Hindawi
Mobile Information Systems
Volume 2021, Article ID 5575249, 10 pages
https://doi.org/10.1155/2021/5575249

Research Article

Deep Field-Aware Interaction Machine for Click-Through Rate Prediction

Gaofeng Qi[1] **and Ping Li**[1,2]

[1]*School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha 410014, China*
[2]*Hunan Provincial Key Laboratory of Intelligent Processing of Big Data on Transportation, Changsha University of Science and Technology, Changsha 410014, China*

Correspondence should be addressed to Ping Li; lping9188@163.com

Received 30 January 2021; Revised 22 March 2021; Accepted 12 April 2021; Published 21 April 2021

Academic Editor: Salvatore Carta

Modeling feature interactions is of crucial importance to predict click-through rate (CTR) in industrial recommender systems. Because of great performance and efficiency, the factorization machine (FM) has been a popular approach to learn feature interaction. Recently, several variants of FM are proposed to improve its performance, and they have proven the field information to play an important role. However, feature-length in a field is usually small; we observe that when there are multiple nonzero features within a field, the interaction between fields is not enough to represent the feature interaction between different fields due to the problem of short feature-length. In this work, we propose a novel neural CTR model named DeepFIM by introducing Field-aware Interaction Machine (FIM), which provides a layered structure form to describe intrafield and interfield feature interaction, to solve the short-expression problem caused by the short feature-length in the field. Experiments show that our model achieves comparable and even materially better results than the state-of-the-art methods.

1. Introduction

The click-through rate (CTR) prediction is crucial for many online services, including personalized recommendation and advertising; its main task is to predict the probability of users clicking on items or ads. For example, when the user retrieves some keywords purchased by the advertiser, the search engine will display the corresponding advertisement; then, the advertiser will pay for each click of the users. Therefore, how to predict CTR accurately and efficiently is a core issue to improve advertising revenue and corporate profits.

The data in the CTR prediction task is usually multifield categorical data, that is, Male, Female, Nike, Adidas; each feature belongs to a unique field. For example, feature "Male" belongs to the field "gender", and feature "Adidas" is in the field "ads." Prior efforts have shown the importance of modeling interactions in CTR prediction [1, 2], and an example of informative feature interactions such as "Male" and "Basketball," "Female" and "Cosmetic" usually have a positive impact on users' clicks.

Among plenty of prediction algorithms, the factorization machine (FM) [1] is one of the most popular ones due to its excellent performance, and many variants have been derived. The key of FM is the vectorization of features and the use of an inner product of two vectors to model the effect of pairwise feature interactions. FFM [3] and FwFM [4] extended the ideas of factorization machines by additionally leveraging the field information and explicitly model field-aware feature interactions. The IFM [5] learns feature interaction through the similarity of feature interaction vectors and corresponding field interaction prototypes, but each feature still only learns a unique implicit vector. FAT-DeepFFM [6] adopts the field expression of features and estimates the importance of different feature interactions between fields; they argue that it is also important to estimate the importance of features before they are crossed, but the existing models [7–10] are after the feature is crossed.

Recently, ONN (operation-aware neural network) [11] combines FFM and MLP to learn the different embedding of different operations (such as convolution operations and product operations) informing interactions. Moreover, AUFM [12] and FNFM [13] propose to combine FFM with SA (stacked autoencoder) and DNNs (deep neural networks) to enhance the ability of high-order feature expression. CFM (convolutional factorization machine) [14] uses convolutional neural networks (CNNs) rather than DNNs to improve high-order and nonlinear expression between fields. Note that usually, the size of the embedding vector in the field is smaller (e.g., $k_{\mathrm{FFM}} \ll k_{\mathrm{FM}}$, k is the embedding size of features) because each embedding vector in the field only needs to learn the effect with a specific field.

Although the above models have achieved promising results, the challenge in real-world online advertising or recommender systems is that there may be multiple dense nonzero features in a field of an instance sample. In a model such as FAT-DeepFFM [6], each column vector of the field corresponds to the influence of other fields, and one column corresponds to itself. Then, the expression ability of feature interaction in the field may be limited by short feature-length (e.g., $k_{\mathrm{FFM}} \ll k_{\mathrm{FM}}$). This means that the short feature-length makes it easy to produce "oversimilarity" between feature interactions that should not be so similar, thus reducing model performance.

To solve the above problems, we propose a new CTR model DeepFIM to solve the "short-expression" problem and better capture the multidense feature interactions. Specifically, we propose a new feature interaction expression based on field identifier, which is called "hierarchical-structure expression." On this basis, we design a cross-interaction layer to identify the intrafield and interfield interaction and use an attention mechanism to distinguish the importance of different features. Particularly, we employ two embedding modes for field-aware embedding queries; this will be more conducive to learning interfield and intrafield characteristics. Inspired by [2], we introduce a dynamically Bi pool layer to enhance the acquisition of high-order features, which can maximize the retention of information and will be beneficial to the subsequent learning of deep neural networks. We conduct extensive experiments on the public dataset to confirm our observations; the results show that our model has a stronger expression ability than the current deep learning models such as NFM [2], DeepFM [15], and DCN [16]. The code and data are publicly accessible at https://github.com/qigaofeng/deepfim for researchers to validate and conduct further research.

To sum up, our contributions are as follows:

(i) We propose a new model named DeepFIM to exploit the field information of features and better learn the interaction by addressing the problem of short-expression.

(ii) We highlight the importance of feature-length for interactions in the field. To the best of our knowledge, this is the first framework in incorporating different embedding modes into the hybrid component learning feature interactions.

(iii) Experiments show the superiority of our proposed DeepFIM over the existing deep learning approaches on the public benchmark dataset.

