TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 08/22 pp86–98
DOI: 10.26599/TST.2022.9010054
Volume 29, Number 1, February 2024

Modeling Long- and Short-Term Service Recommendations with a Deep Multi-Interest Network for Edge Computing

Rui Yuan, Shunmei Meng*, Ruihan Dou, and Xinna Wang

Abstract: Edge computing platforms enable application developers and content providers to provide context-aware services (such as service recommendations) using real-time wireless access network information. How to recommend the most suitable candidate from these numerous available services is an urgent task. Click-through rate (CTR) prediction is a core task of traditional service recommendation. However, many existing service recommender systems do not exploit user mobility for prediction, particularly in an edge computing environment. In this paper, we propose a model named long and short-term user preferences modeling with a multi-interest network based on user behavior. It uses a logarithmic network to capture multiple interests in different fields, enriching the representations of user short-term preferences. In terms of long-term preferences, users' comprehensive preferences are extracted in different periods and are fused using a nonlocal network. Extensive experiments on three datasets demonstrate that our model relying on user mobility can substantially improve the accuracy of service recommendation in edge computing compared with the state-of-the-art models.

Key words: recommender system; logarithmic network; nonlocal network

1 Introduction

In an edge computing environment, an edge server[1–3] is deployed between clients and the server near the mobile terminal, which can improve the service according to the device information and analyze user behaviors. On the basis of the increasing popularity of mobile devices, many mobile services have been developed that run on mobile devices and are often invoked by people accessing edge servers in edge computing. The selection of suitable services from many service candidates is an urgent task when users travel across the edges of different networks.

Recently, feature engineering[4–6] has been introduced to solve the service prediction problem and has obtained good prediction accuracy[7–9]. Click-through rate (CTR) prediction is an essential task in recommender systems, such as online advertising. To exploit the original information, finding effective transformations of the original information to improve prediction performance is essential[10]. Moreover, deep analysis of behavior data is another direction for improving service quality, where sequences of user behaviors are modeled to detect user preferences[11–13]. Recently, many CTR models have evolved from traditional methods to deep learning models. Generally, a model learns hidden data behind original information with difficulty because original data in a recommender system are often sparse and have high dimensions. Thus, if we want to build a good recommender system, feature combination is a better approach. Most deep learning models[14–18] focus on capturing interactions between

• Rui Yuan, Shunmei Meng, Ruihan Dou, and Xinna Wang are with the Department of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210056, China. E-mail: yuanrui49@126.com; mengshunmei@njust.edu.cn; wangxinnna@163.com.
• Ruihan Dou is also with the Faculty of Mathematics, University of Waterloo, Waterloo N2L 3G1, Canada. E-mail: rdou@uwaterloo.ca.
* To whom correspondence should be addressed.
Manuscript received: 2022-09-07; revised: 2022-10-16; accepted: 2022-11-04

original information in different fields. However, these models focus less on user behaviors[19–21]. The deep interest network (DIN)[22] captures the relevant interests of users and derives interest representations using an attention mechanism for CTR prediction. The deep interest evaluation network (DIEN)[23] uses GRU to fully explore actual interests and designs a new recurrent neural network, AUGRU, by integrating an attention mechanism. Although RNN-based models have shown inspiring results in characterizing sequence dependencies, they have two inherent limitations. That is, they can only model sequential activities[24], and due to this sequence dependency, they cannot be computed in a parallel manner, for instance, given a sequence of user actions $\{l_1, l_2, l_3, l_4, l_5\}$, assume that l_1 is a record of a user's office item purchases, l_2 is a laptop he bought, l_3 and l_4 are books and mugs, respectively, and l_5 is a mouse he bought for the laptop. The sequence is then modeled in strict time-order by RNN. However, in this case, the user's movement from l_4 to l_5 depends mainly on the correlation between l_5 and l_2, rather than the semantic relationship between l_4 and l_5. Actually, not all of user actions are correlated[25–27], and most existing RNN-based models typically use a sequence to represent the evolution of user interests, which in practice is often insufficient for capuring the diverse interests of users.

To solve the above problems, we propose a novel model called long- and short-term user preferences modeling with a multi-interest network (LSTMIN) for service recommendation in an edge computing environment, which learns its long- and short-term preferences from users' explicit historical behaviors and automatically adapts to their different preferences. To capture user preferences at a finer granularity from a wide range of behaviors, we combine a deep network and a logarithmic neural network. We integrate the attention mechanism into a multi-interest network to capture users' short-term preferences. In addition, we extract long-term preferences from historical behaviors using a nonlocal network. Finally, we combine long- and short-term preferences to make recommendations. In conclusion, we make the subsequent contributions.

- We design a multi-interest network to model users' short-term preferences, which learns arbitrary subsequences from a list of behaviors and their aggregations. In addition, we adopt an attention mechanism to enhance the learning process by avoiding the effects of noise generated from unrelated fields.
- Furthermore, we explore users' long-term preferences using GRU and nonlocal neural networks to design a long-term preference evolution network.
- Finally, we perform massive experiments on three real datasets, and our model is proven to outperform the state-of-the-art models.

The remainder of this paper is organized as follows: Section 2 presents our related work. Section 3 introduces factorization machines and the advantages and effectiveness of logarithmic neurons. Section 4 presents our recommendation method based on short- and long-term user preferences. Section 5 describes our implementation for experiments and performance comparisons. Section 6 concludes our paper.

2 Related Work

Early studies on CTR particularly heed feature interaction techniques, which include factorization machines (FMs), which obtain feature embeddings by casting features to a low-dimensional embedding. However, FM-based models are gradually being replaced by deep learning models because of their computational inefficiency and manual involvement in feature interaction. CTR models gradually build models through deep neural networks. Researchers apply DNNs to automatically learn the models of feature interactions. Adding an FM as prior knowledge to the model, FNN[28] uses the parameters of an FM as initial embeddings, introducing DNN for the higher-order interactions of features. FNN reduces feature engineering and enhances the learning ability of an FM. PNN[29] proposes the idea of the product layer, which is based on multiplication to represent the structure of DNN with crossing features. The major downside of FNN and PNN is that they focus more on high-order feature interactions while capturing little low-order interactions. NFM[30] improves the interaction process to enhance second-order feature interactions. The disadvantage of the above models is that they solely concentrate on second-order feature interactions and ignore higher-order interactions. DCN[31] solves this problem by introducing a cross-network and combining it with DNN to build a deep cross-network to improve the interaction between features. DeepFM[32] and Wide & Deep[33] use an FM as automatic feature engineering and use DNN to enhance generalization. However, DNNs implicitly model high-order feature interactions. The final function learned by DNNs can be arbitrary, and no theoretical conclusion has been reached on what the maximum degree of feature interaction is. These

CTR models substantially reduce the effort of manual work; however, most of these models rely on the user's side information or click events and ignore the behavior features of the users themselves.

Attention mechanisms[34] originate from NLP and have also been used in recommender systems. AFM[35] adds an attention layer to NFM, reducing the noise that can be generated by useless interactions. DIN[22] carefully considers the user's past and present items and uses an attention mechanism to model interests dynamically, capturing changes in user preferences. DIEN[23] models user interest evolution by using two GRU layers and combines attention operations into a GRU update gate to create a novel AUGRU structure. DIEN uses GRU as GRU exposes the complete memory and hidden layers, unlike LSTM. Although they do not use DNN as an abstract function of features, they use RNN as the main structure to combine the information in behavior sequences.

