parameter scale. This design yields tangible training efficiency gains. On the other hand, the Mamba-based encoder backbone brings inherent efficiency advantages. As validated by a series of recent works on Mamba-based recommender systems [53,64], the selective state-space model enjoys linear time complexity and constant memory footprint relative to sequence length, which is superior to the quadratic complexity of self-attention architectures.

4.8. Visualized case study

Contrastive learning concentrates on learning a general feature function which maps items to features residing on a hypersphere space [33]. As suggested by Wang and Isola [25], alignment and uniformity of feature distributions are two key properties related to the contrastive loss. In particular, uniformity prefers a feature distribution that preserves maximal information, i.e., the uniform distribution on the unit hypersphere. In this section, the visualized case study addresses RQ6, illustrating the strong capacity for feature learning by our Conan. Specifically, the item embedding weights of each model are projected into two-dimensional feature space by SVD decomposition. Due to the space limit, we conduct all the visualized experiments on the Diginetica dataset, as shown in Fig. 7. The similar trends can be found in two other datasets.

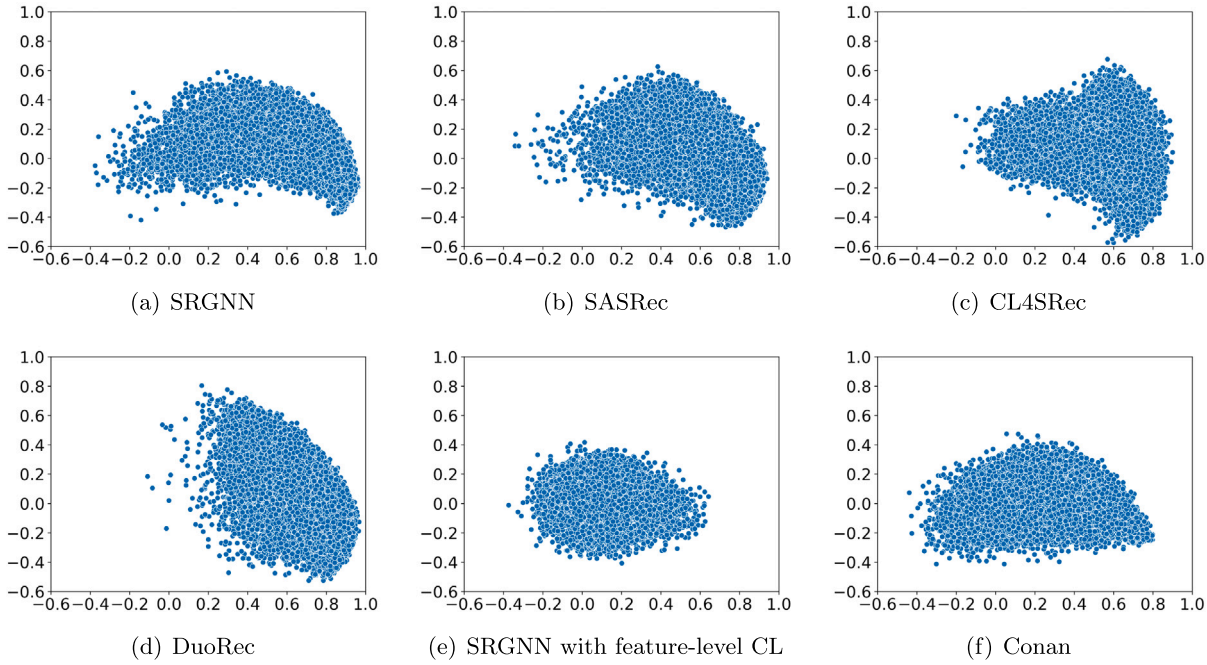


Fig. 7. Distribution visualizations of normalized item embeddings on diginetica dataset. SRGNN with feature-level CL adopts the feature-level contrastive learning [63] to enhance vanilla SRGNN [8].

Comparing (b) to (c) and (b) to (d) in Fig. 7, we observe the distinct feature shift in (c) and (d), showing that traditional CL-enhanced SASRec struggles to exhibit better uniformity than vanilla SASRec. This is because short session sequences are sensitive to the data-level augmentations, while some neural layers are sensitive to the perturbations of neurons. Moreover, comparing (a) to (e), we observe that the item features tend to be concentrated in a limited feature space, demonstrating that the CL-enhanced SRGNN shows worse uniformity than vanilla SRGNN. Although the effective feature-level augmentation [63] is applied, the recommender system is still sensitive to the disturbed representations of items. Furthermore, comparing (e) to (f), we observe that our Conan achieves better uniformity than CL-enhanced SRGNN, indicating that the session-level feature disturbance is more effective at generating the uniform distribution of item features.

5. Related work

This section reviews SBRS based on graph learning, sequence learning, graph-sequence learning, and contrastive learning. Table 13 summarizes these studies in terms of backbone network, session processing strategy, architecture, and the use of contrastive learning.

5.1. Graph learning-based methods

To capture the non-sequential item transition patterns, the GNN-based methods [8,29,35–37,65,67] have been proposed for session-based recommendation. Specifically, the raw session sequences are transformed into various session graphs to represent the complex transitions among items, such as the vanilla session graph [78], the session star graph [79], the global session graph [80], and the session hypergraph [81]. Specifically, the vanilla session graph [78] views items as nodes and constructs edges based on the adjacent item transitions. Furthermore, the session star graph [79] introduces the virtual star node for each session to solve the long-range information propagation problem. Moreover, the global session graph [80] considers the rich inter-session information, while the session hypergraph [81] regards the item transitions as high-order relations. Subsequently, GNNs are adopted to update the features of each item node, such as GGNN [82], GAT [83], and GCN [12]. After that, the readout functions are introduced to learn the session-level user preference by gathering item node features. The popular approach [8] is soft attention which assigns weights to different items and aggregates item features to adaptively infer the comprehensive user intention. Recently, MSERS [68] captures both long-term and latent item dependencies by employing supernode-enhanced GGNN, while CapsGSR [66] combines GraphSAGE with the capsule network to learn multi-faceted item features. Despite their success, it is challenging for these models to capture the sequential item transition patterns because the session-level sequential cues are overlooked during session graph construction.

5.2. Sequence learning-based methods

Considering that user-item interactions are chronologically ordered, various sequential methods are applied to model sequential dependencies. Specifically, RNN-based methods, such as GRU4Rec [6] and HRNN [84], model each ordered session context as a sequence of interactions within the context. Moreover, CNN-based methods, such as Caser [85] and NextItNet [10], treat a batch of session sequences as a session image and focus on capturing local item transition patterns with convolutional filters. Furthermore, attention-based methods, such as SASRec [7] and Bert4Rec [86], assign large weights to items of interest and aggregate item features to learn user preferences. From the neurodynamic optimization perspective, Kong et al. [87] propose a general linear three-step predictor–corrector discretized zeroing neural network for time-varying optimization, which achieves a wider stable stepsize domain and provides methodological reference for stable time-varying preference modeling in sequential recommendation.

However, these methods face several disadvantages. Specifically, compared to Markov chains [88], RNN-based methods [6,84] handle temporal features across more time steps. Unfortunately, due to the recurrent structure, these methods still suffer from sub-optimal training efficiency and gradient problems. CNN-based methods [10,85] are friendly to the modern accelerators, which maintain the high training efficiency for large-scale deployment. However, these methods tend to excessively emphasize the local contextual features, overlooking the session-level user behavioral patterns. Attention-based methods [7,86]

Table 13
Overview of representative related works. We summarize them along five dimensions: view (graph learning, sequence learning, or graph-sequence learning), main backbone network, data processing method, main architecture (stacking or parallel), and whether it adopts contrastive learning. CNN denotes convolutional neural network. GRU denotes gate recurrent unit. LSTM denotes long short-term memory network. GCN denotes graph convolutional network. GAT denotes graph attention network. GGNN denotes gated graph neural network. SSM denotes state space model.

