Deep Situation-Aware Interaction Network for Click-Through Rate Prediction

Yimin Lv
Institute of Software, Chinese Academy of Sciences, University of Chinese Academy of Sciences
Beijing, China
lvyimin21@mails.ucas.ac.cn

Shuli Wang
Meituan
Beijing, China
shuliw1996@gmail.com

Beihong Jin*
Institute of Software, Chinese Academy of Sciences, University of Chinese Academy of Sciences
Beijing, China
Beihong@iscas.ac.cn

Yisong Yu
Institute of Software, Chinese Academy of Sciences, University of Chinese Academy of Sciences
Beijing, China
yuyisong20@mails.ucas.ac.cn

Yapeng Zhang
Meituan
Beijing, China
zhangyapeng05@meituan.com

Jian Dong
Meituan
Beijing, China
dongjian03@meituan.com

Yongkang Wang
Meituan
Beijing, China
wangyongkang03@meituan.com

Xingxing Wang
Meituan
Beijing, China
wangxingxing04@meituan.com

Dong Wang
Meituan
Beijing, China
wangdong07@meituan.com

ABSTRACT

User behavior sequence modeling plays a significant role in Click-Through Rate (CTR) prediction on e-commerce platforms. Except for the interacted items, user behaviors contain rich interaction information, such as the behavior type, time, location, etc. However, so far, the information related to user behaviors has not yet been fully exploited. In the paper, we propose the concept of a situation and situational features for distinguishing interaction behaviors and then design a CTR model named Deep Situation-Aware Interaction Network (DSAIN). DSAIN first adopts the reparameterization trick to reduce noise in the original user behavior sequences. Then it learns the embeddings of situational features by feature embedding parameterization and tri-directional correlation fusion. Finally, it obtains the embedding of behavior sequence via heterogeneous situation aggregation. We conduct extensive offline experiments on three real-world datasets. Experimental results demonstrate the superiority of the proposed DSAIN model. More importantly, DSAIN has increased the CTR by 2.70%, the CPM by 2.62%, and the GMV by 2.16% in the online A/B test. Now, DSAIN has been deployed on the Meituan food delivery platform and serves the main traffic of the Meituan takeout app. Our source code is available at https://github.com/W-void/DSAIN.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; **Online advertising**.

KEYWORDS

situation-aware, click-through rate prediction, user behavior modeling

ACM Reference Format:
Yimin Lv, Shuli Wang, Beihong Jin, Yisong Yu, Yapeng Zhang, Jian Dong, Yongkang Wang, Xingxing Wang, and Dong Wang. 2023. Deep Situation-Aware Interaction Network for Click-Through Rate Prediction. In *Seventeenth ACM Conference on Recommender Systems (RecSys '23), September 18–22, 2023, Singapore, Singapore.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3604915.3608793

1 INTRODUCTION

CTR prediction is a critical task in both recommendation systems and online advertising, and improving the accuracy of the CTR prediction has been receiving attention from both academia and industry.

Recently, modeling user behavior sequences [5, 9, 19, 20, 24, 25, 37, 38] has been introduced into CTR prediction. Generally, a user behavior sequence refers to a sequence of traces of a user performing actions on the app provided by an online service. For an e-commerce platform, a user behavior sequence at least contains the items that the user interacts with. For a specific scenario, a

*Beihong Jin is the corresponding author.

RecSys '23, September 18–22, 2023, Singapore, Singapore

ACM ISBN 979-8-4007-0241-9/23/09.
https://doi.org/10.1145/3604915.3608793

user behavior sequence can be augmented to include the side information of the interaction. Besides item-related and user-related side information, there are three types of behavior-related side information, i.e., the type of the specific behavior, and the time and (physical or virtual) location the specific behavior occurs.

Early work on user behaviors (e.g., GRU4Rec [16], Caser [31]) adopts Recurrent Neural Networks (RNNs) or Convolutional Neural Networks (CNNs) to capture the dependency relationships implied in the user behavior sequences. However, these models cannot obtain the semantics of user behaviors due to limited behavior information (e.g., only item ID is available). Recent work introduces the side information of an interaction behavior to understand the user behaviors. Some work (e.g., [8, 22, 23, 30, 34, 39]) exploits temporal and spatial information of behaviors to help model user interest. Some work (e.g., [10, 11, 32]) extracts behavior patterns and captures complicated correlations among different behaviors from a multi-behavior sequence. However, existing models focus on modeling a certain aspect of side information (such as behavior type or temporal information). No work can simultaneously incorporate behavior types and behavior spatio-temporal information in modeling user behavior sequences, which leads to the waste of some valuable side-information and neglects complex interdependencies among different side information of behaviors.

We argue that for an interaction of a user with an item, the behavior type, temporal features, and spatial features make up a situation of the interaction, and they are collectively called situational features. Taking a takeout scenario as an example, temporal features, which depict the temporal context in which the behavior occurred, include the hour of the day, the meal time period of day (e.g., breakfast, lunch, dinner, etc.), and the period of week (e.g., weekday and weekend). Nevertheless, when the geographic locations of users are not available, we can treat an app as a virtual space, thus features of the location where a user triggers an action can be listed under spatial features. In particular, we have one feature to record the location of triggering an action (on the search results page or the recommendation list page) and another feature to indicate whether the location of performing an action is an advertising place. Obviously, each situation has its own semantics.

For modeling the sequences of user behaviors with situational features, we have to face the following challenge. Given a user behavior and the corresponding situational features, how do we generate the high-quality embeddings for these situational features and the behavior, if taking into account multiple internal correlations, including but not limited to the correlations between different situational features of the same behavior? In this paper, we devise a novel CTR model named DSAIN (Deep Situation-Aware Interaction Network), which deals with the above challenge efficiently and elegantly. Our main contributions can be summarized as follows.

- We identify the noise in user behavior sequences and utilize the reparameterization trick to reduce noise interference.
- We combine commonalities and differences in situational features of the same type into their embeddings, and parameterize the embeddings of situational features to learn the approximated interaction between item ID and each situational feature, obtaining refined embeddings of situational features.
- We devise a light-weighted scheme of modeling tri-directional correlations to coherently capture cross-behavior, cross-situational feature, and cross-channel correlations, obtaining enhanced embeddings of situational features.
- We conduct extensive offline experiments on three real-world datasets (i.e., two public and one industrial datasets) and an online A/B test. Offline experimental results demonstrate the proposed DSAIN model achieves state-of-the-art performance. Results from the online A/B test show that DSAIN increases the CTR by 2.70%, the CPM by 2.62%, and the GMV by 2.16%.

The rest of the paper is organized as follows. Section 2 introduces the related work. Section 3 describes the DSAIN model in detail. Section 4 gives the experimental evaluation. Finally, the paper is concluded in Section 5.

2 RELATED WORK

Our work is related to the research under two closely related topics: CTR prediction and user behavior modeling.

2.1 CTR Prediction

Traditional CTR models learn feature interactions with the aid of deep neural networks. The representative work at this stage includes Wide&Deep [7], PNN [27], DeepFM [12], xDeepFM [21], ONN [35] and CAN [2].

Benefiting from the increase in computing power and model capacity, many efforts are put into sequential behavior modeling, which aims to infer the user interest with respect to the candidate based on the user historical behavior sequences. Specifically, DIN [38] employs an attention mechanism to dynamically reweight a user's historical behavior w.r.t. the candidate. Following this line, DIEN [37] introduces an interest-evolving layer to capture the interest-evolving process with the combination of attention and GRU. To further consider the intrinsic session structure of user behaviors, DSIN [9] uses a self-attention and a Bi-LSTM to extract in-session interest representations and cross-session evolution patterns, respectively. Later, Transformer-based models such as BST [6] also achieve great success in CTR prediction.

With the ever-growing volume of user interactions, many researchers pay attention to capturing user interest from long behavior sequences, which enables recommender systems to mine long-term behavior dependencies and the periodicity of user behaviors. Representative work in this field can be roughly divided into two categories, i.e., memory-augmented methods (e.g., HPMN [29] and MIMN [24]) and user behavior retrieval-based methods (e.g., UBR [26], two-phase model SIM [25], end-to-end behavior retrieval model ETA [5], and simple hash sampling-based approach SDIM [3]).

2.2 User Behavior Modeling

Originating from the conventional user behavior research on simple user behaviors, researchers have extended to further utilize more diverse and richer user behavior data to improve user behavior modeling performance. Recent research on behavior modeling can be roughly divided into two directions, and then we briefly introduce the main work in these two directions as follows.

The first direction is *increased multiplicity* - oriented user behavior modeling, i.e., multi-behavior recommendation. For instance, NMTR [10] first separately predicts the user-item interactions concerning different behavior types, then assembles them in a cascading manner to account for the cross-type behavior relations. Based on the concatenation of representations of single-type sequences, DMT [11] devises the deep multifaceted Transformers to simultaneously model users' multiple types of behaviors and utilizes MMoE (Multi-gate Mixture of Experts) to optimize multiple objectives. Taking into account complicated relationships among different behaviors as well as the behavior priority, FeedRec [32] proposes a strong-to-weak attention network that uses strong feedback to distill accurate positive and negative user interest from weak feedback. Furthermore, researchers attempt to explore the multiplex user-item interactive semantics with graph learning techniques in MBGCN [18], GHCF [4], MB-GMN [33].

