A Dual Input-aware Factorization Machine for CTR Prediction

Wantong Lu[1] , **Yantao Yu**[1] , **Yongzhe Chang**[1,2] , **Zhen Wang**[3] , **Chenhui Li**[1] and **Bo Yuan**[1*]
[1]Shenzhen International Graduate School, Tsinghua University
[2]Data 61 CSIRO, Sydney, Australia
[3]UBTECH Sydney AI Centre, The University of Sydney, Australia
{luwt18, yyt17, lch17}@mails.tsinghua.edu.cn, yongzhe.chang@data61.csiro.au,
zwan4121@uni.sydney.edu.au, yuanb@sz.tsinghua.edu.cn

Abstract

Factorization Machines (FMs) refer to a class of general predictors working with real valued feature vectors, which are well-known for their ability to estimate model parameters under significant sparsity and have found successful applications in many areas such as the click-through rate (CTR) prediction. However, standard FMs only produce a single fixed representation for each feature across different input instances, which may limit the CTR model's expressive and predictive power. Inspired by the success of *Input-aware Factorization Machines* (IFMs), which aim to learn more flexible and informative representations of a given feature according to different input instances, we propose a novel model named *Dual Input-aware Factorization Machines* (DIFMs) that can adaptively reweight the original feature representations at the bit-wise and vector-wise levels simultaneously. Furthermore, DIFMs strategically integrate various components including Multi-Head Self-Attention, Residual Networks and DNNs into a unified end-to-end model. Comprehensive experiments on two real-world CTR prediction datasets show that the DIFM model can outperform several state-of-the-art models consistently.

1 Introduction

The prediction of click-through rate (CTR) is crucial in online advertising [McMahan *et al.*, 2013; Juan *et al.*, 2016; Wen *et al.*, 2019], where the task is to estimate the probability that a user will click on a recommended ad or item. In online advertising, advertisers pay publishers to display their ads on publishers' sites. One popular payment model is the cost-per-click (CPC) model [Zhou *et al.*, 2018; Zhou *et al.*, 2019], where advertisers are charged only when a click occurs. As a consequence, a publisher's revenue relies heavily on the ability to predict CTR accurately [Wang *et al.*, 2017].

Representing features accurately is important for CTR prediction. Typically, each feature is represented as a scalar weight and a k-dimensional embedding vector in Factorization Machines (FMs)-based models [Rendle, 2010], such as AFM [Xiao *et al.*, 2017], NFM [He and Chua, 2017], DeepFM [Guo *et al.*, 2017] and xDeepFM [Lian *et al.*, 2018]. However, in all the above models, the same representation of a given feature is shared among different input instances (a.k.a. different scenes), which may limit the CTR model's predictive power. For example, the feature *female* is apparently crucial for click probability in the instance: {*young, female, student, pink, skirt*}. However, in another instance: {*young, female, student, blue, notebook*}, the feature *female* is relatively less crucial. As such, the representation of a given feature in different input instances (e.g., the feature *female* in the above two instances) should be assigned different levels of predictive power. To this end, IFMs were proposed [Yu *et al.*, 2019], which consider the uniqueness of each input instance and learn a unique *input-aware factor* (used to reweight the original feature representations) for the same feature in different input instances via DNNs.

Despite great promise, we argue that, in IFMs, plain DNNs implicitly learn the input-aware factors at the bit-wise level. That is to say, all the elements within the same embedding vector will affect each other. Meanwhile, in the field of CTR prediction, whether DNNs are the most effective model in learning the input-aware factors remains an open question. After all, the bit-wise level learning itself may be insufficient for achieving optimal prediction.

Inspired by the success of *Transformer* in machine translation tasks [Vaswani *et al.*, 2017], which is highly efficient and capable of uncovering syntactic and semantic patterns among words in a sentence, we adopt its core idea, the Multi-Head Self-Attention mechanism, to exploit the implicit relationship among features in an instance, and to adaptively learn the input-aware factors at the vector-wise level.

In this paper, in light of the uniqueness of each instance, by allowing the same feature to have different levels of predictive power in different instances, we propose a competent model for CTR prediction tasks named *Dual Input-aware Factorization Machines* (DIFMs). It shares the same benefits as the DNNs model (i.e., the IFM model) and introduces a novel *vector-wise network* that is more effective in learning the input-aware factors for reweighting the original feature representations. To summarize, we make the following key contributions:

*Corresponding Author

- We propose a novel network model DIFMs, which can adaptively learn different representations of a given feature according to different input instances effectively.
- Compared to the IFM model, our proposed DIFM model can effectively learn the *input-aware factors* (used to reweight the original feature representations) at the bit-wise and vector-wise levels simultaneously.
- The DIFM model strategically integrates various components including Multi-Head Self-Attention, Residual Networks and DNNs into a unified end-to-end model.
- We conduct comprehensive experiments on two real-world CTR prediction datasets, and the results confirm that our DIFM model can outperform major state-of-the-art models consistently.

