xDeepFIG: An eXtreme Deep Model with Feature Interactions and Generation for CTR Prediction

Bokai Xu
Division of Science and Technology, BNU-HKBU United International College, China

Shihan Bu
Division of Science and Technology, BNU-HKBU United International College, China

Xinyue Li
Division of Science and Technology, BNU-HKBU United International College, China

Yanzhi Lin
Division of Science and Technology, BNU-HKBU United International College, China

Shengxin Zhu*
Research Center for Mathematics, Beijing Normal University at Zhuhai, China
Division of Science and Technology, BNU-HKBU United International College, China

ABSTRACT

In this paper, we propose an eXtreme deep model with feature interactions and generation for CTR prediction, called xDeepFIG. The feature generation module fully leverages some advantages of convolutional neural network (CNN) to generate new local and global features, and concatenates them with raw features. Such new fully fused features are shared by both the deep neural network (DNN) and compressed interaction network (CIN), which can learn both implicit and explicit high-order feature interactions automatically. Numerical results on two benchmark datasets for CTR demonstrates such feature fusion can bring some advantages and the xDeepFIG outperforms recent baseline models.

CCS CONCEPTS

• **Computing methodologies**; • **Neural networks**; • **Factorization methods**;

KEYWORDS

Explicit and implicit feature interaction, Feature generation, CIN, DNN, CTR

ACM Reference Format:
Bokai Xu, Shihan Bu, Xinyue Li, Yanzhi Lin, and Shengxin Zhu. 2021. xDeepFIG: An eXtreme Deep Model with Feature Interactions and Generation for CTR Prediction. In *2021 3rd International Conference on Big-data Service and Intelligent Computation (BDSIC 2021), November 19–21, 2021, Xiamen, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3502300.3502306

*Corresponding Author email address: Shengxin.Zhu@bnu.edu.cn

BDSIC 2021, November 19–21, 2021, Xiamen, China

ACM ISBN 978-1-4503-9055-2/21/11…$15.00
https://doi.org/10.1145/3502300.3502306

1 INTRODUCTION

Recommender system aims at finding candidate items for users based on the user's historical behaviors data, and thus to achieve personalized recommendation. Among many tasks of recommender system, click-through rate (CTR) prediction is an important topic, which estimates the probability of a recommended item clicked by a user. Ranking those items that are more likely to be clicked by users at the top can improve the number of user clicks and thus greatly increase the business revenue. That why accurate prediction for CTR plays a key role in recommender system.

The principle of recommendation is to figure out the relationship between different features. Thus, engineering for feature interactions is a central task in CTR prediction. In CTR prediction, both low- and high-order feature interactions are important. However, one of the challenges in CTR prediction is how to model feature interactions sufficiently and effectively. The easy-to-managed generalized linear models have limited ability to learn feature interactions. Some other models like factorization machine learn the pairwise feature interactions. Recently, some deep learning models has been proposed and they are able to learn high-order feature interactions.

However, one of the challenges in CTR prediction is how to model feature interactions sufficiently and effectively. The easy-to-managed generalized linear models are frequently used in many online platforms due to its flexibility and simplicity. However, the limited ability to learn feature interactions also leads to the comparably low prediction accuracy. Some other models like factorization machine and the corresponding variants learn the pairwise feature interactions. Recently, some deep learning models has been proposed and they are able to learn high-order feature interactions from raw data whereas the underlying sparse matrix may cause a large number of parameters. In addition, the function proposed by DNN might be arbitrary and feature interactions of DNN focus on the bit-wise level rather than vector-wise level.

Our approach is to design a hybrid architecture called e**X**treme **deep** model with **F**eature **I**nteractions and **G**eneration (xDeepFIG). It takes the advantages of xDeepFM [11] and FGCNN [12], which aim at efficiently and automatically learning both explicit and implicit feature interactions and generating new features respectively.

Enlightened by their ideas, xDeepFIG contains a feature generation module, an explicit feature interactions module and an implicit feature interactions module to better learn both explicit and implicit feature interactions.

In short, the major contributions we made in this paper are as follows:

1.The proposed xDeepFIG models contains a feature generation module, an explicit feature interactions module and an implicit feature interactions module. It can generate significant features automatically as well as capture both low- and high-order feature interactions implicitly and explicitly.

2.Different from the commonly used wide and deep structure, xDeepFIG uses linear and compressed interaction network (CIN) part as a "Wide" part to learn both low- and high-order explicit feature interactions, whilst takes the DNN as the "Deep" part to learn implicit high-order feature interactions. Specifically, unlike xDeepFM or DeepFM which feeds raw features into model directly, we propose a process of feature generation to obtain new feature combinations. Combining these new features with the raw features, we endow xDeepFIG with a more comprehensive ability of perceiving feature interactions.

3.About the process of feature generation, dissimilar with FGCNN which only puts the new and raw features into deep component, xDeepFIG shares the same new and raw features into linear, CIN and deep components for a more significant feature expression.

4.Experiments on two standard benchmark datasets show that xDeepFIG outperforms the existing popular models in terms of Logloss and area under ROC (AUC).

2 RELATED WORK

Because of the efficiency and simplicity of the traditional machine learning methods, many classical CTR prediction models using Logistic Regression (LR) [10][17] and Follow the Regularized Leader (FTRL) [14] are well received and widely applied in practice. These kinds of linear models are easy to establish and maintain, so they are widely employed in the web-level recommender systems. Recently, linear mixed models have also been considered [22][6][2]. However, linear models have their limitations on learning the feature interactions. A conventional method is to model pairwise feature interactions in the feature space manually, whereas it is difficult to model high-order feature interactions and is often time-consuming. Some other models like Factorization Machine (FM) [16] also take feature interactions into consideration. FM proposes a latent and low-dimensional vector to embed each feature and calculates inner product of feature vectors so that it gives a similarity measure for feature interactions. FM greatly improves the performance on sparse embedding vectors compared with previous models.