The second direction is *growing heterogeneity* - oriented user behavior modeling. Most of the above-mentioned behavior modeling methods underuse the rich side information of interaction behaviors. To fill this gap, some work further takes into consideration the heterogeneous and diverse features associated with interactions. For instance, Trans2D [30] utilizes a modified Transformer with 2D self-attention to capture changes in attributes over time. To avoid fusing side information straightforward, which may deteriorate the original item ID representations, NOVA [23] proposes to leverage side information as an auxiliary for the self-attention module to learn a better attention distribution, instead of being fused into item representations. DIF-SR [34] argues that integrating side information before the attention will limit the learning of attention matrices, thus decoupling various side information with separate attention calculations to further improve NOVA. Moreover, CARCA [28], FDSA [36], S3Rec [39], and MISS [13] also achieve satisfied performance in behavior modeling with heterogeneous features.

Compared to existing work, DSAIN introduces the situation and situational features for each behavior, providing a new perspective on user behaviors. Further, DSAIN gives a simple yet effective method to integrate situational features into behavior embedding.

3 METHODOLOGY

In this section, we first give the problem formulation and then describe the proposed DSAIN model in detail. At the end of this section, we present the optimization objective and analyze the complexity of DSAIN.

Fig. 1 gives the architecture of DSAIN. Except for the embedding layer, DSAIN includes four modules, i.e., the Behavior Denoising Module (BDM), the Situational Feature Encoder (SFE), the Correlation Fusion Module (CFM), and the Situation Aggregation Module (SAM).

3.1 Problem Formulation

Let U, V denote the user set and the item set, respectively. For a user $u \in U$, his/her historical behavior sequence is denoted as $S = \{s_i\}_{i=1}^{L}$, where L is the length of the behavior sequence, $s_i = (v_i, B_i)$ is the i-th behavior, $v_i \in V$ is the interacted item, and B_i is the situation w.r.t. the behavior s_i. Supposing we have n types of situational features in total, then the situation B_i can further be represented as $\{f_i^k\}_{k=1}^{n}$, where f_i^k represents the k-th situational feature of the behavior s_i. Furthermore, we adopt the context of the current request to represent the current situation for clicking on the candidate v_c, denoted as $C = \{f_c^k\}_{k=1}^{n}$.

Given $\mathcal{X} = \{(u, S)\} \cup \{v_c, C\}$ as input and $y \in \{0, 1\}$ as the label of clicking the candidate, our CTR prediction task can be formalized as:

$$\mathcal{P}(y = 1|\mathcal{X}) = F(\mathcal{X}; \theta) \tag{1}$$

where F is the probability value output by the network we will develop and θ represents the parameters of the network.

3.2 Embedding Layer

There exist four groups of features, i.e., user profiles, item profiles, behavior sequences, and the current request context, used as the input of the embedding layer. Specifically, the user profile includes user ID, user age, user gender, etc. The item profile refers to item ID, category ID, etc. The user behavior sequence consists of interacted items and behaviors with corresponding situational features as defined above. Further, the current request context refers to the situation of the target behavior, i.e., the click on the candidate. Specifically, we utilize two different embedding tables to generate embeddings for situational features and common features, respectively. By looking up embedding tables, the embedding layer outputs the initial embedding vectors including user embedding $\mathbf{u}$, historical interacted items embeddings $\mathbf{v}_i$, behavior situation embeddings $\mathbf{B}_i = \{\mathbf{f}_i^k\}_{k=1}^{n}$, candidate item embedding $\mathbf{v}_c$, and the current situation embedding $\mathbf{C} = \{\mathbf{f}_c^k\}_{k=1}^{n}$. Specifically, the dimensions of situational feature embeddings are d_1, while others are d_2.

3.3 Behavior Denoising Module

Except for clicked items, the user historical behavior sequence considered in this paper is heavily mixed with abundant exposed items, which are imposed on the user by online e-commerce services rather than stemming from the user's active selection. Although these exposed items can reflect the user interest to a certain extent, there exists much noise which may deteriorate the performance of the CTR model. Therefore, decreasing noise interference in the user behavior sequence by some mechanism is of vital importance.

We start by analyzing a simple filter mechanism. Instead of taking some features such as the category and brand to pick out the top-k items most relevant to the candidate from the user historical behavior sequence, we calculate the correlation between the item and candidate in a soft mode and then sample items based on probabilities. The calculation process of the correlation score is shown in Eq. 2.

$$p_i = \text{sigmoid}\left(\mathbf{W}_2\left(\mathbf{v}_i + \mathbf{W}_1\left((\mathbf{f}_c^k)\|_{k=1}^{n} \otimes (\mathbf{f}_i^k)\|_{k=1}^{n}\right) + \mathbf{v}_c\right)\right) \tag{2}$$

where $\mathbf{W}_1 \in \mathbb{R}^{d_2 \times (n \times d_1)}$ and $\mathbf{W}_2 \in \mathbb{R}^{1 \times d_2}$ are learnable parameters. $\otimes$ means the element-wise product. $(\mathbf{f}_c^k)\|_{k=1}^{n}$ and $(\mathbf{f}_i^k)\|_{k=1}^{n}$ concatenate n situational features of the target click and historical behavior s_i, respectively. p_i denotes the correlation score of the item v_i w.r.t. the candidate v_c, which is adopted as the keeping probability of v_i. Then, the probability of discarding the item v_i is $1 - p_i$. Furthermore, we generate a binary variable named selection factor, denoted as

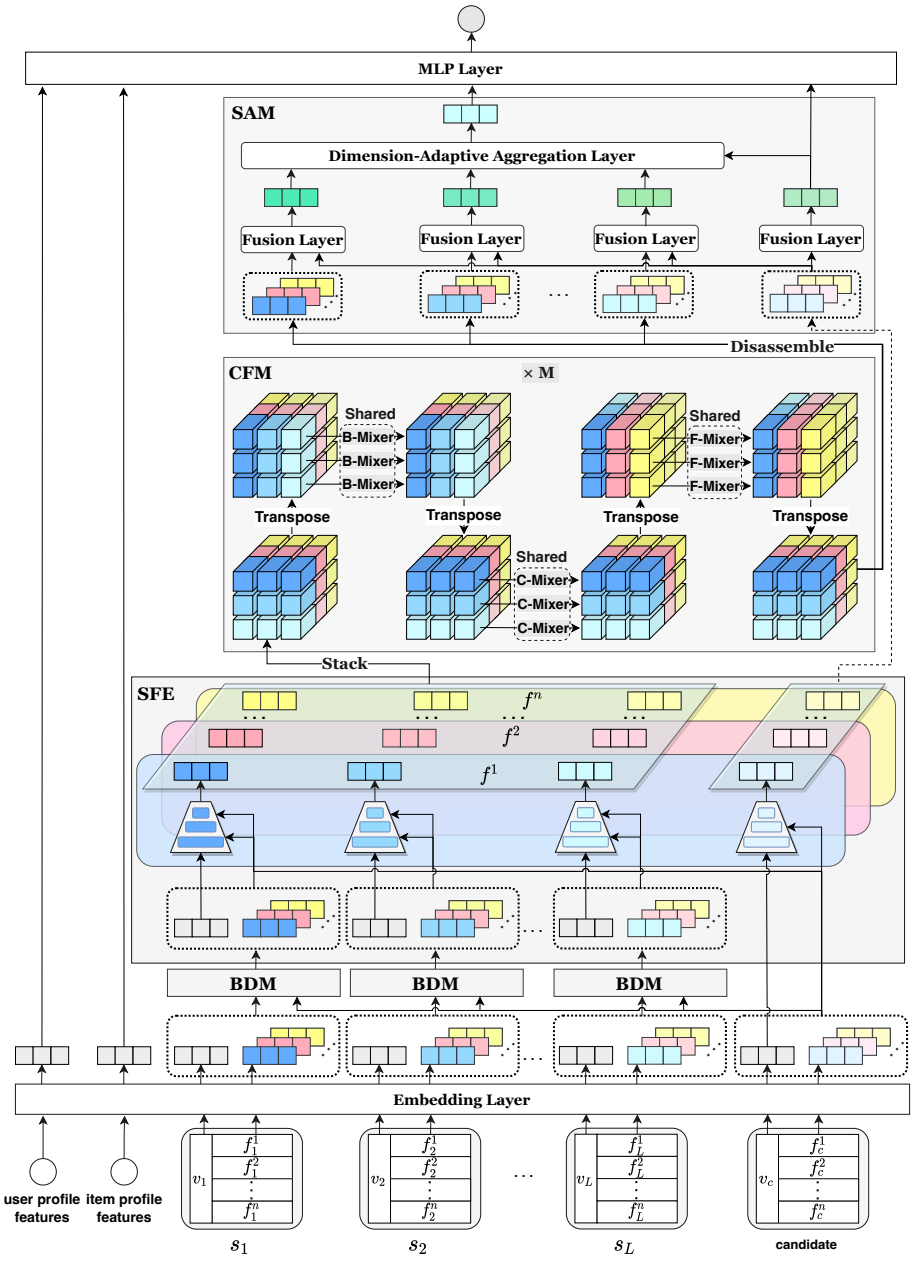


Figure 1: Architecture of our DSAIN (Deep Situation-Aware Interaction Network).

$d_i \in \{0, 1\}$, $d_i = 1$ represents that v_i is reserved and $d_i = 0$ means that v_i is discarded.

