IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 30 August 2022, accepted 24 September 2022, date of publication 3 October 2022, date of current version 7 October 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3211656

RESEARCH ARTICLE

A CTR Prediction Model With Double Matrix-Level Cross-Features

WEI ZHANG, YAHUI HAN, ZHAOBIN KANG, AND KAIYUAN QU
Department of Artificial Intelligence Education, Central China Normal University, Wuhan 430079, China

Corresponding author: Zhaobin Kang (kangzhaobin@mails.ccnu.edu.cn)

This work was supported by the Bing-Tuan Science and Technology Public Relations Project, a Data-Driven Regional Smart Education Service Key Technology Research and Application Demonstration, under Grant 2021AB023.

ABSTRACT CTR prediction is an important task in recommender systems, which is used to estimate the likelihood of a user clicking on an advertisement.In the past, the CTR prediction model based on the deep neural network mainly obtains the implicit feature combination of the model at the bit-wise level, and the interpretability and generalization of the model are poor. At the same time, the prediction accuracy of the model is poor. For the above problems, We propose a click-through rate prediction model (DTM) with double matrix-level cross-features. The model integrates various components such as multi-head self-attention, residual network and interaction network into an end-to-end model, and automatically obtains explicit feature combinations at the vector-wise level and bit-wise level, which not only has better interpretability, generalization and memory, and reduce the inherent flaws and engineering complexity of multi-modules. The experimental results show that on the datasets Criteo and Avazu, compared with other state-of-the-art CTR prediction models, the AUC values of the DTM model are increased by 4% and 3% on average, and the loss values are decreased by 3.5% and 2.8% on average, respectively.

INDEX TERMS Click-through rate prediction, recommender systems, feature interaction, multi-head self-attention, cross network.

I. INTRODUCTION

Accurately representing features is very important for CTR prediction. Features play a central role in the success of many CTR prediction systems [1], [2]. Since using raw features rarely yields the best results, data scientists often transform raw features heavily in order to obtain the best combination of features. A major type of feature transformation is the cross-product transformation of categorical features [3], [4]. These features are called cross-features, and they measure the interaction between multiple original features. For example, if a user works at Alibaba and presents a technical article on deep learning on Monday. Then a third-order cross-feature (user_organization = Alibaba, item_category = deep learning, time = Monday) has a value of 1 [5], [6], [7].

Traditional cross-feature engineering suffers from three main drawbacks. First, there is a high cost to obtain high-quality features. Because correct features are usually based on specific tasks, data scientists need to spend a lot of time exploring the potential information in the data before becoming domain experts and extracting meaningful cross features [8], [9]. Second, in a large-scale recommendation system, a large number of original features make it impossible to extract all the cross-features manually [10], [11]. Third, handcrafted cross-features do not generalize to interactions not seen in the training data [12]. Therefore, learning interaction features without human engineering is a meaningful task.

Factorization Machine(FM) [12] is often used to obtain explicit low-order features. FM embeds each feature i into a latent factor vector v_i, and paired features are modeled as the inner product of latent vectors. Classical FM can be extended to arbitrary high-order feature interactions, but in practice, because of combinatorial expansion, it has high complexity. In recent years, deep neural networks (DNNs) have achieved success in computer vision [13], speech recognition [14], and natural language processing [15] due to their powerful feature representation learning capabilities.

The associate editor coordinating the review of this manuscript and approving it for publication was Jon Atli Benediktsson.

Leveraging deep neural networks to learn complex and selective feature interactions is promising. FNN [16] proposed a factorization machine-supported neural network to learn implicit high-order feature interactions, which used pre-trained factorization machines for domain embedding before DNNs. PNN [17] further proposes a product-based neural network, which introduces a product layer between the embedding layer and the DNN layer without relying on a pre-trained FM. The main disadvantage of FNNs and PNNs is that they mainly acquire implicit high-order feature interactions, while ignoring the effect of low-order feature combinations. The Wide&Deep [18] and DeepFM [19] models achieve the simultaneous acquisition of low-order and high-order feature combinations by introducing a hybrid architecture that contains shallow and deep components for learning memory and generalization. However, this hybrid structure adopts implicit bit-wise feature interaction for high-order features, and the model's memory and generalization are not strong. All of the above models utilize DNNs to learn high-order feature interactions. However, deep neural networks model high-order feature interactions in an implicit way, and the model is less interpretable. The DNNs model models feature interactions at the bit level. Although it can be learned in more detail, it will increase the risk of overfitting and lose certain generalization ability.

To sum up, the current CTR prediction model often obtains implicit bit-level high-order feature combinations when acquiring feature combinations, which leads to poor interpretability and generalization of the model, and the model is very easy to overfit. In addition, the performance of the model is also degraded due to the loss of low-order features. Therefore, we combine both bit-wise and vector-wise aspects, and propose a click-through rate prediction model (DTM) with double matrix-level cross-features. The main purpose of this model is to explicitly and automatically capture high-order and low-level interaction features. In terms of vector feature interaction, we use multi-head self-attention mechanism and residual network to achieve the acquisition of vector cross features of the model. This module can explicitly learn feature interaction, and the degree of interaction increases with the depth of the network. In terms of bit interaction features, we use a cross network to achieve it, which can adaptively learn cross features of any order and their weights from the data. In terms of low-order cross-features, we adopt the form of factorization machine to obtain the low-order feature combination of vector-wise level. The DTM model not only can explicitly obtain low-order and high-order feature matrices, but also has excellent interpretability, memory and generalization.