The rest of this paper is organized as follows: in Section 2, we present a brief review of each of the important historical works of our predecessors. In Section 3, we first describe problem formulation and preliminaries in our model. In Section 4, we introduce our proposed DeepFIM model in detail. In Section 5, we design the experiment and verify the prediction effect of the method by comparison experiment. We also analyze the experimental results in this section. Section 6 concludes the paper and lists possible future work.

2. Related Work

In this section, we discuss existing CTR prediction models. Generally speaking, they can be roughly grouped into two categories, namely, shallow models and deep models.

First, shallow models include Field-aware Factorization Machine (FFM) [3], Field-weighted Factorization Machine (FwFM) [4], and Attention Factorization Machine (AFM) [17]. Juan et al. proposed the FFM [3] model in 2016. FFM associates multiple embedding vectors for each feature. They argued that different implicit vectors represent different fields, and different feature interaction effects are modeled from different fields. This greatly improves prediction performance. The Field-weighted Factorization Machine [4] proposes a more efficient way to model the interaction between different fields in memory. The results show that FwFM can achieve the equivalent prediction performance as FFM with fewer parameters. Besides, Xiao et al. noted the importance of FM's inability to distinguish between features. Therefore, they propose a model called Attentional Factorization Machine (AFM) [17], which uses attention mechanisms to estimate the weight of feature interaction, to mine the important features. However, these methods are only one of the shallow-layer models; the main limitation is that they cannot extract high-order features well.

Secondly, deep models include NFM (Neural Factorization Machines) [2], Wide and Deep [18], DeepFM [15], DCN (Deep and Cross Network) [16], and XDeepFM (eXtreme Deep Factorization Machine) [19]. He et al. [2] pointed out that the inner product of the embedded vector is a linear combination of the implicit vector, and they developed a new model to deepen the FM under the neural network framework to learn high-order and nonlinear features. What is more, Wide and Deep [18] proposed a fusion network of a linear model and DNNs model to learn both low-order and high-order combined features. Unfortunately, features in linear models still require feature engineering, and it is not easy to adapt to new data sets. DeepFM [15] handles this issue by modeling low-order feature interaction through FM components rather than linear models. However, treating every feature interaction fairly may produce noise and degrade the performance. Since then, a variety of embedded-based neural networks had been proposed to learn high-order interactions. Shan et al. proposed DCN [16] to automatically learning high-

order combined features. Moreover, xDeepFM [19] proposes a compress interaction network (CIN) and combines with DNNs to automatically learn high-order feature interaction in both explicit and implicit manner. Recently, Yang et al. had proposed an ONN (operation-aware neural network) [11], which uses FFM [3] to learn the different embedding of different operations (such as convolution operations and product operations) informing interactions. And another relevant work includes [20, 21].

3. Problem Formulation

We begin by describing the CTR prediction problem and introducing some basic notations. Suppose we have a data set for training that consists of m instances $\mathcal{S} = \{(x, y)\}$, where x is n-fields data record usually involving a pair of user and item and y indicates whether the item is clicked ($y = 1$ indicates that a click has occurred and $y = 0$ otherwise). Normally, x can include category fields (e.g., gender) and continuous fields (e.g., age). Each category field is represented as a one-hot encoded vector, and each continuous field is represented as a value itself, or as a discrete one-hot encoded vector. Our task is to build a prediction model $\widetilde{y} = \text{CTR\_model}(x)$ to estimate the probability that the user has a potential interest in the item with which he/she has not engaged before. Additionally, throughout the paper, we use the italic and bold uppercase letters (e.g., $\mathbf{Q}$) to denote a matrix, bold lowercase letters to denote a vector (e.g., $\mathbf{x}$), and italic uppercase letters to denote a tensor (e.g., C). Scalar is represented by lowercase letters (e.g., y).

3.1. Preliminaries. FM (factorization machine) [1] is a widely used model that uses the dot product of two embedding vectors to model the effect of pairwise feature interactions:

$$y_{\text{FM}}(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} w_{ij} x_i x_j, \tag{1}$$

where w_0 is the global bias, w_i is the weight of the i-th feature, and x_i is the feature value of the i-th feature. Besides, FM captures pairwise (order-2) feature interactions effectively as $w_{ij} = \langle \mathbf{v}_i, \mathbf{v}_j \rangle$, where $<, >$ denotes the inner product of two vectors; therefore, the parameters for unobserved cross features can also be estimated.

3.1.1. Field-Aware Factorization Machine. FFM (Field-aware Factorization Machine) [3] extended the ideas of Factorization Machines by additionally leveraging the field information and won two competitions hosted by Criteo and Avazu:

$$y_{\text{FFM}}(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} < \mathbf{v}_{i,f_j}, \mathbf{v}_{j,f_i} > x_i x_j, \tag{2}$$

where $\mathbf{v}_{i,f_j}$ is the embedding vector of the i-th feature for field j. The inclusion of field importance on feature interactions can enhance prediction performance. Note that the size of the embedding vector in FFM is usually smaller than that in FM because each embedding vector in FFM only needs to learn the effect of a specific field ($k_{\text{FFM}} \ll k_{\text{FM}}$).

3.2. Our Approach. In this section, we present the details and the training procedure of the proposed model. Figure 1 illustrates the neural network architecture of DeepFIM. The input layer is similar to FFM [3], which converts the features of users and advertisements to feature vectors that are spliced from the input features into different fields. Then, the field-aware embedding layer embeds each nonzero feature into multiple dense vectors. Next, the FIM layer is the key component for processing feature interaction and the DNN (f_{BI}) layer is used to better capture the nonlinear and high-order features. At last, the output layer gives the final prediction of the target.