Then, a feasible way of behavior denoising is sampling from $[p_i, 1 - p_i]$ and drawing a conclusion of keeping or dropping the item v_i to generate d_i. However, it should be noted that the sampling process is discontinuous and non-differentiable, which is a stumbling block in the model training process based on gradient back-propagation. Inspired by [17], we leverage the reparameterization trick with Gumbel-Max and then transition to its smooth approximation Gumbel-Softmax, thereby transferring the non-differentiable sampling process into a continuous and differentiable sampling from the Gumbel distribution. The calculation equation is as follows.

$$\hat{d}_i = \frac{\exp\left((g_0 + \log p_i)/\tau\right)}{\exp\left((g_0 + \log p_i)/\tau\right) + \exp\left((g_1 + \log(1 - p_i))/\tau\right)} \tag{3}$$

where τ is the softmax temperature. g_0 and g_1 are i.i.d. samples drawn from the standard Gumbel distribution G(0,1). Taking g_0 as an example, sampling a value γ from the standard uniform distribution U(0,1) and then taking $-\log(-\log \gamma)$ as the final sample value g_0.

Based on the above reparameterization process with Gumbel-Softmax, we update the embedding $\mathbf{v}_i$ of the item v_i with corresponding selection factor $\hat{d}_i$ to decrease noise interference in original behavior sequence, as shown in Eq. 4.

$$\hat{\mathbf{v}}_i = \hat{d}_i \mathbf{v}_i \tag{4}$$

3.4 Situational Feature Encoder

For one behavior of a user interacting with an item, Situational Feature Encoder (SFE) generates the embeddings for situational features of the behavior by combining commonalities and differences in the situational features of the same type, and parameterizes the embeddings of situational features in the form of MLP, approximating the interaction between item ID and each situational feature.

Concretely, for the situational feature f^k, $1 \le k \le n$, we generate the general vector $\mathbf{f}^k$ for this feature (e.g., period of the week) and the specific vector list $\{\mathbf{f}^{k,t}\}_{t=1}^{m_k}$ for all values of this situational feature (e.g., weekday or weekend), where m_k is the number of values of the situational feature f^k (e.g., $m_k = 2$ for the above example). We think that the general vector and the specific vector list describe the commonalities and differences in the situational features of the same type, respectively.

Specifically, for situational feature f_i^k, we adopt its embedding $\mathbf{f}_i^k$ as the specific vector. Then, taking into consideration the candidate information, SFE fuses the general vector $\mathbf{f}^k$ and the specific vector $\mathbf{f}_i^k$ as shown in Eq. 5 and Eq. 6.

$$\text{gate}_{i,k} = \text{sigmoid}\left(\text{MLP}((\mathbf{v}_c \| \mathbf{f}_c^k) \otimes (\hat{\mathbf{v}}_i \| \mathbf{f}_i^k))\right) \tag{5}$$

$$\hat{\mathbf{f}}_i^k = \text{gate}_{i,k}\, \mathbf{f}_i^k + \left(1 - \text{gate}_{i,k}\right) \mathbf{f}^k \tag{6}$$

where $\|$ indicates the concatenation operation and it is easy to find that the dimension of $\hat{\mathbf{f}}_i^k$ is equal to $\mathbf{f}_i^k$, that is, d_1.

Then, $\hat{\mathbf{f}}_i^k$ is reshaped and split into the weight matrix and bias vector as the parameters of a micro-MLP named MLP_i^k w.r.t. f_i^k. This process can be formularized as follows.

$$\left(w_{i,j}^k \| b_{i,j}^k\right) \|_{j=0}^{D-1} = \hat{\mathbf{f}}_i^k \tag{7}$$

$$\sum_{j=0}^{D-1} \left(\left|w_{i,j}^k\right| + \left|b_{i,j}^k\right|\right) = \left|\hat{\mathbf{f}}_i^k\right| = d_1 \tag{8}$$

where $w_{i,j}^k$ and $b_{i,j}^k$ denote the weight and bias of j-th layer of MLP_i^k, D determines the depth of the micro-MLP and $|\cdot|$ gets the size of the variables. Then, we feed $\hat{\mathbf{v}}_i$ into MLP_i^k as Eq. 9 to refine the embedding of the situational feature f_i^k.

$$\mathbf{v}_{i,k} = \text{MLP}_i^k(\hat{\mathbf{v}}_i) \tag{9}$$

Following the above process, we can obtain the refined situational features representations $\{\mathbf{v}_{i,k}\}_{k=1}^n$ for each historical behavior s_i and $\{\mathbf{v}_{c,k}\}_{k=1}^n$ for the target behavior (i.e., the click on the candidate v_c).

3.5 Correlation Fusion Module

In this section, we devise the Correlation Fusion Module (CFM) to coherently learn tri-directional correlations. Specifically, the first direction is along the behavior sequence. The information in the second direction refers to the relationships among situational features of the same behavior. Moreover, the third direction information describes the correlations among different dimensions (i.e., channels) of the situational feature embedding, where distinct channels contains various latent semantics.

Concretely, CFM consists of stacked M layers of the identical block. Each block is equipped with three MLP-based mixers, i.e., a *behavior-mixer*, a *channel-mixer*, and a *feature-mixer*, capturing the cross-behavior, cross-channel and cross-feature (i.e., cross-situational feature) correlations, respectively. Within each block, we first apply the *behavior-mixer* on the behavior sequence and then utilize *channel-mixer* to learn the latent semantics of the embedding for each situational feature. Then, *feature-mixer* is adopted to learn intrinsic correlations among all situational features of the same behavior. Specifically, the proposed CFM can learn tri-directional information without introducing the position information, unlike the series of methods based on Transformer, since CFM is an MLP-based method and the MLP is inherently sensitive to the position. It is worth noting that the parameters of each block are shared between M layers to lighten the burden of model training and online inference.

3.5.1 *Preprocessing.*

Based on the refined representation $\mathbf{v}_{i,k} \in \mathbb{R}^{d_2}$ of the situational feature f_i^k obtained by the SFE module, we stack $\{\mathbf{v}_{i,k}\}_{i=1}^L$ and generate a two-dimensional matrix $\mathbf{V}^k \in \mathbb{R}^{L \times d_2}$ corresponding to f^k. Following this line, we further stack $\{\mathbf{V}^k\}_{k=1}^n$ to construct a three-dimensional matrix $\mathbf{V} \in \mathbb{R}^{n \times L \times d_2}$.

3.5.2 *Behavior-Mixer.*

The *behavior-mixer* aims to capture the correlations among behaviors in the user historical behavior sequence. As aforementioned, we first apply the *behavior-mixer* and the *channel-mixer* successively for each situational feature and then adopt the *feature-mixer*. Therefore, we detail the *behavior-mixer* based on the two-dimensional matrix $\mathbf{V}^k \in \mathbb{R}^{L \times d_2}$ corresponding to the situational feature f^k.

The simplest implementation of *behavior-mixer* is feeding the column vector of $\mathbf{V}^k$ into a MLP block, which contains two fully-connected layers with a nonlinear activation function, and then outputting an embedding with the same dimension as the input. For easy understanding, the above process is essentially equivalent to transposing the matrix $\mathbf{V}^k$ and then performing the transformation on the row vector of the ${\mathbf{V}^k}^\top$ as shown in Fig. 1. The *behavior-mixer* maps from $\mathbb{R}^L$ to $\mathbb{R}^L$ and is shared across all columns of $\mathbf{V}^k$. Formally, the output $\mathbf{V}^{k,\text{Beh}}$ of the *behavior-mixer* for the situational feature f^k can be represented as Eq. 18.

$$\mathbf{V}^{k,\text{Beh}}_{*,\alpha} = \mathbf{V}^{k}_{*,\alpha} + \mathbf{W}_2\ \sigma\left(\mathbf{W}_1\ \text{LayerNorm}\left(\mathbf{V}^{k}\right)_{*,\alpha}\right), \text{for } \alpha = 1, \ldots, d_2 \tag{10}$$

where $\mathbf{V}^{k,\text{Beh}}_{*,\alpha}$ is the α-th column of $\mathbf{V}^{k,\text{Beh}}$ and σ is an activation function named GELU [15]. $\mathbf{W}_1 \in \mathbb{R}^{D_L \times L}$ and $\mathbf{W}_2 \in \mathbb{R}^{L \times D_L}$ denote two trainable matrices corresponding to two fully-connected layers, respectively. D_L is the tunable hidden size of the *behavior-mixer*. Aside from the MLP layers, we further employ other standard architectural components, i.e., layer normalization [1] and residual connection [14]. It can be easily found that the *behavior-mixer* is sensitive to the order of behaviors in a sequence by adopting the MLP-based architecture, which frees the *behavior-mixer* from the position embedding.

Although the above process can capture the correlations among all behaviors, there exist three critical issues. First, this method indiscriminately feeds the entire behavior sequence of length L into the same MLP block, which may lead to noise interference. Second, the intrinsic and complex relationships of the behavior subsequence, consisting of adjacent behaviors within a certain range, are neglected by the above method. Third, the behavior subsequence is not strictly independent of each other and the latent correlations among them should be modeled by some mechanism.

To deal with the above three issues, we optimize the *behavior-mixer* with a partition-and-fusion pattern. To materialize this idea, we divide the entire sequence of length L into three types of subsequences according to different partition strategies. Then, we derive three distinct *behavior-mixer* to capture the correlations implied in the above three types of subsequences, respectively. We elaborate on these three *behavior-mixers* as follows.