In summary, the contributions of this paper are as follows:

- A novel click-through rate prediction model (DTM) with double matrix-level cross-features is proposed, which can effectively combine high-order feature combinations with explicit vector-level and bit-level features and explicit vector-level low-order feature combinations.
- A multi-head self-attention mechanism and residual network are designed in DTM, which can learn feature interactions explicitly. Experiments show that the model can explicitly obtain high-order feature combinations, which makes the model have good interpretability and improves the accuracy of the model.
- A cross network is also designed in DTM, which can adaptively and explicitly learn arbitrary order crossover features and their weights from the data, and the acquisition of these crossover features is at the bit level, which makes the model have good memory ability.
- Extensive experiments on two datasets, Criteo and Avazu, show that our DTM model significantly outperforms several other state-of-the-art models.

The rest of this paper is organized as follows. In Section 2, we review related works which are relevant to our proposed model, followed by introducing our proposed model in Section 3. In addition, we give an experimental comparison and analysis with other models in Section 4. Finally, we summarize this work in Section 5.

II. RELATED WORK

Traditional CTR prediction models mainly have the following two ways: Logistic regression (LR) only models the linear combination of the original features predicted by CTR [21], [22]. In 2010, Factorization Machines (FM) used factorization techniques to model second-order feature interactions and achieved good performance on large sparse data. Due to the dramatic increase in the size and dimensionality of datasets, these methods have not been able to meet people's needs well [18], [23], [24].

With the success of deep learning in computer vision, speech recognition, and natural language processing, more and more deep learning-based CTR prediction models have been proposed [20], [25], and [26]. In 2016, the University of London proposed the Factorization-supported Neural Network (FNN) model, which is a fully connected neural network that uses FM to pre-train the embedding layer, which can speed up the convergence of the model [16]. In 2016, Shanghai Jiao Tong University proposed the Product-based Neural Network (PNN) model, which introduced a product layer between the embedding layer and the DNN layer to explore the interaction of high-order features. The model can more specifically emphasize the interaction between different features, making it easier for the model to capture the interaction information between features [17]. However, both the FNN model and the PNN model only focus on high-order features, ignoring the effect of low-order feature combinations, resulting in less information available to the model and a decrease in recommendation accuracy. In 2016, Google proposed the Wide&Deep model, which includes wide part and deep part. The wide part models linear low-order feature interaction, while the deep part models nonlinear high-order feature interaction. The model achieves the simultaneous acquisition of low-order and high-order feature matrices [18]. In 2017, Tsinghua University proposed a DeepFM model that uses FM to replace the wide part of Wide&Deep on the basis of Wide&Deep. This model mainly solves the problem that the wide part of the Wide&Deep model cannot automatically

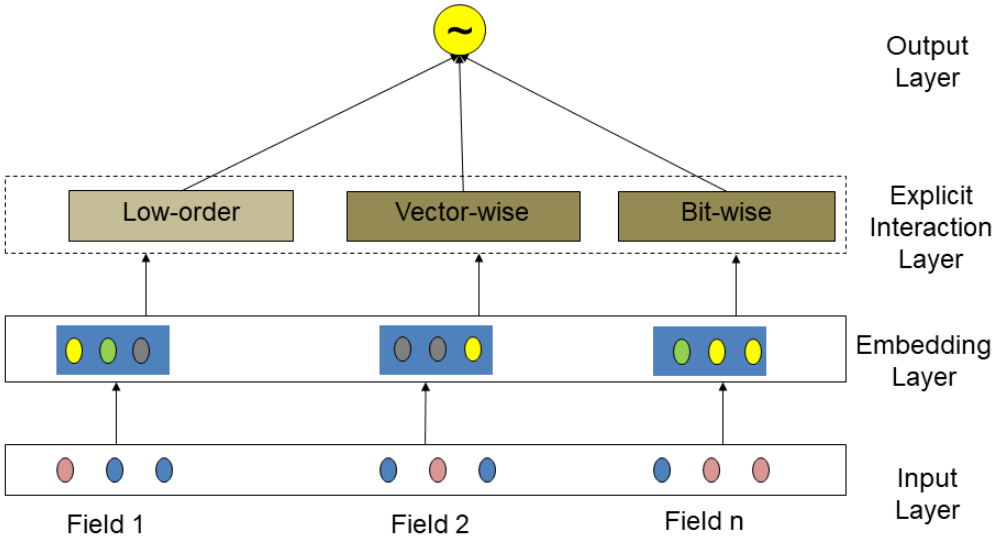


FIGURE 1. The structure diagram of DTM.

obtain feature combinations [19]. Although the Wide&Deep model and DeepFM model realize the simultaneous acquisition of high-order and low-order feature combinations, they both acquire high-order feature combinations in an implicit bit-level way, which leads to poor interpretability and generalization of the model. In 2018, the university of science and technology of China proposed the xDeepFM model, which designed a compressed cross network to acquire high-order feature combination by vector-wise, realizing the simultaneous acquisition of implicit feature combination and explicit feature combination, but the model did not acquire explicit low-order feature combination [28]. In 2018, Peking University proposed the AutoInt [29] model, which combined multi-head self-attention and residual networks to acquire high-order feature combinations with weight information, with strong interpretability. In 2019, Sina Weibo proposed the FiBiNET [30] model, which combined inner product and Hadamard product and introduced an additional parameter matrix W to learn feature crossover, ensuring that the model could still effectively combine features on sparse models. Although the AutoInt and FibiNET models acquire explicit features in vector-wise fashion, the memory of the model will decline without the combination of features at the bit-wise level. In 2020, Shanghai Jiao Tong University proposed the Adaptive Factorization Network (AFN) [31] model, which utilizes logarithmic change layer to learn high-order cross features with weighted information, and can learn arbitrary combination of features from data. However, the model does not acquire explicit low-order features.