The above models only use the user's behavior or the user's features for interaction without considering what the behaviors represent. TDSSM[36] optimizes long- and short-term user preferences to improve recommender systems. CARA[37] captures users' short-term preferences by exploiting GRU's gate mechanism. DeepMove[38] models the sequential transition using multi-modal RNN. However, most RNN-based approaches place behaviors in the same field and do not explore the users' diversity of interests in different fields and the association between fields.

In addition, the above models do not explore the long-term preferences of users. Therefore, user behavior sequences are crucial features in recommender systems, and how to make good use of user history remains an important issue[39–42].

3 Preliminary

3.1 Factorization machines

An FM[43] is proposed for modeling interactions for features. Formally, the formula for a d-way FM is illustrated as Eq. (1):

$$\hat{y}(x) := \omega_0 + \sum_{i=1}^{n} \omega_i x_i + \sum_{l=2}^{d} \sum_{i_1=1}^{n} \cdots \sum_{i_l=i_{l-1}+1}^{n} \left(\prod_{j=1}^{l} x_{i_j} \right) \left(\sum_{f=1}^{k_l} \prod_{j=1}^{l} v_{i_j,f}^{l} \right) \tag{1}$$

where the feature interaction is expressed as an inner product, particularly the sum of the elements after multiplying both embeddings of feature pairs. ω is the weight of each feature, and n is the maximum order. The time complexity of the direct calculation Eq. (1) is $O(m^n)$, where k is the dimension of feature embedding. The FM is widely used in feature engineering, and is considered the beginning of higher-order feature interaction. In this paper, we adaptively understand the multi-interests of behaviors to obtain the most relevant fields of behaviors in terms of current behavior. The maximum order and the features are learned adaptively by the model, thus improving efficiency in computing.

3.2 Logarithmic neural network (LOGNN)

Compared with MLP, which tends to be affected by large fluctuations in exceptional inputs and results in slower convergence, LOGNN[44] can suppress exceptions and fit the data distribution better. Converting the MLP network into a logarithmic network can transform multiplication into addition, division into subtraction, and powers into multiplication exponentiation, thus making the network more adapted to power arithmetic and accelerating convergence. The logarithmic space takes Euler's number e or another appropriate number as a basis, converts the input to a logarithmic number, then multiplies the converted input by its weight vector, and finally outputs it after an activation function followed by a power operation. Figure 1a shows the structure of a logarithmic neuron.

The equation for logarithmic neurons is given in Eq. (2). ω_i is the weight of each feature x_i.

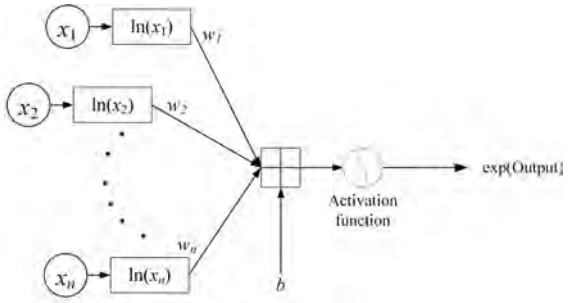
$$y = \exp\left(\sum_i \omega_i \ln x_i\right) = \prod_i x_i^{\omega_i} \tag{2}$$

On the basis of LOGN, we can create a structure named LOGNN, as shown in Fig. 1b. In LOGNN, a logarithmic layer composed of logarithmic neurons processes the raw data and feeds it into DNN, which performs deeper feature extraction on the data. In this article, we use LOGNN to adaptively learn the cross-weights of behaviors from historical user behavior.

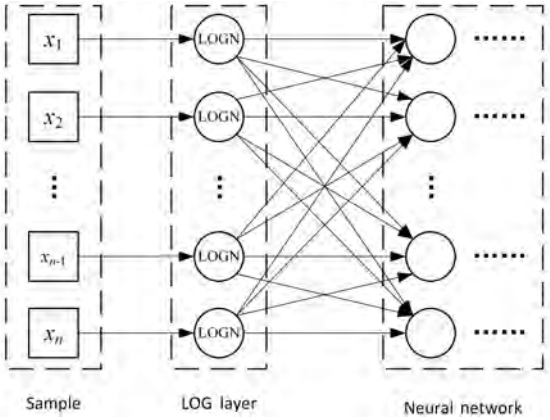
3.3 Nonlocal neural networks

In this section, we provide a general definition of nonlocal operations. Following the nonlocal mean operation, we define a generic nonlocal operation in deep neural networks as:

$$\boldsymbol{y}_i = \frac{1}{\mathcal{C}(\boldsymbol{x})} \sum_{\forall j} f\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) g\left(\boldsymbol{x}_j\right) \tag{3}$$



(a) Logarithmic neuron in LOGNN



(b) LOGNN network structure

Fig. 1 Description of LOGNN network.

where i is the index of an output position whose response is to be computed, and j is the index that enumerates all possible positions. $\boldsymbol{x}$ is the input signal (their features), and $\boldsymbol{y}$ is the output signal of the same size as $\boldsymbol{x}$. Pairwise function f computes a scalar (representing a relationship such as an affinity) between i and all j. Function g computes a representation of the input signal at position j. The response is normalized by factor $\mathcal{C}(\boldsymbol{x})$. The nonlocal behavior in Eq. (3) is due to all positions ($\forall j$) being considered in the operation. As a comparison, a convolutional operation sums the weighted input in a local neighborhood, and a recurrent operation at time i is often based only on the current and the latest time steps.

A nonlocal operation also differs from a fully connected (FC) layer. Equation (3) computes responses based on relationships between different locations, where the FC layer uses learned weights. In other words, the relationship between $\boldsymbol{X}_j$ and $\boldsymbol{X}_i$ is not a function of the input data in the FC layer,unlike in nonlocal layers. Furthermore, our formulation in Eq. (3) supports inputs of variable sizes and maintains the corresponding size in the output. In contrast, an FC layer requires a fixed-size input/output and loses positional correspondence.

Above all, a nonlocal operation allows us to build a richer hierarchy that combines nonlocal and local information.

4 LSTMIN Model for CTR Prediction

Our model has two main components, a long-term preference model and a short-term model with multi-interest extraction. Our main contribution lies in interest extraction, where the long-term preference modeling uses a nonlocal network to capture the comprehensive preferences of users and uses a LOGNN to model short-term preferences in a parallel manner to overcome the inefficiency in computation caused by serial models (e.g., RNN).

4.1 BaseModel

4.1.1 Feature representation

We use four types of features in the system: User Profiles, User Behavior, Item Features, and Contextual Information. Each feature has several types. For example User Profile contains age, gender, etc. The fields of user behavior are lists of item IDs that the user has interacted with; items are characterized by their item IDs, category IDs, etc. Contextual Information is the IDs of devices, time, location, etc. Each feature has its own one-hot encoding, and in user behavior each user contains a list of behaviors, which is represented in Eq. (4).

$$\boldsymbol{x}_B = [\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_t] \in \mathbb{R}^{K \times T}, \boldsymbol{b}_t \in \{0, 1\}^K \quad (4)$$

where $\boldsymbol{b}_t$ represents the t-th action, T is the number of user historical behaviors, and K is the number of items.