View	Models	Main Architecture	Contrastive Learning	Main Backbone Network	Data Processing Method
Graph Learning	SRGNN [8]	Stacking	✗	GGNN	Vanilla Session Graph
	PosRec [29]	Stacking	✗	GGNN	Vanilla Session Graph
	FGNN [65]	Stacking	✗	GAT	Global Session Graph, Vanilla Session Graph
	CapsGSR [66]	Stacking	✗	GraphSAGE, Capsule network	Vanilla Session Graph
	CGL [67]	Parallel	✓	GGNN, Attention	Global Session Graph, Vanilla Session Graph
	LESSR [37]	Parallel	✗	GGNN, GAT, Attention	Edge-order Preserving Session Graph, Shortcut Session Graph
	MSERS [68]	Parallel	✗	Supernode-enhanced GGNN, GGNN, Attention	Global Session Graph, Supernode-enhanced Global Graph
Sequence Learning	GRU4Rec [6]	Stacking	✗	GRU	Session Sequence
	NextItNet [10]	Stacking	✗	Dilated CNN	Session Sequence
	NARM [39]	Stacking	✗	GRU, Attention	Session Sequence
	GLINT-RU [40]	Stacking	✗	Dense Selective GRU, Linear Attention, Gated MLP	Session Sequence
	MMSBR [69]	Stacking	✓	Attention	Session Sequence
	SLIME4Rec [49]	Stacking	✓	Fourier Transform	Session Sequence
	Mamba4Rec [13]	Stacking	✗	Mamba	Session Sequence
	RecMamba [50]	Stacking	✗	Mamba	Session Sequence
	EchoMamba4Rec [51]	Stacking	✗	Bi-Mamba, Fourier Transform-based Filter	Session Sequence
	MLSA4Rec [52]	Stacking	✗	Mamba, Mamba-LightSANs	Session Sequence
	SIGMA [53]	Stacking	✗	Partially Flipped Mamba, Feature Extract GRU	Session Sequence
	SS4Rec [70]	Stacking	✗	Time-aware SSM, Relation-aware SSM	Session Sequence, Timestamp sequence
	MAINT [71]	Parallel	✗	LSTM, Attention	Session Sequence
	CL4SRec [4]	Parallel	✓	Attention	Session Sequence
	DuoRec [5]	Parallel	✓	Attention	Session Sequence
	Student [72]	Parallel	✓	Attention	Session Sequence
	FEARec [48]	Parallel	✓	Attention, Fourier Transform	Session Sequence
Graph-Sequence Learning	FAPAT [9]	Stacking	✗	GAT	Heterogeneous Session Graph, Session Sequence
	HIDE [73]	Stacking	✗	GAT, Attention	Heterogeneous Session Hypergraph, Session Sequence
	SPARE [11]	Stacking	✓	GCN, Attention	Global Session Graph with Shortest-path
	GES-SASRec [3]	Stacking	✗	LightGCN, Attention	Vanilla Session Graph, Session Sequence, Co-occurrent Item Graph
	SGNN-HN [2]	Stacking	✗	GGNN, Attention	Session Star Graph, Session Sequence
	CoHHN [74]	Parallel	✗	GAT, Attention	Heterogeneous Session Hypergraph, Session Sequence
	GCARM [28]	Parallel	✓	GAT, Attention	Global Session Graph, Session Sequence, Vanilla Session Graph
	RESTC [75]	Parallel	✓	GAT, Attention	Global Session Graph, Session Sequence, Vanilla Session Graph
	STGCR [76]	Parallel	✓	GCN, Attention	Session Hypergraph, Session Sequence, Vanilla Session Graph
	MI-GNN [77]	Parallel	✗	GGNN, Attention	Global Session Graph, Vanilla Session Graph, Session Hypergraph, Session Sequence
	CM-GNN [55]	Parallel	✓	GAT, GCN, Attention	Global Session Graph, Vanilla Session Graph, Session Hypergraph, Session Sequence
	GSAU [56]	Parallel	✓	LightGCN, Attention	Vanilla Session Graph, Session Sequence
	Ours	Parallel	✓	GAT, Attention	Vanilla Session Graph, Session Sequence

focus on relatively interesting items to capture the long-term semantic features and maintain the high training efficiency. However, these methods rely on explicitly storing the entire context for context-based reasoning, resulting in quadratic computational complexity. There is still a lack of efforts to propose more effective and efficient models to solve the session-based recommendation task.

Recently, Mamba [19,89] has already shown its potential in different applications, such as image segmentation [90], in-context learning [14], and sequential recommendation [13]. The Mamba architecture is a linear time varying SSM which overcomes this limit via a selection mechanism allowing the latent state dynamics to change with the current input, thus enabling selective retention of information [91]. In particular, it simultaneously maintains the high training and evaluation efficiency for long sequence modeling [92]. For example, SIGMA [53] leverages a partially flipped Mamba to construct a bidirectional architecture specifically tailored to improve contextual modeling, while an input-sensitive dense selective gate is employed to optimize directional weights and enhance the processing of sequential information. Therefore, it is promising to consider Mamba-based methods for session-based recommendation.

5.3. Graph-sequence learning-based methods

Considering that graph learning and sequence learning focus on different aspects respectively, some efforts are devoted to aggregating their complementary information into a uniform parallel or stacking framework. Specifically, SGNN-HN [2] and FAPAT [9] introduced the stacking framework to learn the sequential and non-sequential item transition patterns, which updates item representations via graph learning and subsequently captures the temporal features via sequence learning. Moreover, MRGSRec [93] adopted the parallel framework to capture the local and global enriched item features by sequential encoder and graph encoder. Recently, IMNE [94] first designs global hypergraph and local heterogeneous hypergraph to capture various higher-order correlations between items, then adopts position-aware attention network to learn session representations. Furthermore, TSREC [95] adaptively transfers semantic information preferences into the item transition module, while the two introduced forms of self-supervised learning further enrich the learned representations.

However, the existing stacking architecture-based methods latently assume that the sequential item transitions captured from sequence learning depend on the non-sequential item transitions captured from graph learning, which leads to the insufficient extraction of temporal features and the biased user preferences. Additionally, the insufficient sequential feature learning [55] and the sub-optimal contrastive learning [75,93] limit the satisfactory performance achievable by many parallel architecture-based methods. Specifically, it is challenging for the mean-pooling approach to provide meaningful session-level representations [55], which restricts the quality of positive and negative samples generated from it. Analysis regarding various contrastive learning methods will be explored in Section 5.4.

5.4. Session-based recommender systems based on contrastive learning

Nowadays, contrastive learning, which deals with massive unlabeled data, has been adopted as an aid to improve session-based recommendation performance. It aims to improve the quality of representations by reducing the distance between positive samples while separating them from negative views in the latent feature space. Therefore, the quality of positive and negative samples influences the effectiveness of contrastive learning. The popular data augmentation methods [4,16] are item masking, item reordering, and cropping. However, TiCoSeRec [96,97] suggested that directly applying invasive augmentation to standard contrastive learning may be useless and even harmful in some scenarios. It is assumed that these data-level augmentations may reduce the amount of critical information, leading to the large difference in quality between the augmented session sequence and the original session sequence. Moreover, some works [5,49] adopted model-level augmentations, such as neuron masking [5], which dynamically modifies the model architecture to augment view pairs on-the-fly. However, these methods face challenges with parameter tuning, as their performance heavily relies on the dropout rate and the number of neurons in each layer. Furthermore, the feature-level augmentation has been proposed to smoothly adjust the uniformity of learned representations via noise injection [63,98]. However, this method also suffers from challenging parameter tuning, particularly regarding noise intensity and the prior distribution of noise. Recently, KAMVG [99] exploits item attributes and keywords through a multi-view graph neural network for session-based recommendation, demonstrating the value of incorporating diverse information sources. Additionally, SCL [100] applies self-supervised contrastive learning to itinerary recommendation, validating the broad applicability of contrastive learning paradigms in recommendation tasks.

To solve these problems, the session-level augmentation [55,101] is introduced for effective contrastive learning. These methods shuffle the session features to generate the session-level negative samples, avoiding tedious parameter tuning and repetitive inefficient session-level feature learning.

6. Conclusion, implications, and future work

This paper presents Conan, a lightweight session-based recommendation framework built on a graph-sequence asymmetric contrastive learning paradigm. It integrates a Mamba-enhanced graph learning module and a bidirectional sequence learning module, and aligns the two views via session-level augmentation to learn comprehensive user preference representations. Extensive experiments on four real-world datasets verify that Conan achieves state-of-the-art recommendation performance while maintaining competitive computational efficiency, and ablation studies further confirm the effectiveness of each core component.

In terms of research implications, this work provides two-fold theoretical insights for session-based recommendation. On one hand, the unified graph-sequence learning framework aligned by asymmetric contrastive learning jointly models sequential and non-sequential item dependencies within a single forward pass, which mitigates the preference bias of single-view modeling and avoids information loss in stacked cascaded architectures. On the other hand, the application of state-space models in both sequence and graph modeling verifies that Mamba's selective mechanism and bidirectional state propagation can serve as an efficient alternative backbone to RNNs and Transformers, supporting linear-complexity long-sequence modeling with competitive representation capacity. In practical applications, the proposed Conan framework achieves a favorable balance between recommendation accuracy and computational efficiency on multiple real-world datasets, making it well suited for large-scale dynamic recommendation scenarios such as e-commerce and streaming media platforms that require low inference latency and high throughput, and it can provide technical support for optimizing core business metrics of industrial recommendation systems.

Despite the promising performance, this work still has several limitations that need to be further addressed. First, the current model is designed for ID-only session recommendation and does not incorporate multi-modal item attributes such as text and images, which limits its ability to leverage rich semantic information when such data is available. Moreover, all experiments are conducted on public benchmark datasets, and the stability and efficiency of the model under an industrial-scale item corpus and real-time traffic still need to be validated through online A/B testing. In the future, we plan to explore the impact of multi-behavior and multi-modality on the performance of recommender systems in session-based recommendation tasks. Additionally, we are interested in evaluating the performance of our Conan in real-world application scenarios through A/B testing.

CRediT authorship contribution statement

Weiyue Li: Writing – original draft, Visualization, Software, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Ming Gao:** Writing – review & editing, Supervision, Resources, Funding acquisition. **Bowei Chen:** Writing – review & editing. **Jingmin An:** Writing – review & editing. **Hao Dong:** Writing – review & editing, Visualization.

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.