In the above model, the combination of bit-level implicit high-order features is simply acquired and the role of low-order features is not taken into account, which leads to the invisibility of the calculation process of the model and the lack of interpretability, which increases the risk of model overfitting and reduces the accuracy and generalization of the model. In view of the above problems, this paper explicitly obtains feature combinations from the perspective of vector-wise, which makes the model have excellent interpretability and generalization. In order to ensure better memorability of the model, explicit bit-wise module is added to obtain high-order feature combinations.

III. THE PROPOSED METHOD

In this section, the DTM model will be shown in detail, as shown in Figure 1. The model mainly includes input layer, embedding layer, explicit interaction layer and output layer. In the explicit interaction layer, the multi-head attention mechanism, residual network and cross network are used to obtain feature combinations. Our goal is to obtain explicit vector-wise low-order feature combinations and high-order feature combinations as well as explicit bit-wise high-order feature combinations.

A. INPUT AND EMBEDDING LAYER

The main purpose of the input layer and the embedding layer is to transform the input features into embedding vectors that meet the requirements of the model. The input layer uses a sparse representation of the original input features, and the embedding layer can embed the sparse features into a low-dimensional, dense real-valued vector. For example, an example input [user_id = 02, gender = male, age = 26, hobbies = comedy and rock] can be converted to high-dimensional sparse features by one-hot encoding:

Sparse high-dimensional data will reduce the recommendation accuracy of the model, so it is necessary to use the embedding layer to map it to the same low-dimensional space. Assuming that there are h feature domains in the input sample, only one bit in each domain takes a value of 1, and the rest take a value of 0, then after the embedding layer, a total of h embedding vectors of length k are obtained. Transpose these vectors and stitch them horizontally to get E_x:

$$E_x = \left[V_1^T, V_2^T, \ldots, V_h^T\right] \tag{1}$$

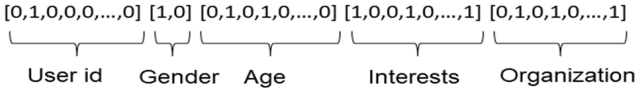


FIGURE 2. One-hot encoded feature vectors.

where h represents the number of feature domains, $V_i \in R^k$ represents the embedding vector of the i-th feature domain, and k represents the embedding dimension.

B. EXPLICIT INTERACTION LAYER

The explicit interaction layer realizes the acquisition of low-level and high-level feature combinations of vector-wise and bit-wise, and is mainly divided into three modules: acquiring explicit vector-wise low-order feature combination modules, acquiring explicit vector-wise high-order feature combination module and acquiring explicit bit-wise high-order feature combination module. We will introduce these three modules in detail below.

1) VECTOR-WISE LOW-ORDER FEATURE COMBINATION

In this paper, a classical Factorization Machine (FM) is adopted to efficiently obtain explicit low-order cross-feature. In addition to obtaining the linear (first-order feature) interactions between features, the FM algorithm can also learn second-order feature interactions by means of vector inner products.

Compared with other low-order feature acquisition methods, the FM algorithm can still capture the second-order feature interactions very effectively when the dataset is sparse. In traditional recommendation algorithms, if you want to obtain the interaction between feature i and feature j, you need to satisfy feature i and feature j in the same sample. The factorization machine solves the problem of needing to exist in the same sample by using the inner product of latent vectors. Therefore, the factorization machine can better and more fully learn the feature interaction information in the data samples. The calculation formula of the factorization machine is shown in Equation 2:

$$y_{low} = w_0 + \sum_{i=1}^{M} w_i E_i + \sum_{i=1}^{M} \sum_{j=i+1}^{M} < V_i, V_j > E_i \cdot E_j \tag{2}$$

where M represents the number of samples, V_i represents the hidden vector corresponding to feature i, V_j represents the hidden vector corresponding to feature j, and w_0 represents the algorithm bias.

2) VECTOR-WISE HIGH-ORDER FEATURE COMBINATION

The vector-vise part adopts the block structure of the encoder in the transformer. The core idea of this module is to use the multi-head self-attention mechanism to explicitly learn high-order feature combinations at the vector level. The structure is shown in figure 3:

In order to calculate the vector-wise feature weight, it is necessary to transform the k × h-dimensional embedding matrix E_x into the h × k-dimensional matrix U_{vec}, as shown

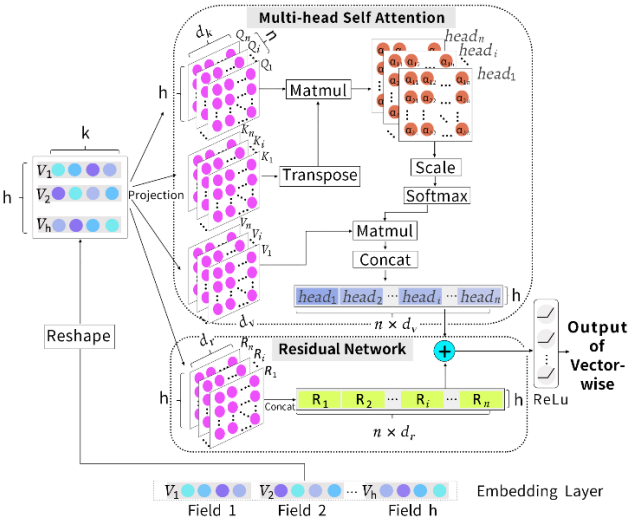


FIGURE 3. The vector-wise part of the DTM model.