4.1.2 Structure of BaseModel

Before we introduce our model, we will take a look at the structure of BaseModel, as shown in Fig. 2. The structure includes the following parts:

- **Embedding.** This part is used to convert dense and sparse features into low-dimensional features. Each feature is associated with an embedding matrix. All features are converted into a shared latent space. For example, the embedding matrix of items is represented as $E_{\text{item}} = [i_1, i_2, \ldots, i_k] \in \mathbb{R}^{K \times d_{\text{item}}}$, where $i_k \in \mathbb{R}^{d_{\text{item}}}$ is a certain item's embedding vector with dimension d_{item}. User Behavior $\boldsymbol{x}_B$, which contains a sequence of items, can be represented by an ordered embedding vector list of behaviors $e_b = [i_{k1}, i_{k2}, \ldots, i_{kt}]$.
- **MLP.** All the embeddings from different categories are concatenated and fed into a DNN layer for final prediction.

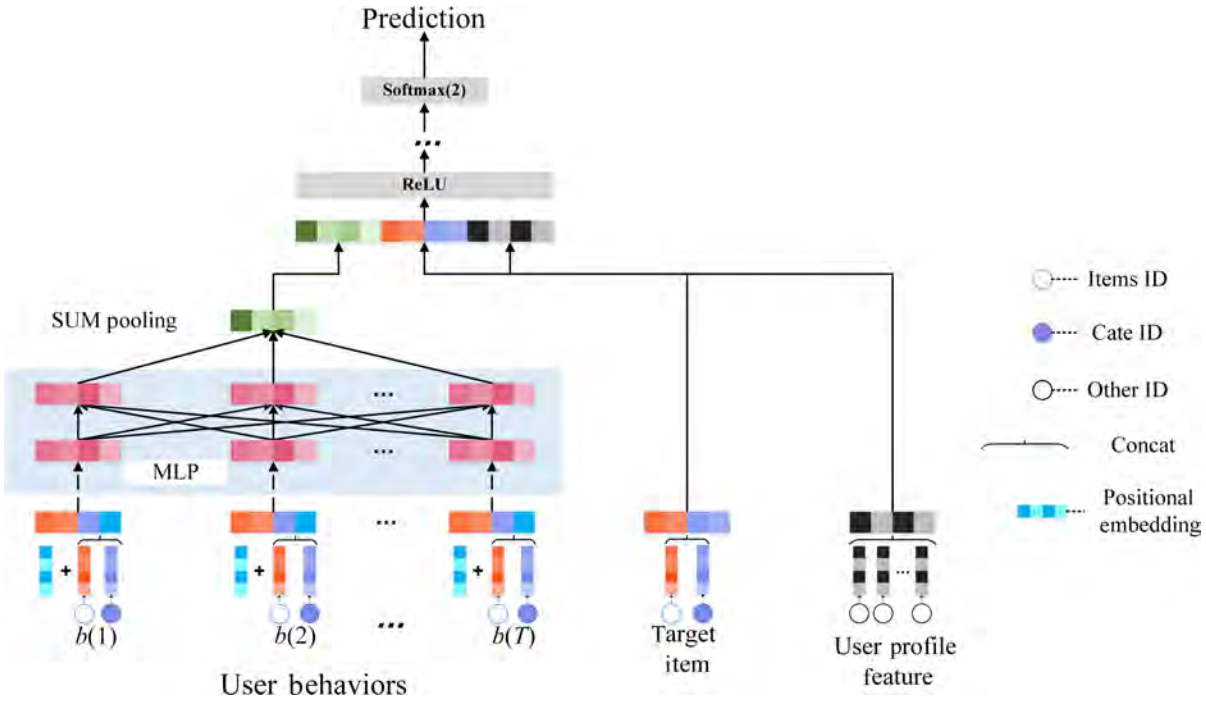


Fig. 2 Base Model.

4.1.3 Loss function

CTR models extensively use the negative log-likelihood function as the loss function, which uses the label of the target element to calculate the loss:

$$Loss = -\frac{1}{N}\sum_{j=1}^{N}(y_j \log p(x_j)+(1-y_j)\log(1-p(x_j))) \tag{5}$$

where $x_j = [x_{\mathrm{U}j}, x_{\mathrm{B}j}, x_{\mathrm{I}j}, x_{\mathrm{C}j}] \in \mathcal{D}$, $\mathcal{D}$ is a training set of size N. $y \in \{0, 1\}$ represents whether the user interacts with the target item. $p(x_j)$ is the output of the model, which represents the probability that the user interacts with the target item.

4.2 Long- and short-term user preference modeling with a multi-interest network

LSTMIN is dedicated to capturing a user's multiple interests and generating a comprehensive list of preferences. As shown in Fig. 3, LSTMIN has three components: an input and an embedding layer, which converts user behaviors and side information to a fixed-length vector space; a short-term preference extractor

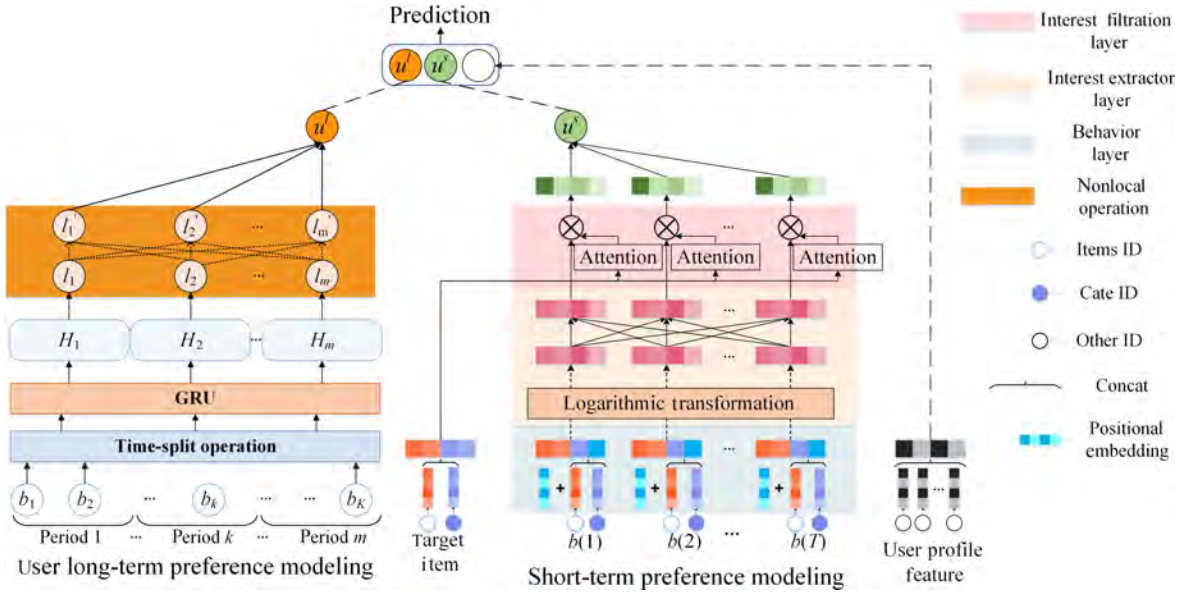


Fig. 3 Target of LSTMIN.

model, which captures user short-term multiple interests in different fields using a LOGNN; and a long-term preference extractor model, which divides user behavior into different periods and extracts user preferences in each period and then uses a nonlocal network to fuse information from each period.