Acknowledgements

This research was supported and funded by the National Natural Science Foundation of China (No. 72293563, 72501053); the Basic Scientific Research Project of Liaoning Provincial Department of Education (No. JYTZD2023050); the Natural Science Foundation of Liaoning Province (No. 2024-MS-175); Doctoral Research Startup Fund Plan Project of Liaoning Province (No. 2026-BS-0754); and the Dalian Scientific and Technological Talents Innovation Support Plan (No. 2022RG17). The authors would also like to thank Hongyu Wang of the Institute of Computing Technology, Chinese Academy of Sciences, for helpful discussions related to this work.

Appendix A. The learning process of conan

Algorithm 1 The forward propagation flow of conan.

Input: The session sequences S and the unique item set V.
Output: Top-k recommendation items at the next time step.

Initialize the item embedding matrix E;
/ * *Graph Learning Process* * /
Construct the session graph $G_s = (\mathcal{V}_s, \mathcal{E}_s)$
for each layer of GaMaForemr **do**
  for each item node v_i in G_s **do**
    $e'_i \leftarrow$ Update item node features based on the self-attention layer; ▷ Eqs. (10)-(13)
    $e''_i \leftarrow$ Update item node features based on the Mamba layer; ▷ Eq. (14)
    $e'''_i \leftarrow$ Update item node features based on the stable MLP; ▷ Eq. (15)
  end for
end for
$e^g_i \leftarrow$ Summarize all the GaMaFormer layers; ▷ Eq. (18)
for each session sequence s **do**
  $h^g_{com} \leftarrow$ Learn the session-level user preference based on dual soft attention network; ▷ Eqs. (19)-(23)
end for
/ * *Sequence Learning Process* * /
for each layer of Bi-Mason **do**
  for each session sequence s **do**
    $E'_s \leftarrow$ Update item features based on the forward and reverse Mamba blocks; ▷ Eq. (24)
    $E''_s \leftarrow$ Update item features based on the Fourier Kolmogorov-Arnold network; ▷ Eq. (25)
  end for
end for
Summarize all the GaMaFormer layers to get the refined item features;
for each session sequence s **do**
  Learn the session-level user preference based on another dual soft attention network;
end for
/ * *Prediction layer* * /
for each candidate item $v_i \in V$ **do**
  $y_i \leftarrow$ Calculate the interaction probability; ▷ Eq. (26)
end for
/ * *Training loss calculation* * /
$\mathscr{L}_{main} \leftarrow$ Calculate the recommendation loss; ▷ Eq. (27)
$\mathscr{L}_{cl} \leftarrow$ Calculate the contrastive learning loss; ▷ Eq. (28)
$\mathscr{L} \leftarrow$ Aggregate to get the total loss; ▷ Eq. (29)
Get predicted interaction probability list: $[y_1, y_2, \dots]$;
Select the items with top-K predicted interaction probabilities to form the recommendation list.

(Continued on next column)

Appendix B. Baseline models

1. Traditional Methods
   - **POP** is a non-neural approach recommending the most popular items across the entire training set. Despite its simplicity, POP often serves as a formidable benchmark.
   - **Item-KNN [34]** recommends items similar to the last session item based on cosine similarity between binary item vectors.
   - **CORE [24]** is a simple yet effective framework for session-based recommendation that maintains a consistent representation space throughout encoding and decoding, addressing the issue of inconsistent predictions.
2. Graph Learning-based Methods
   - **SRGNN [8]** transforms session sequences into session graphs and applies a gated graph neural network to capture pairwise item transition relations.
   - **GCSAN [35]** enhances SRGNN by adopting self-attention for capturing long-range item dependencies.
   - **TAGNN [36]** strengthens SRGNN with a target-aware attention network to generate user preferences tailored to different candidate items.
   - **LESSR [37]** addresses information loss in GNN-based SBRSs with lossless edge-order preserving aggregation and shortcut graph attention.
   - **COTREC [38]** is a self-supervised graph co-training framework that iteratively selects evolving pseudo-labels as informative self-supervision examples for enhancing session-based recommendations.
3. Sequence Learning-based Methods
   - RNN-based Methods
     - **GRU4Rec [6]** stacks multiple gated recurrent unit (GRU) layers to encode the session sequence into a final state. It also applies the ranking loss to train the model.
     - **NARM [39]** is the RNN-based method using GRUs for sequential signal capture and attention for emphasizing user intent.
     - **GLINT-RU [40]** leverages a single-layer dense selective GRU module to accelerate inference.
   - Time-unaware Methods
     - **STAMP [41]** replaces all RNN encoders with attention layers, focusing on the representation of the last item in the session. It does not use any kind of positional encoding.
     - **NextItNet [10]** is the CNN-based method that adopts dilated convolutions to increase receptive fields instead of suboptimal pooling operation.

- Transformer Encoder-based Methods
  - **SASRec [7]** is the self-attention based sequential recommender capturing long-term semantic features.
  - **SR-PLR [42]** combines deep learning with symbolic learning and employs Beta embeddings to model logical relationships. Here we adopt the SASRec-based version.
  - **TiSASRec [43]** improves upon SASRec by incorporating both position information and time interval information.
  - **STAR [44]** improves recommendations by leveraging temporal information in session data, embedding items based on co-occurrence in sub-sessions, and adjusting item weights according to time intervals between events.
  - **AC-TSR [45]** calibrates unreliable attention weights from existing Transformer-based models. Here we adopt the TiSASRec-based version.
  - **MiaSRec [46]** captures various user intents by generating multiple session representations for each item and dynamically selecting the most relevant ones.
  - **CL4SRec [4]** leverages contrastive learning with various data augment methods to capture item content-collaborative signal dependencies.
  - **DuoRec [5]** introduces the contrastive regularization to reshape sequence representation distributions.
  - **AdaMCT [47]** explicitly learns personalized behavior preferences to effectively utilize locality inductive bias and model global interactions.
- Fourier Transform-based Methods
  - **FEARec [48]** is the contrastive learning based model adopted time domain attention and auto-correlation.
  - **SLIME4Rec [49]** proposes the dynamic frequency selection and the static frequency split module to capture user dynamic preferences.
- Mamba-based Methods
  - **Mamba4Rec [13]** enhances the basic Mamba block with Transformer techniques for efficient sequential recommendation.
  - **RecMamba [50]** adopts the basic Mamba block to capture long-term user preferences.
  - **EchoMamba4Rec [51]** enhances Mamba-based models with Fourier transform layer, GLU, and bi-directional mechanism.
  - **MLSA4Rec [52]** combines the basic Mamba block with low-rank decomposition self-attention to leverage complementary advantages.
  - **SIGMA [53]** introduces a bidirectional PF-Mamba to allocate the weights for each direction and address the context modeling challenge.

4. Graph-Sequence Learning-based Methods
   - **SGNN-HN [2]** extends SRGNN by introducing session star graph for long-distance information exploration and mitigates over-fitting based on highway networks.
   - **GCEGNN [54]** unifies global-level and session-level transition patterns between items in both the global and session graphs.
   - **CM-GNN [55]** introduces a contrastive multi-level framework to learn session representations from various perspectives.
   - **GSAU [56]** adopts two sub-modules based on graph learning and sequence learning, respectively. Moreover, the two sub-modules share a unified embedding space which is optimized jointly.

Appendix C. Overall performance with baseline models on four datasets

The comparative performance of various competitive baseline models and our Conan on Diginetica, Retailrocket, Dressipi, and KuaiRand are presented in Tables C.14, C.15, C.16, and C.17, respectively.

Table C.14
Recommendation performance of examined models on the Diginetica dataset. For each column, the best performance and the second best performance methods are denoted in **bold** and underlined fonts respectively. * indicates that the improvement over the strongest baseline is statistically significant ($p < 0.05$) based on a paired t-test.