in Equation 3. Then the matrix is input to the multi-head self-attention layer to calculate the attention weight of the feature.

$$U_{vec} = reshape\,(E_x) \tag{3}$$

The multi-head attention layer includes four parts: attention space mapping, self-attention calculation, multi-head attention and residual network fusion, and weight calculation. These four parts are described in detail below.

a: ATTENTION SPACE MAPPING

As shown in formulas (4) to (6), this step is to map the matrix U_{vec} to the Q (query), K (key), V (value) spaces through the matrices $W^{Q_i}, W^{K_i} \in R^{k \times d_k}, W^{V_i} \in R^{k \times d_v}$ respectively to obtain a new matrix representation Q_i, K_i, V_i. Among them, i represents the ith attention space, and d_k represents the size of the attention factor, usually $d_k = d_v$.

$$Q_i = U_{vec} W^{Q_i} \tag{4}$$
$$K_i = U_{vec} W^{K_i} \tag{5}$$
$$V_i = U_{vec} W^{V_i} \tag{6}$$

b: SELF-ATTENTION CALCULATION

The correlation of matrices Q_i and K_i is calculated by the inner product and normalized to obtain the attention score, and then the matrix V_i is weighted to obtain the representation $head_i$ of the matrix U_{vec} in a single attention space. Its calculation process is shown in formula (7). It is worth noting that the normalization operation is also a method to capture global information, which can obtain the importance weight of a feature in all features.

$$\begin{aligned} head_i &= Attention\,(Q_i, K_i, V_i) \\ &= softmax\left(\frac{Q_i K_i^T}{\sqrt{d_k}}\right) V_i \end{aligned}$$

$$= \frac{\exp\left(\frac{Q_i K_i^T}{\sqrt{d_k}}\right)}{\sum_{j=1}^{n} \exp\left(\frac{Q_j K_j^T}{\sqrt{d_k}}\right)} \tag{7}$$

where $head_i \in R^{h \times d_v}$ denotes the output of the i-th single head.

c: MULTI-HEAD ATTENTION AND RESIDUAL NETWORK FUSION

As shown in formula (8), by splicing n heads, the representation of the matrix U_{vec} under the weighting of multi-head attention is obtained.

$$MultiHead\ (U_{vec}) = Concat\ (head_1, \ldots, head_n) \tag{8}$$

where n is the number of heads and $MultiHead\ (U_{vec}) \in R^{h \times (d_v \times n)}$ denotes the output of Multi-Head Self-Attention. In order to retain the original input information, the vector U_{vec} is passed through the residual matrix W^{R_i} to obtain the matrix R_i

$$Resdual\ (U_{vec}) = Concat\ (R_1, \ldots\ldots, R_n) \tag{9}$$

$$Where\ R_i = U_{vec} W^{R_i} \tag{10}$$

where n is the number of heads and $MultiHead\ (U_{vec}) \in R^{h \times (d_v \times n)}$ denotes the output of Multi-Head Self-Attention. In order to retain the original input information, the vector U_{vec} is passed through the residual matrix $W^{R_i} \in R^{k \times d_r}$ to obtain the matrix $R_i \in R^{h \times d_r}$. Note that, to keep dimensions consistent, we set $d_r = d_k = d_v$.

d: WEIGHT CALCULATION

Add the two parts in pairs and activate them with the activation function to get the output of the vector-wise part, denoted as O_{vec}.

$$O_{vec} = max\ (0, MultiHead\ (U_{vec}) + Residual\ (U_{vec})) \tag{11}$$

where $O_{vec} \in R^{h \times (d_v \times n)}$.

3) BIT-WISE COMBINATION OF HIGH-ORDER FEATURES

Bit-wise part adopts a new type of cross network, and its key idea is to effectively apply explicit feature crossover. The cross network is composed of cross layers, and the formula of each layer is as follows:

$$x_{l+1} = x_0 x_l^T w_l + b_l + x_l = f\ (x_l, w_l, b_l) + x_l \tag{12}$$

where x_l and x_{l+1} are the outputs of the cross layer at l and $l+1$ respectively, and w_l and b_l are the connection parameters between these two layers. Notice that all the variables in the above equation are column vectors, and w_l is also a column vector, not a matrix. $x_{l+1} = f\ (x_l, w_l, b_l) + x_l$ means that the output of each layer is the output of the previous layer plus the function f, and f is the residual in fitting the output of this layer and the output of the previous layer. Figure 4 shows a visualization of a simple cross network.

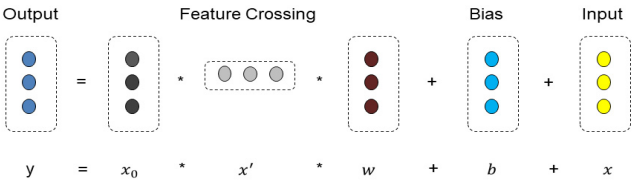


FIGURE 4. Cross network computing process.