3.3. Field-Aware Embedding Layer. First, we propose a new embedding layer, which contains two embedding modes: one is matrix embedding mode and the other is the vector embedding mode. Specifically, the matrix embedding mode uses a field embedding method similar to the FFM [3] model to convert features into low-dimensional dense representations. Additionally, the vector embedding mode adopts a method similar to the FM [1] model. Formally, let $\mathbf{V}_i$ denote the embedding matrix of i-th feature, where each row $\mathbf{v}_i$ is the embedding vector for a field; let $\mathbf{e}_{f_i}$ denote the embedding vector of i-th field. Then, we have a set of field-aware embedding vector $\mathcal{P}_{\mathbf{X}} = \{\mathbf{v}_1 x_1, \mathbf{v}_2 x_2, \ldots, \mathbf{v}_n x_n\} \cup \{\mathbf{e}_{f_1} x_1, \mathbf{e}_{f_2} x_2, \ldots, \mathbf{e}_{f_n} x_n\}$, and each feature x_i also belongs to its field i. Moreover, we denote $\mathcal{T}_{\mathbf{X}} = \{\mathbf{e}_{f_1} x_1, \mathbf{e}_{f_2} x_2, \ldots, \mathbf{e}_{f_n} x_n\}$ as the input of the high-order interaction layer.

4. FIM Component

After the field-aware embedding layer, we feed the embedding vectors $\mathcal{P}_{\mathbf{X}}$ into the FIM layer, which is used to capture low-order feature interactions. Figure 2 shows the structure of the FIM component; for clarity purposes, we omit the linear regression part in the figure. Formally, let the nonzero feature set in the feature vector $\mathbf{x}$ be $\mathcal{X}$. In the interaction part of FIM layer, we can represent the output as a set of vectors:

$$f_{\text{PI}}(\mathcal{P}_{\mathbf{X}}) = \left\{ \left(F_{i,j} \odot M_{i,j} \right) x_i x_j \right\}_{i,j \epsilon R_x}, \tag{3}$$

where F, M is a k-dimensional tensor and $\odot$ denotes the Hadamard product. $R_x = \{(i, j)\}_{i,j \epsilon \mathcal{X}, j > i}$. We compress $f_{\text{PI}}(\mathcal{P}_{\mathbf{X}})$ with a sum pooling. Then, we use the full-connection layer to establish it and get the prediction score:

$$\overline{y} = \mathbf{p}^T \sum_{(i,j) \epsilon R_x} \left(\langle \underbrace{F_{i,j}}_{\text{field level}} \rangle \odot \langle \underbrace{M_{i,j}}_{\text{feature level}} \rangle \right) x_i x_j + b, \tag{4}$$

where $\mathbf{p} \epsilon \mathbb{R}^k$ denotes the weights and $b \epsilon \mathbb{R}$ denotes the bias for the prediction layer. $F_{i,j}$ represents field-aware interaction vector between field i and field j; $M_{i,j}$ denotes the

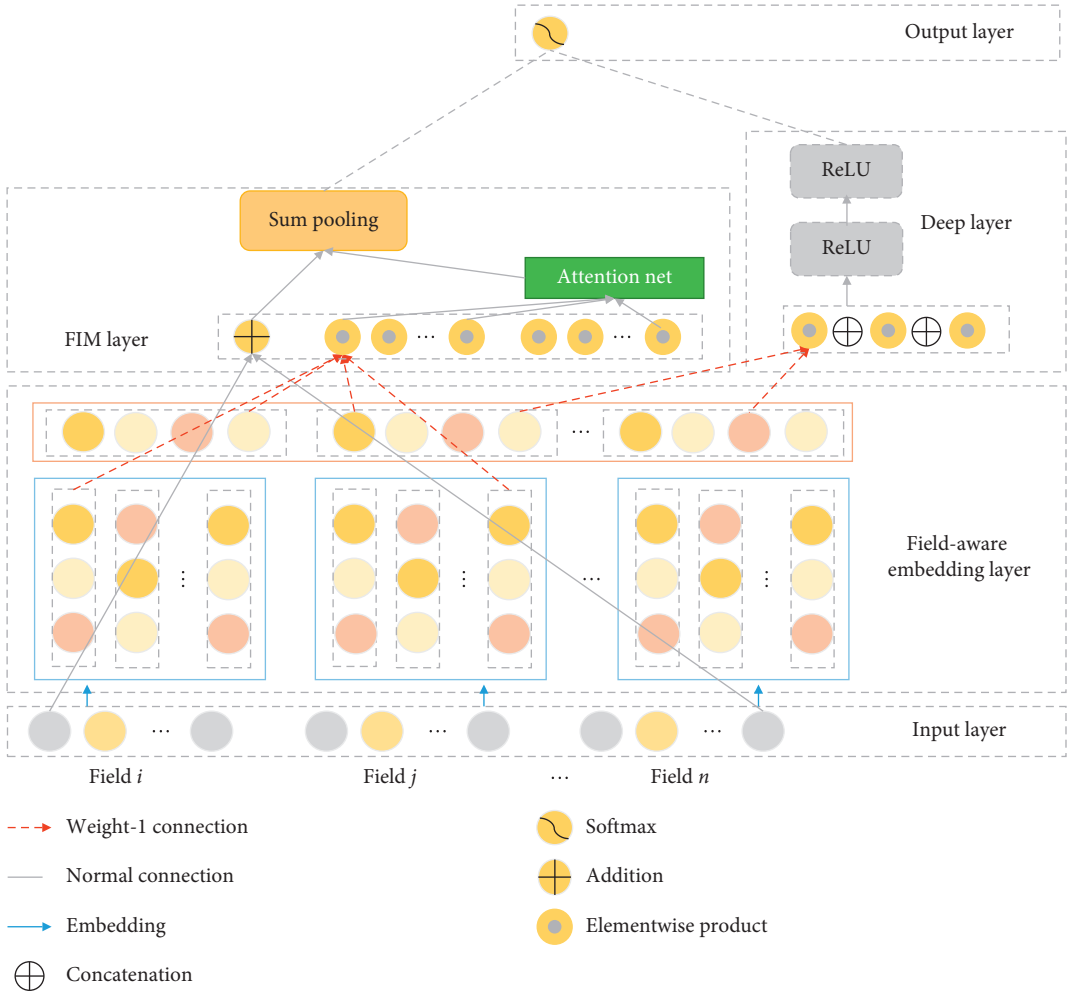

Figure 1: The architecture of DeepFIM.

