FLEN: Leveraging Field for Scalable CTR Prediction

Wenqiang Chen[1], Lizhang Zhan[2], Yuanlong Ci[1], Chen Lin[3]

[1]Data Intelligence, Meitu Inc.
[2]Advertisement Recommendation Platform, Technology Engineering Group, Tencent Inc.
[3]Department of Computer Science, Xiamen University, Xiamen, Fujian, China
{cwq1, cyl4}@meitu.com, lizhangzhan@tencent.com, chenlin@xmu.edu.cn

Abstract

Click-**T**hrough **R**ate (CTR) prediction has been an indispensable component for many industrial applications, such as recommendation systems and online advertising. CTR prediction systems are usually based on multi-field categorical features, i.e., every feature is categorical and belongs to one and only one field. Modeling feature conjunctions is crucial for CTR prediction accuracy. However, it requires a massive number of parameters to explicitly model all feature conjunctions, which is not scalable for real-world production systems.

In this paper, we describe a novel **F**ield-**L**everaged **E**mbedding **N**etwork (**FLEN**) which has been deployed in the commercial recommender system in Meitu and serves the main traffic. FLEN devises a field-wise bi-interaction pooling technique. By suitably exploiting field information, the field-wise bi-interaction pooling captures both inter-field and intra-field feature conjunctions with a small number of model parameters and an acceptable time complexity for industrial applications. We show that a variety of state-of-the-art CTR models can be expressed under this technique. Furthermore, we develop **Dicefactor**: a dropout technique to prevent independent latent features from co-adapting.

Extensive experiments, including offline evaluations and online A/B testing on real production systems, demonstrate the effectiveness and efficiency of proposed approaches. Notably, FLEN has obtained 5.19% improvement on CTR with $1/6$ of memory usage and computation time, compared to last version (i.e. NFM).

1 Introduction

Click-**T**hrough **R**ate (CTR) prediction is the task of predicting the probabilities of users clicking items or ads. It is a critical problem in recommender systems and online advertising, which provide substantial revenue for Internet companies. As such, CTR prediction has attracted much attention from both academia and industry communities in the past few years (Chapelle, Manavoglu, and Rosales, 2015; McMahan et al., 2013; Graepel et al., 2010).

The data in CTR prediction task is *multi-field categorical data*, i.e., every feature is categorical and belongs to one and only one field. For example, feature "gender=Female" belongs to field "gender", feature "age=24" belongs to field

"age" and feature "item category=cosmetics" belongs to field "item category". The value of feature "gender" is either "male" or"female. Feature "age" is discretized to several age groups: "0-18", "18-25", "25-30", and so on. It is well regarded that, feature conjunctions are essential for accurate CTR prediction (Cheng et al., 2016; Covington, Adams, and Sargin, 2016; He and Chua, 2017; Lian et al., 2018; Guo et al., 2017). An example of informative feature conjunctions is: age group "18-25" combined with gender "female" for item category "cosmetics". It indicates that young girls are more likely to click on cosmetic products.

Modeling sparse feature conjunctions has been continually improved and refined by a number of prior work, including shallow and deep models. Shallow models such as Factorization Machines (FM) (Rendle, 2010) and a lot of inspired extensions, including FFM (Juan, Lefortier, and Chapelle, 2017) and FwFM (Pan et al., 2018). FFM and FwFM explicitly model field-aware feature interactions and have achieved promising results. However, *the challenge in real-world online advertising or recommender system is the strict latency limit at serving time and the scalability for high-dimensionality of features.* The model complexity of FFM and FwFM is $O(N^2)$, where N is the number of features. The drawback of directly applying FFM and FwFM in real-world applications is the dramatically increased use of computational resources. Any uniform increase in the number of features results in a quadratic increase of computation. In real-world system we need to predict hundreds of items for each user in less than 10 milliseconds. Moreover, FFM is restricted by space complexity, thus is impractical. Deep models such as NFFM (Yang et al., 2019) combines FFM and MLP to capture operation-aware feature conjunctions with additional parameters which suffers the same limitation as FFM.

In this paper we describe a novel **F**ield-**L**everaged **E**mbedding **N**etwork (FLEN) which has been successfully deployed in the online recommender system in Meitu, serving the main traffic. FLEN devises a new operation in neural network modeling Field-wise Bilinear Interaction (FwBI) pooling to address the restriction of time and space complexity when applying field-aware feature conjunctions in real industrial system. The field-wise bi-interaction pooling technique is based on the observation that features from various fields interact with each other differently while taking inspi-

ration and guidance from the Inception architecture for computer vision (Szegedy et al., 2016). By encoding field information in a factorized FwFM component, FwBI, FLEN dramatically reduces model complexity. It is worth pointing out that, to our knowledge, this is the first time a variety of state-of-the-art shallow models, including FM (Rendle, 2010), MF (Koren, Bell, and Volinsky, 2009), and FwFM (Pan et al., 2018), have been expressed in a unified framework.

Deep and shallow models that represent features in a latent vector space encounter the coupled gradient issue (Qu et al., 2018), i.e. two supposedly independent features are updated in the same direction during the gradient update process. In FLEN, this issue is partly tackled by leveraging field information. We also propose a novel dropout technology: Dicefactor to decouple independent features. Dicefactor randomly drops bi-linear paths (i.e. cross-feature edge in the field-wise bi-interaction component) to prevent a feature from adapting to other features.

To demonstrate the effectiveness and efficiency of FLEN, we conduct extensive offline evaluations and online A/B testing. In offline evaluations, FLEN outperforms state-of-the-art methods on both a well-known benchmark and an internal dataset consisting of historical click records collected in our system. Online A/B testing shows that FLEN enhances the CTR prediction accuracy (i.e. increases CTR by 5.19%) with a fraction of computation resources (i.e. $1/6$ memory usage and computation time), compared with the last version of our ranking system (i.e. NFM (He and Chua, 2017)).

2 Related work

CTR prediction has been extensively studied in the literature, as online advertising systems have become the financial backbone of most Internet companies. Related literature can be roughly categorized into shallow and deep models.