Model	Category	@5			@10			@15			@20		
		Recall	MRR	NDCG	Recall	MRR	NDCG	Recall	MRR	NDCG	Recall	MRR	NDCG
Pop	Traditional Method	0.52	0.23	0.30	0.95	0.29	0.44	1.29	0.32	0.53	1.59	0.33	0.60
Item-KNN		19.27	11.05	13.01	28.34	12.25	15.94	34.48	12.73	17.56	38.99	12.98	18.63
CORE		30.65	18.27	21.34	41.39	19.70	24.81	48.21	20.23	26.61	53.02	20.51	27.75
SRGNN	Graph Learning Method	27.85	16.23	19.10	38.91	17.70	22.67	45.93	18.25	24.53	51.10	18.54	25.75
GCSAN		29.00	17.16	20.09	39.90	18.61	23.61	46.77	19.15	25.43	51.83	19.43	26.62
TAGNN		29.55	17.30	20.33	40.51	18.76	23.87	47.67	19.61	25.77	52.80	19.61	26.98
LESSR		28.42	16.55	19.49	39.73	18.05	23.13	46.70	18.60	24.98	51.80	18.89	26.19
COTREC		24.23	17.02	18.83	27.62	17.48	19.93	29.41	17.62	20.41	30.66	17.69	20.70
GRU4Rec	Sequence Learning	26.98	15.57	18.39	37.98	17.03	21.94	45.00	17.58	23.80	50.14	17.87	25.01
NARM	Method	27.68	15.99	18.88	38.88	17.48	22.50	45.93	18.04	24.37	51.04	18.33	25.57
GLINT-RU		28.62	16.99	19.87	39.61	18.45	23.41	46.47	18.99	25.23	51.57	19.27	26.43
STAMP		25.55	14.87	17.51	35.76	16.23	20.81	42.38	16.75	22.56	47.33	17.03	23.73
NextItNet		20.11	11.10	13.33	29.74	12.38	16.43	36.16	12.89	18.13	40.92	13.15	19.25
SASRec		28.76	16.91	19.85	39.93	18.40	23.45	47.04	18.96	25.33	52.23	19.25	26.56
SR-PLR		27.65	16.38	19.17	38.18	17.78	22.57	45.03	18.32	24.38	50.02	18.60	25.56
TiSASRec		28.42	16.59	19.52	39.70	18.09	23.16	46.92	18.66	25.07	52.26	18.96	26.33
STAR		29.30	18.49	21.17	38.66	19.74	24.19	44.50	20.20	25.74	48.75	20.44	26.74
AC-TSR		26.63	15.80	18.48	37.37	17.22	21.94	44.39	17.77	23.79	49.59	18.06	25.02
MiaSRec		25.16	15.53	17.92	33.20	16.60	20.52	38.28	17.00	21.86	41.95	17.21	22.73
CL4SRec		26.15	15.49	18.13	36.42	16.85	21.44	43.30	17.39	23.26	48.38	17.68	24.46
DuoRec		27.68	16.32	19.13	38.40	17.75	22.59	45.50	18.30	24.47	50.70	18.60	25.70
FEARec		28.36	16.78	19.64	39.18	18.21	23.13	46.20	18.76	24.99	51.38	19.05	26.21
SLIME4Rec		29.93	17.51	20.59	41.50	19.05	24.32	48.74	19.62	26.24	54.05	19.92	27.49
AdaMCT		28.71	16.97	19.87	39.79	18.43	23.44	46.83	18.99	25.31	51.96	19.28	26.52
Mamba4Rec		28.31	16.46	19.39	39.43	17.94	22.98	46.36	18.48	24.81	51.44	18.77	26.01
RecMamba		30.07	17.71	20.77	41.17	19.18	24.35	48.19	19.73	26.21	53.12	20.01	27.37
EchoMamba4Rec		29.49	17.67	20.60	40.03	19.07	24.00	46.62	19.59	25.74	51.38	19.85	26.87
MLSA4Rec		28.62	16.86	19.77	39.37	18.29	23.24	46.11	18.82	25.03	50.94	19.09	26.17
SIGMA		27.05	15.88	18.64	37.84	17.31	22.12	44.92	17.87	23.99	50.03	18.15	25.20
GCEGNN	Graph-Sequence	30.42	17.84	20.96	41.71	19.34	24.60	48.66	19.89	26.44	53.72	20.17	27.64
SGNN-HN	Learning Method	30.31	17.65	20.78	41.42	19.13	24.38	48.39	19.68	26.22	53.44	19.97	27.42
CM-GNN		29.34	17.06	20.10	40.63	18.56	23.75	47.65	19.12	25.60	52.73	19.40	26.80
GSAU		27.55	16.08	18.92	38.48	17.53	22.45	45.62	18.09	24.34	50.79	18.39	25.56
Conan (ours)		**32.02***	**18.95***	**22.19***	**43.42***	**20.47***	**25.88***	**50.59***	**21.04***	**27.77***	**55.69***	**21.32***	**28.98***

Table C.15
Recommendation performance of examined models on the Retailrocket dataset. For each column, the best performance and the second best performance methods are denoted in **bold** and underlined fonts respectively. * indicates that the improvement over the strongest baseline is statistically significant ($p < 0.05$) based on a paired t-test.

Model	Category	@5			@10			@15			@20		
		Recall	MRR	NDCG	Recall	MRR	NDCG	Recall	MRR	NDCG	Recall	MRR	NDCG
Pop	Traditional Method	0.44	0.23	0.28	0.72	0.27	0.37	0.90	0.29	0.42	1.24	0.31	0.50
Item-KNN		9.75	6.36	7.08	13.35	6.83	8.22	15.53	7.00	8.80	17.05	7.09	9.15
CORE		54.60	40.49	44.03	61.92	41.48	46.40	65.73	41.78	47.41	68.22	41.92	48.00
SRGNN	Graph Learning Method	54.65	40.90	44.35	61.56	41.84	46.60	65.05	42.11	47.52	67.40	42.25	48.07
GCSAN		55.56	42.09	45.47	61.92	42.94	47.53	65.21	43.20	48.40	67.44	43.33	48.93
TAGNN		56.72	42.52	46.08	63.35	43.42	48.24	66.58	43.67	49.10	68.72	43.79	49.60
LESSR		55.79	41.76	45.28	62.33	42.64	47.40	65.67	42.91	48.29	67.88	43.03	48.81
COTREC		44.32	40.92	41.79	45.21	40.99	41.95	45.77	41.01	42.03	46.20	41.03	42.08
GRU4Rec	Sequence Learning	54.44	41.17	44.50	61.14	42.08	46.67	64.58	42.35	47.59	66.89	42.48	48.13
NARM	Method	54.88	41.43	44.80	61.63	42.34	46.99	65.19	42.62	47.94	67.49	42.75	48.48
GLINT-RU		56.06	42.69	46.04	62.54	43.57	48.15	65.91	43.83	49.04	68.12	43.96	49.56
STAMP		49.02	38.70	41.27	55.57	39.58	43.40	59.24	39.87	44.37	61.77	40.01	44.97
NextItNet		45.36	33.61	36.54	52.45	34.56	38.84	56.27	34.86	39.85	58.79	35.00	40.45
SASRec		53.18	41.11	44.12	60.45	42.08	46.48	64.37	42.40	47.52	67.02	42.54	48.15
SR-PLR		52.64	40.82	43.78	59.59	41.76	46.03	63.32	42.05	47.02	65.80	42.19	47.60
TiSASRec		56.14	42.33	45.79	62.96	43.25	48.01	66.48	43.53	48.94	68.80	43.66	49.49
STAR		48.73	39.13	41.53	54.26	39.87	43.32	57.37	40.12	44.15	59.35	40.23	44.61
AC-TSR		52.33	40.65	43.57	59.50	41.61	45.89	63.34	41.92	46.91	66.05	42.07	47.55
MiaSRec		51.81	39.76	42.79	56.94	40.45	44.46	59.65	40.67	45.18	61.38	40.77	45.59
CL4SRec		51.29	39.78	42.65	58.22	40.71	44.90	61.95	41.01	45.89	64.53	41.15	46.50
DuoRec		52.53	40.66	43.62	59.80	41.64	45.98	63.79	41.95	47.04	66.45	42.10	47.66
FEARec		52.87	40.77	43.80	60.18	41.76	46.17	64.08	42.06	47.20	66.69	42.21	47.82
SLIME4Rec		55.71	42.17	45.56	62.97	43.14	47.91	66.76	43.44	48.92	69.28	43.59	49.51
AdaMCT		54.20	41.82	44.92	61.20	42.77	47.19	64.90	43.06	48.17	67.34	43.20	48.74
Mamba4Rec		55.80	42.27	45.66	62.26	43.14	47.76	65.66	43.41	48.66	67.87	43.54	49.18
RecMamba		56.65	43.17	46.55	63.11	44.05	48.65	66.37	44.31	49.52	68.55	44.43	50.03
EchoMamba4Rec		55.51	42.77	45.97	61.26	43.54	47.83	64.24	43.78	48.63	66.19	43.89	49.09
MLSA4Rec		55.70	42.43	45.76	61.91	43.27	47.78	65.04	43.52	48.61	67.14	43.63	49.10
SIGMA		55.25	41.77	45.15	61.87	42.66	47.30	65.31	42.93	48.21	67.55	43.06	48.74
GCEGNN	Graph-Sequence Learning Method	56.43	42.02	45.64	63.26	42.94	47.85	66.74	43.22	48.78	69.07	43.35	49.33
SGNN-HN		55.84	41.01	44.73	63.00	41.98	47.06	66.59	42.26	48.01	68.92	42.40	48.56
CM-GNN		55.46	41.14	44.73	62.40	42.07	46.98	65.92	42.35	47.91	68.30	42.48	48.47
GSAU		49.97	37.95	40.95	57.47	38.96	43.38	61.45	39.27	44.43	64.14	39.43	45.07
Conan (ours)		**57.02***	**43.30***	**46.74***	**63.83***	**44.22***	**48.95***	**67.21***	**44.49***	**49.85***	**69.45***	**44.61***	**50.38***

Table C.16
Recommendation performance of examined models on the Dressipi dataset. For each column, the best performance and the second best performance methods are denoted in **bold** and underlined fonts respectively. * indicates that the improvement over the strongest baseline is statistically significant ($p < 0.05$) based on a paired t-test.