4.2.1 Input layer and embedding layer

We embed the features of items themselves and their categories and then concatenate them to represent the embedding of a behavior. We must heed three points in the embedding layer. First, as we log-transform the user's behavior vector at the next level, we must keep all embeddings positive. Second, we must add a very small perturbation to zero embeddings to prevent overflow. In addition, as we use a parallel computational model (as opposed to serial models such as RNN-based models) to maintain the sequential order between behaviors, a positional embedding must be added after behavior embeddings to maintain the positional distance between two behaviors. We choose positional encoding (Eq. (6)) from Transformer[34] to this end. Finally, we obtain the set of feature embeddings $\boldsymbol{e} = \{\boldsymbol{e}_{\text{user}}, \boldsymbol{e}_{\text{item}}, \boldsymbol{e}_{\text{behavior}}\}$.

$$\begin{aligned} PE_{(pos,2i)} &= \sin\left(pos/10000^{2i/d_{\text{behavior}}}\right), \\ PE_{(pos,2i+1)} &= \cos\left(pos/10000^{2i/d_{\text{behavior}}}\right) \end{aligned} \tag{6}$$

where i is the position of each behavior in the sequence.

Next, we divide user behaviors into long- and short-term behaviors. We select the most recent T actions as the user's short-term actions, T varies in each dataset, and we will discuss how to choose T in experiments. We then divide time into m periods (in our experiment, we set $m = 48$) and extract user preferences from each period, representing the preferences of users in each period. Finally, we obtain long- and short-term preferences from behaviors to predict next behavior.

4.2.2 Short-term preference modeling

In sequential recommendation, existing RNN-based models regard a user's behavior sequence as a continuous path, considering only transfers between behaviors while neglecting the effects of multiple interests; moreover, RNN can only learn time-dependent behaviors that strictly fit the sequence, omitting the features of the behaviors themselves. Although some RNN-based methods, such as LSTM and GRU, can model across longer periods, they suffer from gradient explosion risk in which inputs do not have contextual relationships. In real life, users' short-term points of interest are usually scattered.

We argue that a sequence of a user's behaviors is a mixture of the user's interests in different fields. We introduce a novel structure to obtain several representation vectors to separately represent the user's interests in different fields. By separately considering the different interests of the users, we can better reflect their short-term preferences. We use a LOGNN to achieve this goal. We treat each user behavior as a feature and feed a fixed length of user behavior into a LOGNN which is called a multi-interest extractor layer. A single neuron in each layer of a LOGNN represents each field in which the behaviors/features are involved. Formally, the output of the j-th vector logarithmic neuron can be expressed as:

$$\boldsymbol{s}^{(j+1)} = \exp\left(\sum_{i=1}^{m} \omega_{ij} \ln \boldsymbol{s}_i\right) = \boldsymbol{s}_1^{\omega_{1j}} \circ \boldsymbol{s}_2^{\omega_{2j}} \circ \cdots \circ \boldsymbol{s}_m^{\omega_{mj}} \tag{7}$$

where m is the number of neurons in the last layer, and w_{ij} is the coefficient of the i-th neuron on the j-th layer. The functions ln and exp, as well as w_{ij}, are applied to the corresponding vectors at the vector-wise level and $\circ$ represent vector-wise product operations. Theoretically, the output of each logarithmic neuron in the next layer $s^{(j+1)}$ can represent any interest field. For example, when w_{1j} and w_{2j} are set to 1, it indicates that the first and second features are involved in this field, and the intensity of engagement is 1. Thus, we can stack multiple logarithmic neurons, which can extract any fields of features from a sequence as the output of this layer. Although intuitively, 0 and 1 represent whether the behavior is or is not engaged in this field, respectively, we decide to make the parameters in the coefficient matrix learned by neural network $\boldsymbol{W}_{\log nn} \in \boldsymbol{R}^{m \times L}$ (where L is the logarithmic neurons' number in this layer) not need to converge to 0 and 1, allowing the network to automatically learn this coefficient (the intensity of engagement of the feature), helping the model to better extract useful information.

To avoid the underfitting problem caused by noise from invalid fields, we introduced an attention layer after the LOGNN layer to improve this extraction process. The introduction of an attention mechanism also solves the problem that RNNs cannot be computed in parallel. We will assign weights to embeddings for each field. Formally,

$$\boldsymbol{h}_i = \text{RELU}(\boldsymbol{W}_{\text{att}}[s_i, s_i \otimes i_{\text{target\_item}}, i_{\text{target\_item}}] + \boldsymbol{bias}) \tag{8}$$

$$a_i = \frac{\exp\left(\boldsymbol{u}^\top \boldsymbol{h}_i\right)}{\sum_{p \in S} \exp\left(\boldsymbol{u}^\top \boldsymbol{h}_p\right)} \tag{9}$$

where $\boldsymbol{W}_{\text{att}} \in \mathbb{R}^{d_{\text{att}} \times (d_{\text{item}}^2 + 2d_{\text{item}})}$ and $\boldsymbol{bias} \in \mathbb{R}^{d_{\text{att}} \times 1}$ are model parameters, and $\boldsymbol{s}_i$ represents the embedding of behaviors $\boldsymbol{s} \in S$ in each field where S is the output of LOGNN. After obtaining normalized attention scores, the short-term preference is calculated as follows:

$$\boldsymbol{s} = [a_0 s_0, \ldots, a_i s_i, \ldots, a_{|S|} s_{|S|}], \text{where } \boldsymbol{s} \in S \tag{10}$$

where S is the logarithmic neural layer's output units.

After the filtration layer, we feed the resulting vectors into DNN for a better fit, and the output is a vector representing users' short-term preferences $\boldsymbol{u}^{\text{short}}$:

$$\boldsymbol{x}^{(l+1)} = \text{relu}(W^{(l)} x^{(l)} + b^{(l)}) \tag{11}$$

where $x^{(l)}$, $W^{(l)}$, and $b^{(l)}$ are inputs, the DNN layer's weight, and the bias of the l-th layer. Finally, a user's short-term preference representation can be represented as $\boldsymbol{u}^s = W x^{(\text{last})} + b$.

4.2.3 Long-term preference modeling

We input the user's behavior over all periods into the long-term preference model. As illustrated in Fig. 3, we use a nonlocal newtork structure to learn a hidden representation of the target user's long-term preferences u^{long}. First, given a user u, we initially split long-term behaviors in terms of time. People's interests will change over time in real scenarios. For example, people tend to store food during the New Year holidays, with a tendency to buy clothes in the spring and winter. Thus, we divide the entire time span into 48 periods (24 periods for hours on weekdays and 24 periods for hours on weekends).

We use GRU to model the dependency between behaviors within each period. The formulation of GRU is listed as follows:

$$\begin{aligned}
\boldsymbol{u}_t &= \sigma\left(W^u \boldsymbol{b}_t + U^u \boldsymbol{h}_{t-1} + \boldsymbol{bias}^u\right), \\
\boldsymbol{r}_t &= \sigma\left(W^r \boldsymbol{b}_t + U^r \boldsymbol{h}_{t-1} + \boldsymbol{bias}^r\right), \\
\tilde{\boldsymbol{h}}_t &= \tanh\left(W^h \boldsymbol{b}_t + \boldsymbol{r}_t \circ U^h \boldsymbol{h}_{t-1} + \boldsymbol{bias}^h\right), \\
\boldsymbol{h}_t &= (\mathbf{1} - \boldsymbol{u}_t) \circ \boldsymbol{h}_{t-1} + \boldsymbol{u}_t \circ \tilde{\boldsymbol{h}}_t
\end{aligned} \tag{12}$$

where σ is the activation function, $\circ$ is an element-wise product, $W^u, W^r, W^h \in \mathbb{R}^{d_{\text{hid}} \times d_{\text{beh}}}$, U^u, $U^r, U^h \in \mathbb{R}^{d_{\text{hid}} \times d_{\text{hid}}}$ is the hidden weight matrix, and d_{beh} is the dimension of the behavior embedding, which equals d_{item}. $\boldsymbol{h}_t$ is the t-th hidden states.