feature-level importance vector of interaction between feature i and feature j. Here, tensor F, M is further factorized by using the canonical decomposition [22]:

$$\begin{cases} F_{i,j} = \mathbf{e}_{f_i} \odot \mathbf{e}_{f_j}, \\ M_{i,j} = \mathbf{v}_{i,f_i} \odot \mathbf{v}_{j,f_i}, \end{cases} \tag{5}$$

where $f \epsilon \mathbb{R}^n$ denotes the field, $\mathbf{e}_{f_i}$ is the embedding vector of field i, and $\mathbf{v}_{i,f_j}$ is the embedding vector of the i-th feature for field j. More importantly, we utilize the attention mechanism to compromise the feature interaction on *field-level* and *feature-level*, which is defined as follows:

$$a'_{ij} = h^T \mathrm{ReLU}\Big(W\big(F_{i,j} \odot M_{i,j}\big)x_i x_j + b\Big), \tag{6}$$

$$a_{ij} = \frac{\exp\big(a'_{ij}\big)}{\Big[\sum_{(i,j)\epsilon R_x} \exp\big(a'_{ij}\big)\Big]^{\varrho}}, \tag{7}$$

where ϱ is the smoothing exponent, a hyperparameter to be set in the range of (0, 1). When ϱ is set to 1, it recovers the softmax function [23]; when ϱ is smaller than 1, the value of the denominator will be suppressed; as a result, the attention weights will not be overly punished for active users [24]. Here, we utilize ϱ to fine-tune the attention score so that we can better control the effective strength of the interactions. For the attention network, ReLU [25] is an activation function; we also perform $L2$ regularization on its weight matrix to prevent possible overfitting. As a result, the overall formulation of FIM is

$$\overline{y}_{\mathrm{FIM}}(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + \mathbf{p}^T \sum_{i=1}^{n} \sum_{j=i+1}^{n} a_{ij} \Bigg(\underbrace{< \mathbf{e}_{f_i} \odot \mathbf{e}_{f_j} > \odot < \mathbf{v}_{i,f_j} \odot \mathbf{v}_{j,f_i} >}_{\text{hierarchical-structure form}} \Bigg) x_i x_j, \tag{8}$$

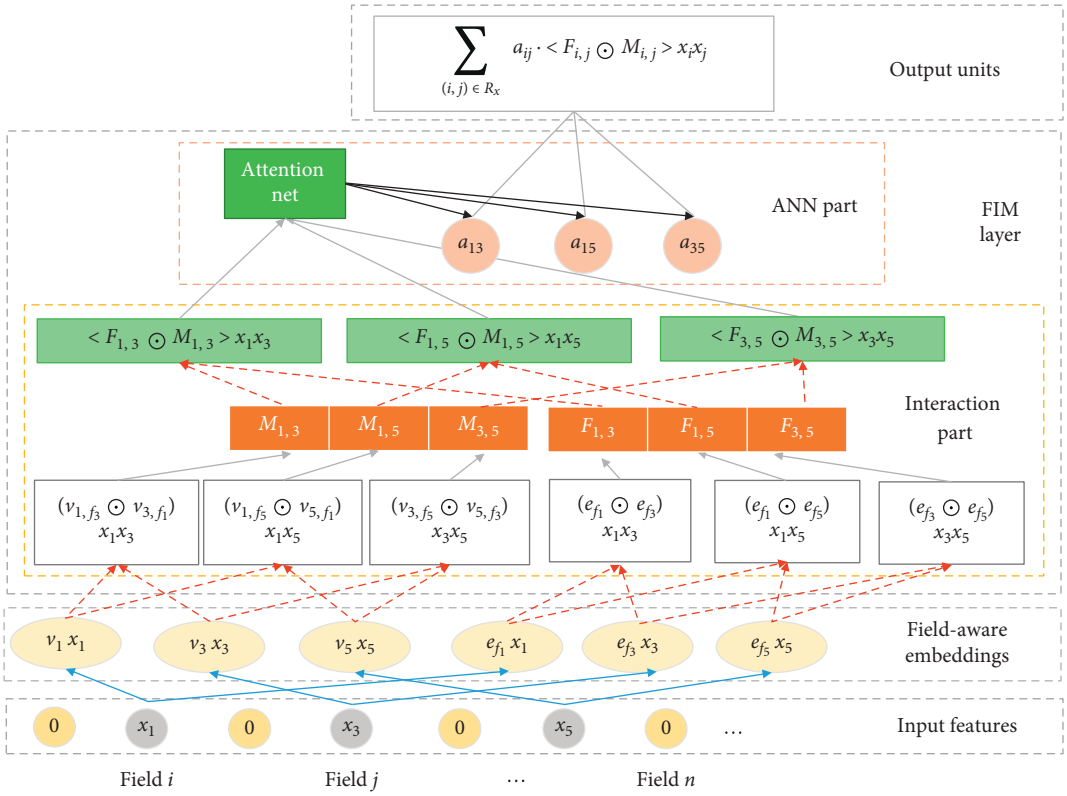

Figure 2: The architecture of the FIM component.

where a_{ij} is the attention score for feature interactions $x_i x_j$; thus the importance of feature interactions can be significantly different. Most notably, here we give a new form of expressing feature interaction, that is, the feature representation by merging vector and matrix forms; we call this the "*hierarchical-structure expression.*"