Deep learning is a powerful approach to capture and learn complex feature interactions. Recently, increasing interests are shown in employing Deep neural network (DNN) for CTR prediction. Deep Neural Networks (DNNs) are firstly developed for CTR prediction and video recommendation where the structure is similar with the traditional multi-layer perceptron (MLP) [4]. Some other models like Attentional Factorization Machine (AFM) use DNN to improve the performance of FM to get more accurate prediction results [20]. AFM aims at distinguishing the importance of different feature interactions. In comparison, Neural Factorization Machine (NFM) [8] proposes the idea of bi-interaction pooling combined with FM and creates a deeper FM model to better learn high-order and nonlinear cross-features interactions. Moreover, Product-based Neural Network (PNN) [15] proposes an idea of product layer by multiplication between embedding layers and the fully connected layers to realize neural network for cross-feature interactions. In order to learn both low- and high-order feature interactions, some hybrid network structures are proposed. The Wide & Deep Learning (Wide & Deep) [3] takes the advantages of the wide model and the deep part. However, Wide & Deep model contains a LR model which needs feature engineering manually to achieve accurate prediction. Based on the idea of hybird structure, Deep Factorization Machine (DeepFM) [7] is proposed, which replaces the LR model of Wide & Deep by FM to prevent feature engineering manually. It models the feature interactions with low-dimension like FM and high-dimension like DNN. And then, Deep&Cross Network (DCN) [19] model is proposed to combine features without increasing network parameters.

As for the state-of-the-art models, Deep Session Interest Network (DSIN) [1][5] considers the behavioral isomorphism of different user behavior sequences and the heterogeneous behaviors between sessions. DSIN also uses the transformer method to record user behavior sequences. In the model AutoFIS [13], it can automatically identify important feature interactions. It contains two stages where some useful feature interactions will be caught in the search stage, and the model with selected feature interactions is re-trained in the re-train part.

Also, recommender system has been widely used in the business. Facebook designs a model GBDT-LR [9] which applies Gradient Boosting Decision Tree (GBDT) to figure out the efficient features as well as the combination of different features. It considers the output of the GBDT model as the input feature vector in the LR model, which can significantly save the time on the feature interactions and selection.

The Deep Interest Network (DIN) [25] and Deep Interest Evolution Network (DIEN) [24] model proposed by Alibaba introduce attention mechanism to set item attributes different weights based on the relevance to the candidate when performing pooling. ByteDance has designed a two-level reinforcement learning framework [23] to jointly optimize recommendation and advertising strategies. As multi-task learning (MTL) has been successfully applied to recommender systems, Tencent has proposed a Progressive Layered Extraction (PLE) model [18] with a novel shared structure to reduce the potential for negative transfer and seesaw phenomenon.

3 METHODOLOGY

In this section, we will elaborate our xDeepFIG model in detail. Firstly, we adopt a feature generation component to obtain a new feature space which can augment and enrich the original raw features. Then, these new features are fed into feature interactions component to perform explicit feature interactions. Utilizing the advantages of Deep Neural Network (DNN) models, the augmented features can also be fed into DNN to learn implicit interactions.

$$\underbrace{[0,1,0,0,\ldots,0]}_{\text{id}}\underbrace{[\ 0,1\ ]}_{\text{gender}}\underbrace{[0,0,1,0,\ldots,0]}_{\text{device}}\underbrace{[1,0,0,1,\ldots,0]}_{\text{hobby}}$$

Figure 1: Sample of encoded vectors after field-aware one-hot encoding

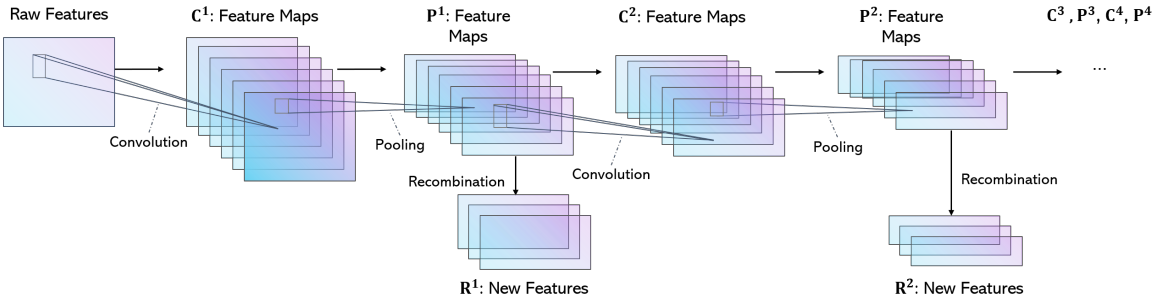

Figure 2: This figure shows one of the parameter combinations of convolution, pooling and recombination (C^i indicates a bunch of processed feature maps after each convolution in the i^{th} cycle, P^i indicates a bunch of processed feature maps after each pooling in the i^{th} cycle, and R^i indicates recombined new feature maps after the i^{th} cycle). The detailed parameter settings of our model (xDeepFIG) will be shown in section 5.2.

3.1 Embedding Layer

In recommender systems, there are two types of features: categorical features and continuous features. If the feature is continuous, we can directly use it as a dense feature. However, if the feature is categorical, we need some transformations. A common method in recommender systems to process categorical feature is the field-aware one-hot encoding [21], which aims at transforming the raw input features to high-dimensional sparse features. Only the value of a field appears in an input instance is marked as 1, and others are marked as 0.

For example, given an input instance [id = 02, gender = female, device = 3, hobby = basketball & piano], the encoded vector after field-aware one-hot encoding will be shown in Figure 1.

After getting the one-hot encoded vectors, the embedding layer is used for transforming these high-dimensional sparse vectors to low-dimension dense vectors. For a categorical feature, if the field is univalent, then feature embedding will be considered as field embedding. For instance, in the above example, feature embedding 3 is used as the field embedding device. And if the field is multi-valent, field embedding uses the sum of feature embedding. In the general form of embedded layers, the results are concatenated vectors:

$$\boldsymbol{E} = [\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_{\mathrm{m}}]$$

where m is the number of field, and $e_i \in R^D$ indicates the embedding of one field, and D is the embedding dimension. One important character of embedding layer is that although the lengths of raw input features are different in most cases, their embedded vector e_i all have the same size D which is the embedding dimension.

3.2 Feature Generation

In addition to learning feature interactions from raw input feature spaces arbitrarily, predicting performance can also be improved if more useful feature interactions could be extracted from the highly sparse raw data and trained with the original features. Recent researches show that machine learning models can generate complicated feature interactions automatically with efficient practicality and robustness. A typical example is Feature Generation by Convolutional Neural Network (FGCNN) [12] which can generate new significant features from the raw feature spaces without any manual feature engineering.