The key difference between GRU and LSTM is that GRU's bag has two gates, reset and update, while LSTM has three gates, input, output, and forget. GRU is less complex than LSTM because it has fewer gates, so we choose GRU as an abstract function in the long-term preference model.

Then, we obtain all h_t hidden states from the hidden layer. In this way, the behaviors $b_1^1, \ldots, b_1^t, \ldots, b_m^t$ in each period m are encoded into $h_1^1, \ldots, h_1^t, \ldots, h_m^t$, and the hidden states in each period make up hidden matrices $H_1, H_2, \ldots, H_m$.

To obtain a fixed-length vector for each period from variable numbers of behaviors, we propose to transform them through a pooling layer. Common pooling operations are sum pooling and average pooling. In our model, we use an average pooling layer, and the average operation is listed as follows:

$$l_m = \frac{1}{|P_m|} \sum_{i=1}^{|P_m|} h_i \tag{13}$$

After learning all the representations $l_1, l_2, \ldots, l_m$, we use the nonlocal operation to derive a user's long-term preferences $\boldsymbol{u}^{\text{long}} \in \mathbb{R}^{d \times 1}$ to identify similar behaviors sufficiently to represent long-term preferences.

The nonlocal network[45] aims to model nonlocal, remote dependencies by considering the features at all locations. In our experiments, we tend to use it to capture the effect of each historical period $P \in \{P_1, P_2, \ldots, P_m\}$. Formally, we use Eq. (14) to calculate l'.

$$l_i' = \frac{1}{C(S)} \sum_{i=1}^{m} \sum_{j=1}^{m} f\left(l_i, l_j\right) g\left(l_i\right) \tag{14}$$

where $C(S) = \sum_{i=1}^{m} f\left(l_i, l_j\right)$ is the normalization factor, function $g(l_i)$ calculates the representation of l_i, and function $f(\cdot)$ calculates scores for the i-th and j-th periods. Formally, $f\left(l_i, l_j\right)$ and $g\left(l_i\right)$ are defined as follows:

$$\begin{aligned}
f\left(l_i, l_j\right) &= \exp\left(l_i^\top l_j\right), \\
g\left(l_i\right) &= W_{nl} l_i
\end{aligned} \tag{15}$$

where l_i and l_j are representatives of the i-th period and the j-th historical period, respectively. W_{nl} is a trainable weight matrix in a nonlocal network. Additionally, we obtain the representation of the long-term preferences of the user. $\boldsymbol{u}^l = \frac{1}{m} \sum_{i=1}^{m} l_i'$.

Finally, we integrate $\boldsymbol{u}^s$, $\boldsymbol{u}^l$, and the profile feature, forwarding through an MLP layer and a sigmoid activation function to obtain the final result[46]. Then, we use the loss function in BaseModel to train the network.

4.3 Training techniques

Dropout layers are often used to avoid the overfitting of neural networks. In our model, we choose a novel dropout layer for behavior sequences to prevent further overfitting. Dropout is applied after the behavior layer,

where some visit results are randomly removed to reduce the model's sensitivity to noise. For example, users sometimes accidentally visit items they dislike or interact with unhelpful behaviors, so the dropout layer can be considered a way to augment data to avoid overfitting and improve the robustness of the model.

For example, as shown in Fig. 4, the user has visited a total of four items in a particular sequence of user actions. The unit marked with a question mark is the output result corresponding to its training sequence, and the unit with a dashed outline is randomly dropped during the training process.

5 Experimental Result and Analysis

In this section, we tend to measure our model by competing against several methods on three real-world datasets.

5.1 Experimental settings

5.1.1 Datasets

We conducted experiments on three usual datasets, Amazon[47], and Movielens[48]. We used a subset of Amazon's electronics data and movies and TV data, and in this dataset, we treated reviews as behaviors. The Movielens datase comprises user-tagged records of films. We treat ratings by users as interactions and sort them by timestamp. Assuming that user u has t behaviors, we aim to use $t-1$ behaviors to predict the user's t-th behavior. Statistics for datasets after preprocessing are shown in Table 1.

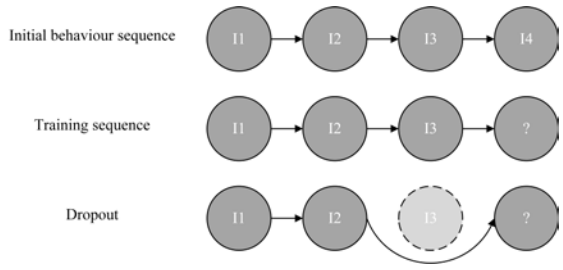


Fig. 4 Dropout after behavior sequence.

5.1.2 Evaluation metrics

Two metrics were used for performance evaluation: AUC and Logloss. Improvements in AUC and Logloss are considered important in CTR prediction.

5.1.3 Comparison methods

We compare our model to the following CTR prediction methods:

- **BaseModel** takes the same embeddings and uses a deep neural network to predict preferences.
- **Wide & Deep**[33] is composed of two parts: The deep part corresponds to the base model. The wide part is a linear model that uses manually crossed features for better interaction.
- **PNN**[29] designs a unique product layer for a wider crossover between features.
- **xDeepFM**[49] generates higher-order cross-features by computing the outer products of different feature vectors.
- **Two-layer GRU Attention**[50] uses two layers of GRU to model the users' whole preference and uses attention mechanisms to improve the model.

5.2 Performance comparison

5.2.1 Comparing with individual models

STSIN is a model that uses only short-term interests to make predictions. We present AUC in Table 2, which clearly shows that Wide & Deep relies mainly

Table 1 Statistics of datasets.

Dataset	User	Item	Behavior	Sparity (%)	Average number	Max. number
Electronic	728 719	159 936	6 732 848	99.9942	9.23	604
Movies and TV	297 529	60 111	3 404 812	99.9809	11.44	3508
Movielens-25m	162 541	62 423	25 000 095	99.7536	153.80	32 202

Table 2 Overall performance of models on Movielens and Amazon datasets.

Model	Movielens		Amazon_Electronics		Amazon_Movies and TV	
	AUC	Logloss	AUC	Logloss	AUC	Logloss
BaseModel	0.8391	0.2956	0.8438	0.2859	0.8846	0.3448
Wide & Deep	0.8081	0.3310	0.8146	0.3256	0.8656	0.3784
PNN	0.8169	0.3092	0.8282	0.2884	0.8835	0.3680
xDeepFM	0.8348	0.2917	0.8308	0.2856	0.8974	0.3326
Two-layer GRU attention	0.8438	0.2843	0.8504	0.2837	0.9347	0.3261
STMIN(Ours)	0.8580	0.2739	0.8657	0.2759	**0.9366**	0.3198
LSTMIN(Ours)	**0.8653**	**0.2712**	**0.8673**	**0.2753**	0.9349	**0.3187**

on artificial features and performs poorly, while PNN improves feature interaction and has better performance. xDeepFM further improves PNN to achieve an optimal auto-interaction of features. Two-layer GRU with attention simulates the evolution of interest in sequence and captures the interest evolution more effectively, greatly improving performance. Our model combines the strengths of the above models. The results suggest that behaviors are correlated and that the exploration of interest evolution is necessary, which can be verified because DIN, two-layer GRU with attention and STSIN outperform an FM on the three datasets. In conclusion, STSIN is the best individual model, which validates multi-interest modeling on short-term preferences.