4.1. Deep Component. To model the interaction of low- and high-order features at the same time, we introduce the feed-forward neural network to capture the high-order interactions. However, for a model with n field inputs, simply concatenating feature embedding vectors carries too little information about feature interactions. NFM [2] uses the sum pooling method to compress the $(n^*(n-1))/2$ vectors into one with the model. And this vector is the input of deep neural networks. Here, we use a vector concatenation method [13] to "*concat*" them into a vector of $k^*(n^*(n-1)/2)$ dimension:

$$\mathbf{c}_{i,j} = x_i \mathbf{e}_{f_i} \odot x_j \mathbf{e}_{f_j}, \tag{9}$$

$$f_{BI}(\mathcal{T}_{\mathbf{X}}) = \bigcup_{(i,j)\epsilon n}^{\oplus} \mathbf{c}_{i,j}, \tag{10}$$

where $\mathbf{c}_{i,j}$ is the intersection vector of the features, where $\oplus$ represents the concatenate operator. More important, the Bi-Concatenation will establish a pool, which can retain information as much as possible, beneficial to the later deep neural network learning of high-order feature interaction. After the Bi-Concatenation, DNNs are adopted to extract high-order features and predict them. their input is concatenated vector after batch normalization [26] and each layer with the ReLU [25] function on the last layer output. Finally, the softmax [23] layer is used to complete the task of probability prediction.

To summarize, we give the overall formulation of the DeepFIM predictive model as

$$\tilde{y} = \text{softmax}\left(\overline{y}_{\text{FIM}} + y_{\text{DNN}(f_{BI})}\right), \tag{11}$$

where $\tilde{y} \epsilon (0,1)$ is the predicted CTR. $\overline{y}_{\text{FIM}}$ is the key component for processing feature interaction, and $y_{\text{DNN}(f_{BI})}$ is a nonlinear transformation by MLP to learn high-order interactions.

4.2. Training. To learn the parameters for DeepFIM, we minimize the negative log-likelihood function to train our model. We denote a training sample by $\mathcal{S} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, x_i is the i-th sample clicked by users. For each positive input sample, its label is $y = 1$; also, $y = 0$ is the label for the negative sample. After our model, each input sample has the respective estimated probabilities $p \epsilon [0,1]$ of the user clicking the item. More formally, our loss function is as follows:

$$L_r = -\frac{1}{N}\left\{\sum_{s\epsilon \mathcal{S}^+} y \log p + \sum_{s\epsilon \mathcal{S}^-} (1-y) y \log(1-p)\right\}, \tag{12}$$

where $\mathcal{S}$ is the training data set whose size is N. $\mathcal{S}^+$ and $\mathcal{S}^-$ are the positive sample and negative sample set.

5. Experiment

In this section, we will evaluate the performance of DeepFIM on the real dataset. Our goal is to answer the following questions:

RQ 1. How do the key hyperparameters influence the performance of the proposed model?

RQ 2. Is the FIM better than the based feature embedding FM or FFM?

RQ 3. How does DeepFIM perform compared with the state-of-the-art algorithms?

5.1. Experiment Setting

5.1.1. Date Set. We use Kaggle Avazu (https://www.kaggle.com/c/avazu-ctr-prediction) to display ad click-through rate prediction dataset, which contains 10 days of click-through log on users' mobile behaviors, including user behavior time, ad delivery site information, ad position information, user device information, user IP address, network connection type, and 9 anonymous category features. The goal is to use the data provided to predict whether a user will click the item that is shown to him/her. Avazu contains a total of 40,428,967 samples, including 33,563,901 positive samples and 68,865,66 negative samples. Because of the limited experimental environment, we select 10% of the sample data set as the experimental data; that is, we select the samples of the 9^{th} day as the training data and the last day that divide them into validation data set and test data set by 50%. There were 4042897 sample data sets, including 3620824 training, 211035 validation, and 211037 test data sets.

5.1.2. Evaluation Metrics. We use Logloss and AUC to evaluate the performance of the test set with the best parameters. Logloss is the cross-entropy loss to evaluate the performance of the classification model, and AUC is the area under the ROC curve to measure the probability that the random positive samples rank higher than the random negative samples. The lower the Logloss score, the higher the AUC score and the better the performance.

5.1.3. Baselines. Among the popular CTR models such as FM [1], FFM [3], XDeepFM [19], Wide and Deep [18], FNN [27], DeepFM [15], and FNFM [13], we compare our models with the last three because they have similar architectures to DeepFIM, and they are also the state-of-the-art models for CTR prediction. As a result, the 7 baseline models to evaluate DeepFIM are LR (logistic regression), FM [1], FFM [3], FNN [27], DeepFM [15], FNFM [13], and DCN [16].

5.1.4. Hyperparameters. To fairly compare all the models, we learn all models by optimizing the square loss of predictions and using the Adam [28] optimized method with a learning rate of 0.001. For comprehensive consideration of training time and convergence speed, the batch size is chosen to be 4096. We also adopt the early stopping strategy based on the performance on the validation set and carefully tune the dropout ratios and regularization strength values for all models to prevent overfitting.

5.2. Parameters Analysis (RQ 1). To show the best performance of DeepFIM, we fine-tune the four main parameters: embedding size, hidden size, dropout, and regularization strength. Firstly, we analyze the embedded size and hidden size. Figures 3(a) and 3(b) illustrate the validation errors of FNN, DeepFM, FNFM, and DeepFIM on different parameter sizes. We can observe that when the embedding size of DeepFM and DeepFIM is 8, Logloss reaches the minimum value; this is because the large embedding size brings the model better representation ability. Besides, FNN and FNFM achieve the best performance when the embedding size is 2, respectively. Note that the performance of models is different when the embedding size changes, thus often requiring specific evaluation based on model characteristics and data sets. On the other hand, for the effect of hidden size, we can find that when the hidden size is 64 and 128, FNN and rest model achieve the best performance. As a result, we will set the size of the best performance for all models. Specifically, for the embedded size, FNN and FNFM are set to 32 and 16, and DeepFM and DeepFIM are set to 8, respectively. For the hidden size, FNN is set to 64, and the remaining models are set to 128 in the following parameter analysis.