In xDeepFIG, feature generation component leverages the advantages of Convolutional Neural Network (CNN) and Multi-Layer Perceptron (MLP) for recommender systems, which was proposed in Feature Generation by Convolutional Neural Network (FGCNN) [12]. Figure 2 depicts the workflow of *feature generation*, which consists of three layers: *convolutional layer*, *pooling layer* and *recombination layer*.

3.2.1 Convolutional Layer. For an embedding size k and number of feature fields n_f, each field i is represented by $\boldsymbol{e_i} \in R^{1\times k}$ which is a low-dimensional feature vector compressed from high-dimensional raw features through embedding layers.

Therefore, every embedding matrix instance $\boldsymbol{E} = (\boldsymbol{e}_1^T, \boldsymbol{e}_2^T, \cdot s, \boldsymbol{e}_{\boldsymbol{n_f}}^T)^T \varepsilon R^{n_f\times k}$ can be contributed to feature interactions. Convolutional layer can be represented as follows:

$$\mathrm{C}^1_{a,b,i} = \tanh\left(\sum_{n=1}^{1}\sum_{j=1}^{h^1} E^1_{a+j-1,b,n}\,{}^{\mathbb{W}}\mathbb{C}^1_{j,1,1,i}\right)$$

$$\tanh(x) = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$$

where $C^1_{:,:,\mathrm{i}}$ presents the i^{th} first convolutional layer's feature map, a and b denote the index of row and column in the i^{th} feature map. $\mathbb{W}^1_{\mathbb{C}} \in R^{h^1\times 1\times 1\times m^1_c}$ represents the weight for the i^{th} recombination

layer with h^1 (the kernel height of the first convolutional layer) and feature maps number m_c^1 in the first convolutional layer so as to retrieve the further neighbor local feature interactions.

3.2.2 Pooling Layer. After each convolutional layer, in order to capture the most significant interactions of features, max-pooling layer can be utilized, which can be denoted as:

$$max\left(\mathrm{C}^1_{a\cdot h_a,b,i},\ldots,\mathrm{C}^1_{a\cdot h_a+h_a-1,b,i}\right)$$

where h_a represents the heights of each pooling layer which of width equal to 1. Thus, the first pooling layer will derive a result like this:

$$\mathbf{P}^1 \in R^{(n_f/h_a)\times k\times m_c^1}$$

And the outputs of the i^{th} pooling layer will be the inputs of the $(i+1)^{th}$ convolutional layer. That is, $\boldsymbol{E}^{\mathrm{i}+1} = \boldsymbol{P}^{\mathrm{i}}$.

3.2.3 Recombination Layer. In the CTR prediction, CNN is not of significant local correlation among features like images and texts. To fill the gap of this area of CNN, the following ideas have been proposed. After each cycle, the recombination layer serves as a critical function. CNN can decrease the number of parameters and reduce the complexity in optimization process, as well as obtain the neighbor feature patterns. Based on the extracted features from CNN, each recombination layer can further generate global feature interactions by MLP fully connected layers. The recombination process can be added with a bias and we can denote the formula as below

$$\mathbf{R}^1 = \tanh\left(\mathbf{P}^1\cdot\mathbb{W}^1_{\mathbb{R}} + \mathbb{B}^1_{\mathbb{R}}\right)$$

where $\mathbb{W}^1_{\mathbb{R}} \in R^{(n_f/h_a k m_c^1)\times(n_f/h_a k m_r^1)}$ represents the weight matrix, $\mathbb{B}^1_{\mathbb{R}} \in R^{(n_f/h_a k m_r^1)}$ represents the bias with embedding layer size equals to k, while the feature maps number m_c^1 is the first convolutional layer and the new feature maps number m_r^1 is the first recombination layer.

In this way, CNN part is considered as a combination of convolutional layer and pooling layer, while MLP part is regarded as the recombination layer (fully-connected) accordingly. Local neighborhood feature interactions are learned by performing CNN, while global feature interactions are extracted by recombining the local feature patterns by MLP in this process. One feature extraction cycle is represented as $(\boldsymbol{C^i}, \boldsymbol{P^i})$ for $i = 1, 2, 3, \ldots$, where $\boldsymbol{C^i}$ indicates a bunch of processed feature maps after each convolution in the i^{th} cycle, and $\boldsymbol{P^i}$ indicates a bunch of processed feature maps after each pooling in the i^{th} cycle. These procedures are able to extract useful neighbor feature patterns from the raw features.

After each cycle of convolution, pooling and recombination process, an intermediate new feature is derived. The final recombination new features are obtained by performing several cycles so as to derive a bunch of new features and concatenate these intermediate new features together. In light of this, the feature spaces are effectively augmented by the newly generated features interacted with the original raw features for further usage.

3.3 Feature Interaction

For the feature interaction component, we adapt the Compressed Interaction Network (CIN), which was proposed in xDeepFM [11]. Since the embedding vectors generated from the embedding layers are used for vector-wise based feature interactions, we can rewrite the output of embedding layers as a matrix $\mathbf{X}^0 \in \mathbb{R}^{m\times D}$, where m is the number fields and D is the embedding dimension and is same for all fields regardless the length of the fields. The i^{th} row of $\mathbf{X}^0$ ($\mathbf{X}^0_{i,*}$), is the i^{th} element ($\mathbf{e}_i$) of the embedding vector $\mathbf{E}$. The output of each hidden layer in CIN is still a matrix. Specifically, we define the output of i^{th} layer of CIN as $\mathbf{X}^i$, where $\mathbf{X}^i \in \mathbb{R}^{C_i\times D}$. C_i represents the number of embedding vectors in the i^{th} layer of CIN and D is the embedding layer's dimension. For convenience, we initialize the first layer $C_0 = m$. For the output of i^{th} layer of CIN $\mathbf{X}^i$, it is computed as

$$\mathbf{X}^i_{h,*} = \sum_{a=1}^{C_{i-1}}\sum_{b=1}^{m}\mathbf{W}^{i,h}_{ab}\left(\mathbf{X}^{i-1}_{a,*}\circ\mathbf{X}^0_{b,*}\right)$$

where $1 \le h \le C_i$, $\mathbf{W}^{i,h} \in \mathbb{R}^{C_{i-1}\times m}$ is the trainable weight for h^{th} feature vector in i^{th} layer of CIN. The symbol $\circ$ is denoted as the Hadamard product.