2.1 Shallow Models

Successful shallow models represent features as latent vectors. For example, matrix factorization (MF) (Koren, Bell, and Volinsky, 2009) is successfully applied in recommendation systems. It factorizes a rating matrix into a product of lower-dimensional sub-matrix, where each sub-matrix is the latent feature space for users and items. Factorization machines (FM) (Rendle, 2010) is a well-known model to learn feature interactions. In FM, the effect of feature conjunction is explicitly modeled by inner product of two latent feature vectors (a.k.a. embedding vectors). Many variants have been proposed based on FM. For example, Rendle et al. (2011) proposed a context-aware CTR prediction method which factorized a three-way $< user, ad, context >$ tensor. Oentaryo et al. (2014) developed hierarchical importance-aware factorization machine to model dynamic impacts of ads.

Field information has been acknowledged as crucial in CTR prediction. A number of recent work has exploited field information. For example, Field-aware Factorization Machines (FFM) (Juan, Lefortier, and Chapelle, 2017) represents a feature based on separate latent vectors, depending on the multiplying feature field. GBFM (Cheng et al., 2014) and AFM (Xiao et al., 2017) considered the importance of different field feature interactions. Field-weighted Factorization Machines (FwFM) (Pan et al., 2018) assigns interaction weights on each field pair.

The field-wise bi-interaction technique of FLEN is inspired by FwFM. It can be viewed as a special case of factoring FwFM in a computationally efficient manner. However, the field-wise bi-interaction technique of FLEN generalizes and ensembles MF, FM and FwFM. Furthermore, shallow models are limited as they focus on modeling linear, low-order feature interactions. FLEN is capable of capturing not only low-order but also high-order, nonlinear interactions.

2.2 Deep Models

An increased interest in designing deep models for CTR prediction has emerged in recent years. The majority of them utilize feature bi-interactions. To name a few, NFM (He and Chua, 2017) generalizes FM by stacking neural network layers on top of a bi-interaction pooling layer. The architecture of DeepFM (Guo et al., 2017) resembles with Wide&Deep (Cheng et al., 2016), which also has a shared raw feature input to both its "wide" (i.e. for bi-interaction) and "deep" (i.e. for high-order interaction) components. DCN (Wang et al., 2017) learns certain bounded-degree feature interactions. xDeepFM (Lian et al., 2018) improves over DeepFM and DCN by generating feature interactions in an explicit fashion and at the field-wise level. NFFM (Yang et al., 2019) learns different feature representations for convolutional operations and product operations. However, the space complexity of NFFM and the time complexity of xDeepFM restrict them from applying in industrial systems. FGCNN (Liu et al., 2019) leverages the strength of CNN to generate local patterns and recombine them to generate new features. Then deep classifier are built upon the augmented feature space. PIN (Qu et al., 2018) generalizes the kernel product of feature bi-interactions in a net-in-net architecture.

The rest of literature learns the high-order feature interactions in an implicit way, e.g. PNN (Qu et al., 2018) FNN (Zhang, Du, and Wang, 2016), DeepCrossing (Shan et al., 2016), and so on. These approaches lack explainability. Some tree-based methods (Zhu et al., 2017; Wang et al., 2018) combine the power of embedding-based models and tree-based models to boost explainability. One drawback of these approaches is having to break training procedure into multiple stages. A recent work FPENN (Liu et al., 2018) also group feature embedding vectors based on field information in deep neural network structure. It estimates the probability distribution of the field-aware embedding rather than using the single point estimation (the maximum a posteriori estimation) to prevent overfitting. However, as FPENN assigns several latent vectors to each field (i.e. one for a field which is not equivalent as the multiplying field), it requires much more model parameters than FLEN.

3 Model

An overview of the model architecture is illustrated in Figure 1. Suppose there are a set of samples $\mathcal{X} = \{\mathbf{x}^s\}$, FLEN
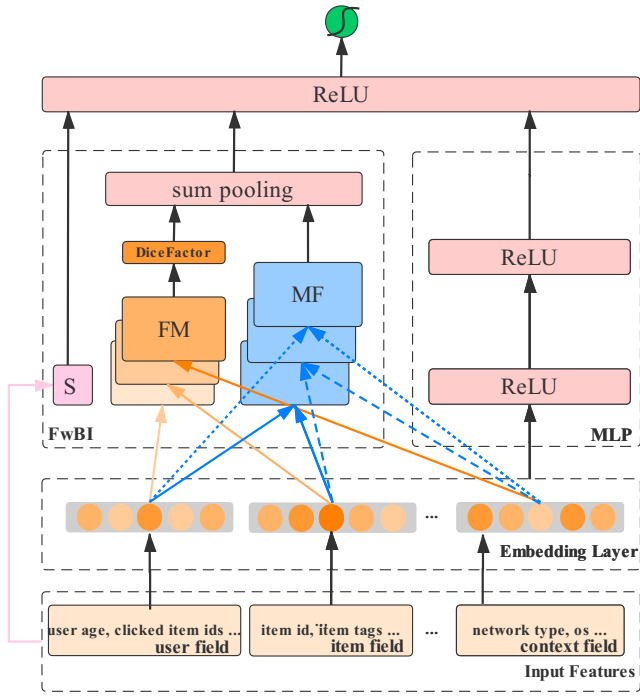
Figure 1: Architecture overview of FLEN

takes as input the s-th sample $\mathbf{x}^s$, which is a vector of features $\mathbf{x}^s = [\mathbf{x}_1^s, \cdots, \mathbf{x}_N^s]$, where $\mathbf{x}_n^s \in \mathcal{R}^{K_n}$ represents a categorical feature (details described in Section 3.1). First, all features pass through an embedding layer, which outputs a concatenation of field-wise feature embedding vectors $\mathbf{e}^s = [\mathbf{e}_1^s, \cdots, \mathbf{e}_M^s]$, where $\mathbf{e}_m^s \in \mathcal{R}^{Ke}$ represents a top level field (details described in Section 3.2). Then the embedding vectors flow a Field-Wise Bi-Interaction component (FwBI) and an MLP component. The field-wise bi-interaction component (Sec 3.3) consists of three submodules which capture all single and field-wise feature interactions (degree one or two) and outputs $\mathbf{h}_{FwBI} \in \mathcal{R}^{Ke+1}$. We show that previous CTR predictions such as MF (Koren, Bell, and Volinsky, 2009), FM (Rendle, 2010) and FwFM (Pan et al., 2018) can be expressed and generalized under the proposed framework. Alternatively, the component can be regarded as a combination of single feature, MF-based inter-field feature interactions and FM-based intra-field feature interactions.