The special structure of cross network makes the order of feature crossover increase with the increase of the number of networks. With respect to the input x_0, the order of feature crossover of an l-layer crossover network is $l + 1$.

Assuming that there are L_c layer cross networks in total, the initial input of x_0 dimension is d, then the number of parameters of the entire cross network is:

$$d \times L_c \times 2 \tag{13}$$

Since the w and b of each layer are d-dimensional, it can be seen from the above equation that complexity is a linear function of the input dimension d. Therefore, compared with the fully connected neural network, the complexity introduced by the cross network is negligible, which ensures that the complexity of the high-order bit-wise feature cross module is at the same level as that of the fully connected network. The reason why the cross network can efficiently learn the combined features is because the rank of $x_0 * x_T$ is 1, so that the model can get all the feature combination items without calculating and storing the entire matrix.

However, it is precisely because of the few parameters of cross network that its expression ability is limited. In order to learn highly nonlinear combination features, modules are introduced into the fully connected network in parallel.

Fully connected network is a fully connected feedforward neural network, and each deep layer has the following formula:

$$h_{l+1} = f\ (W_l h_l + b_l) \tag{14}$$

where h_l and h_{l+1} are the l layer and $l + 1$ hidden layer respectively, W_l, b_l is the parameter of the deep layer l, $f(\cdot)$ is ReLU function.

Next, the complexity can be estimated by calculating the number of parameters. For simplicity, assume that all layers are of equal size, the input x_0 dimension is d, there are L_d layers of neural network, and the number of neurons in each layer is m. Then the total parameters or complexity is:

$$d \times m + m + \left(m^2 + m\right) \times (L_d - 1) \tag{15}$$

Then, concatenate the output of the interaction layer and the output of the deep network.

$$O_{bit} = x_{L_1}^T \oplus h_{L_2}^T \tag{16}$$

where $x_{L_1}^T \in R^d$, $h_{L_2}^T \in R^m$ represent the output of the interaction layer and the deep network.

Finally, vector level feature combination and bit level feature combination are added in bitwise.

C. OUTPUT LAYER

The core idea of the output layer is to combine the results of the above modules and output the final prediction results. The output layer adopts the way of fully connected neural network to output, and the calculation formula is as follows:

$$\hat{y} = \sigma\left(w_{low}^T y_{low} + w_{high}^T y_{high} + \mathrm{b}\right) = \frac{1}{1 + e^{-(w_{low}^T y_{low} + w_{high}^T y_{high} + \mathrm{b})}} \quad (17)$$

where σ is the sigmoid function, y_{low}, y_{high} represents the output of the low-order feature combination module and the high-order feature combination module, w_{low}^T, w_{high}^T represents the parameter matrix.

The loss function adopts the cross entropy loss function in CTR, and the calculation formula is as follows:

$$loss = -\frac{1}{N}\sum_{j=1}^{N}\left(y_j \log\left(\hat{y}_j\right) + \left(1 - y_j\right)\log\left(1 - \hat{y}_j\right)\right) \quad (18)$$

where N is the number of total samples, and y_j is the true value of the *jth* sample, and $\hat{y}_j$ is the predicted value of the *jth* sample by the model.

IV. EXPERIMENT

In this section, we conduct sufficient experiments to verify the following three questions:

(Q1) Does the DTM model proposed in this paper have better performance compared with other latest models?

(Q2) For recommendation models, is it necessary to combine vector-wise and bit-wise?

(Q3) How does DTM model performance vary with hyper-parameter settings?

We first introduce some basic experimental settings in detail, and then answer the above questions.

A. EXPERIMENTAL SETUP

1) DATASET

Criteo and Avazu datasets are used in this paper. The statistical data of these datasets are summarized in table 1. Criteo dataset is a benchmark dataset for CTR prediction. It has a record of 45 million users clicking on displayed ads. It contains 26 classification feature fields and 13 numerical feature fields. The Avazu dataset contains the user's mobile behavior, including whether the displayed mobile advertisement is clicked by the user. It has 23 feature fields, including user / device features and advertising attribute features.

TABLE 1. Dataset statistics.

Dataset	Instances	Fields	Feature
Criteo	45,840,617	39	998,960
Avazu	40,428,967	23	1,544,488

2) EVALUATION METRICS

We uses two popular metrics to evaluate the performance of all models.

a: AUC

The area under ROC curve is usually used to judge the performance of binary prediction model. The lower limit of AUC is 0 and the upper limit is 1. The higher the value, the better the performance of the model.

b: LogLoss

LogLoss is a widely used measurement method in binary classification problems. It is used to measure the distance between two distributions, that is, the gap between the true value and the predicted value. The lower bound of logarithmic loss is 0, which means that the two distributions match exactly. The smaller the value, the better the performance.

It is worth noting that a slightly higher AUC or lower LogLoss at the 0.001 level is significant for the CTR prediction task, which has also been pointed out in existing work.