Let T be the CIN depth level. The sum pooling is firstly applied on each hidden layer $\mathbf{X}^k (k \in [1, T])$ as follows

$$p_i^k = \sum_{j=1}^{D}\mathbf{X}^k_{i,j}$$

Where $i \in [1, H_k]$, H_k is the length of the k^{th} layer. Before feeding to the output unit, all the pooling vectors are concatenated as

$$\mathbf{p}^+ = \left[\mathbf{p}^1, \mathbf{p}^2, \ldots, \mathbf{p}^T\right] \in \mathbb{R}^{\sum_{i=1}^{T} H_i}$$

3.4 xDeepFIG

Our proposed e**X**treme **d**eep model with **f**eature **i**nteractions and **g**eneration (xDeepFIG) utilizes the feature generation component to generate more features whilst CIN for explicit feature interactions and DNN for implicit feature interactions. Figure 3 shows the overview architecture of our model.

Given the raw input m-field features $\chi \in \mathbb{R}^m$, the embedding layer is firstly applied to χ to get the embedding vector $\mathbf{X}^0 \in \mathbb{R}^{m\times D}$, where D is the embedding size which is all the same regardless the dimension of each field. Then, we use feature generation method described in 3.2 to generate new features from the raw embedding features $\mathbf{X}^0$. We denote these new features as $\mathbf{X}^{fg} \in \mathbb{R}^{fg_{length}\times D}$, where fg_{length} is the dimension of the generated new features and D is the dimension of the raw embedding features. Then we concatenate $\mathbf{X}^{fg}$ and $\mathbf{X}^0$ to denote the concatenated vector as $\mathbf{X}^{concat} = [\mathbf{X}^{fg}, \mathbf{X}^0] \in \mathbb{R}^{(fg_{length}+m)\times D}$.

For the explicit feature interactions by using CIN, the input of it is the output of feature generation component ($\mathbf{X}^{concat}$), which is the same as $\mathbf{X}^0$ in 3.3. And the output of CIN is denoted as $\mathbf{p}^+$.

For the implicit feature interaction part, we employ the idea of DNN which uses fully connected layers. Before feeding to DNN, we use pairwise feature interactions (inner product) for $\mathbf{X}^{concat}$ to obtain specific explicit feature interactions. We denote the results after inner product as $\mathbf{X}^{ip} \in \mathbb{R}^{ip_{length}\times D}$. And finally, we concatenate $\mathbf{X}^{ip}$ and $\mathbf{X}^{concat}$ together to derive the final new features $\mathbf{X}^{new} = [\mathbf{X}^{concat}, \mathbf{X}^{ip}]$.

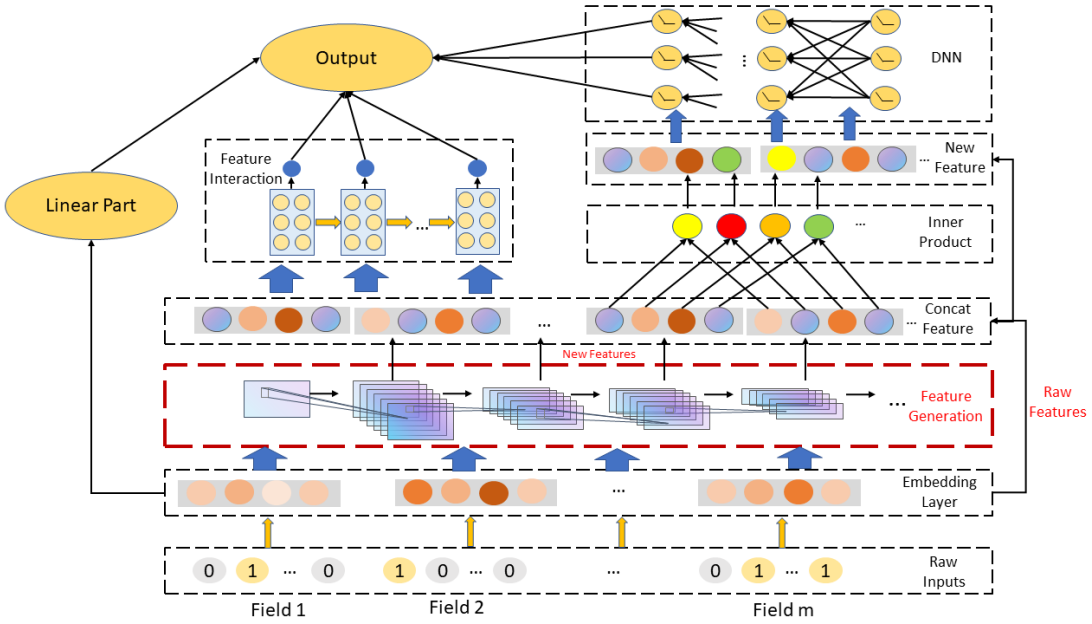

Figure 3: The architecture of xDeepFIG. The model has a feature generation component (red block) which differs from other traditional models, and all the linear, feature interaction and deep components share the new features and raw features together.

Then, we can feed the newly concatenated features $\mathbf{X}^{new}$ to DNN for implicit feature interactions. For the first layer of DNN, the formula is

$$\mathbf{L}^{(1)} = \sigma\left(\mathbf{W}^{(0)^T}\mathbf{X}^{new} + b^{(0)}\right)$$

where $\mathbf{L}^{(1)}$ is the first layer's output of DNN. $\mathbf{W}^{(0)}$ and $b^{(0)}$ are the trainable weight and bias of the first layer respectively.

For the $k^{th}(k > 1)$ hidden layers of DNN

$$\mathbf{L}^{(k)} = \sigma\left(\mathbf{W}^{(k-1)^T}\mathbf{L}^{(k-1)} + b^{(k-1)}\right)$$

where $\mathbf{L}^{(k)}$ is the k^{th} layer's output of DNN. $\mathbf{W}^{(k-1)}$ and $b^{(k-1)}$ are the trainable weight and bias of the $(k-1)^{th}$ layer respectively.

The output of DNN, denoted as $\mathbf{O}^{DNN}$, is

$$\mathbf{O}^{DNN} = \sigma\left(\mathbf{W}^{(last)^T}\mathbf{L}^{(last)} + b^{(last)}\right)$$

where $\mathbf{L}^{(last)}$ is the output of the last hidden layer of DNN. $\mathbf{W}^{(last)}$ and $b^{(last)}$ are the trainable weight and bias of the last hidden layer respectively.