The MLP component captures non-linear, high-order feature interactions (Details described in Section 3.5). The output of field-wise bi-interaction component and output of MLP component are concatenated to feed the last prediction layer (Section 3.6).

Hereafter, unless stated otherwise, we use lower-case letters for indices, upper-case letters for universal constants, lower-case bold-face letters for vectors and upper-case bold-face letters for matrices, calligraphic letters for sets. We use square brackets to denote elements in a vector or a matrix, e.g. $x[j]$ denotes the $j-$th elements of $\mathbf{x}$. We will omit superscripts whenever no ambiguity results.

3.1 Feature Representation

It is natural to represent categorical features as one-hot or multi-hot vectors. Suppose there are N unique features and each feature n has K_n unique values, we represent each instance as $\mathbf{x} = [\mathbf{x}_1, \cdots, \mathbf{x}_N]$, where $\mathbf{x}_n \in \mathcal{R}^{K_n}$. $x_n[j] \in \{0, 1\}, j = 1, \cdots, K_n$. A non-zero entry $x_n[j] = 1$ indicates that the corresponding feature value in instance x falls in the category j.

Suppose there are M fields, $F(n)$ denotes the field of feature n, we organize the feature representations in a field-wise manner for complexity reduction. Specifically, $\mathbf{x} = concat(\mathbf{f}_1, \cdots, \mathbf{f}_M)$, where $\mathbf{f}_m$ is the concatenation of feature vectors in field m, i.e. $\mathbf{f}_m = concat(\mathbf{x}_n|F(n) = m)$. The organized instance is illustrated in the example below. Note that there is a hierarchical structure of fields. For example, "item tags field" and "item id field" belong to the more general "item field". In practice we also use the top level fields (Covington, Adams, and Sargin, 2016) (namely user field, item field and context field) and achieve maximal performance enhancement and complexity reduction. A practically useful aspect of this hierarchical design is that it aligns with the intuition that the output of top level fields are highly inter-correlated.

$$\underbrace{\underbrace{[0,1,0,...,0]}_{age\ field} ... \underbrace{[\ 1,0\ ]}_{gender\ field}}_{user\ field} \underbrace{\underbrace{[0,1,0,...,0]}_{item\ id\ field} \underbrace{[0,1,0,1,...,0]}_{item\ tags\ field}}_{item\ field}$$

3.2 Embedding Layer

Since the feature representations of the categorical features are very sparse and high-dimensional, we employ an embedding procedure to transform them into low dimensional, dense real-value vectors. Firstly, we transform each feature $\mathbf{x}_n$ to $\mathbf{f}_n$.

$$\mathbf{f}_n = \mathbf{V}_n \mathbf{x}_n, \tag{1}$$

where $\mathbf{V}_n \in \mathcal{R}^{K_e \times K_n}$ is an embedding matrix for the corresponding feature that will be optimized together with other parameters in the network. Note that feature size can be various.

Next we apply sum-pooling to $\mathbf{f}_n$ to obtain the field-wise embedding vectors.

$$\mathbf{e}_m = \sum_{n|F(n)=m} \mathbf{f}_n \tag{2}$$

Finally, we concatenate all field-wise embedding vectors to build $\mathbf{e}$, as illustrated in Figure 2.
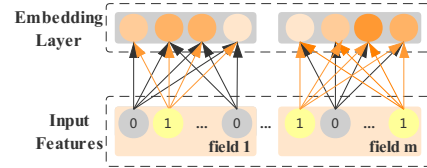

Figure 2: Illustration of embedding layer with $K_e = 4$.

3.3 Field-Wise Bi-Interaction Component

The field-wise bi-interaction component learns a mapping $\Phi_{FwBI}(\mathbf{W}_{FwBI}, \mathbf{R}) : (\mathcal{R}^N, \mathcal{R}^{MK_e}) \rightarrow \mathcal{R}^{K_e+1}$. There are three sub-module in this component.

The first sub-module (denoted as S) is a sub-network with one hidden layer and one output neuron. It operates on the feature representation vectors, i.e. $\mathbf{h}_S = \sum_{i=0}^{\sum_n^N K_n} w[i]x[i]$ where $x[0] = 1, w[0]$ is bias term and $K_0 = 1$.

The second sub-module is called an MF module, which focuses on learning inter-field feature interactions between each pair of top level fields. It first operates element-wise product on all pairs of field-wise embedding vectors, i.e. $\mathbf{h}_{MF} = \sum_{i=1}^{M} \sum_{j=i+1}^{M} \mathbf{e}_i \odot \mathbf{e}_j r[i][j]$. Where $r[i][j] \in \mathcal{R}$ is a weight to model the interaction strength between field i and j. We use $\odot$ to denote the element-wise product of two vectors, that is, $(\boldsymbol{v}_i \odot \boldsymbol{v}_j)[k] = \boldsymbol{v}_i[k]\boldsymbol{v}_j[k]$.

In real industrial system, M is usually less than 4 for reducing computation and avoiding overfitting. As shown in Figure 1, there are 3 MF models learning field-wise feature interactions on all pairs of top level fields.

The third sub-module is called an FM module, which focuses on learning intra-field feature interactions in each MF module. It first computes self element-wise product on each field embedding vectors, i.e. $\mathbf{hf}_m = \mathbf{e}_m \odot \mathbf{e}_m$. Similar operations are also conducted on each feature embedding vectors, i.e. $\mathbf{ht}_m = \sum_{n, F(n)=m} \mathbf{f}_n \odot \mathbf{f}_n$. Finally, the two vectors are merged over all field-wise subtraction of the two intermediate vectors, i.e. $\mathbf{h}_{FM} = \sum_m (\mathbf{hf}_m - \mathbf{ht}_m)r[m][m]$. Where $r[m][m] \in \mathcal{R}$ is a weight for each field m, discriminating the importance of each field that contribute to the final prediction.