2 Factorization Machines

As a class of general predictors working with real valued feature vectors, Factorization Machines (FMs) are able to estimate parameters under significant sparsity effectively. Formally, FMs predict the target as:

$$\hat{y}_{FM}(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle \, x_i x_j \quad (1)$$

where w_0 is the global bias, w_i and $\mathbf{v}_i \in \mathbb{R}^k$ denote the scalar weight and the k-dimensional embedding vector of the i-th feature, respectively. $\langle \mathbf{v}_i, \mathbf{v}_j \rangle$ is the dot product of two vectors of size k, which models the interaction between the i-th and j-th features.

3 Our Approach

In this section, we present the structure of our proposed DIFM model as shown in Figure 1.

3.1 The DIFM Model

The motivation is to adaptively learn the *input-aware factors* (used to reweight the original feature representations: w and $\mathbf{v}$) both at the bit-wise and vector-wise levels simultaneously. To this end, we propose the novel Dual Input-aware Factorization Machines (DIFMs) for CTR prediction.

The DIFM model consists of the following components: (1) Sparse Input and Embedding Layer; (2) Dual-Factor Estimating Networks (Dual-FEN) Layer; (3) Combination Layer; (4) Reweighting Layer; (5) Prediction Layer.

Sparse Input and Embedding Layer

The sparse input layer and embedding layer are widely used in deep learning based CTR models such as DeepFM [Guo *et al.*, 2017] and AFM [Xiao *et al.*, 2017]. The sparse input layer adopts a sparse representation for raw input features. The embedding layer is able to embed the sparse feature into a low dimensional, dense real-value vector. The output of the embedding layer is a concatenated field embedding vector: $\mathbf{E_x} = [\mathbf{v}_1^T, \mathbf{v}_2^T, \cdots, \mathbf{v}_i^T, \cdots, \mathbf{v}_h^T]$, where h denotes the number of fields while $\mathbf{v}_i \in \mathbb{R}^k$ denotes the embedding vector of the i-th field , and k is the embedding size.

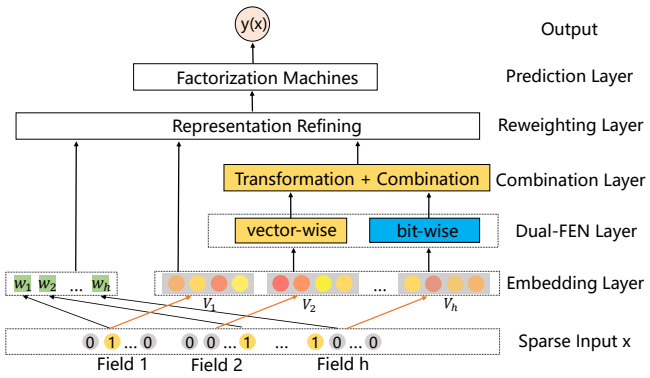


Figure 1: The network structure of the proposed Dual Input-aware Factorization Machines model.

Dual-FEN Layer

The Dual-FEN Layer consists of two components[1]: the vector-wise part and the bit-wise part, which both aim to learn a unique input-aware factor for the same feature in different input instances.

The vector-wise part is inspired by the success of *Transformer* for machine translation in natural language processing (NLP) [Vaswani *et al.*, 2017]. In this work, we adopt its core idea, the Multi-Head Self-Attention mechanism, to learn the input-aware factors at the vector-wise level. Figure 2 shows the structure of the vector-wise part.

First, we need to reshape the embedding vector $\mathbf{E_x}$ into a $h \times k$ matrix, which is the input of the vector-wise part.