Adjacent Behavior-Mixer sets the window size as L_w, where L is divisible by L_w, and then divides the entire sequence into $\frac{L}{L_w}$ segments from the beginning, i.e., each segment consists of L_w behaviors. For the φ-th segment, where $\varphi \in \{1, \ldots, \frac{L}{L_w}\}$, the consecutive L_w behaviors are mapped by the MLP block, which has a nonlinear activation function and two fully connected layers denoted as $\mathbf{W}_1^{\varphi} \in \mathbb{R}^{D_{L_w} \times L_w}$ and $\mathbf{W}_2^{\varphi} \in \mathbb{R}^{L_w \times D_{L_w}}$, respectively. It is worth noting that the MLP blocks adopted by different segments are shared. The calculation result of the *adjacent behavior-mixer* for the φ-th segment is denoted as $\mathbf{V}^{k,\text{Adj}}\langle\varphi\rangle$, which can be formalized as follows.

$$\mathbf{V}^{k,\text{Adj}}_{*,\alpha}\langle\varphi\rangle = \mathbf{V}^{k,\text{Div}}_{*,\alpha}\langle\varphi\rangle + \mathbf{W}_2^{\varphi}\ \sigma\left(\mathbf{W}_1^{\varphi}\ \text{LayerNorm}\left(\mathbf{V}^{k,\text{Div}}\langle\varphi\rangle\right)_{*,\alpha}\right), \quad \text{for } \alpha = 1, \ldots, d_2 \tag{11}$$

where $\mathbf{V}^{k,\text{Adj}}\langle\varphi\rangle \in \mathbb{R}^{L_w \times d_2}$ and $\mathbf{V}^{k,\text{Div}}\langle\varphi\rangle$ refers to the φ-th segment obtained by dividing the matrix $\mathbf{V}^{k} \in \mathbb{R}^{L \times d_2}$ along the L-axis. Then, we concat all of $\mathbf{V}^{k,\text{Adj}}_{*,\alpha}\langle\varphi\rangle \in \mathbb{R}^{L_w \times d_2}$ with φ from 1 to $\frac{L}{L_w}$ to derive the final output of the *adjacent behavior-mixer* as shown in Eq. 12.

$$\mathbf{V}^{k,\text{Adj}}_{*,\alpha} = \left(\mathbf{V}^{k,\text{Adj}}_{*,\alpha}\langle\varphi\rangle\right) \|_{\varphi=1}^{\frac{L}{L_w}}, \qquad \text{for } \alpha = 1, \ldots, d_2 \tag{12}$$

Dilated Behavior-Mixer is devised for capturing long-range correlations. Specifically, inspired by the dilated convolution, *dilated behavior-mixer* gathers behaviors from the entire sequence in intervals of $\frac{L}{L_w}$ and generate $\frac{L}{L_w}$ segments. Apparently, each segment is composed of L_w inconsecutive behaviors. The *dilated behavior-mixer* concentrates on learning the long-term dependency while disregarding the local patterns across adjacent behaviors, which is complementary to the *adjacent behavior-mixer*. For L_w behaviors in the φ'-th segment, where $\varphi' \in \{1, \ldots, \frac{L}{L_w}\}$, we feed them into a MLP block with the same structure as that of the *adjacent behavior-mixer*. Similarly, the MLP blocks are shared among different segments. We implement the *dilated behavior-mixer* as Eq. 13.

$$\mathbf{V}^{k,\text{Dil}}_{*,\alpha}\langle\varphi'\rangle = \mathbf{V}^{k,\text{Gat}}_{*,\alpha}\langle\varphi'\rangle + \mathbf{W}_2^{\varphi'}\sigma\left(\mathbf{W}_1^{\varphi'}\ \text{LayerNorm}\left(\mathbf{V}^{k,\text{Gat}}\langle\varphi'\rangle\right)_{*,\alpha}\right), \quad \text{for } \alpha = 1, \ldots, d_2 \tag{13}$$

where $\mathbf{V}^{k,\text{Dil}}\langle\varphi'\rangle \in \mathbb{R}^{L_w \times d_2}$ is the output of *dilated behavior-mixer* with respect to the φ'-th segment, and $\mathbf{V}^{k,\text{Gat}}\langle\varphi'\rangle$ refers to the φ'-th segment generated by gathering behaviors from $\mathbf{V}^{k} \in \mathbb{R}^{L \times d_2}$ along the L-axis in intervals of $\frac{L}{L_w}$, starting with the φ'-th row of $\mathbf{V}^{k}$. Then, we concat all of $\mathbf{V}^{k,\text{Dil}}_{*,\alpha}\langle\varphi'\rangle \in \mathbb{R}^{L_w \times d_2}$ with φ' from 1 to $\frac{L}{L_w}$ as Eq. 14.

$$\mathbf{V}^{k,\text{Dil}}_{*,\alpha} = \left(\mathbf{V}^{k,\text{Dil}}_{*,\alpha}\langle\varphi'\rangle\right) \|_{\varphi'=1}^{\frac{L}{L_w}}, \qquad \text{for } \alpha = 1, \ldots, d_2 \tag{14}$$

Shifted Behavior-Mixer aims to learn the correlations of two adjacent segments, which is another supplementary operation of the *adjacent behavior-mixer*. Specifically, we shift the partition of segments in the *adjacent behavior-mixer* by an offset of $\lfloor \frac{L_w}{2} \rfloor$. For several behaviors near the beginning and end of the sequence, since they are not covered by the shifted window, we concatenate them into $\mathbf{V}^{k,\text{Remain}} \in \mathbb{R}^{L_w \times d_2}$ and skip them when performing the MLP mapping. Apparently, shift-window operation generates $\frac{L}{L_w} - 1$ segments. Then, for the φ''-th segments, where $\varphi'' \in \{1, \ldots, \frac{L}{L_w} - 1\}$, we feed the consecutive L_w behaviors in the φ''-th segments into a specific MLP block, which is shared between different segments. The *shifted behavior-mixer* can be formalized as Eq. 15.

$$\mathbf{V}^{k,\text{Shi}'}_{*,\alpha}\langle\varphi''\rangle = \mathbf{V}^{k,\text{Sft}}_{*,\alpha}\langle\varphi''\rangle + \mathbf{W}_2^{\varphi''}\sigma\left(\mathbf{W}_1^{\varphi''}\text{LayerNorm}\left(\mathbf{V}^{k,\text{Sft}}\langle\varphi''\rangle\right)_{*,\alpha}\right), \quad \text{for } \alpha = 1, \ldots, d_2 \tag{15}$$

where $\mathbf{V}^{k,\text{Shi}'}\langle\varphi''\rangle \in \mathbb{R}^{L_w \times d_2}$ is the output of *shifted behavior-mixer* with respect to the φ''-th segment, and $\mathbf{V}^{k,\text{Sft}}\langle\varphi''\rangle$ refers to the φ''-th segment obtained by shifting the $\mathbf{V}^{k,\text{Div}}\langle\varphi''\rangle$ by an offset of $\lfloor \frac{L_w}{2} \rfloor$ along the L-axis. Then, we concat all of $\mathbf{V}^{k,\text{Shi}'}_{*,\alpha}\langle\varphi''\rangle \in \mathbb{R}^{L_w \times d_2}$ with φ'' from 1 to $\frac{L}{L_w} - 1$ and the remaining behaviors representations $\mathbf{V}^{k,\text{Remain}} \in \mathbb{R}^{L_w \times d_2}$ to generate the final output of the *shifted behavior-mixer* as shown in Eq. 16.

$$\mathbf{V}^{k,\text{Shi}}_{*,\alpha} = \mathbf{V}^{k,\text{Remain}}_{*,\alpha} \| \left(\left(\mathbf{V}^{k,\text{Shi}'}_{*,\alpha}\langle\varphi''\rangle\right) \|_{\varphi''=1}^{\frac{L}{L_w}-1}\right), \text{for } \alpha = 1, \ldots, d_2 \tag{16}$$

Now, three types of complementary correlations among all behaviors have been captured by the *adjacent behavior-mixer*, *dilated behavior-mixer* and *shifted behavior-mixer*, respectively. To obtain $\mathbf{V}^{k,\text{Beh}}$, which incorporates above three types of behavior

correlations, we aggregate $\mathbf{V}^{k,\text{Adj}} \in \mathbb{R}^{L\times d_2}$, $\mathbf{V}^{k,\text{Dil}} \in \mathbb{R}^{L\times d_2}$ and $\mathbf{V}^{k,\text{Shi}} \in \mathbb{R}^{L\times d_2}$ in a weight-adaptive manner as shown in Eq. 17.

$$\mathbf{V}^{k,\text{Beh}} = w_1\mathbf{V}^{k,\text{Adj}} + w_2\mathbf{V}^{k,\text{Dil}} + w_3\mathbf{V}^{k,\text{Shi}} \tag{17}$$

where w_1, w_2 and w_3 are learnable weight parameters.

3.5.3 Channel-Mixer.

Aside from the correlations among different behaviors, the internal relationships among different channels of the situational feature embedding deserve more attention. Specifically, the embedding of the situational feature is multi-dimensional and usually implies latent semantics on each dimension. Modeling the intrinsic relationships among these dimensions by some mechanism can further improve performance.

After performing the behavior mixing, the correlations along the user behavior sequence have been injected into $\mathbf{V}^{k,\text{Beh}}$ for each situational feature f^k as shown in Eq. 17. Then, the *channel-mixer* takes the row vector of $\mathbf{V}^{k,\text{Beh}} \in \mathbb{R}^{L\times d_2}$ as input, feeds it into a MLP block and maps from $\mathbb{R}^{d_2}$ to $\mathbb{R}^{d_2}$ as shown in Eq. **??**. All rows of $\mathbf{V}^{k,\text{Beh}}$ share the *channel-mixer*.

$$\mathbf{V}^{k,\text{Cha}}_{\beta,*} = \mathbf{V}^{k,\text{Beh}}_{\beta,*} + \left(\mathbf{W}_4\,\sigma\left(\mathbf{W}_3\,\left(\text{LayerNorm}\left(\mathbf{V}^{k,\text{Beh}}\right)_{\beta,*}\right)^{\text{T}}\right)\right)^{\text{T}}, \quad \text{for } \beta = 1,\ldots,L \tag{18}$$

where $\mathbf{V}^{k,\text{Cha}}_{\beta,*}$ is the β-th row of $\mathbf{V}^{k,\text{Cha}}$, σ is GELU, $\mathbf{W}_3 \in \mathbb{R}^{D_{d_2}\times d_2}$ and $\mathbf{W}_4 \in \mathbb{R}^{d_2\times D_{d_2}}$ denote two trainable matrices. D_{d_2} is the tunable hidden size of the *channel-mixer*. After Eq. **??**, the internal relationships among different channels have been incorporated into $\mathbf{V}^{k,\text{Cha}}$.