Model	Category	@5			@10			@15			@20		
		Recall	MRR	NDCG	Recall	MRR	NDCG	Recall	MRR	NDCG	Recall	MRR	NDCG
Pop	Traditional Method	1.30	0.74	0.87	2.17	0.85	1.14	2.84	0.91	1.32	3.39	0.94	1.45
Item-KNN		13.30	7.94	9.06	19.35	8.76	11.03	23.38	9.09	12.11	26.40	9.26	12.83
CORE		20.44	11.89	14.01	27.99	12.90	16.45	32.66	13.27	17.69	36.10	13.46	18.50
SRGNN	Graph Learning Method	22.19	13.33	15.52	29.93	14.36	18.02	34.81	14.74	19.31	38.34	14.94	20.15
GCSAN		23.05	13.74	16.05	31.06	14.81	18.64	36.11	15.21	19.97	39.77	15.41	20.84
TAGNN		23.35	13.72	16.10	31.60	14.82	18.77	36.69	15.22	20.11	40.38	15.42	20.99
LESSR		21.55	13.01	15.13	29.18	14.03	17.59	33.95	14.40	18.85	37.52	14.60	19.70
COTREC		17.08	12.06	13.31	21.12	12.59	14.61	23.84	12.80	15.33	25.97	12.92	15.83
GRU4Rec	Sequence Learning	22.18	13.12	15.36	30.36	14.21	18.01	35.49	14.61	19.36	39.23	14.82	20.24
NARM	Method	21.79	12.90	15.10	29.90	13.98	17.72	35.02	14.38	19.07	38.79	14.59	19.96
GLINT-RU		23.12	13.95	16.22	31.17	15.02	18.82	36.22	15.42	20.16	39.91	15.63	21.03
STAMP		20.91	12.78	14.80	27.98	13.73	17.08	32.52	14.08	18.28	35.87	14.27	19.07
NextItNet		19.68	11.67	13.66	27.11	12.66	16.05	31.91	13.04	17.32	35.47	13.24	18.17
SASRec		21.58	12.96	15.09	29.41	14.00	17.62	34.47	14.40	18.96	38.21	14.61	19.84
SR-PLR		18.53	11.08	12.92	25.72	12.03	15.24	30.43	12.40	16.49	33.98	12.60	17.33
TiSASRec		23.08	13.99	16.24	31.05	15.05	18.82	36.11	15.45	20.16	39.82	15.66	21.03
STAR		19.89	12.48	14.32	26.16	13.31	16.34	30.20	13.63	17.41	33.24	13.80	18.13
AC-TSR		20.93	12.95	14.93	28.25	13.92	17.29	33.03	14.30	18.56	17.29	14.50	19.40
MiaSRec		20.01	11.94	13.94	27.45	12.93	16.34	32.18	13.30	17.59	35.72	13.50	18.43
CL4SRec		19.30	11.73	13.60	26.33	12.66	15.87	30.99	13.03	17.10	34.48	13.22	17.93
DuoRec		21.17	13.15	15.14	28.42	14.11	17.48	33.16	14.49	18.73	36.66	14.68	19.56
FEARec		20.47	12.71	14.63	27.39	13.62	16.86	31.90	13.98	18.06	35.26	14.17	18.85
SLIME4Rec		22.40	13.61	15.79	30.18	14.64	18.30	35.13	15.03	19.61	38.79	15.24	20.47
AdaMCT		21.26	13.14	15.15	28.70	14.13	17.55	33.59	14.51	18.85	37.19	14.72	19.70
Mamba4Rec		23.08	13.71	16.03	31.40	14.82	18.72	36.59	15.23	20.09	40.36	15.44	20.98
RecMamba		22.47	13.44	15.67	30.54	14.51	18.28	35.57	14.91	19.61	39.31	15.12	20.49
EchoMamba4Rec		23.17	13.85	16.16	31.34	14.94	18.80	36.45	15.34	20.15	40.10	15.55	21.01
MLSA4Rec		21.96	13.09	15.29	29.87	14.15	17.84	34.84	14.54	19.16	38.47	14.74	20.01
SIGMA		22.90	13.60	15.91	31.20	14.71	18.59	36.28	15.11	19.93	40.04	15.32	20.82
GCEGNN	Graph-Sequence Learning Method	**23.81**	**14.10**	**16.51**	**31.95**	**15.18**	**19.14**	36.92	**15.58**	20.45	40.58	**15.78**	21.32
SGNN-HN		22.64	13.14	15.49	30.88	14.24	18.15	36.03	14.64	19.52	39.78	14.85	20.40
CM-GNN		23.61	13.93	16.35	31.78	15.01	18.97	36.78	15.41	20.29	40.48	15.62	21.17
GSAU		20.22	12.53	14.43	27.00	13.43	16.62	31.42	13.78	17.79	34.77	13.97	18.58
Conan (ours)		23.62	14.05	16.42	31.94	15.16	19.11	**37.16***	15.57	**20.49***	**40.90***	**15.78***	**21.37***

Table C.17
Recommendation performance of examined models on the KuaiRand dataset. For each column, the best performance and the second best performance methods are denoted in **bold** and underlined fonts respectively. * indicates that the improvement over the strongest baseline is statistically significant ($p < 0.05$) based on a paired t-test.

Model	Category	@5			@10			@15			@20		
		Recall	MRR	NDCG	Recall	MRR	NDCG	Recall	MRR	NDCG	Recall	MRR	NDCG
Pop	Traditional Method	2.77	1.54	1.84	4.15	1.71	2.28	5.82	1.84	2.72	6.94	1.91	2.98
Item-KNN		5.86	3.11	3.79	9.68	3.61	5.01	12.79	3.85	5.83	15.42	4.00	6.45
CORE		3.72	1.60	2.12	7.36	2.07	3.28	10.84	2.34	4.20	13.67	2.50	4.87
SRGNN	Graph Learning Method	6.32	3.90	4.49	9.84	4.36	5.62	12.61	4.57	6.35	15.14	4.72	6.95
GCSAN		5.85	3.61	4.16	9.47	4.08	5.32	12.47	4.32	6.11	15.02	4.46	6.72
TAGNN		6.39	3.80	4.44	10.15	4.29	5.64	13.10	4.51	6.41	15.98	4.68	7.09
LESSR		6.52	3.85	4.50	10.45	4.36	5.76	13.64	4.61	6.60	16.50	4.77	7.28
COTREC		2.68	1.29	1.63	4.89	1.58	2.34	6.61	1.71	2.79	8.24	1.80	3.18
GRU4Rec	Sequence Learning	6.55	3.91	4.56	10.49	4.43	5.83	13.86	4.70	6.72	16.65	4.85	7.38
NARM	Method	6.51	3.86	4.51	10.27	4.35	5.72	13.58	4.61	6.59	16.38	4.77	7.25
GLINT-RU		6.67	**3.98**	4.64	10.63	4.49	5.91	13.94	4.75	6.78	16.94	4.92	7.49
STAMP		6.28	3.83	4.43	9.83	4.30	5.57	13.03	4.55	6.42	15.67	4.69	7.04
NextItNet		5.83	3.35	3.96	9.24	3.79	5.05	12.22	4.02	5.84	14.77	4.17	6.44
SASRec		6.49	3.93	4.56	10.43	4.44	5.82	13.85	4.71	6.72	16.92	4.88	7.45
SR-PLR		5.71	3.27	3.87	9.03	3.70	4.93	11.79	3.92	5.66	14.43	4.06	6.28
TiSASRec		6.19	3.57	4.21	9.66	4.02	5.32	12.73	4.26	6.13	15.59	4.42	6.81
STAR		4.73	3.20	3.58	6.83	3.47	4.25	8.77	3.62	4.76	10.70	3.73	5.21
AC-TSR		6.56	3.85	4.52	10.29	4.34	5.71	13.60	4.60	6.59	16.38	4.75	7.24
MiaSRec		5.23	2.83	3.42	8.64	3.28	4.51	11.28	3.48	5.21	13.89	3.63	5.82
CL4SRec		4.43	2.20	2.75	7.52	2.61	3.74	10.06	2.81	4.41	12.41	2.94	4.97
DuoRec		6.27	3.70	4.33	9.94	4.18	5.51	12.94	4.42	6.30	15.74	4.57	6.96
FEARec		6.07	3.64	4.24	9.45	4.09	5.32	12.15	4.30	6.04	14.54	4.43	6.61
SLIME4Rec		6.30	3.67	4.32	10.05	4.16	5.52	13.24	4.41	6.37	16.18	4.58	7.06
AdaMCT		6.38	3.85	4.47	9.99	4.32	5.63	13.13	4.56	6.46	15.92	4.72	7.11
Mamba4Rec		6.46	3.90	4.53	10.41	4.41	5.79	13.83	4.68	6.69	17.02	4.85	7.44
RecMamba		6.28	3.81	4.42	9.90	4.29	5.58	12.99	4.53	6.40	15.87	4.69	7.08
EchoMamba4Rec		6.27	3.85	4.44	9.92	4.33	5.62	13.17	4.58	6.48	15.74	4.73	7.08
MLSA4Rec		6.34	3.83	4.45	9.99	4.31	5.62	12.99	4.54	6.41	15.61	4.69	7.03
SIGMA		6.56	3.87	4.53	10.47	4.38	5.79	13.78	4.64	6.66	16.54	4.80	7.31
GCEGNN	Graph-Sequence Learning Method	5.59	3.22	3.80	9.51	3.73	5.06	12.81	3.99	5.93	15.61	4.15	6.59
SGNN-HN		6.32	3.77	4.39	10.41	4.30	5.71	13.53	4.55	6.53	16.32	4.71	7.19
CM-GNN		5.70	3.31	3.90	9.40	3.80	5.08	12.67	4.05	5.95	15.71	4.22	6.67
GSAU		5.79	3.29	3.90	9.42	3.76	5.06	12.56	4.01	5.89	15.37	4.17	6.56
Conan (ours)		**6.73***	**3.98***	**4.66***	**10.80***	**4.51***	**5.96***	**14.05***	**4.77***	**6.82***	**17.09***	**4.94***	**7.54***

Appendix D. Detailed results of recommendation performance

Table D.18
Performance comparison (mean ± standard deviation) between the proposed conan and the strongest baseline for each dataset across 5 independent runs.