$$\mathbf{U}_{vec} = \text{reshape}(\mathbf{E_x}) \quad (2)$$

Next, we map the input matrix into three different matrices: $\mathbf{Q}_i$ (Queries), $\mathbf{K}_i$ (Keys), $\mathbf{V}_i$ (Values) for the i-th head (a.k.a. the i-th subspace):

$$\begin{aligned} \mathbf{Q}_i &= \mathbf{U}_{vec}\mathbf{W}^{\mathbf{Q}_i} \\ \mathbf{K}_i &= \mathbf{U}_{vec}\mathbf{W}^{\mathbf{K}_i} \\ \mathbf{V}_i &= \mathbf{U}_{vec}\mathbf{W}^{\mathbf{V}_i} \end{aligned} \quad (3)$$

where the projection matrices $\mathbf{W}^{\mathbf{Q}_i}, \mathbf{W}^{\mathbf{K}_i} \in \mathbb{R}^{k \times d_k}$, $\mathbf{W}^{\mathbf{V}_i} \in \mathbb{R}^{k \times d_v}$, and d_k denotes the size of the attention factor (in general, $d_k = d_v$). Then, according to *Scaled Dot-Product Attention* [Vaswani *et al.*, 2017], we conduct the dot products of the Query with all Keys divided by $\sqrt{d_k}$, and apply a softmax function to obtain the weights on Values:

$$\text{Attention}(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i) = \text{softmax}(\frac{\mathbf{Q}_i {\mathbf{K}_i}^T}{\sqrt{d_k}})\mathbf{V}_i \quad (4)$$

As the original paper [Vaswani *et al.*, 2017] suggests, it is beneficial to linearly project the Queries, Keys and Values n times with different, learned linear projections to d_k, d_k and d_v dimensions, respectively. On each of these projected Queries, Keys and Values, we perform the attention function

[1]For simplicity of writing, we use "*vec*" denotes "vector-wise" and "*bit*" denotes "bit-wise" in formulas.

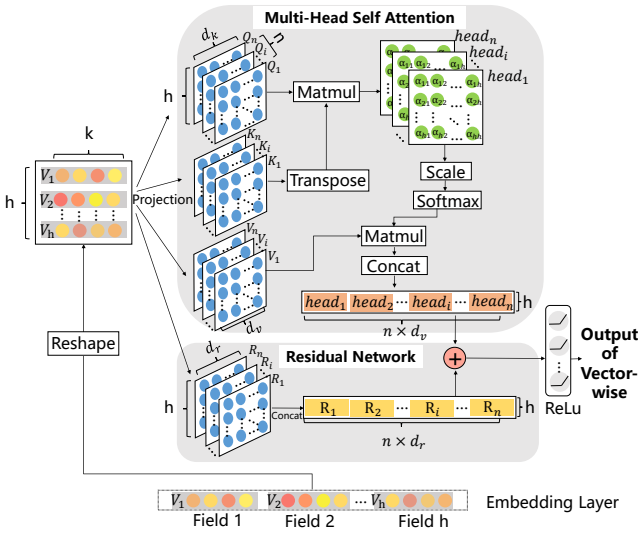


Figure 2: The network structure of the vector-wise part in DIFM.

in parallel, yielding a set of d_v-dimensional vectors and concatenate them:

$$\begin{aligned} \text{MultiHead}(\mathbf{U}_{vec}) &= \text{Concat}(\text{head}_1, \ldots, \text{head}_n) \\ \text{where head}_i &= \text{Attention}(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i) \end{aligned} \tag{5}$$

where $\text{head}_i \in \mathbb{R}^{h \times d_v}$ denotes the output of the i-th single head and n is the number of heads. $\text{MultiHead}(\mathbf{U}_{vec}) \in \mathbb{R}^{h \times (d_v \times n)}$ denotes the output of Multi-Head Self-Attention.

To preserve some information about the original embedding vector, after the Multi-Head Self-Attention, we use the Residual Network to add the original feature embedding vector to the output of Multi-Head Self-Attention:

$$\begin{aligned} \text{Resdual}(\mathbf{U}_{vec}) &= \text{Concat}(\text{R}_1, \ldots, \text{R}_n) \\ \text{where R}_i &= \mathbf{U}_{vec}\mathbf{W}^{\mathbf{R}_i} \end{aligned} \tag{6}$$

where $\mathbf{W}^{\mathbf{R}_i} \in \mathbb{R}^{k \times d_r}$ denotes the linear projection by the shortcut connections to match the dimensions. $\text{R}_i \in \mathbb{R}^{h \times d_r}$ is the i-th residual block related to head_i. Note that, to keep dimensions consistent, we set $d_r = d_k = d_v$. Finally, the output of the vector-wise part can be formulated as:

$$\mathbf{O}_{vec} = \sigma(\text{MultiHead}(\mathbf{U}_{vec}) + \text{Resdual}(\mathbf{U}_{vec})) \tag{7}$$

where $\mathbf{O}_{vec} \in \mathbb{R}^{h \times (d_v \times n)}$ and σ is the activation function.