Clearly, the output of MF module and FM module is a K_e-dimension vector that encodes the inter-field and intra-field feature interactions in the embedding space, respectively.

We concatenate the output of S and the sum pooling of MF and FM module, i.e. $\mathbf{h}_{in} = [\mathbf{h}_S, \mathbf{h}_{MF} + \mathbf{h}_{FM}]$. Then $\mathbf{h}_{in}$ is fed to a hidden layer, i.e. $\mathbf{h}_{FwBI} = \sigma(\mathbf{W}_{FwBI}^T \mathbf{h}_{in})$. In practice, we use the ReLU as the active function σ.

By leveraging multiple MF and FM modules to learn both the inter-field and intra-field feature interactions, we end up with more disentangled parameters and therefore with faster training. This aligns with the guiding principles in Inception that applies multiple "Inception modules" to learn disentangled features. Furthermore, the parallel structures of FwBI allows the computational budget be distributed in a distributed environment.

It is worth pointing out that the FwBI component is much more memory-efficient than FFM and furthermore, it can be efficiently computed in $O(MK_eN + K_eM^2)$. In real industrial system, M is usually less than 4. When $M \ll N$, FwBI can be efficiently trained and serverd online in linear time, which is attractive in industrial systems.

3.4 Relation to Previous CTR Prediction Systems

A variety of shallow models can be expressed and generalized under the field-wise bi-interaction technique. To start with, we set the active function σ as an identify function, the weight matrix $\mathbf{W}_{FwBI}^T = \mathbf{I}$, where $\mathbf{I}$ is a unit matrix.

Since the embedding vectors $\mathbf{e}, \mathbf{f}$ are transformed from the original feature representations, $\mathbf{e}_m = \sum_{n|F(n)=m} \mathbf{V}_n \mathbf{x}_n$, we can re-write the computation in field-wise bi-interaction layer as:

$$
\Phi_{FwBI} = \mathbf{W}_{FwBI}^T \Big[\underbrace{w_0 + \sum_{i=1}^{\sum_n^N K_n} w[i]x[i]}_{S}
$$
$$
, \underbrace{\sum_{i=1}^{M} \sum_{j=i+1}^{M} \big[(\sum_{n,F(n)=i} \mathbf{V}_i\mathbf{x}_n)(\sum_{n,F(n)=j} \mathbf{V}_j\mathbf{x}_n)r[i][j] \big]}_{MF}
$$
$$
+ \underbrace{\sum_m^M [(\sum_{n,F(n)=m} \mathbf{V}_n\mathbf{x}_n)^2 - (\sum_{n,F(n)=m} (\mathbf{V}_n\mathbf{x}_n)^2)]r[m][m]}_{FM} \Big]
$$
$$(3)$$

where we use the symbol $(\mathbf{Vx})^2$ to denote $\mathbf{Vx} \odot \mathbf{Vx}$. The first term in Equation 3 corresponds to the first sub-module which is based on single feature values. Note that we use indicator indexing, i.e. $x_i, i = 1, \cdots, \sum_n^N K_n$ is the binary value indictor for all possible values in all features n. The second term corresponds to the sum pooling over the second sub-module, which resembles the matrix factorization form when written in feature representations, and the third sub-module, which resembles the factorization machine form.

Clearly, if there are only one field, i.e., $M = 1$ and $r_{m,m} = \frac{1}{2}$, then we can exactly recover the FM model.

$$
\Phi_{FwBI} = \Phi_{FM}
$$
$$
= \Big[w_0 + \sum_{i=1}^{N} w[i]x[i], \frac{1}{2}[(\sum_{i=1}^{N} v[i]x[i])^2 - \sum_{i=1}^{N} ((v[i]x[i])^2)] \Big]
$$
$$(4)$$

If there are N fields, i.e., $M = N$, there is not any intra-field feature interactions, then we can recover the FwFM model.

$$
\Phi_{FwBI} = \Phi_{FwFM}
$$
$$
= \Big[w_0 + \sum_{i=1}^{M} w[i]x[i], \sum_{i=1}^{M} \sum_{j>i}^{M} (v[i]x[i])(v[j]x[j])r[i][j] \Big]
$$
$$(5)$$

3.5 MLP component

We employ an MLP component to capture non-linear, high-order feature interactions. The input is simply a concatenation of all field-wise embedding vectors, i.e. $\mathbf{h}_0 = concat(\mathbf{e}_1, \cdots, \mathbf{e}_M)$. A stack of fully connected layers is constructed on the input $\mathbf{h}_0$. Formally, the definition of fully connected layers are as follows:

$$
\begin{aligned}
\mathbf{h}_1 &= \sigma_1(\mathbf{W}_1\mathbf{h}_0 + \mathbf{b}_1), \\
\mathbf{h}_2 &= \sigma_2(\mathbf{W}_2\mathbf{h}_1 + \mathbf{b}_2), \\
&\cdots\cdots \\
\mathbf{h}_L &= \sigma_L(\mathbf{W}_L\mathbf{h}_{L-1} + \mathbf{b}_L),
\end{aligned}
$$
$$(6)$$

where L denotes the number of hidden layers, $\mathbf{W}_l$, $\mathbf{b}_l$ and σ_l denote the weight matrix, bias vector and activation function for the l-th layer, respectively. We use ReLU as the active function for each layer.

3.6 Prediction Layer

The output vector of the last hidden MLP layer $\mathbf{h}_L$ is concatenated with the output vector of field-wise bi-interaction component Φ_{FwBI} to form $\mathbf{h}_F = concat(\mathbf{h}_{FwBI}, \mathbf{h}_L)$. The concatenation $\mathbf{h}_F$ goes through one last hidden layer and is transformed to

$$z = \sigma(\mathbf{w}_F^T \mathbf{h}_F), \quad (7)$$

where vector $\mathbf{w}_F$ denotes the neuron weights of the final hidden layer.