Besides explicit and implicit feature interactions, we also apply a linear part to capture the linear pattern of data. The output of linear part is denoted as $\mathbf{O}^{linear}$ and is calculated through

$$\mathbf{O}^{linear} = \mathbf{W}_{lp}^T\mathbf{X}^0 + b_{lp}$$

where $\mathbf{W}_{\text{lp}}$ and b_{lp} are the weight and bias of linear part respectively.

Finally, we concatenate the results of explicit feature interaction part $\mathbf{p}^+$, implicit feature interaction part $\mathbf{O}^{DNN}$, and linear part $\mathbf{O}^{linear}$ to attain our final predicted value $\hat{y}$:

$$\hat{y} = \sigma\left(\boldsymbol{w}_{\text{linear}}^T\,\mathbf{O} + \boldsymbol{w}_{dnn}^T\mathbf{O}^{DNN} + \boldsymbol{w}_{\text{cin}}^T\mathbf{p}^+ + b\right)$$

As CTR prediction is a task of binary classification, the log loss function is defined as follows:

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

where y_i is the label and $\hat{y}$ is the predicted value. N is the total number of training instances. We need to minimize the following objective function during the training process:

$$\mathcal{J} = \mathcal{L} + \lambda_*\|\Theta\|$$

where λ_* is the regularization term to avoid overfitting problem and Θ is the collection of trainable parameters including parameters for linear part, explicit feature interaction part by using CIN and implicit feature interaction part by using DNN.

4 COMPARISONS WITH OTHER MODELS

4.1 DeepFM

DeepFM [7] is regarded as a hybrid algorithm which combines the Deep component and FM component. FM is used as a low-order feature interaction, while DNN part is used as a high-order feature interaction. By combining the two methods in a parallel way, the final architecture has the following characteristics:

- No need to pre-train FM to get implicit vector
- No manual feature engineering is required
- Able to learn the feature interactions of both high-order and low-order simultaneously

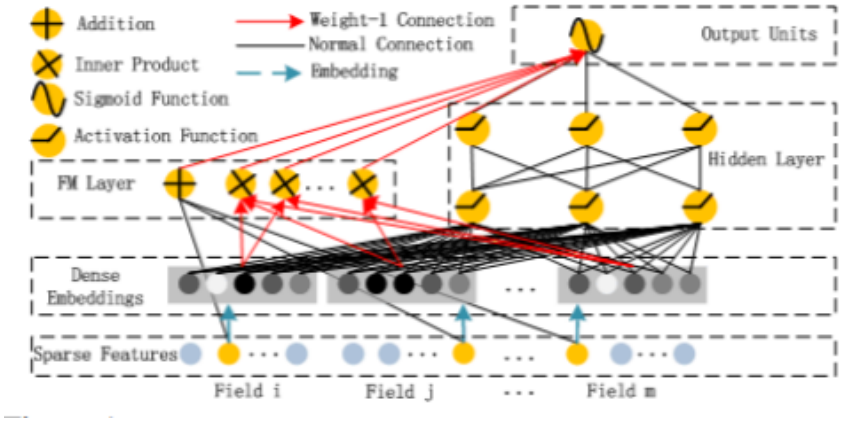

Figure 4: The structure of DeepFM [7]

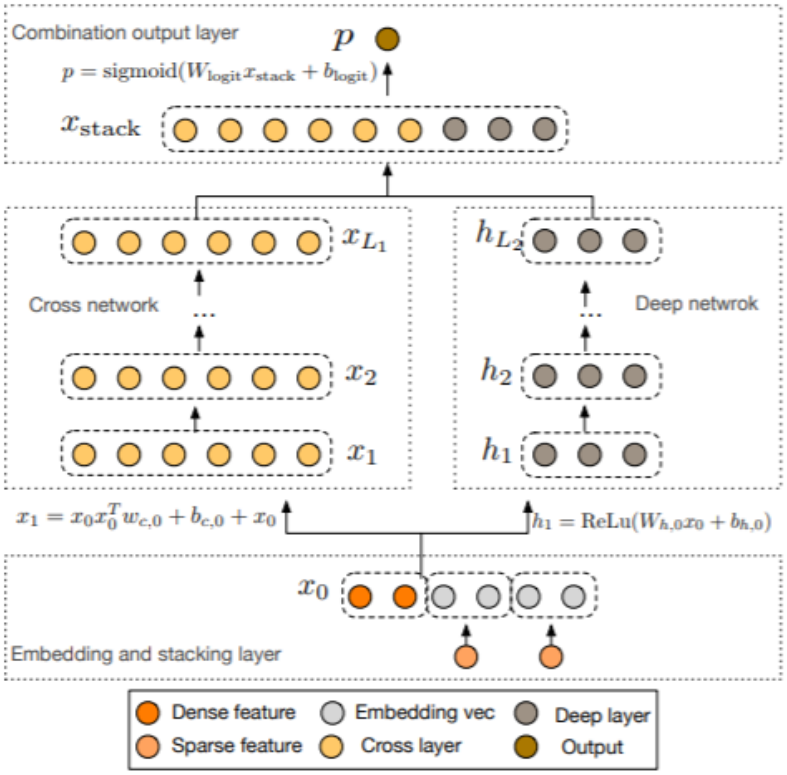

Figure 5: The structure of Deep & Cross Network [19]

- FM component and Deep component share the same feature embedding

In our model, we replace FM part with CIN component and make use of raw features to feature generation. And then, the raw and new features will be combined and later fed to the feature interaction components.

4.2 DCN

Deep & Cross Network (DCN) [19] can efficiently learn specific order of feature interactions, and does not require manual feature engineering. DCN employs cross network and implements feature crossing in each layer. Instead of using cross network, we use CIN for vector-wise feature interactions and learn high-order feature interactions explicitly.