The bit-wise part is a stack of fully connected layers. The input of the bit-wise part: $\mathbf{I}_{bit} = \mathbf{E}_{\mathbf{x}}$. Figure 3 shows the structure of the bit-wise part.

$$\begin{aligned} \mathbf{a}_1 &= \sigma_1(\mathbf{W}_1\mathbf{I}_{bit} + \mathbf{b}_1), \\ &\cdots \\ \mathbf{a}_i &= \sigma_i(\mathbf{W}_i\mathbf{a}_{i-1} + \mathbf{b}_i) \end{aligned} \tag{8}$$

where $\mathbf{I}_{bit} \in \mathbb{R}^{(h \times k)}$ is the input of the bit-wise part, and $\mathbf{W}_i$, $\mathbf{b}_i$, σ_i, $\mathbf{a}_i$ denote the weight matrix, bias vector, activation function and the output of the i-th layer, respectively. The output of the bit-wise part $\mathbf{O}_{bit} \in \mathbb{R}^t$ where t is the number of neurons of the last hidden layer is defined as:

$$\mathbf{O}_{bit} = \mathbf{a}_L = \sigma_L(\mathbf{W}_L\mathbf{a}_{L-1} + \mathbf{b}_L) \tag{9}$$

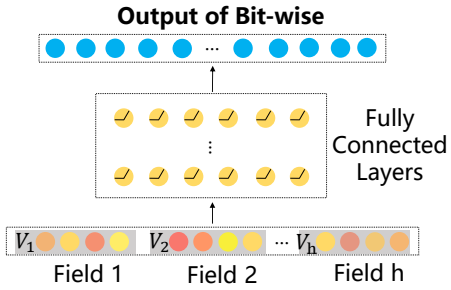


Figure 3: The network structure of the bit-wise part in DIFM.

Combination Layer

In this layer, the output of the vector-wise part is reshaped into the form of a single vector (i.e., $\mathbf{O}_{vec} \in \mathbb{R}^{h_1}$, $\mathbf{O}_{bit} \in \mathbb{R}^{h_2}$, where $h_1 = h \times d_v \times n$ and $h_2 = t$).

Then, the two input-aware factors based on the vector-wise and bit-wise parts are calculated as:

$$\begin{aligned} \mathbf{m}_{vec} &= \mathbf{O}_{vec}\mathbf{P}_{vec} \\ \mathbf{m}_{bit} &= \mathbf{O}_{bit}\mathbf{P}_{bit} \end{aligned} \tag{10}$$

where $\mathbf{P}_{vec} \in \mathbb{R}^{h_1 \times h}$ and $\mathbf{P}_{bit} \in \mathbb{R}^{h_2 \times h}$ are the weight matrices that transform $\mathbf{O}_{vec}$ and $\mathbf{O}_{bit}$ into a h-dimensional vector, respectively. Finally, we combine the two intermediate input-aware factors:

$$\mathbf{m}_{\mathbf{x}} = \mathbf{m}_{vec} + \mathbf{m}_{bit} \tag{11}$$

where $\mathbf{m}_{\mathbf{x}} \in \mathbb{R}^h$ is the complete input-aware factor corresponding to the sparse input $\mathbf{x}$, which considers both the vector-wise and bit-wise parts.

Reweighting Layer

The definition of the reweighting layer is as follows:

$$\begin{aligned} w_{\mathbf{x},i} &= m_{\mathbf{x},i}w_i \\ \mathbf{v}_{\mathbf{x},i} &= m_{\mathbf{x},i}\mathbf{v}_i \end{aligned} \tag{12}$$

where $m_{\mathbf{x},i}$ is the i-th element in $\mathbf{m}_{\mathbf{x}}$, which denotes the i-th input-aware factor for each non-zero feature in $\mathbf{x}$. $w_{\mathbf{x},i}$, $\mathbf{v}_{\mathbf{x},i}$ are the reweighted representations of features for the specific input $\mathbf{x}$, which are more accurate and informative.

Prediction Layer

With the reweighted input-aware weight $w_{\mathbf{x},i}$ and embedding vector $\mathbf{v}_{\mathbf{x},i}$, the final FM prediction score is obtained as:

$$\hat{y}_{DIFM}(x) = w_0 + \sum_{i=1}^{n} w_{\mathbf{x},i}x_i + \sum_{i=1}^{n}\sum_{j=i+1}^{n} \langle \mathbf{v}_{\mathbf{x},i}, \mathbf{v}_{\mathbf{x},j} \rangle x_i x_j \tag{13}$$

3.2 Learning

For binary classification, the learning process aims to minimize the following objective function (log loss):

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(\sigma(\hat{y_i})) + (1 - y_i)\log(1 - \sigma(\hat{y_i}))) \tag{14}$$

where $y_i \in \{0, 1\}$ is the ground truth of the i-th instance while $\sigma(\hat{y_i}) \in (0, 1)$ is the predicted CTR (here σ is the $sigmoid$ function) and N is the total number of training instances.