5.2.2 Comparing with integrated models

LSTMIN combines long- and short-term behavior for prediction. We compare LSTMIN with several integrated models. As illustrated in Table 2, LSTMIN achieves the optimal performance in all three datasets. This result suggests that discrepancies exist between users' long- and short-term preferences. LSTMIN considers both long- and short-term user preferences and achieves better performance.

After analyzing the above metrics, let us look at the classification effect of the model. We extracted 3000 pieces from test data, used LSTSIN to make classification predictions, and plotted the results in a scatter plot, as shown in Fig. 5a. The closer a point is to 1, the higher the possibility that the user will act on the item; the abscissa is a random floating point number from 0 to 1, which is intended to scatter the points. The blue and red points in the graph indicate the positive and negative samples of users respectively. In Fig. 5a, the blue and red points are mostly concentrated in the upper and lower areas, respectively. When the threshold is 0.3998, the model obtains the optimal AUC of 0.8673.

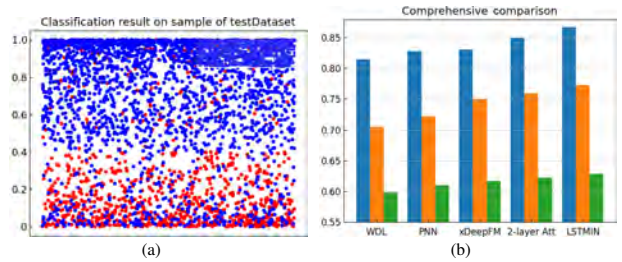


Fig. 5 Comparison results of LSTMIN with several integrated models. (a) Depicts of the model's intuitive classification effect (red: negative points, and blue: positive points), and (b) comparison of the comprehensive indicators of each model (blue: AUC, orange: F1, and green: G-Mean).

The division of the threshold in Fig. 5a is also consistent with intuitive perception.

Figure 5b shows that LSTSIN outperforms all other models in all comprehensive indicators. This result is mainly due to the combination of long- and short-term preferences in LSTSIN, which makes the final prediction results more accurate; short-term preference modeling can extract interests from different fields, which makes the model more robust in interest representation; long-term preference modeling can extract the interests of users in different periods and derive the comprehensive preferences.

5.3 Hyperparameter investigation

We next consider the impact of four hyperparameters on performance, which we summarize for the Amazon_Electronics dataset, as the remaining two datasets perform similarly.

5.3.1 Length of history records

A critical contribution of our model is that it considers past information to capture long- and short-term user preferences. We investigate the impact of different history lengths on modeling short-term preferences. We conduct our experiments on the Amazon dataset because of its abundance of user behaviors, and the number of behaviors per user fluctuates slightly, facilitating our analysis. The result is illustrated in Fig. 6, which indicates that the performance tends to rise and then fall as the length of the history increases. This result is obtained because the model must learn more features to model people's behavior patterns, but more user behaviors mean that the model tends to long-term user preferences more rather than short-term preferences, which leads to a decrease in prediction accuracy.

5.3.2 Number of logarithmic neurons

Each logarithmic neuron represents the different fields of interest of users. Figure 7 shows the results for different

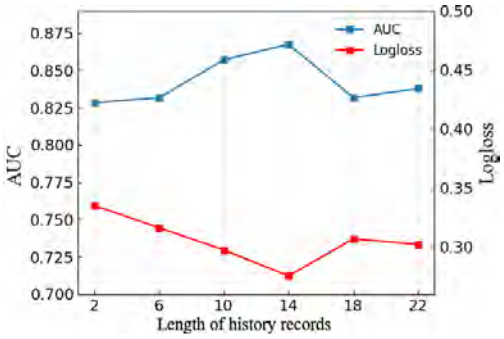


Fig. 6 AUC against length of history records.

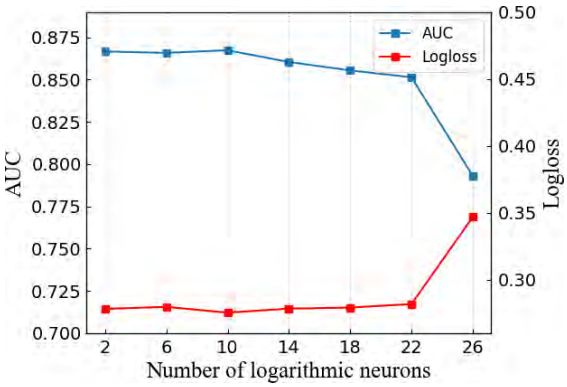


Fig. 7 AUC against number of logarithmic neurons.

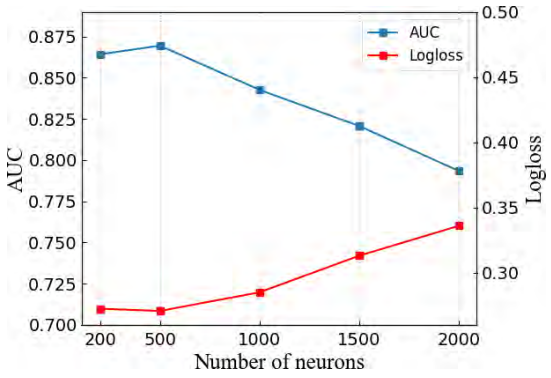


Fig. 9 AUC against number of neurons.

numbers of neurons in the logarithmic transformation layer. The performance of the model remains stable as the number of logarithmic neurons increases. However, the model shows a downward trend as this number becomes very large. This result shows that the user's interest is finite, and we should choose the correct number of logarithmic neurons to reach the best performance. When the model has only two fields of interest, our model still performs well but not optimally, which suggests that finding reliable fields is crucial for prediction accuracy.

5.3.3 Depth of hidden layers and the number of neurons

Figure 8 shows the effect of the depth of the hidden layers. The performance of LSTMIN first increases with the network's depth but then decreases because of overfitting when the depth exceeds three layers. Importantly, when the depth is 0, our model still achieves good results by directly learning the extractions. This finding proves the effectiveness of the LOGNN layer in learning multi-interest extractions. We fixed the depth to three and increased the number of neurons in the hidden layer. The results, illustrated in Fig. 9, indicate that the performance first increases with neuron number as more parameters improve the fitting ability. When this number is greater than 500, performance begins to decrease because of overfitting.

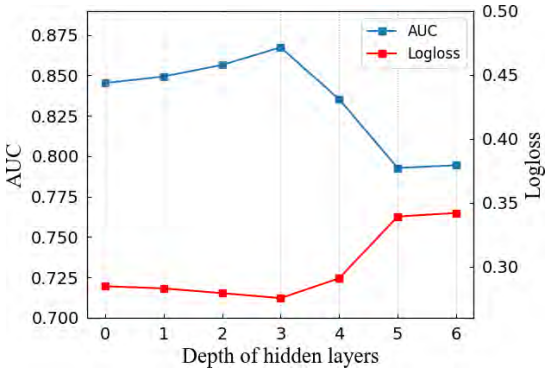


Fig. 8 AUC against depth of hidden layers.

6 Conclusion

In contrast to traditional service recommendations based on original information, in this paper, we introduce a CTR prediction method for service recommendation in an edge computing environment, with a novel long- and short-term user preferences model with a multi-interest network, which learns user long- and short-term preferences from sequence data. In short-term preference extraction, LSTMIN can automatically generate subsequences related to target items to extract the user's multi-interests in different fields and then use an attention mechanism to improve this extraction process. Additionally, LSTMIN extends the long-term preference network, using nonlocal operations to capture preferences from long-term historical behaviors for click prediction. Extensive experiments on three datasets prove that LSTMIN has excellent performance compared to the prior state-of-the-art models. For our future work, we will follow two directions. One is to find the impact of time when the service is recommended after entering a new edge station, such as behavior time. As in edge computing, the configurations and running status of devices probably are unstable and may change with the environment. We will continuously improve our model in the future.