3.5.4 Feature-Mixer.

After behavior mixing and channel mixing, the cross-behavior and cross-channel correlations have been incorporated into the embedding of each situational feature. However, the inherent and complex relationships among all the situational features, which jointly portray the situation of the behavior, have not been effectively captured. Thus, we derive the *feature-mixer* to fuse the cross-feature correlation into the embedding of the situational feature. Concretely, for each situational feature f^k, the cross-behavior and cross-channel relationships have been injected into $\mathbf{V}^{k,\text{Cha}}$ along the path $\mathbf{V}^k \rightarrow \mathbf{V}^{k,\text{Beh}} \rightarrow \mathbf{V}^{k,\text{Cha}}$. We re-express $\mathbf{V}^{k,\text{Cha}}$ as $\widetilde{\mathbf{V}}^k$ and concat all $\widetilde{\mathbf{V}}^k \in \mathbb{R}^{L\times d_2}$, where $k = 1,\ldots,n$, corresponding to n situational features, to generate a three-dimensional matrix $\widetilde{\mathbf{V}} \in \mathbb{R}^{n\times L\times d_2}$. We transpose it into $\widetilde{\mathbf{V}} \in \mathbb{R}^{L\times d_2\times n}$ and feed $\widetilde{\mathbf{V}}^i \in \mathbb{R}^{d_2\times n}$ for each behavior s_i into the *feature-mixer* separately. The *feature-mixer* maps from $\mathbb{R}^n$ to $\mathbb{R}^n$ for each row vector of $\widetilde{\mathbf{V}}^i$ with a shared MLP block, which has a similar marco architecture to the *channel-mixer*. This process is shown in Eq. 19.

$$\widetilde{\mathbf{V}}^{i,\text{Fea}}_{\alpha,*} = \widetilde{\mathbf{V}}^{i}_{\alpha,*} + \left(\mathbf{W}_6\,\sigma\left(\mathbf{W}_5\,\left(\text{LayerNorm}(\widetilde{\mathbf{V}}^i)_{\alpha,*}\right)^{\text{T}}\right)\right)^{\text{T}}, \quad \text{for } \alpha = 1,\ldots,d_2 \tag{19}$$

where σ is GELU, $\mathbf{W}_5 \in \mathbb{R}^{D_n\times n}$ and $\mathbf{W}_6 \in \mathbb{R}^{n\times D_n}$ denote two trainable matrices. D_n is the tunable hidden size of the *feature-mixer*. Since the *feature-mixer* is subsequent to the *behavior-mixer* and *channel-mixer*, empowers the *feature-mixer* not only to model the cross-feature correlations but also to build a bridge for information communication of cross-behavior and cross-channel correlations among all situational features. Therefore, the *feature-mixer* coherently connects the tri-directional information.

3.6 Situation Aggregation Module

We have incorporated the tri-directional information, i.e., cross-behavior, cross-channel, and cross-feature correlations, into $\widetilde{\mathbf{V}}^{i,\text{Fea}} \in \mathbb{R}^{d_2\times n}$ for each historical behavior s_i as shown in Eq. 19. Specifically, we slice $\widetilde{\mathbf{V}}^{i,\text{Fea}}$ along the n-axis and obtain the $\{\tilde{\mathbf{v}}_{i,k}\}_{k=1}^n$ corresponding to n situational features of the historical behavior s_i. It should be noted that we do not apply the CFM module on the situational features of the target behavior, thus the embeddings $\{\mathbf{v}_{c,k}\}_{k=1}^n$ generated by the SFE module are maintained. Subsequently, we adopt two different methods to aggregate $\{\tilde{\mathbf{v}}_{i,k}\}_{k=1}^n$ and $\{\mathbf{v}_{c,k}\}_{k=1}^n$, respectively, thus generating the embeddings for the historical behavior s_i and the target behavior, respectively.

For the target behavior, with the consideration of importance difference among distinct situational features, we aggregate $\{\mathbf{v}_{c,k}\}_{k=1}^n$ into the embedding $\mathbf{b}_c$ of the target behavior in a weight-adaptive way, as shown in Eq. 20.

$$\mathbf{b}_c = \text{MLP}(\mathbf{v}_c + \sum_{k=1}^{n} w_{c,k}\mathbf{v}_{c,k}) \tag{20}$$

where $w_{c,k}$ is a trainable weight parameter.

Further, we dynamically reweight the embeddings of situational features for each historical behavior s_i with respect to the situational features of the target behavior, and then generate the embedding $\mathbf{b}_i$ of the historical behavior s_i as follows.

$$w_{i,k} = \text{MLP}(\tilde{\mathbf{v}}_{i,k} + \mathbf{v}_{c,k} \otimes \tilde{\mathbf{v}}_{i,k}) \tag{21}$$

$$w'_{i,k} = \frac{\exp(w_{i,k})}{\sum_{k=1}^{n}\exp(w_{i,k})} \tag{22}$$

$$\mathbf{b}_i = \text{MLP}(\mathbf{v}_i + \sum_{k=1}^{n} w'_{i,k}\tilde{\mathbf{v}}_{i,k}) \tag{23}$$

Then, to adjust the contributions of historical behavior representations at the channel level, we devise a channel-adaptive unit and give up the method in DIN [38], since the method in DIN simply calculates the similarity score at the granularity of the item. For each historical behavior embedding $\mathbf{b}_i$, the channel-adaptive factor $\mathbf{g}_i$, which is a vector with the same dimension as $\mathbf{b}_i$ rather than a scalar, is computed as Eq. 24.

$$\mathbf{g}_i = \text{sigmoid}\left(\mathbf{W}_g(\mathbf{b}_i \| (\mathbf{b}_i \otimes \mathbf{b}_c) \| \mathbf{b}_c)\right) \tag{24}$$

where $\mathbf{W}_g \in \mathbb{R}^{d_2\times 3d_2}$ is a learnable weight matrix. Finally, the embedding $\mathbf{e}_s$ of the behavior sequence is obtained by the following aggregation based on $\mathbf{g}_i$ as Eq. 25.

$$\mathbf{e}_s = \frac{1}{L}\sum_{i=1}^{L}\mathbf{g}_i \otimes \mathbf{b}_i \tag{25}$$

3.7 Model Optimization

Finally, the user embedding $\mathbf{u}$, the embedding $\mathbf{e}_s$ of the behavior sequence, and the embedding $\mathbf{b}_c$ of the target behavior (i.e., the click on the candidate), along with other feature embeddings denoted as $\mathbf{e}_o$, are concatenated and fed into an MLP to perform the CTR prediction, as shown in Eq. 26.

$$\hat{y} = \text{sigmoid}\left(\text{MLP}(\mathbf{u} \parallel \mathbf{e}_s \parallel \mathbf{b}_c \parallel \mathbf{e}_o)\right) \tag{26}$$

DSAIN is optimized by minimizing the following negative log-likelihood function defined as Eq. 27.

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)\right) \tag{27}$$

where N denotes the batch size, $y_i \in \{0, 1\}$ is the label and $\hat{y}_i$ is the predicted CTR value.

3.8 Complexity Analysis

Space Complexity. The learnable parameters in DSAIN are from the item embeddings ($O(|\mathcal{V}|d_2)$), situational features embeddings $O(nd_1)$, and parameters in the BDM($O(nd_1d_2)$), CFM($O(nL_wD_{L_w} + nd_2D_{d_2} + nD_n)$), and SAM($O(d_2^2)$). There are no extra parameters in SFE. Specifically, d_1 can be represented as δd_2, where δ is a constant with a certain range. Furthermore, L_w, D_{L_w}, D_{d_2} and D_n are hyperparameters, usually set to relatively small values. Thus, the total number of parameters is $O(|\mathcal{V}|d_2+nd_2^2)$, which is moderate compared to other classic and effective methods (e.g., $O(|\mathcal{V}|d + d^2)$ for FeedRec[32]) since n is quite limited.

Time Complexity. The computation amount of DSAIN is composed of four parts: the BDM module accounts for $O(Lnd_1d_2)$, the SFE module accounts for $O(Ln(d_1 + d_2 + d_2^2))$, the CFM module accounts for $O(MLnd_2)$, and the SAM module accounts for $O(Ld_2^2)$. As aforementioned, d_1 can be represented as δd_2. Therefore, the total computational complexity is $O(Lnd_2^2 + MLnd_2)$. Since n and M are relatively small, our model does not increase the computational cost, compared to other CTR models (e.g., $O(nd^2 + n^2d)$ for FeedRec[32]). A convenient property in our model is that the computation of all items in the behavior sequence is fully parallelizable, which is amenable to GPU acceleration.

4 EXPERIMENTS

In this section, we conduct extensive offline experiments on three real-world datasets and an online A/B test to answer the following research questions.