3.3 Relationship with FM and IFM

By removing the vector-wise part and Combination Layer, it is easy to see that our DIFM model will be degraded to the IFM model. If we further remove the DNNs (bit-wise part) , and at the same time fix all $m_{\mathbf{x},i}$ to 1, $w_{\mathbf{x},i}$ and $\mathbf{v}_{\mathbf{x},i}$ will only depend on the i-th feature, and the shallow DIFM model is downgraded to the traditional FM model.

4 Related Work

Logistic Regression (LR) only models the linear combination of raw features for CTR prediction. FMs [Rendle, 2010] use factorization techniques to model second-order feature interactions and work well on large sparse data. As a variant of FMs, Field-aware FM (FFM) [Juan *et al.*, 2016] allows each feature to have multiple embedding vectors for feature interactions but suffers from the memory requirement.

With the success of deep learning in computer vision, speech recognition, and natural language processing [He *et al.*, 2016; Amodei *et al.*, 2016; Cho *et al.*, 2014], an increasing number of DNN-based models for CTR prediction have been proposed. How to effectively model feature interactions has become the key challenge for most of these models. FNN [Zhang *et al.*, 2016] is a forward neural network using FM to pre-train the embedding layer. PNN [Qu *et al.*, 2016] introduces a product layer between the embedding layer and DNN layers to explore the high-order feature interactions. NFM [He and Chua, 2017] devises a bilinear interaction layer to obtain the element-wise product of pairwise feature interactions and then stacks multiple non-linear layers over the bilinear interaction layer to capture the high-order feature interactions. Wide&Deep [Cheng *et al.*, 2016] comprises the wide and deep parts, where the wide part models the linear low-order feature interactions, and the deep part models the non-linear high-order feature interactions. However, feature engineering is still needed in the wide part. DeepFM [Guo *et al.*, 2017] uses FM to replace the wide part in Wide&Deep without any feature engineering. DCN [Wang *et al.*, 2017] efficiently captures certain bounded-degree feature interactions explicitly. xDeepFM [Lian *et al.*, 2018] also models the low-order and high-order feature interactions in an explicit way by proposing a novel CIN component.

Meanwhile, the attention mechanism is also introduced for CTR prediction. AFM [Xiao *et al.*, 2017] adopts the attention mechanism to model the importance of all feature interactions. AutoInt [Song *et al.*, 2019] combines the multi-head attention and residual networks for feature interactions.

However, in all the above models, the same representation of a given feature is shared in different instances, which may limit the CTR model's predictive power [Yu *et al.*, 2019]. To overcome this limit, IFM [Yu *et al.*, 2019] learns a unique input-aware factor for the same feature in different instances via DNNs. Inspired by the success of IFM, our proposed Dual Input-aware Factorization Machines (DIFMs) can adaptively learn the *input-aware factors* (used to reweight the original feature representations) at the bit-wise and vector-wise levels simultaneously and effectively.

5 Experiments

In this section, we conduct extensive experiments to answer the following questions:

RQ1 How do the key hyper-parameters of DIFM (i.e., the number of heads, the attention size d_k, activation function and the number of hidden layers) impact its performance?

RQ2 Is it necessary to combine the bit-wise and vector-wise levels for learning the input-aware factors? Which is the most important component in DIFM?

RQ3 How does DIFM perform compared to state-of-the-art methods for CTR prediction?

We will answer these questions after presenting some fundamental experimental settings.

5.1 Experimental Settings

Datasets. We evaluate our proposed DIFM model on the following two datasets. **Avazu**[2] dataset was published in the contest of Avazu Click-Through Rate Prediction in 2014. It contains click logs with **40** millions of data instances. For each click data, there are 24 fields which indicate elements of a single ad impression. We follow the data processing details of IFM [Yu *et al.*, 2019] and split it randomly into two parts: 80% is for training and 20% is for testing. **Criteo**[3] dataset contains one month of ad click logs. There are 13 continuous features and 26 categorical ones. We select 7 consecutive days of samples for training, and the next 1 day for evaluation. At last, it contains click logs with **100** millions of data instances.