Finally, we apply a sigmoid layer to make predictions.

$$\sigma(\mathbf{z}) = \frac{1}{1 + e^z} \quad (8)$$

Our loss function is *negative log-likelihood*, which is defined as follows:

$$\mathbb{L} = -\frac{1}{T} \sum_{s=1}^{|\mathcal{X}|} (y^s log(\Phi(\mathbf{x}^s)) + (1 - y^s) log(1 - \Phi(\mathbf{x}^s))), \quad (9)$$

where $y^s \in \{0, 1\}$ as the label, $\Phi(\mathbf{x}^s)$ is the output of the network, representing the estimated probability of the instance $\mathbf{x}^s$ being clicked. The parameters to learn in our model are represented as $\mathbf{V}, \mathbf{W}, \mathbf{R}$, which are updated via minimizing the total negative log-likelihood using gradient descent.

4 Dicefactor: Dropout Technique

Dropout (Srivastava et al., 2014) is a simple yet effective regularization to prevent DNNs from overfitting. The idea behind Dropout is that it provides an efficient way to train and ensemble exponentially many neural networks. Inspired by this technique, we propose a specific *DiceFactor* method in our FLEN model.

We first reformulate Eq. (4) as:

$$\Phi_{FM} = \left[w_0 + \sum_{i=1}^{N} w[i]x[i], \sum_{i=1}^{M} \sum_{j=1 \& i \neq j}^{M} \mathbf{e}_i \odot \mathbf{e}_j \right], \quad (10)$$

where $\mathbf{e}_i$ represents the i-th field's embedding vector. Based on Eq. (10), Fig. 3 shows the expanding structure of the bi-linear interaction in two field embedding vectors. There are K_e bi-linear paths, each bi-linear path connects corresponding elements in two embedding vectors. The key idea of DiceFactor is to randomly drop the bilinear paths during the training. This prevents $\mathbf{e}_i$ from co-adapting to $\mathbf{e}_j$, i.e., $\mathbf{e}_i$ is updated through the direction of $\mathbf{e}_j$.

In our implementation, each factor is retained with a predefined probability β during training. With the DiceFactor, the formulation of bi-linear interaction of FM part in the training becomes:

$$\Phi_{FM} = \left[w_0 + \sum_{i=1}^{N} w[i]x[i], \sum_{i=1}^{M} \sum_{j=1 \& i \neq j}^{M} \mathbf{p}\mathbf{e}_i \odot \mathbf{e}_j \right], \quad (11)$$

where $\mathbf{p} \in \mathcal{R}^{K_e}$ and $p[i] \sim Bernoulli(\beta)$. With the DiceFactor, the network can be seen as a set of 2^{K_e} thinned networks with shared weights. In each iteration, one thinned network is sampled randomly and trained by back-propagation as shown in Fig. 3a.
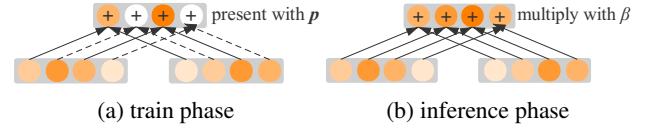

(a) train phase   (b) inference phase

Figure 3: Dicefactor keeps a bi-linear path with probability β at train phase. At inference phase, every bi-linear path is kept and the output is multiplied by β.

For inference, instead of explicitly averaging the outputs from all 2^{K_e} thinned networks, we use the approximate "Mean Network" scheme in (Srivastava et al., 2014). As shown in Fig. 3b, each factor term $x_i \boldsymbol{v}_i \odot x_j \boldsymbol{v}_j$ is multiplied by β at inference phase:

$$\Phi_{FM} = \left[w_0 + \sum_{i=1}^{N} w[i]x[i], \sum_{i}^{M} \sum_{j \& i \neq j}^{M} \beta \mathbf{e}_i \odot \mathbf{e}_j \right] \quad (12)$$

In this way, the output of each neuron at inference phase is the same as the expectation of output of 2^{K_e} different networks at train phase.

5 Offline Model Evaluation

To assess the validity of our CTR prediction models, we run a traditional offline evaluation based on a well-known public benchmark and an internal dataset.

Avazu[1] The first dataset we adopt is originally used in the Kaggle CTR prediction competition. It contains users' mobile behaviors, i.e. whether a displayed mobile ad is clicked by a user. It has 23 feature fields spanning from user/device features to ad attributes. The data set is collected during a time span of 10 days. We use 9 days of clicks for training and the last 1 day of data for test. Unlike competition, we take more attention to the model rather than features. So we simply use the original 22 features besides id and device ip.

Meitu The second dataset used to run the experiments is a uniformly generated sample of our internal historical data. As a training set we extract a sample of one million items shown during a seven-day time period from 2019-07-26 to 2019-08-01 to users of a photo and video-sharing social networking app, namely Meitu. We collect 70 millions users' click records. The test set contains a sample of half one million items shown the next day, i.e. 2019-08-02. There are around two million features (e.g., user age, clicked feed ids, and etc) organized in a hierarchy of fields, with top level fields "user", "item" and "context". Some features used in our system are described in Table 1.

The statistics of the data sets are summarized in Table 2.

Table 1: Example features in Meitu dataset.

Field	Feature	Dimemsionality	Type	AverageNonzero Ids per Instance
User Field	gender	2	one-hot	1
	age	~ 10	one-hot	1
	clicked_feed_ids	$\sim 10^7$	multi-hot	$\sim 10^2$
	liked_feed_ids	$\sim 10^5$	multi-hot	$\sim 10^2$
Item Field	feed_id	$\sim 10^7$	one-hot	1
	tags	$\sim 10^5$	multi-hot	$\sim 10^1$
	clicked_rate	10	one-hot	10
	liked_rate	10	one-hot	10
	author_id	$\sim 10^7$	one-hot	1
	author_gender	2	one-hot	1
	author_age	~ 10	one-hot	1
	author_clicked_rate	10	one-hot	10
	author_liked_rate	10	one-hot	10
Context Field	network_type	~ 4	one-hot	1
	time	~ 10	one-hot	1
	brand	~ 15	one-hot	1

Table 2: Statistics of evaluation data sets.