3) BASELINE METHODS

a: FM [12]

FM utilizes factorization techniques to model second-order feature interactions.

b: Wide&Deep [18]

Wide&Deep model uses wide part to realize the memorization of model, and deep part to realize the generalization of model.

c: DeepFM [19]

DeepFM uses deep layer to obtain high order crossover features, FM obtains low order crossover features, and achieves high and low order crossover features at the same time.

d: AFM [27]

AFM is one of the most advanced models for capturing second-order feature interactions. It extends FM using the attentional mechanism to distinguish the different importance of second-order combinatorial features.

e: AutoInt [29]

AutoInt combines high-order features through the attention mechanism, evaluates different combinations using the multi-head mechanism, and maps features to different subspaces. By superimposing multiple interaction layers, the combination characteristics of different orders can be simulated.

f: xDeepFM [28]

xDeepFM is an improved version of Wide & Deep, on which CIN layer is added to explicitly construct finite order feature combinations.

g: FiBiNET [30]

FiBiNET model is a CTR prediction model combining feature importance and bilinear feature interaction.

h: AFN [31]

AFN is a new method of adaptive eigennorm without parameters is proposed, and it is proved that the significant

migration performance can be achieved by gradually adapting the eigennorm of two domains to a wide range of values.

B. PERFORMANCE COMPARISON (Q1)

DTM integrates multi-head self-attention mechanism and cross network and DNN into an end-to-end model. Although self-attention mechanism, cross network and DNN cover two different attributes in learning feature interaction, it is necessary to verify whether combining vector crossover feature and bit crossover feature is indeed necessary and effective. Here, we compare several comparison models that are not limited to a single model, and the results are shown in Table 2. It was observed that FM was much worse than all other models, which indicated that although FM alone could deal with vector crossover feature, the results of the model were relatively poor due to its simple structure and lack of bit crossover feature. The performance of Wide&Deep, AFM, AutoInt, FiBiNET, AFN, DeepFM and xDeepFM is obviously better than FM, which directly reflects that the high-order cross feature combination can improve the model performance in the recommender system. Compared with the contrast model, the AUC value of DTM model increased by 4% and 3% on average, and the LOSS value decreased by 3.5% and 2.8% on average, respectively.

TABLE 2. Performance comparison of different models.

Model	Criteo		Avazu	
	AUC	LogLoss	AUC	LogLoss
FM	0.6869	0.5286	0.6437	0.5221
Wide&deep	0.7066	0.4984	0.7324	0.4217
DeepFM	0.7071	0.4907	0.7396	0.4185
AFM	0.7120	0.4896	0.7467	0.4112
AutoInt	0.7360	0.4849	0.7401	0.4121
xDeepFM	0.7403	0.4846	0.7436	0.4118
FiBiNET	0.7434	0.4621	0.7499	0.4108
AFN	0.7465	0.4608	0.7502	0.4087
DTM	**0.7625**	**0.4488**	**0.7607**	**0.3988**

Compared with the latest models such as AFN, FibiNET and AutoInt, DTM still has better AUC and LOSS performance, which verifies the superiority of our model. Compared with other models, DTM model has better performance, indicating that the combination of explicit vector-wise feature matrix and explicit bit-wise feature matrix can improve the performance of the recommended model, and the model has better interpretability, generalization and memorability.

C. ABLATION STUDY (Q2)

The DTM model innovatively obtains feature combinations by crossing vector-wise features and complementing the main bit-wise feature combinations, which increases the generalization and memorability of the model and realizes an end-to-end model. So is it really necessary and effective to combine them for joint forecasting? In our DTM model, which is the most important component? In order to verify and understand the DTM model, validity analysis experiments are carried out on DTM. Table 3 shows the performance of DTM model under different components, and verifies the effectiveness of combining vector-wise feature combination and bit-wise feature combination.

TABLE 3. Comparison of performance between different variants of DTM.

Model	Criteo		Avazu	
	AUC	LogLoss	AUC	LogLoss
FM	0.6869	0.5286	0.6437	0.5221
DeepFM	0.7071	0.4907	0.7396	0.4185
No vector	0.7508	0.4575	0.7560	0.4018
No FM	0.7537	0.4560	0.7569	0.4015
No cross net	0.7587	0.4548	0.7577	0.4012
DTM	**0.7625**	**0.4488**	**0.7607**	**0.3988**

(1) First, by comparing the performance of DeepFM model and FM model on the two datasets, it can be seen that the DeepFM model with high-order feature combination has better performance. Therefore, introducing high-order cross feature combination into the recommendation model is effective and can improve the performance of the model.

(2) Second, by comparing the performance of various variants of DTM model, it can be seen that the performance of the model considering bit-wise feature combination and vector-wise feature combination at the same time has better performance than that of any model alone. Therefore, it is necessary and effective to jointly predict bit-wise and vector-wise, so that the model can have better memory and generalization.

(3) Finally, among the three variants of the DTM model, the vector-wise features have better performance than the other two variants, which proves that the vector-wise feature vectors play a more important role, mainly because the vector module not only can arbitrary-order feature combinations be obtained, but also weight information can be assigned to these combinations, which further improves the generalization of the model.

D. PARAMETER STUDY (Q3)

In this subsection, we will perform some hyperparameter studies on our model. Specifically, we study the following hyperparameters: (1) the Dropout value; (2) the number of hidden layers; (3) the activation function. During the experiment, only one hyperparameter is changed at a time, and the other hyperparameters remain unchanged.

1) DROPOUT

The main function of dropout is to prevent parameters from relying too much on training data and increase the generalization ability of parameters to datasets. Some neurons are ignored at random to reduce overfitting. The size of the dropout value determines how many neurons need to sleep. If the Dropout value is too small, the effect of reducing overfitting and increasing generalization cannot be achieved. If the dropout value is too high, important information will be lost, making the recommendation inaccurate. As shown in figure 5, on the Criteo and Avazu datasets, the AUC and Loss of the DTM model increase first and then decrease. At dropout 0.4, the maximum AUC and minimum Loss values

FIGURE 5. Performance of DTM models at different dropout values.