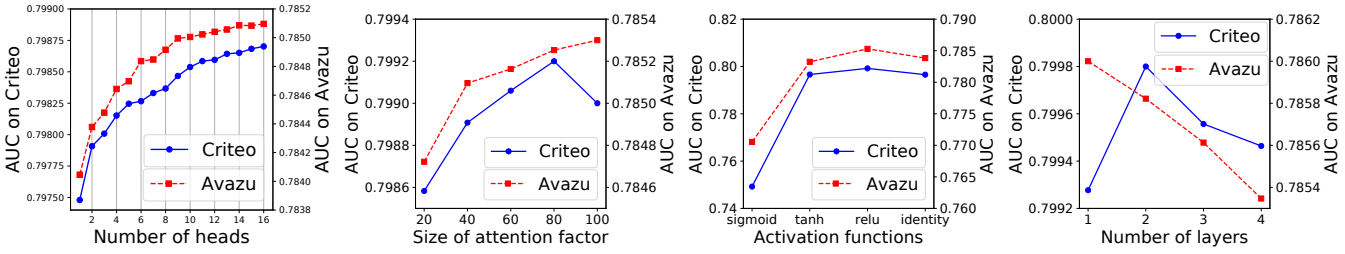
Evaluation Metrics. We use two evaluation metrics in our experiments: **AUC** (Area Under ROC) and **Logloss** (cross entropy). Note that **an improvement of 0.001-level in AUC or Logloss is usually regarded as being significant for CTR prediction**, because it will bring a large increase in a company's revenue if the company has a large user base, which has also been pointed out in many existing works [Cheng *et al.*, 2016; Guo *et al.*, 2017; Wang *et al.*, 2017; Song *et al.*, 2019].

Baselines. We compare our proposed DIFM model with the following eight competitive models, some of which are state-of-the-art models for CTR prediction.

- **LR** [Lee *et al.*, 2012]: Logistic Regression only models the linear combination of raw features.
- **FM** [Rendle, 2010]: FM uses factorization techniques to model second-order feature interactions.
- **FFM** [Juan *et al.*, 2016]: FFM introduces field-aware embedding vectors mechanism, which gain FFM higher capacity and better performance. Considering its high space complexity, we set $k = 4$ in this paper.
- **NFM** [He and Chua, 2017]: We implement the NeuralFM and set the number of layers in the hidden layer to 1 with 512 neurons as the original paper.

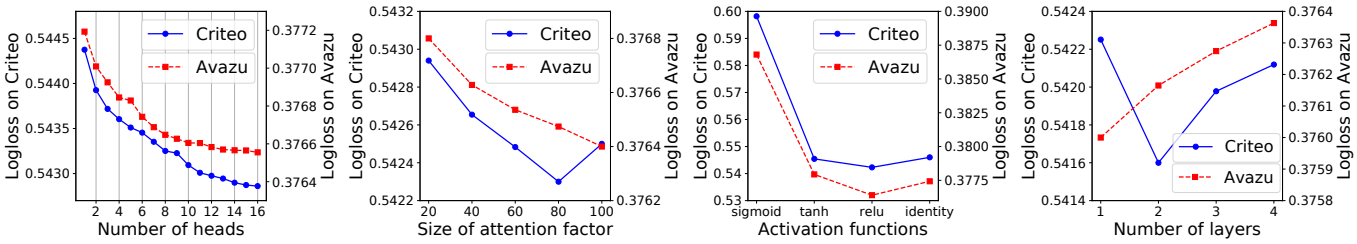
[2] http://www.kaggle.com/c/avazu-ctr-prediction

[3] http://labs.criteo.com/downloads/download-terabyte-click-logs/



(a) Number of heads (b) Size of attention factor (c) Activation functions (d) Number of layers

Figure 4: Impact of network hyper-parameters on AUC performance.



(a) Number of heads (b) Size of attention factor (c) Activation functions (d) Number of layers

Figure 5: Impact of network hyper-parameters on Logloss performance.

- **AFM** [Xiao *et al.*, 2017]: AFM extends FM by using attention mechanism to distinguish the different importance of second-order feature interactions.
- **DeepFM** [Guo *et al.*, 2017]: The FM part is a factorization machine, and the deep part is a three-layer MLP with layer sizes 256, 256, 256.
- **xDeepFM** [Lian *et al.*, 2018]: xDeepFM considers the implicit and explicit modeling of high-order feature interactions simultaneously, which are implemented with the DNN and CIN components.
- **IFM** [Yu *et al.*, 2019]: We implement the method with the same structure of the original paper. The hidden dimension in Factor Estimating Network (FEN) is set to 256, 256, 256 as recommended.