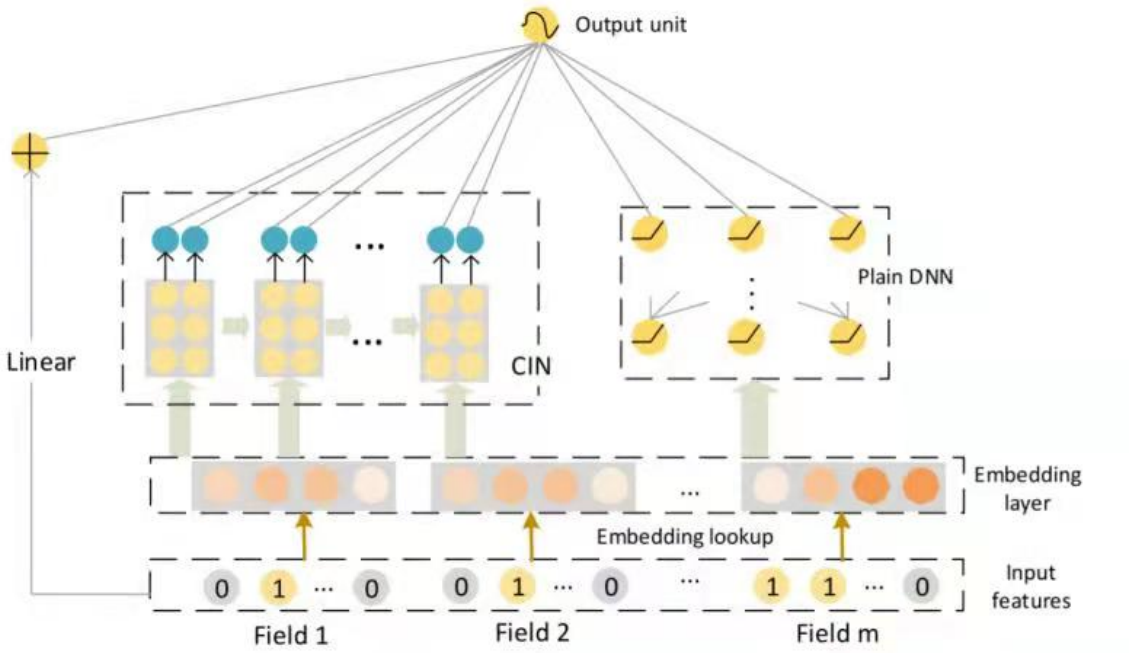

Figure 6: The structure of xDeepFM [11]

Table 1: Comparison of different models. (Note: Explicit and implicit mean explicit feature interactions and implicit feature interactions respectively; High-order explicit means whether high-order feature interactions used in explicit feature interactions; Share means whether the raw and new features are shared inputs for both "Wide" and "Deep" parts.)

Model	Published Year	Institute	Characteristics				
			Feature Generation	Explicit	Implicit	High-order Explicit	Share
DeepFM	2016	Huawei	×	✓	✓	×	×
DCN	2017	Stanford & Google	×	✓	✓	✓	×
FGCNN	2019	Huawei	✓	×	✓	×	×
xDeepFM	2018	Microsoft	×	✓	✓	✓	×
xDeepFIG	--	--	✓	✓	✓	✓	✓

4.3 xDeepFM

xDeepFM [11] makes improvements based on DCN which comes up with a new network using CNN and RNN instead of using cross network. Inspired by xDeepFM, we take the advantage of CIN to learn explicit feature interactions whilst DNN to learn implicit feature interactions. In addition, we apply feature generation component to obtain new features from raw features without any manual feature engineering.

4.4 FGCNN

FGCNN [12] consists of two modules: feature generation and a deep classifier. Feature generation uses CNN and recombination layer to generate new features. These new features are combined with the raw features together and be fed to arbitrary deep classifiers. Apart from the feature generation procedure, we take the advantage of CIN component to learn the high-order explicit feature interactions additionally. With feeding the new-generated features into CIN and DNN component, we can maintain advantages of the FGCNN and xDeepFM.

In this section, we will compare the differences among xDeepFIG and other well-known models, which is summarized in Table 1. We could see that xDeepFIG is the only model that requires feature generation, high-order explicit and implicit feature interactions and the inputs of "Wide" and "Deep" parts are shared.

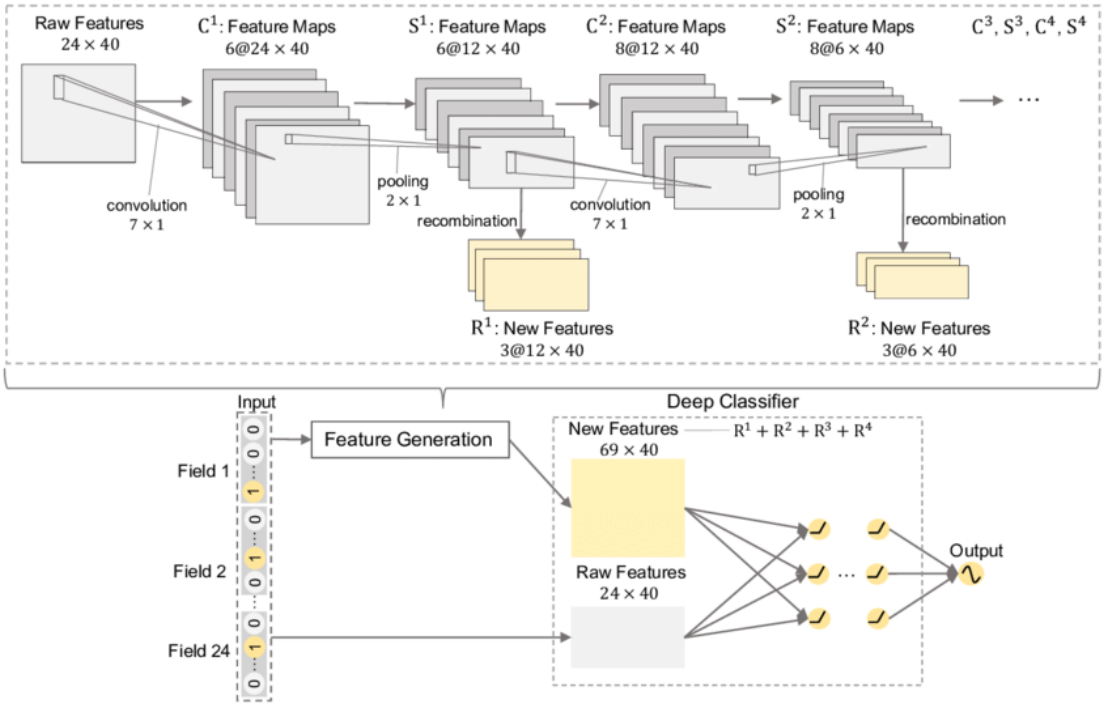

Figure 7: The structure of FGCNN [12]

Table 2: Dataset Statistics

Dataset	#Instances	#Fields	Ratio ofNeg (non-clicked): Pos (clicked)
Avazu	4×10^6	24	5:1
Criteo	2×10^6	39	3:1

5 EXPERIMENTS

5.1 Dataset

5.1.1 Avazu. Avazu dataset[1] contains 23 attributes and 1 label in total which specifically includes 9 anonymized categorical variables, advertisement id, hour, site information, app information, device information and whether an advertisement is clicked or not as a binary variable. In order to predict the click-through rate of items on certain webpages by analyzing online system logs and user behavior information, the train dataset provides 10 days of click-through information in time order and the test dataset provides 1 day of all information except click, the binary variable attribute for prediction. Since the original dataset is relatively large, we decided to do a stratified sampling of the dataset for our efficient utilization.