Acknowledgment

This paper was partially supported by the Open Research Project of the State Key Laboratory of Novel Software Technology (Nanjing University) (No. KFKT2022B28), National Key R&D Program of China

(No. 2020YFB1804604), the National Natural Science Foundation of China (Nos. 61702264, 62076130, and 61872219), the 2020 Industrial Internet Innovation and Development Project from Ministry of Industry and Information Technology of China, the Fundamental Research Fund for the Central Universities (Nos. 30918012204, 30920041112, and 30919011282), and the Postdoctoral Science Foundation of China (No. 2019M651835).

References

[1] M. H. Rimaz, R. Hosseini, and F. B. Moghaddam, AudioLens: Audio-aware video recommendation for mitigating new item problem, in *Service-Oriented Computing–ICSOC 2020 Workshops*, H. Hacid, F. Outay, H. Y. Paik, A. Alloum, M. Petrocchi, M. R. Bouadjenek, A. Beheshti, X. M. Liu, and A. Maaradji, Eds. Cham, Switzerland: Springer, 2020, pp. 365–378.

[2] F. Wang, G. Li, Y. Wang, W. Rafique, M. R. Khosravi, G. F. Liu, Y. Liu, and L. Qi, Privacy-aware traffic flow prediction based on multi-party sensor data with zero trust in smart city, *ACM Trans. Internet Technol.*, 2022, doi: 10.1145/3511904.

[3] Y. Li, J. Liu, B. Cao, and C. Wang, Joint optimization of radio and virtual machine resources with uncertain user demands in mobile cloud computing, *IEEE Trans. Multimedia*, vol. 20, no. 9, pp. 2427–2438, 2018.

[4] A. Metzger, C. Quinton, Z. Á. Mann, L. Baresi, and K. Pohl, Feature model-guided online reinforcement learning for self-adaptive services, in *Proc. 18th Int. Conf. on Service-Oriented Computing*, Dubai, United Arab Emirates, 2020, pp. 269–286.

[5] X. Xu, Q. Jiang, P. Zhang, X. Cao, M. R. Khosravi, L. T. Alex, L. Qi, and W. Dou, Game theory for distributed IoV task offloading with fuzzy neural network in edge computing, *IEEE Trans. Fuzzy Syst.*, vol. 30, no. 11, pp. 4593–4604, 2022.

[6] X. Xu, H. Li, W. Xu, Z. Liu, L. Yao, and F. Dai, Artificial intelligence for edge service optimization in internet of vehicles: A survey, *Tsinghua Science and Technology*, vol. 27, no. 2, pp. 270–287, 2022.

[7] X. Xu, H. Tian, X. Zhang, L. Qi, Q. He, and W. Dou, DisCOV: Distributed COVID-19 detection on X-ray images with edge-cloud collaboration, *IEEE Trans. Serv. Comput.*, vol. 15, no. 3, pp. 1206–1219, 2022.

[8] H. Dai, J. Yu, M. Li, W. Wang, A. X. Liu, J. Ma, L. Qi, and G. Chen, Bloom filter with noisy coding framework for multi-set membership testing, *IEEE Trans. Knowl. Data Eng.*, pp. 1–14, 2022, doi: 10.1109/TKDE.2022.3199646.

[9] L. Qi, W. Lin, X. Zhang, W. Dou, X. Xu, and J. Chen, A correlation graph based approach for personalized and compatible web APIS recommendation in mobile APP development, *IEEE Trans. Knowl. Data Eng.*, 2022, doi: 10.1109/TKDE.2022.3168611.

[10] J. Liu, Z. Zhao, J. Ji, and M. Hu, Research and application of wireless sensor network technology in power transmission and distribution system, *Intelligent and Converged Networks*, vol. 1, no. 2, pp. 199–220, 2020.

[11] J. Mabrouki, M. Azrour, D. Dhiba, Y. Farhaoui, and S. El Hajjaji, IoT-based data logger for weather monitoring using Arduino-based wireless sensor networks with remote graphical application and alerts, *Big Data Mining and Analytics*, vol. 4, no.1, pp. 25–32, 2021.

[12] C. Catlett, P. Beckman, N. Ferrier, H. Nusbaum, M. E. Papka, M. G. Berman, and R. Sankaran, Measuring cities with software-defined sensors, *Journal of Social Computing*, vol. 1, no. 1, pp. 14–27, 2020.

[13] Y. S. Su, Y. Ruan, S. Sun, and Y. T. Chang, A pattern recognition framework for detecting changes in Chinese internet management system, *Journal of Social Computing*, vol. 1, no. 1, pp. 28–39, 2020.

[14] B. Chen, Y. Wang, Z. Liu, R. Tang, W. Guo, H. Zheng, W. Yao, M. Zhang, and X. He, Enhancing explicit and implicit feature interactions via information sharing for parallel deep CTR models, in *Proc. 30th ACM Int. Conf. on Information & Knowledge Management*, Gold Coast, Australia, 2021, pp. 3757–3766.

[15] K. Zhang, H. Qian, Q. Cui, Q. Liu, L. Li, J. Zhou, J. Ma, and E. H. Chen, Multi-interactive attention network for fine-grained feature learning in CTR prediction, in *Proc. 14th ACM Int. Conf. on Web Search and Data Mining*, virtual, 2021, pp. 984–992.

[16] H. Fei, J. Zhang, X. Zhou, J. Zhao, X. Qi, and P. Li, GemNN: Gating-enhanced multi-task neural networks with feature interaction learning for CTR prediction, in *Proc. 44th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, virtual, 2021, pp. 2166–2171.

[17] C. Wu, F. Wu, L. Lyu, Y. Huang, and X. Xie, FedCTR: Federated native ad CTR prediction with cross-platform user behavior data, *ACM Trans. Intell. Syst. Technol.*, vol. 13, no. 4, pp. 62, 2022.

[18] Z. Xu, D. Li, W. Zhao, X. Shen, T. Huang, X. Li, and P. Li, Agile and accurate CTR prediction model training for massive-scale online advertising systems, in *Proc. 2021 Int. Conf. on Management of Data*, virtual, 2021, pp. 2404–2409.

[19] M. Z. Siddiqi and T. Mir, Reconfigurable intelligent surface-aided wireless communications: An overview, *Intelligent and Converged Networks*, vol. 3, no. 1, pp. 33–63, 2022.

[20] A. K. Sandhu, Big data with cloud computing: Discussions and challenges, *Big Data Mining and Analytics*, vol. 5, no. 1, pp. 32–40, 2022.

[21] H. Zhao, Z. Liu, J. Tang, B. Gao, Y. Zhang, H. Qian, and H. Wu, Memristor-based signal processing for edge computing, *Tsinghua Science and Technology*, vol. 27, no. 3, pp. 455–471, 2022.

[22] G. Zhou, X. Zhu, C. Song, Y. Fan, H. Zhu, X. Ma, Y. Yan, J. Jin, H. Li, and K. Gai, Deep interest network for click-through rate prediction, in *Proc. 24th ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining*, London, UK, 2018, pp. 1059–1068.

[23] G. Zhou, N. Mou, Y. Fan, Q. Pi, W. Bian, C. Zhou, X. Zhu, and K. Gai, Deep interest evolution network for click-through rate prediction, in *Proc. 31st AAAI Conf. on Artificial Intelligence*, Honolulu, HA, USA, 2019, pp. 5941–5948.