Parameter Settings. We implement all models using Tensorflow[4]. To enable a fair comparison, all models are learned by optimizing the log loss (Equation 14) using the Adam (Learning rate: 0.001) optimizer [Kingma and Ba, 2014]. For all methods, the embedding sizes for Criteo and Avazu are set to 20 and 40, respectively, which is the same as the parameter settings in IFM [Yu *et al.*, 2019]. The batch size is set to 2000 for both datasets. The default setting for the number of neurons per layer is: (1) 256 for DNNs layers in DeepFM, xDeepFM, IFM and DIFM; (2) 200 for CIN layers in xDeepFM. For AFM, the attention factor is also set to 256 as the original paper recommended [Xiao *et al.*, 2017]. To be fair and achieve the best performance for each model, hyper-parameters of each model are tuned by grid-searching carefully. The best settings for each model have been shown in corresponding sections.

[4]https://www.tensorflow.org/

5.2 Hyper-parameter Study (RQ1)

First, we study the impact of hyper-parameters on DIFM, including (1) the number of attention heads n; (2) the attention key size d_k; (3) activation functions (the vector-wise part); (4) the number of hidden layers in DNNs.

To avoid the analysis being affected by the bit-wise part, we conducted experiments (1), (2), (3) via removing the DNNs (bit-wise) part while varying the settings for the vector-wise part. Subsequently, we conducted experiment (4) via holding the best settings for the vector-wise part while varying the settings for the bit-wise part.

Number of Attention Heads. Figures 4a and 5a demonstrate the impact of the number of self-attention heads. We can observe that model performance on two datasets increases steadily when the number of heads increases from 1 to 16. We also conducted experiments with heads greater than 16, but found that the improvement of further increasing head number is not significant, even the quality will drop off with too many heads. Consequently, we fixed the number of heads to 16 for the following experiments.

Size of Attention Factor. Adding the size of attention factor (d_k) equals increasing the number of feature maps in Eq.3. As shown in Figures 4b and 5b, model performance on Avazu dataset increases steadily when we increase the size of attention factor from 20 to 100, while on Criteo dataset, 80 is a more suitable setting for the size of attention factor. As such, To avoid the model being too complicated, we fixed the size of attention factor to 100 for Avazu and 80 for Criteo.

Activation Function. Note that we exploited *relu* as activation function on neurons of the vector-wise part, as shown in Eq.7. A common practice in deep learning literature is to em-

	Avazu		Criteo	
Model	AUC	Logloss	AUC	Logloss
FM	0.7758	0.3822	0.7837	0.5600
bit-wise	0.7849	0.3771	0.7981	0.5437
vector-wise-no-res	0.7851	0.3766	0.7990	0.5425
vector-wise	0.7853	0.3764	0.7992	0.5423
complete-no-res	0.7857	0.3762	0.7996	0.5418
complete	0.7860	0.3760	0.7998	0.5416

Table 1: The performance of different components in our DIFM. (Avazu: 40M instances and Criteo: 100M instances)

ploy non-linear activation functions on hidden neurons. We thus compared the performance of different activation functions on the vector-wise part (for neurons of the bit-wise part, we keep the activation function with *relu*). As shown in Figures 4c and 5c, *relu* is indeed more appropriate for neurons of the vector-wise part.

Depth of Network. Figures 4d and 5d demonstrate the impact of the number of hidden layers in DNNs. For Criteo dataset, the performance of DIFM increases with the depth of network at the beginning. However, the model performance starts to degrade when the depth of network is set to greater than 2. It is caused by overfitting evidenced by the observation that the training error still keeps decreasing. For Avazu dataset, when we stack more layers, the performance is not further improved, and the best performance is when we use only one hidden layer.

5.3 Ablation Study (RQ2)

The DIFM model strategically integrates the bit-wise and vector-wise levels for learning the input-aware factors into a unified end-to-end model. As such, whether it is indeed necessary and effective to combine them for joint prediction? Which is the most important component in our DIFM model? To validate and gain deep insights into the DIFM model, we conducted ablation experiments over DIFM. Table 1 shows the performance of DIFM $w.r.t.$ different components (i.e. different variants). We have the following key observations:

- First, introducing the input-aware factors mechanism for reweighting the original feature representations is effective, which can be verified through the fact that all variants of the DIFM model outperform FM by a large margin on both datasets.
- Second, it is necessary and effective to combine the bit-wise and vector-wise levels for joint prediction. Considering the bit-wise and vector-wise levels simultaneously for learning the input-aware factors brings additional improvement over the cases of considering either alone.
- Lastly, the vector-wise component is the best individual variant over DIFM, and the residual unit can further improve performance, which demonstrates the importance of preserving some information about the original embedding vector.