5.1.2 Criteo. Criteo dataset[2] is an online advertising data set published by Criteo labs. It contains 39 attributes including 13 continuous variables and 26 categorical variables. As we stated before, we still use the stratified sampling method to obtain a relatively smaller dataset. Table 2 summarizes the dataset statistics.

[1]https://www.kaggle.com/c/avazu-ctr-prediction/data
[2]https://www.kaggle.com/c/criteo-display-ad-challenge/data

5.2 Parameters Setup

The setting of the parameters is shown in Table 3

5.3 Experiment and Results

5.3.1 Evaluation. The evaluation metrics are AUC (area under ROC) and Logloss (cross entropy) to evaluate the performance of different models. A larger AUC score and a lower Logloss score mean a better performance of the model.

5.3.2 Result. The dataset is split into training and testing set, with the ratio of 4:1. DeepFM and DCN has already shown its better performance over traditional linear models and machine learning models. xDeepFM has an overwhelming advantage in Wide & Deep frameworks like DeepFM. FGCNN is almost the best model in CTR prediction field. In this case, we only compare our model with these models to show our model have better performance in CTR prediction. The results are shown in Table 4, which we can draw several conclusions:

- Comparing to DeepFM and DCN, xDeepFIG is significantly better than these two baseline models with 0.71% and 0.65% in AUC (0.0036 and 0.0038 in Logloss) in Avazu dataset,

Table 3: Parameters Setting. (Note: opt=optimizer, Loss=loss function, hidden=number of hidden units, CIN=number of layer of CIN, cross_layer_num=number of cross layers, kernel=number of convolution kernels, Conv=shape of convolution, pool=shape of max pooling, new=shape of new features)

	Avazu	Criteo
General	Opt = Adam	Opt = Adam
	Loss = binary_crossentropy	Loss = binary_crossentropy
	Regularizer = L2 (1e-5)	Regularizer = L2 (1e-5)
DeepFM	hidden = (128,64,1)	hidden = (128,64,1)
DCN	hidden = (128,64,1)	hidden = (128,64,1)
	cross_layer_num = 3	cross_layer_num = 3
xDeepFM	hidden = (128,64,1)	hidden = (128,64,1)
	CIN = (128,64)	CIN = (200,200,200)
FGCNN	Kernel = (14, 16, 18, 20)	Kernel = (38,40,42,44)
	Conv = 7*1	Conv = 9*1
	pool = (2, 2, 2, 2)	pool = (2, 2, 2, 2)
	new = (3, 3, 3, 3)	new = (3, 3, 3, 3)
xDeepFIG	Kernel = (14, 16, 18, 20)	Kernel = (38,40,42,44)
	Conv= 7*1	Conv= 9*1
	pool = (2, 2, 2, 2)	pool = (2, 2, 2, 2)
	new = (3, 3, 3, 3)	new = (3, 3, 3, 3)
	hidden = (128,64,1)	hidden = (128,64,1)
	CIN = (128,64)	CIN = (200,200,200)

Table 4: Overall performance of Avazu and Criteo dataset

	Avazu		Criteo	
Model	AUC	Logloss	AUC	Logloss
DeepFM	0.7466	0.3973	0.7661	0.4796
DCN	0.7472	0.3975	0.7695	0.4769
FGCNN	0.7478	0.3986	0.7733	0.4744
xDeepFM	0.7530	0.3950	0.7773	0.4696
xDeepFIG	**0.7537***	**0.3937***	**0.7781***	**0.4689***

whilst with 1.20% and 0.86% in AUC (0.0107 and 0.0080 in Logloss) in Criteo dataset.

- With the component of feature interaction, the result of FGCNN is improved by xDeepFIG by 0.59% in AUC (0.0049 in Logloss) in Avazu dataset, whilst by 0.48% in AUC (0.0055 in Logloss) in Criteo dataset.
- With the component of feature generation, the result of xDeepFM is improved by xDeepFIG by 0.07% in AUC (0.0013 in Logloss) in Avazu dataset, whilst by 0.08% in AUC (0.0007 in Logloss) in Criteo dataset.

In general, xDeepFIG integrates the advantages of feature generation and feature interaction, which shows its superiority in CTR prediction.

6 CONCLUSION

In conclusion, CTR prediction is an essential task in recommender system, where the accurate prediction can help for better advertising. In our paper, we take advantages of both xDeepFM and FGCNN and propose a hybrid architecture called xDeepFIG which contains three main modules: feature generation, feature interaction and deep component. Feature generation module generates features automatically from raw data by CNN and recombines new features as the input for explicit feature interaction and implicit feature interaction by deep component. The experiment results show that xDeepFIG has the best performance on both Avazu and Criteo dataset compared with some other existing frequently used models, by using the evaluation metrics of Logloss and AUC score. Our proposed model can lead to some potential applications in other fields, such as data augmentation (because of the feature generation component), reinforcement learning, etc. The success of this feature generation and interactions model in CTR indicates that such a model can be used in other fields with deep learning.

ACKNOWLEDGMENTS

The project is supported by special projects for key fields of new generation information technology of Guangdong province China (No.2021ZDX1046) and UIC's start-up research fund(R72021114). The project is supported by UIC's start-up research fund (R72021114).