- **RQ1**: Does DSAIN outperform the existing CTR models on performance?
- **RQ2**: How do four components in the DSAIN contribute to the recommendation performance?
- **RQ3**: How do the key hyper-parameters in DSAIN affect its performance?
- **RQ4**: What is the online performance of DSAIN in a live production environment, e.g., the Meituan food delivery platform?

4.1 Experimental Setup

4.1.1 Datasets. We conduct offline experiments on the following three datasets.

- Taobao. It is a public dataset[1] collected from the display advertising system of Taobao. This dataset contains more than 26 million interaction records of 1.14 million users within 8 days. Each sample comprises five features: user ID, timestamp, behavior type, item brand ID, and category ID. It should be noted that this dataset does not include item ID but adopts brand ID as the alternative. We construct temporal features based on timestamp information, which together with the behavior type constitute the situational features of this dataset.
- Eleme. It is a public dataset[2] constructed from logs of ele.me service. This dataset contains interactions within 8 days. Aside from the behavior type, this dataset naturally contains rich temporal features (e.g., the hour of day, weekday or weekend, etc.), which jointly form the situational features.
- Meituan. It is an industrial dataset collected from the Meituan food delivery platform, which contains 1.3 billion interaction records of 130 million users within 30 days. This industrial dataset not only includes behavior type and temporal features (e.g., the period of the day corresponding to breakfast/lunch/dinner, and weekday/weekend, etc.) but also incorporates behavior-related spatial information, such as the page (e.g., search result page and recommendation list page) in which the behavior occurs.

Table 1: Statistics of datasets.

Datasets	#Users	#Items	#Categories	#Behaviors
Taobao	1,141,729	99,815	6,769	26,557,961
Eleme	14,427,689	7,446,116	10	177,114,244
Meituan	130,648,310	14,054,691	184	1,331,247,488

The statistics of the three datasets are shown in Table 1.

4.1.2 Comparison Models. The following eight baselines are chosen for comparative experiments and they are divided into three groups.

Group I includes three models that treat behavior-related side information as common features.

- **DIN**[38] employs a local activation unit to dynamically reweight users' historical behaviors w.r.t. the candidate.
- **DIEN**[37] introduces an interest-evolving layer to capture the evolution of interest on the basis of DIN.
- **CAN**[2] disentangles representation learning and feature interaction modeling via a co-action unit, which is also equipped with multi-order enhancement and multi-level independence to enhance the feature interaction.

Group II includes two models that conduct modeling for the behavior type.

- **DMT**[11] exploits Deep Multifaceted Transformers to model users' diverse behavior sequences, utilizes MMoE to jointly optimize multi-objectives and uses the Bias Deep Neural Networks to reduce the selection bias in implicit feedback.
- **FeedRec**[32] gives a user modeling framework to incorporate various explicit and implicit user feedback to infer user interest.

Group III includes three models that focus on modeling for behavior temporal and/or spatial information.

- **CARCA**[28] captures the dynamic nature of the user profiles in terms of contextual features and item attributes via dedicated multi-head self-attention blocks.
- **Trans2D**[30] devises a novel extended attention mechanism named attention2D to learn item-item, attribute-attribute and

[1] https://tianchi.aliyun.com/dataset/56

[2] https://tianchi.aliyun.com/dataset/131047

Table 2: Performance comparison. The best result and the second-best result in each row are in bold and underlined, respectively.

Dataset	Metric	Group I			Group II		Group III			DSAIN
		DIN	DIEN	CAN	DMT	FeedRec	CARCA	Trans2D	DIF-SR	
Taobao	AUC	0.6223	0.6241	0.6276	0.6268	0.6312	0.6289	0.6303	<u>0.6330</u>	**0.6452** ± 0.5‰
	Logloss	0.2622	0.2618	0.2607	0.2611	0.2592	0.2603	0.2599	<u>0.2588</u>	**0.2571** ± 0.1‰
Eleme	AUC	0.6368	0.6420	0.6450	0.6435	0.6477	0.6455	0.6464	<u>0.6502</u>	**0.6634** ± 0.4‰
	Logloss	0.1245	0.1198	0.1157	0.1173	0.1143	0.1152	0.1149	<u>0.1135</u>	**0.1116** ± 0.1‰
Meituan	AUC	0.6577	0.6612	0.6645	0.6635	0.6715	0.6683	0.6709	<u>0.6740</u>	**0.6823** ± 0.4‰
	Logloss	0.1922	0.1884	0.1855	0.1861	0.1819	0.1842	0.1825	<u>0.1799</u>	**0.1763** ± 0.1‰

item-attribute patterns from sequences with multiple item attributes.

- **DIF-SR**[34] decouples the behavior-related side information and the item embedding to prevent information entanglement.

4.1.3 Evaluation Metrics. We adopt two metrics, i.e., **AUC** (Area Under ROC Curve) and **Logloss**, to evaluate our proposed model in offline experiments. A larger AUC indicates better recommendation performance, but Logloss performs the opposite. It is noteworthy that a small improvement in AUC is likely to lead to a significant increase in online CTR [38]. In online experiments, we adopt **CTR** (Click-Through Rate), **CPM** (Cost-Per-Mille) and **GMV** (Gross Merchandise Volume) as evaluation metrics.

4.1.4 Implementation Details. We implement DSAIN by TensorFlow. For all comparison models and our DSAIN model, we adopt Adam as the optimizer with the learning rate fixed to 0.001 and initialize the model parameters with normal distribution by setting the mean and standard deviation to 0 and 0.01, respectively. For our proposed DSAIN model, L, L_w, n, d_1, and d_2 are set to 300, 30, 5, 144, and 8, respectively. The temperature parameter τ in BDM and the layer depth in CFM are set to 1 and 4, respectively. For the hidden dimensions of three mixers, i.e., D_{L_w} for the *behavior-mixer*, D_{d_2} for the *channel-mixer*, and D_n for the *feature-mixer*, in CFM are set to 10, 16, 8, respectively.

4.2 Overall Performance (RQ1)

Table 2 lists the performance results of baselines and our DSAIN model on Taobao, Eleme, and Meituan datasets. All experiments are repeated three times. The best results of compared models and the average results of DSAIN are reported. From Table 2, we find performance trends of all the baselines are consistent on three datasets and have some observations as follows.

(1) FeedRec in Group II outperforms all models in Group I, and DIF-SR in Group III outperforms all the other models in Groups I, II, and III. This demonstrates the significance of fully utilizing the behavior-related side information.

(2) DMT in Group II performs better than DIN and DIEN in Group I, while slightly worse than CAN in Group I. Moreover, CARCA and Trans2D in Group III surpass models in Group I and DMT in Group II, but are slightly worse than FeedRec in Group II. This phenomenon inspires us that the improvement of model performance does not merely depend on utilizing richer information of interaction and the model architecture imposes a significant effect on the performance.

(3) Our DSAIN consistently outperforms all baseline models and achieves state-of-the-art performance on all datasets. We think, compared to DSAIN, all competitors either underutilize the behavior-related side information, or neglect to capture multi-directional correlations implied in the user behavior sequence, especially the dependencies among different situational features, thus suffering the degradation of model performance.

4.3 Ablation Study (RQ2)

In this section, we perform the ablation study on the Meituan dataset to investigate the different impacts of four modules on the performance of DSAIN.

4.3.1 Effect of Behavior Denoising Module. In this section, we investigate the impact of the Behavior Denoising Module (BDM) in DSAIN. We consider the following variants: (1) DSAIN$_1^*$: it simplifies the DSAIN by dropping BDM, and performs nothing on the original user historical behavior sequence and then directly feeds it into the model; (2) DSAIN$_2^*$: it keeps a specified number of exposed items before each clicked item and discards the remaining exposed items, thereby forming a new sequence, which is the subsequence of the original user historical behavior sequence. Specifically, DSAIN$_2^*(x)$ sets the number of reserved exposures to x, where $x \in \{2, 4, 8, 12, 16, 20\}$.

As shown in Table 3, (1) compared to DSAIN$_1^*$ and DSAIN$_2^*(\cdot)$, DSAIN consistently fosters the improvement of recommendation performance, which demonstrates the effectiveness of BDM. (2) We find that the model performance is the worst when only keeping two exposed items before each clicked item. This may be because the historical behavior sequence considered in this paper contains a large number of exposed items while clicked items are relatively sparse. Too much valuable information is discarded when almost all exposures are dropped. Therefore, DSAIN performs better when x increases from 2 to 12. However, the model performance suffers a decrease as x continues to increase from 12, which may be owing to the noise interference introduced by excessive exposures.

4.3.2 Effect of Situational Feature Encoder. To demonstrate the significance of the Situational Feature Encoder (SFE), we consider the following variants: (1) DSAIN$_1^\dagger$: it simplifies DSAIN by replacing SFE with an average pooling over all $\hat{\mathbf{f}}_i^k, k = 1, \ldots, n$, plus the concatenation of the result and the item embedding $\hat{\mathbf{v}}_i$; (2) DSAIN$_2^\dagger$: it performs the average pooling over all $\hat{\mathbf{f}}_i^k, k = 1, \ldots, n$, and obtain a vector $\bar{\mathbf{f}}_i^k$, then splits $\bar{\mathbf{f}}_i^k$ into several vectors adopted as parameters

Table 3: Performance of variants with different BDMs or without BDM.