	Avazu		Criteo	
Model	AUC	Logloss	AUC	Logloss
LR	0.7505	0.3963	0.7766	0.5670
FM	0.7758	0.3822	0.7837	0.5600
FFM	0.7769	0.3813	0.7976	0.5443
AFM	0.7782	0.3805	0.7911	0.5512
NFM	0.7816	0.3785	0.7898	0.5528
IFM	0.7849	0.3771	0.7981	0.5437
DeepFM	0.7821	0.3782	0.7975	0.5446
xDeepFM	0.7830	0.3778	0.7979	0.5439
DIFM	**0.7860**	**0.3760**	**0.7998**	**0.5416**

Table 2: The performance comparison of different CTR models. (Avazu: 40M instances and Criteo: 100M instances)

5.4 Performance Comparison (RQ3)

In this final subsection, we compared our DIFM model with several strong baselines. The performance of different models on Avazu and Criteo datasets is shown in Table 2, from which we have the following key observations:

- First, learning feature interactions improves the performance of CTR prediction models. This observation is from the fact that LR performs far worse than all the rest models. Meanwhile, learning the low-order and high-order feature interactions endows the CTR prediction model better representation ability.
- Second, learning adaptive and different representations of a given feature according to different input instances is crucial in CTR prediction, which can be verified through the result that the shallow DIFM and IFM models achieve even better performance than deep learning methods, such as AFM, NFM, DeepFM and xDeepFM.
- Lastly, our proposed DIFM model consistently achieves the best performance on both datasets. This demonstrates the effectiveness and rationality of the combination of the bit-wise and vector-wise in the DIFM model.

6 Conclusion

In this paper, we presented a novel model named *Dual Input-aware Factorization Machines* (DIFMs) for CTR prediction. It aims to adaptively learn flexible representations of a given feature according to different input instances with the help of the Dual-Factor Estimating Network (Dual-FEN). The major advantage of DIFM is that it can effectively learn the *input-aware factors* (used to reweight the original feature representations) not only at the bit-wise level but also at the vector-wise level simultaneously. Extensive experiments on two real-world CTR prediction datasets were conducted to verify the effectiveness of DIFM for CTR prediction. In terms of two popular evaluation metrics, empirical results indicate that DIFM can outperform the classical LR, FM, FFM and several major state-of-the-art deep learning methods such as AFM, NFM, IFM, DeepFM and xDeepFM consistently.

References

[Amodei *et al.*, 2016] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*, pages 173–182, 2016.

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

[Cho *et al.*, 2014] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[Guo *et al.*, 2017] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: a factorization-machine based neural network for ctr prediction. *arXiv preprint arXiv:1703.04247*, 2017.

[He and Chua, 2017] Xiangnan He and Tat-Seng Chua. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 355–364. ACM, 2017.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on CVPR*, pages 770–778, 2016.

[Juan *et al.*, 2016] Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. Field-aware factorization machines for ctr prediction. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 43–50. ACM, 2016.

[Kingma and Ba, 2014] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Lee *et al.*, 2012] Kuang-chih Lee, Burkay Orten, Ali Dasdan, and Wentong Li. Estimating conversion rate in display advertising from past erformance data. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 768–776. ACM, 2012.

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

[McMahan *et al.*, 2013] H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, et al. Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1222–1230. ACM, 2013.

[Qu *et al.*, 2016] Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. Product-based neural networks for user response prediction. In *2016 IEEE 16th ICDM*, pages 1149–1154. IEEE, 2016.

[Rendle, 2010] Steffen Rendle. Factorization machines. In *2010 IEEE ICDM*, pages 995–1000. IEEE, 2010.

[Song *et al.*, 2019] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *Proceedings of the 28th CIKM*, pages 1161–1170. ACM, 2019.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[Wang *et al.*, 2017] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*, page 12. ACM, 2017.

[Wen *et al.*, 2019] Hong Wen, Jing Zhang, Quan Lin, Keping Yang, and Pipei Huang. Multi-level deep cascade trees for conversion rate prediction in recommendation system. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 338–345, 2019.

[Xiao *et al.*, 2017] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. Attentional factorization machines: Learning the weight of feature interactions via attention networks. *arXiv preprint arXiv:1708.04617*, 2017.

[Yu *et al.*, 2019] Yantao Yu, Zhen Wang, and Bo Yuan. An input-aware factorization machine for sparse prediction. In *Proceedings of the 28th IJCAI*, pages 1466–1472. AAAI Press, 2019.

[Zhang *et al.*, 2016] Weinan Zhang, Tianming Du, and Jun Wang. Deep learning over multi-field categorical data. In *European conference on information retrieval*, pages 45–57. Springer, 2016.

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

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