Secondly, we utilize dropout [29] on a different layer to avoid overfitting. Specifically, the dropout is adopted on the MLP layer for all models. Figure 3(c) illustrates the verification of Logloss results, and we can find, by setting an appropriate dropout rate, the performance of the models has been improved to varying degrees. As shown in Figure 3(c), we set the dropout from 0 to 0.9 with increments of 0.1. When the dropout tends to 1, the performance of both models is poor due to the underfitting issue; when the dropout tends to 0, some models also cannot achieve the best performance. In the dataset, the DeepFIM model performs best when the dropout ratio is 0.3. Therefore, the benefits of applying for a dropout can be verified. For these four models, we adopt their optimal dropout rate in the following analysis.

As for regularization strength parameter L2, we first set the same regularization strength 0.0001 for FNN, DeepFM, FNFM, and DeepFIM; then, we evaluate DeepFIM and rest models on several regularization strengths. As shown in Figure 3(d), the verification error of Logloss continues to increase as the regularization strength increases, and the greater the regularization strength, the lower the model performance. This suggests that increasing the regularization strength does not always result in performance improvement. Without special mention, the regularization strength is set to the best for all models in the following comparison.

5.3. Impact of FIM (RQ 2). To further evaluate the impact of the FIM layer and RQ 2, we first compared FIM with FM, FFM, and Table 1 shows the optimal validation of the Logloss results and the corresponding parameters of the

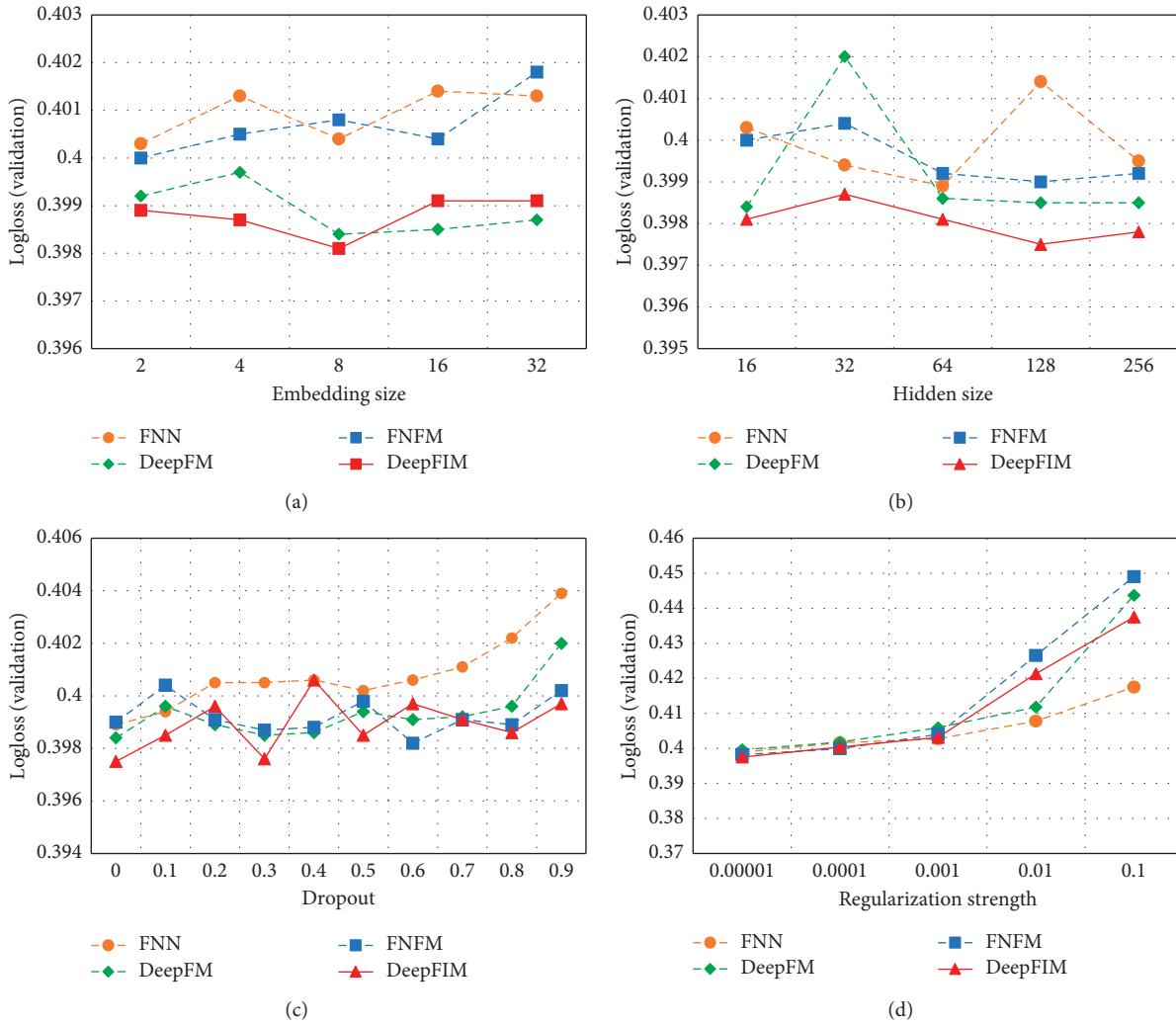

Figure 3: Logloss results of FNN, DeepFM, FNFM, and DeepFIM under different parameters.

Table 1: The comparisons between FM, FFM, AFM, AFFM, and FIM.