Dataset	Model	@5			@10		
		Recall	MRR	NDCG	Recall	MRR	NDCG
Diginetica	STAR	29.3 ± 0.1	18.49 ± 0.7	21.17 ± 0.91	38.66 ± 0.42	19.74 ± 0.43	24.19 ± 0.03
	Conan (ours)	32.02 ± 0.47	18.95 ± 0.23	22.19 ± 0.32	43.42 ± 0.62	20.47 ± 0.44	25.88 ± 0.2
Retailrocket	RecMamba	56.65 ± 0.18	43.17 ± 0.11	46.55 ± 0.23	63.11 ± 0.29	44.05 ± 0.64	48.65 ± 0.45
	Conan (ours)	57.02 ± 0.51	43.3 ± 0.27	46.74 ± 0.7	63.83 ± 0.47	44.22 ± 0.03	48.95 ± 0.12
Dressipi	GCEGNN	23.81 ± 0.77	14.1 ± 0.05	16.51 ± 0.74	31.95 ± 1.06	15.18 ± 0.41	19.14 ± 0.63
	Conan (ours)	23.62 ± 0.55	14.05 ± 0.61	16.42 ± 0.14	31.94 ± 0.69	15.16 ± 0.47	19.11 ± 0.44
KuaiRand	GLINT-RU	6.67 ± 0.31	3.98 ± 0.08	4.64 ± 0.73	10.63 ± 0.73	4.49 ± 0.39	5.91 ± 0.6
	Conan (ours)	6.73 ± 0.58	3.98 ± 0.11	4.66 ± 0.35	10.8 ± 0.51	4.51 ± 0.11	5.96 ± 0.65
Dataset	Model	@15			@20		
		Recall	MRR	NDCG	Recall	MRR	NDCG
Diginetica	STAR	44.5 ± 0.67	20.2 ± 0.05	25.74 ± 0.26	48.75 ± 0.31	20.44 ± 0.31	26.74 ± 0.95
	Conan (ours)	50.59 ± 0.37	21.04 ± 0.58	27.77 ± 0.27	55.69 ± 0.09	21.32 ± 0.16	28.98 ± 0.96
Retailrocket	RecMamba	66.37 ± 0.91	44.31 ± 0.07	49.52 ± 0.83	68.55 ± 1	44.43 ± 0.16	50.03 ± 0.16
	Conan (ours)	67.21 ± 0.92	44.49 ± 0.04	49.85 ± 0.68	69.45 ± 0.21	44.61 ± 0.12	50.38 ± 0.24
Dressipi	GCEGNN	36.92 ± 0.05	15.58 ± 0.85	20.45 ± 0.33	40.58 ± 0.49	15.78 ± 0.05	21.32 ± 0.17
	Conan (ours)	37.16 ± 0.75	15.57 ± 0.02	20.49 ± 0.61	40.9 ± 0.08	15.78 ± 0.36	21.37 ± 0.49
KuaiRand	GLINT-RU	13.94 ± 0.24	4.75 ± 0.42	6.78 ± 0.43	16.94 ± 0.05	4.92 ± 0.19	7.49 ± 0.17
	Conan (ours)	14.05 ± 0.44	4.77 ± 0.46	6.82 ± 0.34	17.09 ± 0.05	4.94 ± 0.68	7.54 ± 0.25

Data availability

Data will be made available on request.

References

[1] Z. Wang, W. Wei, S. Feng, X. Mao, M. Qiu, D. Chen, R. Fang, Exploiting group-level behavior pattern for session-based recommendation, IEEE Trans. Knowl. Data Eng. 36 (1) (2024) 152–166.

[2] Z. Pan, F. Cai, W. Chen, H. Chen, M. de Rijke, Star graph neural networks for session-based recommendation, in: ACM International Conference on Information and Knowledge Management, 2020, pp. 1195–1204.

[3] T. Zhu, L. Sun, G. Chen, Graph-based embedding smoothing for sequential recommendation, IEEE Trans. Knowl. Data Eng. 35 (1) (2023) 496–508.

[4] X. Xie, F. Sun, Z. Liu, S. Wu, J. Gao, J. Zhang, B. Ding, B. Cui, Contrastive learning for sequential recommendation, in: IEEE International Conference on Data Engineering, 2022, pp. 1259–1273.

[5] R. Qiu, Z. Huang, H. Yin, Z. Wang, Contrastive learning for representation degeneration problem in sequential recommendation, in: ACM International Conference on Web Search and Data Mining, 2022, pp. 813–823.

[6] B. Hidasi, A. Karatzoglou, L. Baltrunas, D. Tikk, Session-based recommendations with recurrent neural networks, in: International Conference on Learning Representations, 2016, pp. 1–10.

[7] W. Kang, J.J. McAuley, Self-attentive sequential recommendation, in: IEEE International Conference on Data Mining, 2018, pp. 197–206.

[8] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, T. Tan, Session-based recommendation with graph neural networks, in: AAAI Conference on Artificial Intelligence, 2019, pp. 346–353.

[9] X. Liu, Z. Li, Y. Gao, J. Yang, T. Cao, Z. Wang, B. Yin, Y. Song, Enhancing user intent capture in session-based recommendation with attribute patterns, in: Conference on Neural Information Processing Systems, 2023, pp. 1–19.

[10] F. Yuan, A. Karatzoglou, I. Arapakis, J.M. Jose, X. He, A simple convolutional generative network for next item recommendation, in: ACM International Conference on Web Search and Data Mining, 2019, pp. 582–590.

[11] A. Peintner, A.R. Mohammadi, E. Zangerle, SPARE: shortest path global item relations for efficient session-based recommendation, in: ACM Conference on Recommender Systems, 2023, pp. 58–69.

[12] Y. Yang, C. Huang, L. Xia, C. Huang, D. Luo, K. Lin, Debiased contrastive learning for sequential recommendation, in: International World Wide Web Conference, 2023, pp. 1063–1073.

[13] C. Liu, J. Lin, J. Wang, H. Liu, J. Caverlee, Mamba4Rec: towards efficient sequential recommendation with selective state space models, CoRR arXiv:2403.03900, 2024.

[14] J. Park, J. Park, Z. Xiong, N. Lee, J. Cho, S. Oymak, K. Lee, D. Papailiopoulos, Can Mamba learn how to learn? A comparative study on in-context learning tasks, CoRR arXiv:2402.04248, 2024.

[15] W. Yu, X. Wang, MambaOut: do we really need Mamba for vision? CoRR arXiv:2405.07992, 2024.

[16] M. Yin, H. Wang, X. Xu, L. Wu, S. Zhao, W. Guo, Y. Liu, R. Tang, D. Lian, E. Chen, APGL4SR: a generic framework with adaptive and personalized global collaborative information in sequential recommendation, in: ACM International Conference on Information and Knowledge Management, 2023, pp. 3009–3019.

[17] Z. Zhang, X. Wang, H. Chen, H. Li, W. Zhu, Disentangled dynamic graph attention network for out-of-distribution sequential recommendation, ACM Trans. Inf. Syst. 43 (1) (2025) 19:1–19:42.

[18] H. Tang, S. Wu, X. Sun, J. Zeng, G. Xu, Q. Li, TCGC: temporal collaboration-aware graph co-evolution learning for dynamic recommendation, ACM Trans. Inf. Syst. 43 (1) (2025) 5:1–5:27.

[19] A. Gu, T. Dao, Mamba: linear-time sequence modeling with selective state spaces, CoRR arXiv:2312.00752, 2023.

[20] O. Lieber, B. Lenz, H. Bata, G. Cohen, J. Osin, I. Dalmedigos, E. Safahi, S. Meirom, Y. Belinkov, S. Shalev-Shwartz, O. Abend, R. Alon, T. Asida, A. Bergman, R. Glozman, M. Gokhman, A. Manevich, N. Ratner, N. Rozen, E. Shwartz, M. Zusman, Y. Shoham, Jamba: a hybrid Transformer-Mamba language model, CoRR arXiv:2403.19887, 2024.

[21] J. Xu, Z. Chen, J. Li, S. Yang, W. Wang, X. Hu, E.C.H. Ngai, FourierKAN-GCF: Fourier Kolmogorov-Arnold network - an effective and efficient feature transformation for graph collaborative filtering, CoRR arXiv:2406.01034, 2024.

[22] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljacic, T.Y. Hou, M. Tegmark, KAN: Kolmogorov-Arnold networks, CoRR arXiv:2404.19756, 2024.

[23] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, M. Wang, LightGCN: simplifying and powering graph convolution network for recommendation, in: International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 639–648.

[24] Y. Hou, B. Hu, Z. Zhang, W.X. Zhao, CORE: simple and effective session-based recommendation within consistent representation space, in: International ACM SIGIR Conference on Research and Development in Information Retrieval, 2022, pp. 1796–1801.