Data	#Samples	#Fields	#Features (Sparse)
Avazu	40,428,967	22	32,699
Internal	208,149,301	33	2,051,310

5.1 Competitors

We compare FLEN with 7 state-of-the-art models.
(1) **FFM** (Juan, Lefortier, and Chapelle, 2017): a shallow model in which the prediction is aggregated over inner products of feature vectors. It represents a feature by several separate vectors, depending on the multiplying feature field.
(2) **FwFM** (Pan et al., 2018): a shallow model which also explicitly aggregates over feature products. Interaction weights are assigned for each field pair.
(3) **DCN** (Wang et al., 2017): a deep model that takes the outer product of feature vectors at bit-wise level to a feedforward neural network.
(4) **DeepFM** (Guo et al., 2017): a deep model that consists of a wide component that models factorization machine and a deep component.
(5) **NFM** (He and Chua, 2017): a deep model which stacks MLP on top of a bi-interaction pooling layer.
(6) **xDeepFM** (Lian et al., 2018): a deep model that explicitly generates features with a compressed interaction network.
(7) **NFFM**(Yang et al., 2019): a deep model which learns different feature representations for convolutional operations and product operations.

All methods are implemented in TensorFlow[2]. We use an embedding dimension of 32 and batch size of 512 for all compared methods. Hidden units d' are set to 64,32. We use AdaGrad (Duchi, Hazan, and Singer, 2011) to optimize all deep neural network-based models. DCN has two interaction layers, following by two feed-forward layers. We use one hidden layer of size 200 on top of Bi-Interaction layer for NFM as recommended by their paper. We use 2 layers with size $(64, 32)$ in the MLP component of FLEN. All experiments are executed on one NVIDIA TITAN Xp Card with 128G memory.

---

[2]Codes are available at https://github.com/aimetrics/jarvis

5.2 Evaluation Metrics

We use two commonly adopted evaluation metrics.

AUC Area Under the ROC Curve (AUC) measures the probability that a CTR predictor will assign a higher score to a randomly chosen positive item than a randomly chosen negative item. A higher AUC indicates a better performance.

Logloss Since all models attempt to minimize the *Logloss* defined by Equation 9, we use it as a straightforward metric.

It is now generally accepted that increase in terms of AUC and Logloss at 0.001-level is significant (Cheng et al., 2016; Guo et al., 2017; Wang et al., 2017).

5.3 Comparative Performance

We report the AUC and Logloss performance of different models in Table 3. We distinguish the original FLEN (denoted as FLEN) and the model with Dicefactor implementation (denoted as FLEN+D). We can see that on both datasets, FLEN has achieved the best performance in terms of AUC and Logloss. We point out that FLEN has impressively boosted the AUC performance of the best competitor (i.e. NFFM) by 0.002 on the Internal dataset, which validates the superiority of FLEN on large-scale CTR systems. We also observe that Dicefactor further significantly enhances AUC performance of FLEN on both datasets.

Furthermore, we can find an interesting observation: leveraging field information makes modeling feature interactions more precisely. This observation is derived from the fact that by exploiting field information, FLEN, NFFM and xDeepFM perform better than NFM, DeepFM and DCN do on the Avazu and Meitu datasets. This phenomenon can be found in more literature, including FFM (Juan, Lefortier, and Chapelle, 2017) and FwFM (Pan et al., 2018).

Table 3: AUC and Logloss performance of different models

Model	Avazu		Meitu	
	AUC	Logloss	AUC	Logloss
FFM	0.7400	0.3994	0.6300	0.5625
FwFM	0.7406	0.3988	0.6306	0.5621
DCN	0.7421	0.3981	0.6337	0.5606
DeepFM	0.7438	0.3982	0.6329	0.5612
NFM	0.7449	0.3973	0.6359	0.5596
xDeepFM	0.7509	0.3947	0.6440	0.5576
NFFM	0.7513	0.3945	0.6443	0.5565
FLEN	**0.7519**	**0.3944**	**0.6463**	**0.5558**
FLEN+D	**0.7528**	**0.3944**	**0.6475**	**0.5554**

## 5.4 Memory Consumption and Running Time

To illustrate the scalability of FLEN, we first compare the model complexity and actual parameter size on Avazu dataset of each method in Table 4 . For a fair comparison, in computing parameter size, we assume the number of deep layers denoted as $H = 3$, the number of hidden layers denoted as $L = 3$ and embedding size $K_e = 32$. We do not take into account parameters for the feed-forward neural network. The number of features is N=32,699 and the number of fields is $M = 3$ on Avazu dataset. We can see that FLEN

is one of the models that make use of the smallest number of parameters. This is because FLEN organizes features in a field manner, thus the parameter size depends on the number of fields M, which is usually less than 4 in practice.

Table 4: Model complexity and parameter size in Avazu dataset for different models

.

Model	Model Complexity	Parameter Size
FFM	$O(NMK_e)$	2.30×10^7
FwFM	$O(NK_e + M^2)$	1.05×10^6
DCN	$O(NK_e + MK_eL + MK_eH)$	1.05×10^6
DeepFM	$O(NK_e + MK_eH)$	1.05×10^6
NFM	$O(NK_e + K_eH)$	1.05×10^6
xDeepFM	$O(NK_e + MH^2L + MK_eH)$	1.11×10^6
NFFM	$O(NMK_e)$	2.30×10^7
FLEN	$O(NK_e + M^2 + MK_eH)$	1.05×10^6