Model	Embedding size	Dropout	Regularization	Logloss
FM	16	0	0.00001	0.4011
FFM	8	0	0.00001	0.3999
AFM	4	0	0.00001	0.4005
AFFM	4	0	0.00001	0.3996
FIM	**4**	**0.3**	**0.00001**	**0.3991**

model. We found that FIM has a lower Logloss result than FM and FFM. At the same time, the result of Logloss of FFM is better than that of FM, which shows that adding field information can improve the performance of the model; the performance of the model can be further improved.

Besides, for a fair comparison of FM, FFM, and FIM, we employ the attention mechanism of AFM on the feature embedding of FFM and named the model AFFM; then, we compare FIM with AFM, AFFM on the same dataset. We carefully tune the embedding size, dropout rate, and regularization strength for AFM, AFFM and FIM as we do in the parameter analysis. Finally, as shown in Figure 4, we train each model ten times and obtain their best validation Logloss as the final result. Table 1 shows the best validation Logloss results and corresponding parameters of models; we observe that FIM attains lower Logloss results compared with AFM, and FIM also outperforms AFFM on the dataset. The observation confirms that, by addressing the problem of *short-expression* as we proposed, FIM outperforms the AFM, no matter the based feature embedding is FM or FFM.

5.4. Model Performance (RQ 3). To answer RQ 3, we compared our results with the state-of-the-art algorithms. Table 2 shows the corresponding test set scores for different models when the validation set achieves the lowest Logloss,

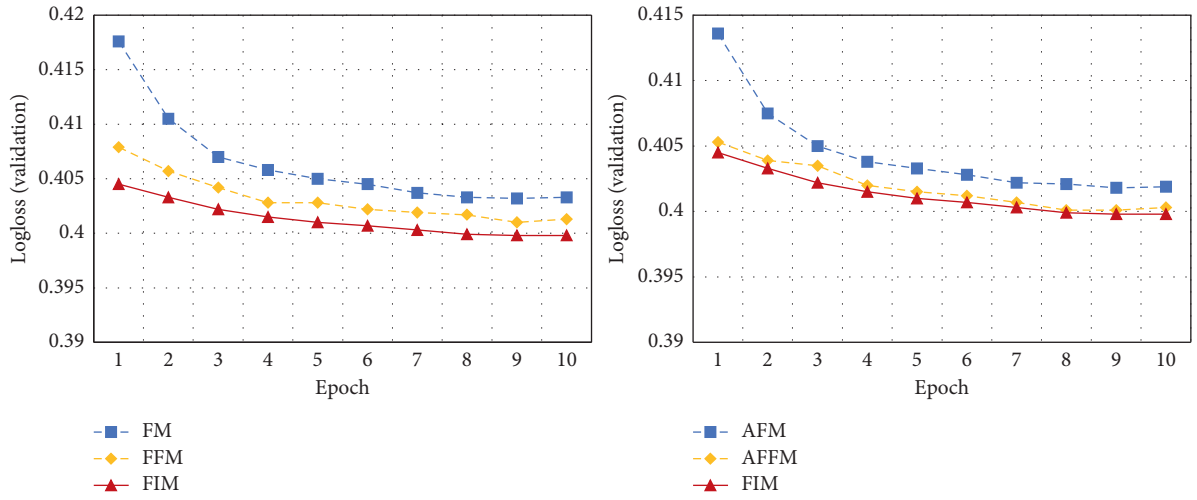

Figure 4: Logloss results on the validation sets of FM, FFM, AFM, AFFM, and FIM.

Table 2: Overall performance of different models.

Model	Logloss	AUC
LR	0.4059	0.7296
FM	0.4011	0.7394
FFM	0.3996	0.7435
DeepFM	0.3984	0.745
FNFM	0.3979	0.7463
DCN	0.3979	0.7456
FNN	0.3991	0.7426
DeepFIM	**0.3973**	**0.7468**

As a report in [18], a small improvement in offline AUC evaluation can lead to a substantial increase in online CTR.

Table 3: Accuracy on cross-validation.

Model	Accuracy	Std.	Fold 1	Fold 2	Fold 3	Fold 4
FNN	0.8335	0.00019	0.8333	0.8336	0.8338	0.8334
DeepFM	0.8354	**0.00011**	0.8354	0.8356	0.8354	0.8353
FNFM	0.8353	0.00022	0.835	0.8354	0.8356	0.8353
DeepFIM	**0.8356**	**0.00012**	**0.8355**	**0.8356**	**0.8358**	**0.8355**

where we have the following observations: (1) we observe that LR underperforms the other methods by at least 1.3% in terms of AUC (1.18% in terms of Logloss), which shows that feature interactions are critical to improving the CTR prediction. (2) Learning the importance of different feature interactions can improve model performance. This observation stems from the fact that AFM- and FIM-based models (DeepFIM) perform better than FM. As the best model, DeepFIM performance is more than FM 0.98% in terms of AUC (0.87% in terms of Logloss). (3) DeepFIM outperforms the models learning high-order and low-order feature interactions simultaneously while sharing the same feature embedding for feature interactions. Compared to these models, our model adopts different field-aware embedding for interaction and achieves the best performance in both Logloss and AUC.

5.5. Cross-Validation. To see how these models can be extended to an independent dataset, we compared the performance of four models (FNN, DeepFM, FNFM, and DeepFIM) with a 4-fold cross-validation evaluation. We do not include LR and FM models here because they do not perform well in accuracy, nor do we show the results of FFM and DCN models here because DeepFIM and FNFM extend their structures and have better performance, respectively. For simplicity, we trained the models over 10 epochs and compared their predictive accuracy.

From Table 3, we can see that DeepFIM has the highest average accuracy and second smallest standard deviation. And in the cross-validation, DeepFIM achieves the highest results in all 4 folds. The results show DeepFIM has potentially good stability and generalizability capacity comparing with other existing deep learning approaches.