[25] T. Wang, P. Isola, Understanding contrastive representation learning through alignment and uniformity on the hypersphere, in: International Conference on Machine Learning, Vol. 119 of Proceedings of Machine Learning Research, 2020, pp. 9929–9939.

[26] X. Chen, K. He, Exploring simple Siamese representation learning, in: IEEE Conference on Computer Vision and Pattern Recognition, 2021, pp. 15750–15758.

[27] J. Grill, F. Strub, F. Altché, C. Tallec, P.H. Richemond, E. Buchatskaya, C. Doersch, B.Á Pires, Z. Guo, M.G. Azar, B. Piot, K. Kavukcuoglu, R. Munos, M. Valko, Bootstrap your own latent - a new approach to self-supervised learning, in: Advances in Neural Information Processing Systems, 2020, pp. 1–35.

[28] Z. Pan, F. Cai, W. Chen, H. Chen, Graph co-attentive session-based recommendation, ACM Trans. Inf. Syst. 40 (4) (2022) 67:1–67:31.

[29] R. Qiu, Z. Huang, T. Chen, H. Yin, Exploiting positional information for session-based recommendation, ACM Trans. Inf. Syst. 40 (2) (2022) 35:1–35:24.

[30] Y.K. Tan, X. Xu, Y. Liu, Improved recurrent neural networks for session-based recommendations, in: Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, 2016, pp. 17–22.

[31] W.X. Zhao, S. Mu, Y. Hou, Z. Lin, Y. Chen, X. Pan, K. Li, Y. Lu, H. Wang, C. Tian, Y. Min, Z. Feng, X. Fan, X. Chen, P. Wang, W. Ji, Y. Li, X. Wang, J. Wen, RecBole: towards a unified, comprehensive and efficient framework for recommendation algorithms, in: ACM International Conference on Information and Knowledge Management, 2021, pp. 4653–4664.

[32] W.X. Zhao, Y. Hou, X. Pan, C. Yang, Z. Zhang, Z. Lin, J. Zhang, S. Bian, J. Tang, W. Sun, Y. Chen, L. Xu, G. Zhang, Z. Tian, C. Tian, S. Mu, X. Fan, X. Chen, J. Wen, RecBole 2.0: towards a more up-to-date recommendation library, in: ACM International Conference on Information & Knowledge Management, 2022, pp. 4722–4726.
[33] F. Wang, H. Liu, Understanding the behaviour of contrastive loss, in: IEEE Conference on Computer Vision and Pattern Recognition, 2021, pp. 2495–2504.
[34] B.M. Sarwar, G. Karypis, J.A. Konstan, J. Riedl, Item-based collaborative filtering recommendation algorithms, in: International World Wide Web Conference, 2001, pp. 285–295.
[35] C. Xu, P. Zhao, Y. Liu, V.S. Sheng, J. Xu, F. Zhuang, J. Fang, X. Zhou, Graph contextualized self-attention network for session-based recommendation, in: International Joint Conference on Artificial Intelligence, 2019, pp. 3940–3946.
[36] F. Yu, Y. Zhu, Q. Liu, S. Wu, L. Wang, T. Tan, TAGNN: target attentive graph neural networks for session-based recommendation, in: International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 1921–1924.
[37] T. Chen, R.C. Wong, Handling information loss of graph neural networks for session-based recommendation, in: ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2020, pp. 1172–1180.
[38] X. Xia, H. Yin, J. Yu, Y. Shao, L. Cui, Self-supervised graph co-training for session-based recommendation, in: ACM International Conference on Information and Knowledge Management, 2021, pp. 2180–2190.
[39] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, J. Ma, Neural attentive session-based recommendation, in: ACM on Conference on Information and Knowledge Management, 2017, pp. 1419–1428.
[40] S. Zhang, M. Wang, W. Wang, J. Gao, X. Zhao, Y. Yang, X. Wei, Z. Liu, T. Xu, GLINT-RU: gated lightweight intelligent recurrent units for sequential recommender systems, in: ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2025, pp. 1948–1959.
[41] Q. Liu, Y. Zeng, R. Mokhosi, H. Zhang, STAMP: short-term Attention/Memory priority model for session-based recommendation, in: ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 1831–1839.
[42] H. Yuan, P. Zhao, X. Xian, G. Liu, Y. Liu, V.S. Sheng, L. Zhao, Sequential recommendation with probabilistic logical reasoning, in: International Joint Conference on Artificial Intelligence, 2023, pp. 2432–2440.
[43] J. Li, Y. Wang, J.J. McAuley, Time interval aware self-attention for sequential recommendation, in: ACM International Conference on Web Search and Data Mining, 2020, pp. 322–330.
[44] R. Yeganegi, S. Haratizadeh, M. Ebrahimi, STAR: a session-based time-aware recommender system, Neurocomputing 573 (2024) 127104.
[45] P. Zhou, Q. Ye, Y. Xie, J. Gao, S. Wang, J.B. Kim, C. You, S. Kim, Attention calibration for transformer-based sequential recommendation, in: ACM International Conference on Information and Knowledge Management, 2023, pp. 3595–3605.
[46] M. Choi, H. Kim, H. Cho, J. Lee, Multi-intent-aware session-based recommendation, in: International ACM SIGIR Conference on Research and Development in Information Retrieval, 2024, pp. 2532–2536.
[47] J. Jiang, P. Zhang, Y. Luo, C. Li, J.B. Kim, K. Zhang, S. Wang, X. Xie, S. Kim, AdaMCT: adaptive mixture of CNN-transformer for sequential recommendation, in: ACM International Conference on Information and Knowledge Management, 2023, pp. 976–986.
[48] X. Du, H. Yuan, P. Zhao, J. Qu, F. Zhuang, G. Liu, Y. Liu, V.S. Sheng, Frequency enhanced hybrid attention network for sequential recommendation, in: International ACM SIGIR Conference on Research and Development in Information Retrieval, 2023, pp. 78–88.
[49] X. Du, H. Yuan, P. Zhao, J. Fang, G. Liu, Y. Liu, V.S. Sheng, X. Zhou, Contrastive enhanced slide filter mixer for sequential recommendation, in: IEEE International Conference on Data Engineering, 2023, pp. 2673–2685.
[50] J. Yang, Y. Li, J. Zhao, H. Wang, M. Ma, J. Ma, Z. Ren, M. Zhang, X. Xin, Z. Chen, P. Ren, Uncovering selective state space model's capabilities in lifelong sequential recommendation, CoRR arXiv:2403.16371, 2024.
[51] Y. Wang, X. He, S. Zhu, EchoMamba4Rec: harmonizing bidirectional state space models with spectral filtering for advanced sequential recommendation, CoRR arXiv:2406.02638, 2024.
[52] J. Su, Z. Huang, MLSA4Rec: Mamba combined with low-rank decomposed self-attention for sequential recommendation, CoRR arXiv:2407.13135, 2024.
[53] Z. Liu, Q. Liu, Y. Wang, W. Wang, P. Jia, M. Wang, Z. Liu, Y. Chang, X. Zhao, SIGMA: selective gated Mamba for sequential recommendation, in: AAAI Conference on Artificial Intelligence, 2025, pp. 12264–12272.
[54] Z. Wang, W. Wei, G. Cong, X. Li, X. Mao, M. Qiu, Global context enhanced graph neural networks for session-based recommendation, in: International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 169–178.
[55] F. Wang, X. Gao, Z. Chen, L. Lyu, Contrastive multi-level graph neural networks for session-based recommendation, IEEE Trans. Multimed. 25 (2023) 9278–9289.
[56] Y. Cao, L. Yang, Z. Liu, Y. Liu, C. Wang, Y. Liang, H. Peng, P.S. Yu, Graph-sequential alignment and uniformity: toward enhanced recommendation systems, in: International World Wide Web Conference, 2025, pp. 888–892.
[57] H. Wang, X. Liu, W. Fan, X. Zhao, V. Kini, D.P. Yadav, F. Wang, Z. Wen, H. Liu, Rethinking large language model architectures for sequential recommendations, in: Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, 2025, pp. 3376–3391.
[58] Y. Jiang, X. Ren, L. Xia, D. Luo, K. Lin, C. Huang, RecGPT: a foundation model for sequential recommendation, in: Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, 2025, pp. 10129–10143.
[59] J. Lin, X. Dai, Y. Xi, W. Liu, B. Chen, H. Zhang, Y. Liu, C. Wu, X. Li, C. Zhu, H. Guo, Y. Yu, R. Tang, W. Zhang, How can recommender systems benefit from large language models: a survey, ACM Trans. Inf. Syst. 43 (2) (2025) 28:1–28:47.
[60] Z. Zhao, W. Fan, J. Li, Y. Liu, X. Mei, Y. Wang, Z. Wen, F. Wang, X. Zhao, J. Tang, Q. Li, Recommender systems in the era of large language models (LLMs), IEEE Trans. Knowl. Data Eng. 36 (11) (2024) 6889–6907.
[61] E.J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, LoRA: low-rank adaptation of large language models, in: The Tenth International Conference on Learning Representations, 2022, pp. 1–26.
[62] S. Qiao, W. Zhou, J. Wen, C. Gao, Q. Luo, P. Chen, Y. Li, Multi-view intent learning and alignment with large language models for session-based recommendation, ACM Trans. Inf. Syst. 43 (4) (2025) 91:1–91:25.
[63] J. Yu, X. Xia, T. Chen, L. Cui, N.Q.V. Hung, H. Yin, XSimGCL: towards extremely simple graph contrastive learning for recommendation, IEEE Trans. Knowl. Data Eng. 36 (2) (2024) 913–926.
[64] Y. Liu, L. Qi, X. Mao, W. Liu, X. Fan, Q. Ni, X. Zhang, Y. Zhang, Y. Tian, A. Beheshti, Hyperbolic-enhanced mixture-of-experts Mamba for sequential recommendation, in: Fortieth AAAI Conference on Artificial Intelligence, 2026, pp. 15403–15411.
[65] R. Qiu, Z. Huang, J. Li, H. Yin, Exploiting cross-session information for session-based recommendation with graph neural networks, ACM Trans. Inf. Syst. 38 (3) (2020) 22:1–22:23.
[66] D.E. Alaoui, J. Riffi, A. Sabri, B. Aghoutane, A. Yahyaouy, H. Tairi, A novel session-based recommendation system using capsule graph neural network, Neural Netw. 185 (2025) 107176.
[67] Z. Pan, F. Cai, W. Chen, C. Chen, H. Chen, Collaborative graph learning for session-based recommendation, ACM Trans. Inf. Syst. 40 (4) (2022) 72:1–72:26.
[68] R. Lin, C. Liu, H. Zhong, C. Yuan, G. Chen, Y. Jiang, Y. Tang, Motif and supernode-enhanced gated graph neural networks for session-based recommendation, Neural Netw. 187 (2025) 107406.
[69] X. Zhang, B. Xu, F. Ma, C. Li, L. Yang, H. Lin, Beyond co-occurrence: multi-modal session-based recommendation, IEEE Trans. Knowl. Data Eng. 36 (4) (2024) 1450–1462.
[70] W. Xiao, H. Wang, Q. Zhou, Q. Wang, SS4Rec: continuous-time sequential recommendation with state space models, CoRR arXiv:2502.08132, 2025.
[71] H. Liu, J. Ding, Y. Zhu, F. Tang, J. Yu, R. Jiang, Z. Guo, Modeling multi-aspect preferences and intents for multi-behavioral sequential recommendation, Knowl.-Based Syst. 280 (2023) 111013.
[72] X. Xia, J. Yu, Q. Wang, C. Yang, N.Q.V. Hung, H. Yin, Efficient on-device session-based recommendation, ACM Trans. Inf. Syst. 41 (4) (2023) 102:1–102:24.
[73] Y. Li, C. Gao, H. Luo, D. Jin, Y. Li, Enhancing hypergraph neural networks with intent disentanglement for session-based recommendation, in: International ACM SIGIR Conference on Research and Development in Information Retrieval, 2022, pp. 1997–2002.
[74] X. Zhang, B. Xu, L. Yang, C. Li, F. Ma, H. Liu, H. Lin, Price DOES matter!: Modeling price and interest preferences in session-based recommendation, in: International ACM SIGIR Conference on Research and Development in Information Retrieval, 2022, pp. 1684–1693.
[75] Z. Wan, X. Liu, B. Wang, J. Qiu, B. Li, T. Guo, G. Chen, Y. Wang, Spatio-temporal contrastive learning-enhanced GNNs for session-based recommendation, ACM Trans. Inf. Syst. 42 (2) (2024) 58:1–58:26.
[76] H. Wang, S. Yan, C. Wu, L. Han, L. Zhou, Cross-view temporal graph contrastive learning for session-based recommendation, Knowl.-Based Syst. 264 (2023) 110304.
[77] T. Wang, C. Chen, J. Huang, S. Huang, Modeling cross-session information with multi-interest graph neural networks for the next-item recommendation, ACM Trans. Knowl. Discov. Data 17 (1) (2023) 1:1–1:28.
[78] S. Qiao, W. Zhou, J. Wen, H. Wang, L. Hu, S. Ni, Multi-perspective enhanced representation for effective session-based recommendation, Knowl.-Based Syst. 263 (2023) 110284.
[79] J. Yuan, W. Ji, D. Zhang, J. Pan, X. Wang, Micro-behavior encoding for session-based recommendation, in: IEEE International Conference on Data Engineering, IEEE, 2022, pp. 2886–2899.
[80] N. Qiu, B. Gao, H. Tu, F. Huang, Q. Guan, W. Luo, LDGC-SR: integrating long-range dependencies and global context information for session-based recommendation, Knowl.-Based Syst. 248 (2022) 108894.
[81] Z. Yin, K. Han, P. Wang, X. Zhu, H3GNN: hybrid hierarchical HyperGraph neural network for personalized session-based recommendation, ACM Trans. Inf. Syst. 42 (3) (2024) 63:1–63:30.
[82] Z. Deng, C. Wang, L. Huang, J. Lai, P.S. Yu, G^3SR: global graph guided session-based recommendation, IEEE Trans. Neural Netw. Learn. Syst. 34 (12) (2023) 9671–9684.
[83] X. Zhang, B. Xu, Y. Wu, Y. Zhong, H. Lin, F. Ma, FineRec: exploring fine-grained sequential recommendation, in: International ACM SIGIR Conference on Research and Development in Information Retrieval, 2024, pp. 1599–1608.
[84] M. Quadrana, A. Karatzoglou, B. Hidasi, P. Cremonesi, Personalizing session-based recommendations with hierarchical recurrent neural networks, in: ACM Conference on Recommender Systems, 2017, pp. 130–137.
[85] J. Tang, K. Wang, Personalized top-N sequential recommendation via convolutional sequence embedding, in: ACM International Conference on Web Search and Data Mining, 2018, pp. 565–573.
[86] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, P. Jiang, BERT4Rec: sequential recommendation with bidirectional encoder representations from transformer, in: ACM