	DSAIN_1^*	$\text{DSAIN}_2^*(2)$	$\text{DSAIN}_2^*(4)$	$\text{DSAIN}_2^*(8)$	$\text{DSAIN}_2^*(12)$	$\text{DSAIN}_2^*(16)$	$\text{DSAIN}_2^*(20)$	DSAIN
AUC	0.6808	0.6778	0.6797	0.6808	0.6811	0.6804	0.6803	**0.6823**
Logloss	0.1765	0.1770	0.1767	0.1764	0.1764	0.1767	0.1765	**0.1763**

of the weight matrix and the bias vector of a micro-MLP, then feeds the item embedding $\hat{\mathbf{v}}_i$ into the micro-MLP; (3) $\text{DSAIN}_3^\dagger$: it concatenates the item embedding $\hat{\mathbf{v}}_i$ with each situational feature embedding $\hat{\mathbf{f}}_i^k$ to generate $\mathbf{v}_{i,k}$; (4) $\text{DSAIN}_4^\dagger$: it only uses the specific vector $\mathbf{f}_i^k$ and drops the general vector $\mathbf{f}^k$ to generate the embedding $\hat{\mathbf{f}}_i^k$ for the situational feature f_i^k.

The results are shown in Table 4. We can infer from Table 4 that (1) the original DSAIN outperforms all four variants, which shows the effectiveness of SFE. (2) Compared to the original DSAIN, the sharp performance degradation of $\text{DSAIN}_1^\dagger$ and $\text{DSAIN}_2^\dagger$ indicates that fusing situational features prematurely may lead to information distraction and deteriorate the model performance. In this case, even with a more complex interaction scheme, such as an micro-MLP structure in $\text{DSAIN}_2^\dagger$, the performance improvement is also very limited in comparison to $\text{DSAIN}_1^\dagger$, which highlights the necessity of considering each situational feature separately. (3) Compared to DSAIN, $\text{DSAIN}_3^\dagger$ drops the micro-MLP and suffers poor performance, which illustrates that the micro-MLP can assist in achieving full-fledged feature interactions between item embedding and situational features. (4) Comparing $\text{DSAIN}_4^\dagger$ to DSAIN, the decrease in AUC indicates that the commonalities of the situational features of the same type are successfully injected into the general vector in SFE.

Table 4: Performance of variants with different SFEs.

	$\text{DSAIN}_1^\dagger$	$\text{DSAIN}_2^\dagger$	$\text{DSAIN}_3^\dagger$	$\text{DSAIN}_4^\dagger$	DSAIN
AUC	0.6790	0.6795	0.6801	0.6810	**0.6823**
Logloss	0.1766	0.1766	0.1765	0.1764	**0.1763**

4.3.3 Effect of Correlation Fusion Module. To illustrate the importance of the Correlation Fusion Module (CFM), we design seven variants as follows: (1) $\text{DSAIN}_1^\ddagger$: it only performs behavior mixing; (2) $\text{DSAIN}_2^\ddagger$: it only performs channel mixing; (3) $\text{DSAIN}_3^\ddagger$: it only performs feature mixing; (4) $\text{DSAIN}_4^\ddagger$: it removes the *feature-mixer* in CFM, i.e., performing behavior mixing and channel mixing; (5) $\text{DSAIN}_5^\ddagger$: it simplifies CFM by removing the *channel-mixer*; (6) $\text{DSAIN}_6^\ddagger$: it simplifies CFM by removing the *behavior-mixer*; (7) $\text{DSAIN}_7^\ddagger$: it removes the nonlinear activation function GELU from the MLP block used in three mixers. The results are shown in Table 5.

From Table 5, we can observe that (1) all seven variants suffer a decline in AUC and an increase in Logloss, compared to the original DSAIN. (2) Comparison between the variants with only a single mixer (i.e., $\text{DSAIN}_1^\ddagger$, $\text{DSAIN}_2^\ddagger$, $\text{DSAIN}_3^\ddagger$) and the original DSAIN shows that the *behavior-mixer* and *feature-mixer* play more important roles than *channel-mixer*, and the *behavior-mixer* is the most significant. (3) Comparing the variants with two mixers (i.e., $\text{DSAIN}_4^\ddagger$, $\text{DSAIN}_5^\ddagger$, $\text{DSAIN}_6^\ddagger$) and the original DSAIN, we find that $\text{DSAIN}_6^\ddagger$ suffers most drastic performance degradation, which shows the *behavior-mixer* has a significant effect on model performance. Moreover, the AUC decrease of $\text{DSAIN}_4^\ddagger$ by more than two thousand points compared to the original DSAIN illustrates that the *feature-mixer* coherently connects the tri-directional information, which fosters the improvement of model performance. (4) Performance decrease of $\text{DSAIN}_7^\ddagger$ compared to the original DSAIN demonstrates the necessity of introducing nonlinearity into the CFM module by adopting the activation function GELU. (5) Compared to using *behavior-mixer* or *feature-mixer* alone, further introducing the *channel-mixer* consistently leads to performance degradation, which may be due to information entanglement. Nevertheless, comparison between $\text{DSAIN}_5^\ddagger$ and the original DSAIN shows the effectiveness of *channel-mixer* in capturing latent semantics across multiple channels while using three mixers together. (6) DSAIN achieves the best performance by jointly leveraging three mixers, thus coherently capturing the tri-directional correlations, i.e., cross-behavior, cross-channel, and cross-feature correlations.

4.3.4 Effect of Situation Aggregation Module. In this section, we investigate the impact of the Situation Aggregation Module (SAM) in DSAIN by considering the following variants: (1) $\text{DSAIN}_1^\diamond$: it performs the average pooling over all $\tilde{\mathbf{v}}_{i,k}, k = 1, \ldots, n$, to aggregate the embeddings of situational features of the same behavior; (2) $\text{DSAIN}_2^\diamond$: it aggregates $\tilde{\mathbf{v}}_{i,k}, k = 1, \ldots, n$ in a weight-adaptive manner; (3) $\text{DSAIN}_3^\diamond$: it calculates the correlation score between the target behavior (a click on the candidate) and each historical behavior to dynamically adjust contributions of historical behaviors.

From Table 6, we find that (1) both $\text{DSAIN}_1^\diamond$ and $\text{DSAIN}_2^\diamond$ suffer a decline in AUC compared to the original DSAIN, which demonstrates the significance of making the utmost of the target behavior to guide the aggregation of embeddings of situational features of the same behavior; (2) The performance degradation of $\text{DSAIN}_3^\diamond$ compared to DSAIN shows the effectiveness of the channel-adaptive unit in the SAM, which empowers DSAIN to modulate contributions at the dimensional level of behavioral embeddings, where the dimensional level is more fine-grained than item level.

4.4 Hyperparameter Analysis (RQ3)

In this section, we conduct the experiments to analyze the sensitivity of critical hyperparameters in DSAIN, including the layer depth M in CFM (Correlation Fusion Module), the hidden dimensions of three mixers (i.e., D_{L_w} of *behavior-mixer*, D_{d_2} of *channel-mixer* and D_n of *feature-mixer*) in CFM, and the temperature τ in BDM (Behavior Denoising Module). As mentioned in the methodology, CFM consists of stacked M layers of the identical block. Thus, we first vary the layer depth M from 1 to 8 to investigate its influence on model performance. As shown in Fig. 2 (a), the model performs

Table 5: Performance of variants with different CFMs.

	$\text{DSAIN}_1^\ddagger$	$\text{DSAIN}_2^\ddagger$	$\text{DSAIN}_3^\ddagger$	$\text{DSAIN}_4^\ddagger$	$\text{DSAIN}_5^\ddagger$	$\text{DSAIN}_6^\ddagger$	$\text{DSAIN}_7^\ddagger$	DSAIN
behavior-mixer	✓			✓	✓		✓	✓
channel-mixer		✓		✓		✓	✓	✓
feature-mixer			✓		✓	✓	✓	✓
GELU	✓	✓	✓	✓	✓	✓		✓
AUC	0.6805	0.6779	0.6793	0.6798	0.6812	0.6770	0.6806	**0.6823**
Logloss	0.1765	0.1767	0.1767	0.1766	0.1764	0.1769	0.1765	**0.1763**

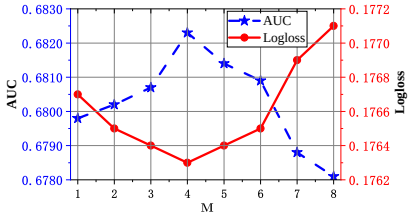
Table 6: Performance of variants with different SAMs.

	$\text{DSAIN}_1^\diamond$	$\text{DSAIN}_2^\diamond$	$\text{DSAIN}_3^\diamond$	DSAIN
AUC	0.6805	0.6800	0.6812	**0.6823**
Logloss	0.1765	0.1767	0.1764	**0.1763**

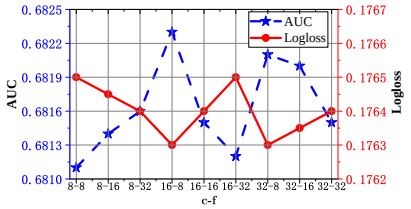
better as M increases from 1 to 4, but then worse as M further increases. We consider the reason for this phenomenon is that stacking more layers moderately contributes to fully exploiting the tri-directional correlations, while excessive layers might also introduce noise interference and blur the representations of situational features. Therefore, we set M to 4 according to the results.