FIGURE 6. Performance of DTM model under different network layers.

are achieved, and the model has the best performance at this moment.

2) NUMBER OF HIDDEN LAYERS

Figure 6 shows the effect of the number of hidden layers in the DTM model. For Criteo and Avazu datasets, the performance of the DTM increased with the increase of the depth of the network. However, when the network depth is set to greater than 4, the performance of the model begins to degrade. This is because with the increase of the number of model layers, the following problems will occur: the model structure becomes more complex, which will lead to the phenomenon of overfitting; extremely high feature combinations cannot effectively improve the model performance.

3) ACTIVATION FUNCTION

The relationship between the features of the recommendation model is often nonlinear, and the activation function can increase the nonlinear expression ability of the recommendation model and improve the accuracy of the recommendation results. The DTM model was compared and analyzed on three main activation functions: sigmoid, tanh and ReLU functions. By comparing the performance of DTM under different activation functions, the following conclusions are drawn: ReLU function is more suitable for DTM model and has better model recommendation accuracy. Considering the different performance of AUC and LOSS under the three activation functions of DTM model, ReLU function is the best, followed by tanh activation function, and sigmoid activation function is the worst. Therefore, ReLU function is selected as the activation function of DTM model in this paper. ReLU activation function has a good performance, possible reasons include: (1) Sparse activation. Nodes with negative outputs are not activated; (2) Gradient propagates efficiently. No gradient disappearance problem or explosion effect; (3) High computational efficiency. Only comparisons, additions and multiplications are done. The performance of DTM model under different activation functions is shown in Figure 7.

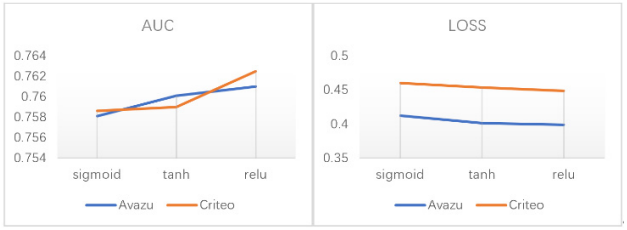


FIGURE 7. Performance of DTM model under different activation functions.

V. CONCLUSION

We propose a novel click-through rate prediction model (DTM) with dual matrix-level feature combinations. Its purpose is to automatically obtain high-order and low-order feature combinations in a matrix-level manner according to the input features. The main advantage of DTM is that it can realize not only explicit feature combination acquisition at the bit level, but also explicit feature acquisition at the matrix level, which increases the interpretability, generalization, and memory of the model. Extensive experiments are carried out on two real CTR datasets to verify that DTM has a good model representation and prediction ability. The two most important evaluation indicators for CTR estimation, AUC and LOSS, are analyzed and compared with other latest deep learning recommendation models, which proves that the DTM model has better recommendation accuracy and is better than other latest recommendation models.

ACKNOWLEDGMENT

The authors would like to thank all the anonymous reviewers for their insightful comments. They would like to thank colleagues and the anonymous reviewers who have provided valuable feedback to help improve the article.

REFERENCES

[1] P. T. Nguyen, J. D. Rocco, R. Rubei, C. Di Sipio, and D. D. Ruscio, "DeepLib: Machine translation techniques to recommend upgrades for third-party libraries," *Expert Syst. Appl.*, vol. 202, Sep. 2022, Art. no. 117267.

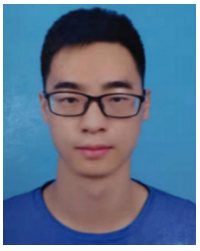
[2] A. Da'u, N. Salim, I. Rabiu, and A. Osman, "Recommendation system exploiting aspect-based opinion mining with deep learning method," *Inf. Sci.*, vol. 512, pp. 1279–1292, Feb. 2020.