International Conference on Information and Knowledge Management, 2019, pp. 1441–1450.
[87] Y. Kong, X. Chen, Y. Jiang, D. Sun, J. Zhang, Novel discretized zeroing neural network models for time-varying optimization aided with predictor-corrector methods, IEEE Trans. Neural Networks Learn. Syst. 36 (8) (2025) 14037–14048.
[88] S. Rendle, C. Freudenthaler, L. Schmidt-Thieme, Factorizing personalized Markov chains for next-basket recommendation, in: International World Wide Web Conference, 2010, pp. 811–820.
[89] T. Dao, A. Gu, Transformers are SSMs: generalized models and efficient algorithms through structured state space duality, CoRR arXiv:2405.21060, 2024.
[90] X. Liu, C. Zhang, L. Zhang, Vision Mamba: a comprehensive survey and taxonomy, CoRR arXiv:2405.04404, 2024.
[91] B.N. Patro, V.S. Agneeswaran, Mamba-360: survey of state space models as transformer alternative for long sequence modelling: methods, applications, and challenges, CoRR arXiv:2404.16112, 2024.
[92] X. Wang, S. Wang, Y. Ding, Y. Li, W. Wu, Y. Rong, W. Kong, J. Huang, S. Li, H. Yang, Z. Wang, B. Jiang, C. Li, Y. Wang, Y. Tian, J. Tang, State space model for new-generation network alternative to transformers: a survey, CoRR arXiv:2404.09516, 2024.
[93] V. Baikalov, E. Frolov, End-to-end graph-sequential representation learning for accurate recommendations, in: International World Wide Web Conference, 2024, pp. 501–504.
[94] L. Guo, S. Zhou, H. Tang, X. Zheng, Y. Luo, Multi-behavior hypergraph contrastive learning for session-based recommendation, IEEE Trans. Knowl. Data Eng. 37 (3) (2025) 1325–1338.
[95] H. Fu, Z. Qin, W. Xue, G. Ding, Fusing temporal and semantic dependencies for session-based recommendation, Inf. Process. Manag. 62 (1) (2025) 103896.
[96] Y. Dang, E. Yang, G. Guo, L. Jiang, X. Wang, X. Xu, Q. Sun, H. Liu, TiCoSeRec: augmenting data to uniform sequences by time intervals for effective recommendation, IEEE Trans. Knowl. Data Eng. 36 (6) (2024) 2686–2700.
[97] Y. Dang, E. Yang, G. Guo, L. Jiang, X. Wang, X. Xu, Q. Sun, H. Liu, Uniform sequence better: time interval aware data augmentation for sequential recommendation, in: AAAI Conference on Artificial Intelligence, 2023, pp. 4225–4232.
[98] C. Liu, C. Yu, N. Gui, Z. Yu, S. Deng, SimGCL: graph contrastive learning by finding homophily in heterophily, Knowl. Inf. Syst. 66 (3) (2024) 2089–2114.
[99] L. Chen, X. Zhu, G. Zhu, Exploiting attributes and keywords for session-based recommendation with multi-view graph neural network, Expert Syst. Appl. 296 (2026) 128990.
[100] L. Chen, G. Zhu, Self-supervised contrastive learning for itinerary recommendation, Expert Syst. Appl. 268 (2025) 126246.
[101] F. Wang, X. Lu, L. Lyu, CGSNet: contrastive graph self-attention network for session-based recommendation, Knowl.-Based Syst. 251 (2022) 109282.