For a detailed study, we report the number of instances being processed by different models per second on the two datasets. As shown in Figure 4, FLEN operates on the most instances on Meitu dataset. On Avazu dataset, FLEN is comparable with other state-of-the-art methods. The high efficiency makes FLEN applicable in real industrial systems to handle large scale and high-dimensional data.
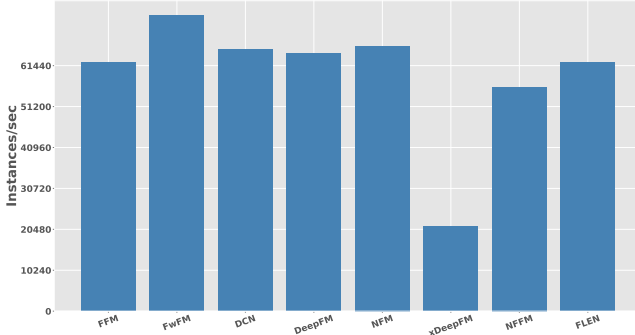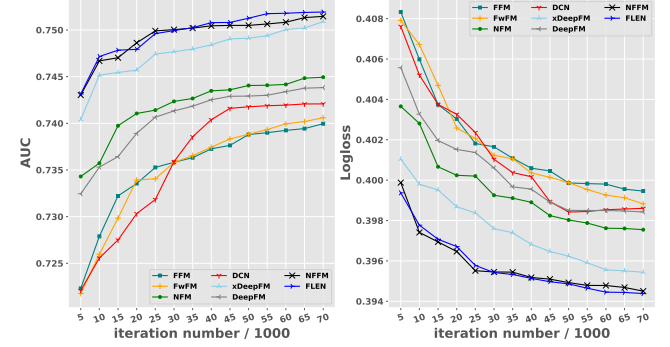

(a) Avazu
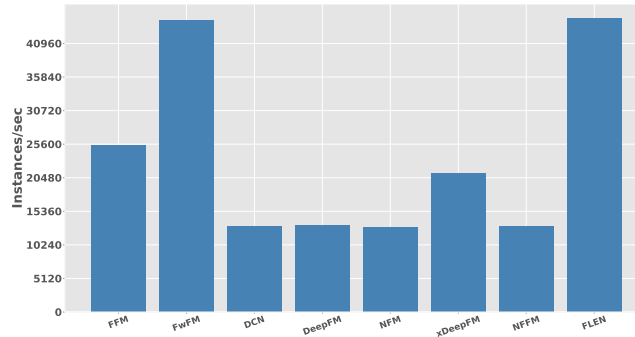
(b) Meitu

Figure 4: Number of instances per second processed by different models

We keep track of AUC and Logloss during the training process after each training "epoch" (i.e., 5,000 iterations through all the training data on Avazu and 20,000 iterations

on Meitu). As shown in Figure 5, FLEN obtains the best AUC (i.e. highest) and Logloss (i.e. lowest) on both datasets in each iteration. Furthermore, we point out that although NFFM (which is the best competitor) is close with FLEN, FLEN achieves faster convergence towards optimization. For example, FLEN has a sharper increase of AUC and more steep decrease of Logloss on Meitu dataset. Thus FLEN requires less training time than NFFM, which is desirable in real-world production systems.

(a) Avazu
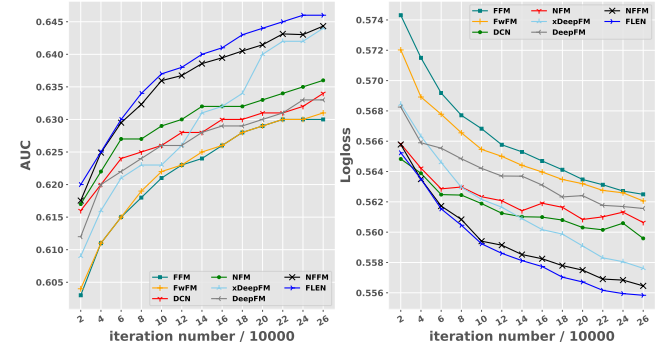
(b) Meitu

Figure 5: AUC and Logloss change in training time.

5.5 Impact of Parameters

We analyze the impact of an important parameter of Dice-factor, the probability β to keep bi-linear paths. We empirically find that a small value $\beta \in (0, 0.5)$ leads to poor performance. Henceforth, we set $\beta = 0.5, 0.6, 0.7, 0.8, 0.9, 1.0$ respectively and report the AUC and Logloss results. As shown in Figure 6, performance of FLEN is affected by β, the keep probability. The best parameter settings for AUC and Logloss are consistent. For example, best AUC and Logloss are both obtained at $\beta = 0.7$ on Avazu dataset. On Meitu dataset, the best $\beta = 0.8$.

6 Online Evaluation

To measure the impact that FLEN has on users we conduct an online evaluation through a 7-day A/B testing on Meitu app. We split 10% of the incoming traffic as experiment group, the rest as control group. The items of the control group in the A/B testing period are provided by the previous
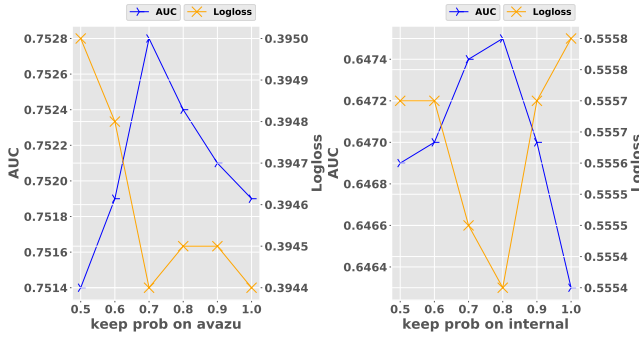
Figure 6: AUC and logloss with different keep probability of dicefactor on two datasets.

version of online ranking system, which is based on NFM. The items offered to the experiment group are items that are predicted by FLEN , i.e. we deliver top 12 items with highest prediction score by FLEN.

We report the CTR different during the A/B testing period in Figure 7. We observe stable and significant increase of CTR during the A/B testing period. The minimal CTR increase is above 4.9%. The mean CTR improvement over seven days is 5.195% with variance 0.282%.
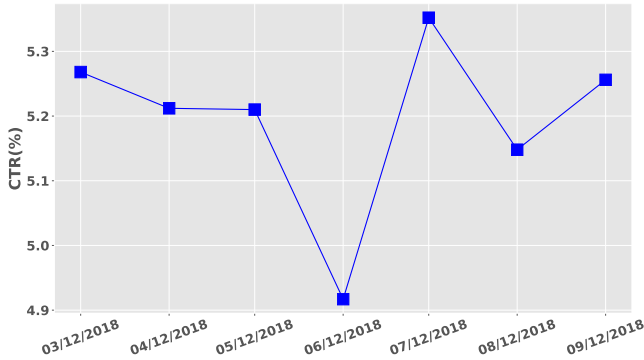

Figure 7: CTR increase during our online A/B testing period.