[3] T. B. Sarwar, N. M. Noor, M. S. U. Miah, M. Rashid, F. A. Farid, and M. N. Husen, "Recommending research articles: A multi-level chronological learning-based approach using unsupervised keyphrase extraction and lexical similarity calculation," *IEEE Access*, vol. 9, pp. 160797–160811, 2021.
[4] L. Sun, X. Liu, Y. Liu, T. Wang, L. Guo, X. Zheng, and Y. Luo, "A novel deep recommend model based on rating matrix and item attributes," *J. Intell. Inf. Syst.*, vol. 57, no. 2, pp. 295–319, Oct. 2021.
[5] P. G. Roetzel, "Information overload in the information age: A review of the literature from business administration, business psychology, and related disciplines with a bibliometric approach and framework development," *Bus. Res.*, vol. 12, no. 2, pp. 479–522, Dec. 2019.
[6] V. Vijayakumar, S. Vairavasundaram, R. Logesh, and A. Sivapathi, "Effective knowledge based recommender system for tailored multiple point of interest recommendation," *Int. J. Web Portals*, vol. 11, no. 1, pp. 1–18, Jan. 2019.
[7] Z. Huang, X. Lin, H. Liu, B. Zhang, Y. Chen, and Y. Tang, "Deep representation learning for location-based recommendation," *IEEE Trans. Computat. Social Syst.*, vol. 7, no. 3, pp. 648–658, Jun. 2020, doi: 10.1109/TCSS.2020.2974534.
[8] K. Ren, W. Zhang, Y. Rong, H. Zhang, Y. Yu, and J. Wang, "User response learning for directly optimizing campaign performance in display advertising," in *Proc. 25th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2016, pp. 679–688.
[9] Q. Liu, F. Yu, S. Wu, and L. Wang, "A convolutional click prediction model," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2015, pp. 1743–1746.
[10] X. He, H. Zhang, M.-Y. Kan, and T.-S. Chua, "Fast matrix factorization for online recommendation with implicit feedback," in *Proc. 39th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2016, pp. 549–558.
[11] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian personalized ranking from implicit feedback," in *Proc. UAI*, 2009, pp. 452–461.
[12] S. Rendle, "Factorization machines," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Dec. 2010, pp. 995–1000.
[13] A. Goel, C. Tung, Y.-H. Lu, and G. K. Thiruvathukal, "A survey of methods for low-power deep learning and computer vision," in *Proc. IEEE 6th World Forum Internet Things (WF-IoT)*, Jun. 2020, pp. 1–6, doi: 10.1109/WF-IoT48130.2020.9221198.
[14] S. Bhatt, A. Jain, and A. Dev, "Continuous speech recognition technologies—A review," in *Recent Developments in Acoustics*, 2021, pp. 85–94.
[15] I. Lauriola, A. Lavelli, and F. Aiolli, "An introduction to deep learning in natural language processing: Models, techniques, and tools," *Neurocomputing*, vol. 470, pp. 443–456, Jan. 2022.
[16] W. Zhang, T. Du, and J. Wang, "Deep learning over multi-field categorical data," in *Proc. Eur. Conf. Inf. Retr.*, 2016, pp. 45–57.
[17] Y. Qu, "Product-based neural networks for user response prediction," in *Proc. IEEE 16th Int. Conf. Data Mining (ICDM)*. Barcelona, Spain, Dec. 2016, pp. 1149–1154.
[18] H. T. Cheng, L. Koc, and J. Harmsen, "Wide & deep learning for recommender systems," in *Proc. 1st Workshop Deep Learn. Recommender Syst.*, 2016, pp. 7–10.
[19] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "DeepFM: A factorization-machine based neural network for CTR prediction," 2017, *arXiv:1703.04247*.
[20] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for Youtube recommendations," in *Proc. 10th ACM Conf. Recommender Syst.*, Sep. 2016, pp. 191–198.
[21] M. Jiang, P. Cui, X. Chen, F. Wang, W. Zhu, and S. Yang, "Social recommendation with cross-domain transferable knowledge," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 11, pp. 3084–3097, Nov. 2015, doi: 10.1109/TKDE.2015.2432811.
[22] H. Liu, J. Wen, L. Jing, and J. Yu, "Deep generative ranking for personalized recommendation," in *Proc. 13th ACM Conf. Recommender Syst.*, Sep. 2019, pp. 34–42.
[23] K.-Y. Kwahk and B. Kim, "Effects of social media on consumers' purchase decisions: Evidence from taobao," *Service Bus.*, vol. 11, no. 4, pp. 803–829, Dec. 2017.
[24] M. Hibat-Allah, M. Ganahl, L. E. Hayward, R. G. Melko, and J. Carrasquilla, "Recurrent neural network wave functions," *Phys. Rev. Res.*, vol. 2, no. 2, Jun. 2020, Art. no. 023358.
[25] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. Chua, "Neural collaborative filtering," in *Proc. WWW*, 2017, pp. 173–182.
[26] M. Naumov, "Deep learning recommendation model for personalization and recommendation systems," 2019, *arXiv:1906.00091*.
[27] 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. Artif. Intell.*, Aug. 2017, pp. 3119–3125.
[28] 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. Knowl. Discovery Data Mining*, Jul. 2018, pp. 1754–1763.
[29] W. Song, C. Shi, Z. Xiao, Z. Duan, Y. Xu, M. Zhang, and J. Tang, "AutoInt: Automatic feature interaction learning via self-attentive neural networks," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, Nov. 2019, pp. 1161–1170.
[30] T. Huang, Z. Zhang, and J. Zhang, "FiBiNET: Combining feature importance and bilinear feature interaction for click-through rate prediction," in *Proc. 13th ACM Conf. Recommender Syst.*, Sep. 2019, pp. 169–177.
[31] W. Cheng, Y. Shen, and L. Huang, "Adaptive factorization network: Learning adaptive-order feature interactions," in *Proc. AAAI*, 2020, pp. 3609–3616.

WEI ZHANG received the Ph.D. degree from the Huazhong University of Science and Technology. He is currently an Associate Professor with the Faculty of Artificial Intelligence in Education, Central China Normal University. He has published more than 44 articles in the academic journals, including 36 papers indexed by SSCI, SCI, EI, CPCI, and CSSCI. His research interests include computer applications, big data analysis, data mining, and application of information technology in education.

YAHUI HAN is currently pursuing the graduate degree with the Department of Artificial Intelligence Education, Central China Normal University. His research interests include machine learning, deep learning, and recommendation systems.

ZHAOBIN KANG received the bachelor's degree from Jishou University, China, in 2020. He is currently pursuing the master's degree in computer science and technology with Central China Normal University. His research interests include recommendation systems, data mining, and deep learning.



KAIYUAN QU is currently pursuing the master's degree in computer science and technology with Central China Normal University. His research interests include data mining, deep learning, and knowledge tracing.