Then, we evaluate the effect of the hidden dimensions of three mixers (i.e., D_{L_w}, D_{d_2}, and D_n) in CFM. Firstly, fixing D_{d_2} to 16 and D_n to 8, we tune the hidden dimension D_{L_w} of the *behavior-mixer* in $\{2, 4, 8, 10, 12, 16, 20\}$ w.r.t. the sequence with a total length of 300 used in our DSAIN. The results present DSAIN performs best when $D_{L_w} = 10$. Due to the limited space, we omit the visualization of the results. Then, we consider different combinations of hidden dimensions $D_{d_2} \in \{8, 16, 32\}$ of the *channel-mixer* and $D_n \in \{8, 16, 32\}$ of the *feature-mixer* with D_{L_w} fixed to 10. Fig. 2 (b) shows the performance of DSAIN with different configurations of D_{d_2} and D_n denoted as "c-f" (e.g., "8-16" represents $D_{d_2} = 8$ and $D_n = 16$). From Fig. 2 (b), we find that DSAIN achieves the peak performance when $D_{d_2} = 16$ and $D_n = 8$.

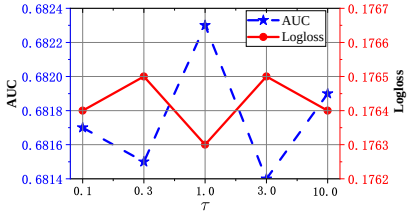
Finally, we tune the temperature τ in $\{0.1, 0.3, 1.0, 3.0, 10.0\}$ to observe its influence on performance. From Fig. 2 (c), we find that a moderate value of τ such as 1.0 is suitable for DSAIN, while the performance is suboptimal when τ is either too small or too large.



(a) Layer depth M in CFM.



(b) Hidden dimensions c-f in CFM.



(c) Temperature τ in BDM.

Figure 2: Influence of critical hyperparameters on the model performance.

4.5 Online A/B Test (RQ4)

We conduct the online A/B test by deploying DSAIN to handle real 40% traffic in the Meituan food delivery platform for seven days beginning on December 6, 2022, where the baseline model is the online-serving CTR model of the Meituan food delivery platform. Compared to the baseline model, DSAIN has increased the CTR by 2.70%, the CPM by 2.62%, and the GMV by 2.16%. Currently, DSAIN has been deployed in the advertisement recommender system of the Meituan food delivery platform and is serving hundreds of millions of users, which contributes to significant business revenue growth.

5 CONCLUSION

In this paper, we propose the concept of the situation and situational features for behaviors and then devise a novel CTR model DSAIN (Deep Situation-Aware Interaction Network). Given a sequence of user behaviors and the corresponding situational features, DSAIN reduces noise behaviors in the sequence, learns the high-quality embeddings for situational features by taking into consideration multiple internal correlations, and aggregates them into the embedding of the behavior sequence for final CTR prediction. The results of offline experiments on three datasets and the online A/B test demonstrate the superiority of our DSAIN model.

ACKNOWLEDGMENTS

This research was supported by Meituan and the Natural Science Foundation of China under Grant No. 62072450.

REFERENCES

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *Stat* 1050 (2016), 21.

[2] Weijie Bian, Kailun Wu, Lejian Ren, Qi Pi, Yujing Zhang, Can Xiao, Xiang-Rong Sheng, Yong-Nan Zhu, Zhangming Chan, Na Mou, et al. 2022. CAN: Feature co-action network for click-through rate prediction. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 57–65.

[3] Yue Cao, Xiaojiang Zhou, Jiaqi Feng, Peihao Huang, Yao Xiao, Dayao Chen, and Sheng Chen. 2022. Sampling is all you need on modeling long-term user behaviors for CTR prediction. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 2974–2983.

[4] Chong Chen, Weizhi Ma, Min Zhang, Zhaowei Wang, Xiuqiang He, Chenyang Wang, Yiqun Liu, and Shaoping Ma. 2021. Graph heterogeneous multi-relational recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 3958–3966.

[5] Qiwei Chen, Changhua Pei, Shanshan Lv, Chao Li, Junfeng Ge, and Wenwu Ou. 2021. End-to-end user behavior retrieval in click-through rate prediction model. *arXiv preprint arXiv:2108.04468* (2021).

[6] Qiwei Chen, Huan Zhao, Wei Li, Pipei Huang, and Wenwu Ou. 2019. Behavior sequence transformer for e-commerce recommendation in alibaba. In *Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data*. 1–4.

[7] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. 7–10.

[8] Qiang Cui, Chenrui Zhang, Yafeng Zhang, Jinpeng Wang, and Mingchen Cai. 2021. ST-PIL: Spatial-temporal periodic interest learning for next point-of-interest recommendation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2960–2964.

[9] Yufei Feng, Fuyu Lv, Weichen Shen, Menghan Wang, Fei Sun, Yu Zhu, and Keping Yang. 2019. Deep session interest network for click-through rate prediction. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. 2301–2307.

[10] Chen Gao, Xiangnan He, Dahua Gan, Xiangning Chen, Fuli Feng, Yong Li, Tat-Seng Chua, and Depeng Jin. 2019. Neural multi-task recommendation from multi-behavior data. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 1554–1557.

[11] Yulong Gu, Zhuoye Ding, Shuaiqiang Wang, Lixin Zou, Yiding Liu, and Dawei Yin. 2020. Deep multifaceted transformers for multi-objective ranking in large-scale e-commerce recommender systems. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2493–2500.

[12] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A factorization-machine based neural network for CTR prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. 1725–1731.

[13] Wei Guo, Can Zhang, Zhicheng He, Jiarui Qin, Huifeng Guo, Bo Chen, Ruiming Tang, Xiuqiang He, and Rui Zhang. 2022. MISS: Multi-interest self-supervised learning framework for click-through rate prediction. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 727–740.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 770–778.

[15] Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415* (2016).

[16] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. In *International Conference on Learning Representations*.

[17] Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*.

[18] Bowen Jin, Chen Gao, Xiangnan He, Depeng Jin, and Yong Li. 2020. Multi-behavior recommendation with graph convolutional networks. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 659–668.

[19] Xiang Li, Shuwei Chen, Jian Dong, Jin Zhang, Yongkang Wang, Xingxing Wang, and Dong Wang. 2023. Context-aware modeling via simulated exposure page for CTR prediction. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1904–1908.

[20] Xiang Li, Shuwei Chen, Jian Dong, Jin Zhang, Yongkang Wang, Xingxing Wang, and Dong Wang. 2023. Decision-making context interaction network for click-through rate prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[21] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. XDeepFM: Combining explicit and implicit feature interactions for recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1754–1763.

[22] Shaochuan Lin, Yicong Yu, Xiyu Ji, Taotao Zhou, Hengxu He, Zisen Sang, Jia Jia, Guodong Cao, and Ning Hu. 2022. Spatiotemporal-enhanced network for click-through rate prediction in location-based services. *arXiv preprint arXiv:2209.09427* (2022).

[23] Chang Liu, Xiaoguang Li, Guohao Cai, Zhenhua Dong, Hong Zhu, and Lifeng Shang. 2021. Noninvasive self-attention for side information fusion in sequential recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 4249–4256.

[24] Qi Pi, Weijie Bian, Guorui Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Practice on long sequential user behavior modeling for click-through rate prediction. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2671–2679.

[25] Qi Pi, Guorui Zhou, Yujing Zhang, Zhe Wang, Lejian Ren, Ying Fan, Xiaoqiang Zhu, and Kun Gai. 2020. Search-based user interest modeling with lifelong sequential behavior data for click-through rate prediction. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2685–2692.

[26] Jiarui Qin, Weinan Zhang, Xin Wu, Jiarui Jin, Yuchen Fang, and Yong Yu. 2020. User behavior retrieval for click-through rate prediction. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2347–2356.

[27] Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. 2016. Product-based neural networks for user response prediction. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 1149–1154.

[28] Ahmed Rashed, Shereen Elsayed, and Lars Schmidt-Thieme. 2022. Context and attribute-aware sequential recommendation via cross-attention. In *Proceedings of the 16th ACM Conference on Recommender Systems*. 71–80.

[29] Kan Ren, Jiarui Qin, Yuchen Fang, Weinan Zhang, Lei Zheng, Weijie Bian, Guorui Zhou, Jian Xu, Yong Yu, Xiaoqiang Zhu, et al. 2019. Lifelong sequential modeling with personalized memorization for user response prediction. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 565–574.

[30] Uriel Singer, Haggai Roitman, Yotam Eshel, Alexander Nus, Ido Guy, Or Levi, Idan Hasson, and Eliyahu Kiperwasser. 2022. Sequential modeling with multiple attributes for watchlist recommendation in e-commerce. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 937–946.

[31] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 565–573.

[32] Chuhan Wu, Fangzhao Wu, Tao Qi, Qi Liu, Xuan Tian, Jie Li, Wei He, Yongfeng Huang, and Xing Xie. 2022. FeedRec: News feed recommendation with various user feedbacks. In *Proceedings of the ACM Web Conference 2022*. 2088–2097.

[33] Lianghao Xia, Yong Xu, Chao Huang, Peng Dai, and Liefeng Bo. 2021. Graph meta network for multi-behavior recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 757–766.

[34] Yueqi Xie, Peilin Zhou, and Sunghun Kim. 2022. Decoupled side information fusion for sequential recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1611–1621.

[35] Yi Yang, Baile Xu, Shaofeng Shen, Furao Shen, and Jian Zhao. 2020. Operation-aware neural networks for user response prediction. *Neural Networks* 121 (2020), 161–168.

[36] Tingting Zhang, Pengpeng Zhao, Yanchi Liu, Victor S Sheng, Jiajie Xu, Deqing Wang, Guanfeng Liu, and Xiaofang Zhou. 2019. Feature-level deeper self-attention network for sequential recommendation. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. 4320–4326.

[37] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep interest evolution network for click-through rate prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 5941–5948.

[38] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1059–1068.

[39] Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. 2020. S^3-Rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 1893–1902.