7 Conclusion

In this paper, we describe the FLEN model that is deployed in the online recommender system in Meitu, serving the main traffic. FLEN has obtained significant performance increase by exploiting field information with an acceptable memory usage and computing latency in the real-time serving system. As future work, we plan to explore the usage of the attention mechanism to attend to important field embedding. Furthermore, we are interested in extending FLEN to multi-task learning, i.e. predict conversions-rate and click-through-rate simultaneously.

8 Acknowledgements

References

Chapelle, O.; Manavoglu, E.; and Rosales, R. 2015. Simple and scalable response prediction for display advertising. *ACM Transactions on Intelligent Systems and Technology (TIST)* 5(4):61.

Cheng, C.; Xia, F.; Zhang, T.; King, I.; and Lyu, M. R. 2014. Gradient boosting factorization machines. In *Proceedings of the 8th ACM Conference on Recommender systems*, 265–272. ACM.

Cheng, H.-T.; Koc, L.; Harmsen, J.; Shaked, T.; Chandra, T.; Aradhye, H.; Anderson, G.; Corrado, G.; Chai, W.; Ispir, M.; et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, 7–10. ACM.

Covington, P.; Adams, J.; and Sargin, E. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, 191–198. ACM.

Duchi, J.; Hazan, E.; and Singer, Y. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12(Jul):2121–2159.

Graepel, T.; Candela, J. Q.; Borchert, T.; and Herbrich, R. 2010. Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine. Omnipress.

Guo, H.; Tang, R.; Ye, Y.; Li, Z.; and He, X. 2017. Deepfm: a factorization-machine based neural network for ctr prediction. *arXiv preprint arXiv:1703.04247*.

He, X., and Chua, T.-S. 2017. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, 355–364. ACM.

Juan, Y.; Lefortier, D.; and Chapelle, O. 2017. Field-aware factorization machines in a real-world online advertising system. In *Proceedings of the 26th International Conference on World Wide Web Companion*, 680–688. International World Wide Web Conferences Steering Committee.

Koren, Y.; Bell, R.; and Volinsky, C. 2009. Matrix factorization techniques for recommender systems. *Computer* 42(8):30–37.

Lian, J.; Zhou, X.; Zhang, F.; Chen, Z.; Xie, X.; and Sun, G. 2018. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. *arXiv preprint arXiv:1803.05170*.

Liu, W.; Tang, R.; Li, J.; Yu, J.; Guo, H.; He, X.; and Zhang, S. 2018. Field-aware probabilistic embedding neural network for ctr prediction. In *Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys '18, 412–416. New York, NY, USA: ACM.

Liu, B.; Tang, R.; Chen, Y.; Yu, J.; Guo, H.; and Zhang, Y. 2019. Feature generation by convolutional neural network for click-through rate prediction. In *The World Wide Web Conference*, WWW '19, 1119–1129. New York, NY, USA: ACM.

McMahan, H. B.; Holt, G.; Sculley, D.; Young, M.; Ebner, D.; Grady, J.; Nie, L.; Phillips, T.; Davydov, E.; Golovin, D.; et al. 2013. Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 1222–1230. ACM.

Oentaryo, R. J.; Lim, E.-P.; Low, J.-W.; Lo, D.; and Finegold, M. 2014. Predicting response in mobile advertising with hierarchical importance-aware factorization machine. In *Proceedings of*

the 7th ACM international conference on Web search and data mining, 123–132. ACM.

Pan, J.; Xu, J.; Ruiz, A. L.; Zhao, W.; Pan, S.; Sun, Y.; and Lu, Q. 2018. Field-weighted factorization machines for click-through rate prediction in display advertising. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, 1349–1357. International World Wide Web Conferences Steering Committee.

Qu, Y.; Fang, B.; Zhang, W.; Tang, R.; Niu, M.; Guo, H.; Yu, Y.; and He, X. 2018. Product-based neural networks for user response prediction over multi-field categorical data. *ACM Transactions on Information Systems (TOIS)* 37(1):5.

Rendle, S.; Gantner, Z.; Freudenthaler, C.; and Schmidt-Thieme, L. 2011. Fast context-aware recommendations with factorization machines. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, 635–644. ACM.

Rendle, S. 2010. Factorization machines. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, 995–1000. IEEE.

Shan, Y.; Hoens, T. R.; Jiao, J.; Wang, H.; Yu, D.; and Mao, J. 2016. Deep crossing: Web-scale modeling without manually crafted combinatorial features. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 255–262. ACM.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15(1):1929–1958.

Wang, R.; Fu, B.; Fu, G.; and Wang, M. 2017. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*, 12. ACM.

Wang, X.; He, X.; Feng, F.; Nie, L.; and Chua, T.-S. 2018. Tem: Tree-enhanced embedding model for explainable recommendation. In *Proceedings of the 2018 World Wide Web Conference*, 1543–1552. International World Wide Web Conferences Steering Committee.

Xiao, J.; Ye, H.; He, X.; Zhang, H.; Wu, F.; and Chua, T.-S. 2017. Attentional factorization machines: Learning the weight of feature interactions via attention networks. *arXiv preprint arXiv:1708.04617*.

Yang, Y.; Xu, B.; Shen, F.; and Zhao, J. 2019. Operation-aware neural networks for user response prediction. *arXiv preprint arXiv:1904.12579*.

Zhang, W.; Du, T.; and Wang, J. 2016. Deep learning over multi-field categorical data. In *European conference on information retrieval*, 45–57. Springer.

Zhu, J.; Shan, Y.; Mao, J.; Yu, D.; Rahmanian, H.; and Zhang, Y. 2017. Deep embedding forest: Forest-based serving with deep embedding features. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1703–1711. ACM.

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2818–2826.