6. Conclusions

In this work, we propose a novel neural CTR model called DeepFIM to better learn the feature interactions. In particular, we employ two different embedding modes to learn flexible interfield and intrafield interactions to solve the "short-expression" problem. Besides, Bi-Concatenation is introduced to enhance the acquisition of higher-order features, which makes it easier to learn the full-connection network parameters. It is worth noting that this paper aims to point out that the features in the field may not be expressed sufficiently due to their short feature-length, so that noise will be formed during feature interaction, thus reducing the performance of the model. We conduct extensive experiments on the public data set to confirm our observation; the results show that our model achieves comparable and even materially better results than the state-of-the-art methods. In the future, we hope to further reduce the parameters of the model by more advanced methods and to extend the DeepFIM to more flexible structures by applying another neural structure search [30] to more data sets.

Data Availability

The data are available at https://github.com/qigaofeng/deepfim.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] S. Rendle, "Factorization machines," in *Proceedings of the IEEE International Conference on Data Mining*, pp. 995–1000, Sydney, Australia, December 2010.

[2] X. He and T. S. Chua, "Neural factorization machines for sparse predictive analytics," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information*, pp. 355–364, Tokyo, Japan, August 2017.

[3] Y. Juan, Y. Zhuang, W.-S. Chin, and C.-J. Lin, "Field-aware factorization machines for ctr prediction," in *Proceedings of the 10th ACM Conference on Recommender Systems*, pp. 43–50, Boston, MA, USA, September 2016.

[4] J. Pan and J. Xu, "Field-weighted factorization machines for click-through rate prediction in display advertising," 2018, https://arxiv.org/abs/1806.03514.

[5] F. Hong, "Interaction-aware factorization machines for recommender systems," in *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 3804–3811, Honolulu, HI, USA, February 2019.

[6] J. Zhang, T. Huang, and Z. Zhang, "FAT-DeepFFM: field attentive deep field-aware factorization machine," 2019, https://arxiv.org/abs/1905.06336.

[7] L. Bin, "AutoFIS: automatic feature interaction selection in factorization models for click-through rate prediction," in *Proceedings of the KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining ACM*, Virtual Event, CA, USA, August 2020.

[8] W. Song, C. Shi, and " AutoInt, "Automatic feature interaction learning via self-attentive neural networks," 2019, http://arxiv.org/abs/1810.11921.

[9] Q.-L. Zhang, L. Rao, and Y. Yang, "DGFFM: generalized field-aware factorization machine based on densenet," in *Proceedings of the 2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, Budapest, Hungary, July 2019.

[10] G. Zhou, C. Song, X. Zhu et al., "Deep interest network for click-through rate prediction," 2017, https://arxiv.org/abs/1706.06978.

[11] Y. Yang, B. Xu, F. Shen, and J. Zhao, "Operation-aware neural networks for user response prediction," 2019, https://arxiv.org/abs/1904.12579.

[12] Q. Wang, F. a. Liu, S. Xing, X. Zhao, and T. Li, "Research on CTR prediction based on deep learning," *IEEE Access*, vol. 7, no. 11, pp. 12779–12789, 2019.

[13] L. Zhang, S. Cheng, J. Huang, S. Li, and G. Pan, "Field-aware neural factorization machine for click-through rate prediction," 2019, https://arxiv.org/abs/1902.09096.

[14] X. Xin, "CFM: convolutional factorization machines for context-aware recommendation," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence IJCAI-19*, Macao, China, August 2019.

[15] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "DeepFM: a factorization-machine based neural network for CTR prediction," 2017, https://arxiv.org/abs/1703.04247.

[16] R. Wang, B. Fu, G. Fu, and M. Wang, "Deep & cross network for ad click predictions," in *Proceedings of the ADKDD*, Halifax, Canada, August 2017.

[17] 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," 2017, https://arxiv.org/abs/1708.04617.

[18] H.-T. Cheng, L. Koc, J. Harmsen et al., "Wide & deep learning for recommender systems," in *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pp. 7–10, Boston, MA, USA, September 2016.

[19] J. Lian and X. Zhou, "xDeepFM: combining explicit and implicit feature interactions for recommender systems," 2018, http://arxiv.org/abs/1803.05170.

[20] R. Saia, L. Boratto, and S. Carta, "A semantic approach to remove incoherent items from a user profile and improve the accuracy of a recommender system," *Journal of Intelligent Information Systems*, vol. 47, no. 1, pp. 111–134, 2016.

[21] S. Carta, A. S. Podda, D. R. Recupero, R. Saia, and G. Usai, "Popularity prediction of instagram posts," *Information*, vol. 11, no. 9, p. 453, 2020.

[22] T. Kolda and B. Bader, "Tensor decompositions and applications," in *Proceedings of the International Conference on SIAM*, pp. 455–500, Sparks, NV, USA, May 2009.

[23] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, https://arxiv.org/abs/1503.02531.

[24] X. He, Z. He, J. Song, Z. Liu, Y.-G. Jiang, and T.-S. Chua, "NAIS: neural attentive item similarity model for recommendation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 12, pp. 2354–2366, 2018.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proceedings of the Advances in Neural Information Processing Systems*, pp. 1097–1105, Lake Tahoe, NV, USA, December 2012.

[26] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," 2015, https://arxiv.org/abs/1502.03167.

[27] W. Zhang, T. Du, and J. Wang, "Deep learning over multi-field categorical data," 2016, http://arxiv.org/abs/1601.02376.
[28] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," 2014, https://arxiv.org/abs/1412.6980.
[29] N. Srivastava, G. Hinton, and A. Krizhevsky, "Dropout: a simple way to prevent neural networks from overfitting," in *Proceedings of the International Conference on Machine Learning*, pp. 1929–1958, Long Beach, CA, USA, June 2014.
[30] C. Liu, B. Zoph, and J. Shlens, "Progressive neural architecture search," 2018, https://arxiv.org/abs/1712.00559.