[24] Y. Li, S. Xia, M. Zheng, B. Cao, and Q. Liu, Lyapunov optimization-based trade-off policy for mobile cloud offloading in heterogeneous wireless networks, *IEEE Trans. Cloud Comput.*, vol. 10, no. 1, pp. 491–505, 2022.

[25] T. Li, C. Li, J. Luo, and L. Song, Wireless recommendations for internet of vehicles: Recent advances, challenges, and opportunities, *Intelligent and Converged Networks*, vol. 1, no. 1, pp. 1–17, 2020.

[26] H. Kou, H. Liu, Y. Duan, W. Gong, Y. Xu, X. Xu, and L. Qi, Building trust/distrust relationships on signed social service network through privacy-aware link prediction process, *Appl. Soft Comput.*, vol. 100, p. 106942, 2021.

[27] Y. Li, C. Liao, Y. Wang, and C. Wang, Energy-efficient optimal relay selection in cooperative cellular networks based on double auction, *IEEE Trans. Wireless Commun.*, vol. 14, no. 8, pp. 4093–4104, 2015.

[28] W. Zhang, T. Du, and J. Wang, Deep learning over multi-field categorical data, in *Proc. 38th European Conf. on Advances in Information Retrieval*, Padua, Italy, 2016, pp. 45–57.

[29] Y. Qu, H. Cai, K. Ren, W. Zhang, Y. Yu, Y. Wen, and J. Wang, Product-based neural networks for user response prediction, in *Proc. 2016 IEEE 16th Int. Conf. on Data Mining*, Barcelona, Spain, 2016, pp. 1149–1154.

[30] X. He and T. S. Chua, Neural factorization machines for sparse predictive analytics, in *Proc. 40th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Tokyo, Japan, 2017, pp. 355–364.

[31] R. Wang, B. Fu, G. Fu, and M. Wang, Deep & cross network for ad click predictions, in *Proc. ADKDD'17*, Halifax, Canada, 2017, p. 12.

[32] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, DeepFM: A factorization-machine based neural network for CTR prediction, in *Proc. 26th Int. Joint Conf. on Artificial Intelligence*, Melbourne, Australia, 2017, pp. 1725–1731.

[33] H. T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, et al., Wide & deep learning for recommender systems, in *Proc. 1st Workshop on Deep Learning for Recommender Systems*, Boston, MA, USA, 2016, pp. 7–10.

[34] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, ? Kaiser, and I. Polosukhin, Attention is all you need, in *Proc. 31st Int. Conf. on Neural Information Processing Systems*, Red Hook, NY, USA, 2017, pp. 6000–6010.

[35] J. Xiao, H. Ye, X. He, H. Zhang, F. Wu, and T. S. Chua, Attentional factorization machines: Learning the weight of feature interactions via attention networks, in *Proc. 26th Int. Joint Conf. on Artificial Intelligence*, Melbourne, Australia, 2017, pp. 3119–3125.

[36] Y. Song, A. M. Elkahky, and X. D. He, Multi-rate deep learning for temporal recommendation, in *Proc. 39th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Pisa, Italy, 2016, pp. 909–912.

[37] J. Manotumruksa, C. Macdonald, and I. Ounis, A contextual attention recurrent architecture for context-aware venue recommendation, in *Proc. 41st Int. ACM SIGIR Conf. on Research & Development in Information Retrieval*, Ann Arbor, MI, USA, 2018, pp. 555–564,

[38] J. Feng, Y. Li, C. Zhang, F. Sun, F. Meng, A. Guo, and D. Jin, DeepMove: Predicting human mobility with attentional recurrent networks, in *Proc. 2018 World Wide Web Conf.*, Lyon, France, 2018, pp. 1459–1468.

[39] W. Gu, F. Gao, R. Li, and J. Zhang, Learning universal network representation via link prediction by graph convolutional neural network, *Journal of Social Computing*, vol. 2, no. 1, pp. 43–51, 2021.

[40] J. Li, A. M. V. V. Sai, X. Cheng, W. Cheng, Z. Tian, and Y. Li, Sampling-based approximate skyline query in sensor equipped IoT networks, *Tsinghua Science and Technology*, vol. 26, no. 2, pp. 219–229, 2021.

[41] M. Azrour, J. Mabrouki, A. Guezzaz, and Y. Farhaoui, New enhanced authentication protocol for internet of things, *Big Data Mining and Analytics*, vol. 4, no. 1, pp. 1–9, 2021.

[42] L. Qi, Y. Liu, Y. Zhang, X. Xu, M. Bilal, and H. Song, Privacy-aware point-of-interest category recommendation in internet of things, *IEEE Internet Things J.*, vol. 9, no. 21, pp. 21398–21408, 2022.

[43] S. Rendle, Factorization machines, in *Proc. 2010 IEEE Int. Conf. on Data Mining*, Sydney, Australia, 2010, pp. 995–1000.

[44] Z. Wang, Z. Xu, D. He, and S. Chan, Deep logarithmic neural network for internet intrusion detection, *Soft Comput.*, vol. 25, no. 15, pp. 10129–10152, 2021.

[45] T. Chen, H. Yin, H. Chen, L. Wu, H. Wang, X. Zhou, and X. Li, TADA: Trend alignment with dual-attention multi-task recurrent neural networks for sales prediction, in *Proc. 2018 IEEE Int. Conf. on Data Mining*, Singapore, 2018, pp. 49–58.

[46] L. Kong, L. Wang, W. Gong, C. Yan, Y. Duan, and L. Qi, LSH-aware multitype health data prediction with privacy preservation in edge environment, *World Wide Web*, vol. 25, no. 5, pp. 1793–1808, 2022.

[47] J. Ni, J. Li, and J. McAuley, Justifying recommendations using distantly-labeled reviews and fine-grained aspects, in *Proc. 2019 Conf. on Empirical Methods in Natural Language Processing and the 9th Int. Joint Conf. on Natural Language Processing*, Hong Kong, China, 2019, pp. 188–197.

[48] F. M. Harper and J. A. Konstan, The MovieLens datasets: History and context, *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, pp. 19, 2015.

[49] J. Lian, X. Zhou, F. Zhang, Z. Chen, X. Xie, and G. Sun, xDeepFM: Combining explicit and implicit feature interactions for recommender systems, in *Proc. 24th ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining*, London, UK, 2018, pp. 1754–1763.

[50] M. Parsana, K. Poola, Y. Wang, and Z. Wang, Improving native ads CTR prediction by large scale event embedding and recurrent networks, arXiv preprint arXiv: 1804.09133, 2018.

Rui Yuan is a graduate student at the Department of Computer Science and Engineering, Nanjing University of Science and Technology, China. His main research interests are data mining, recommender systems, neural networks and big data analysis.

Shunmei Meng received the PhD degree from the Department of Computer Science and Technology, Nanjing University, China, in 2016. She is an assistant professor at the Department of Computer Science and Engineering, Nanjing University of Science and Technology, China. She has published papers in international journals and international conferences, such as TPDS, TII, WWWJ, FGCS, COSE, ICDM, ICWS, and ICSOC. Her research interests include recommender systems, cloud computing, and security and privacy.

Ruihan Dou is currently a bachelor student in Faculty of Mathematics, University of Waterloo, Canada. His research interests include data statistics, financial analysis and risk management.

Xinna Wang is currently pursuing the MA degree at the Department of Computer Science and Engineering, University of Nanjing University of Science and Technology, China. Her current research interests include recommendation systems, machine learning, and neural networks.