REFERENCES

[1] Qiwei Chen, Huan Zhao, Wei Li, Pipei Huang, and Wenwu Ou. 2019. Behavior Sequence Transformer for E-Commerce Recommendation in Alibaba. In Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data (Anchorage, Alaska) (DLP-KDD '19). Association for Computing Machinery, New York, NY, USA, Article 12, 4 pages. https://doi.org/10.1145/3326937.3341261

[2] Zhiyi Chen, Shengxin Zhu, Qiang Niu, and Tianyu Zuo. 2020. Knowledge Discovery and Recommendation With Linear Mixed Model. IEEE Access 8 (2020), 38304–38317. https://doi.org/10.1109/ACCESS.2020.2973170

[3] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & Deep Learning for Recommender Systems. In Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (Boston, MA, USA) (DLRS 2016). Association for Computing Machinery, New York, NY, USA, 7–10. https://doi.org/10.1145/2988450.2988454

[4] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In Proceedings of the 10th ACM Conference on Recommender Systems (Boston, Massachusetts, USA) (RecSys '16). Association for Computing Machinery, New York, NY, USA, 191–198. https://doi.org/10.1145/2959100.2959190

[5] Yufei Feng, Fuyu Lv, Weichen Shen, Menghan Wang, Fei Sun, Yu Zhu, and Keping Yang. 2019. Deep Session Interest Network for Click-Through Rate Prediction. In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19. International Joint Conferences on Artificial Intelligence Organization, Macau, China, 2301–2307.https://doi.org/10.24963/ijcai.2019/319

[6] Baode Gao, Guangpeng Zhan, Hanzhang Wang, Yiming Wang, and Shengxin Zhu. 2019. Learning with Linear Mixed Model for Group Recommendation Systems. In Proceedings of the 2019 11th International Conference on Machine Learning and Computing (Zhuhai, China) (ICMLC '19). Association for Computing Machinery, New York, NY, USA, 81–85. https://doi.org/10.1145/3318299.3318342

[7] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-Machine Based Neural Network for CTR Prediction. In Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17). AAAI Press, Melbourne, Australia, 1725–1731.

[8] Xiangnan He and Tat-Seng Chua. 2017. Neural Factorization Machines for Sparse Predictive Analytics. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (Shinjuku, Tokyo, Japan) (SIGIR '17). Association for Computing Machinery, New York, NY, USA, 355–364. https://doi.org/10.1145/3077136.3080777

[9] Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, and Joaquin Quiñonero Candela. 2014. Practical Lessons from Predicting Clicks on Ads at Facebook. In Proceedings of the Eighth International Workshop on Data Mining for Online Advertising (New York, NY, USA) (ADKDD'14). Association for Computing Machinery, New York, NY, USA, 1–9. https://doi.org/10.1145/2648584.2648589

[10] Kuang-chih Lee, Burkay Orten, Ali Dasdan, and Wentong Li. 2012. 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 (Beijing, China) (KDD '12). Association for Computing Machinery, New York, NY, USA, 768–776. https://doi.org/10.1145/2339530.2339651

[11] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. XDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (London, United Kingdom) (KDD '18). Association for Computing Machinery, New York, NY, USA, 1754–1763. https://doi.org/10.1145/3219819.3220023

[12] Bin Liu, Ruiming Tang, Yingzhi Chen, Jinkai Yu, Huifeng Guo, and Yuzhou Zhang. 2019. Feature Generation by Convolutional Neural Network for Click-Through Rate Prediction. In The World Wide Web Conference (San Francisco, CA, USA) (WWW '19). Association for Computing Machinery, New York, NY, USA, 1119–1129. https://doi.org/10.1145/3308558.3313497

[13] Bin Liu, Chenxu Zhu, Guilin Li, Weinan Zhang, Jincai Lai, Ruiming Tang, Xiuqiang He, Zhenguo Li, and Yong Yu. 2020. AutoFIS: Automatic Feature Interaction Selection in Factorization Models for Click-Through Rate Prediction. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (Virtual Event, CA, USA) (KDD '20). Association for Computing Machinery, New York, NY, USA, 2636–2645. https://doi.org/10.1145/3394486.3403314

[14] H. Brendan McMahan, Gary Holt, D. Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, Sharat Chikkerur, Dan Liu, Martin Wattenberg, Arnar Mar Hrafnkelsson, Tom Boulos, and Jeremy Kubica. 2013. Ad Click Prediction: A View from the Trenches. In Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (Chicago, Illinois, USA) (KDD '13). Association for Computing Machinery, New York, NY, USA, 1222–1230. https://doi.org/10.1145/2487575.2488200

[15] Yanru Qu, Bohui Fang, W. Zhang, Ruiming Tang, Minzhe Niu, H. Guo, Y. Yu, and X. He. 2019. Product-Based Neural Networks for User Response Prediction over Multi-Field Categorical Data. ACM Transactions on Information Systems (TOIS) 37 (2019), 1 – 35

[16] Steffen Rendle. 2010. Factorization Machines. In 2010 IEEE International Conference on Data Mining. IEEE, Sydney, NSW, Australia, 995–1000. https://doi.org/10.1109/ICDM.2010.127

[17] Matthew Richardson, Ewa Dominowska, and Robert Ragno. 2007. Predicting Clicks: Estimating the Click-through Rate for New Ads. In Proceedings of the 16th International Conference on World Wide Web (Banff, Alberta, Canada) (WWW '07). Association for Computing Machinery, New York, NY, USA, 521–530. https://doi.org/10.1145/1242572.1242643

[18] Hongyan Tang, Junning Liu, Ming Zhao, and Xudong Gong. 2020. Progressive Layered Extraction (PLE): A Novel Multi-Task Learning (MTL) Model for Personalized Recommendations. In Fourteenth ACM Conference on Recommender Systems (Virtual Event, Brazil) (RecSys '20). Association for Computing Machinery, New York, NY, USA, 269–278. https://doi.org/10.1145/3383313.3412236

[19] RuoxiWang, Bin Fu, Gang Fu, and MingliangWang. 2017. Deep and Cross Network for Ad Click Predictions. In Proceedings of the ADKDD'17 (Halifax, NS, Canada) (ADKDD'17). Association for Computing Machinery, New York, NY, USA, Article 12, 7 pages. https://doi.org/10.1145/3124749.3124754

[20] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. 2017. Attentional Factorization Machines: Learning the Weight of Feature Interactions via Attention Networks. In Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17). AAAI Press, Melbourne, Australia, 3119–3125.

[21] Weinan Zhang, Tianming Du, and Jun Wang. 2016. Deep Learning over Multi-field Categorical Data. In Advances in Information Retrieval, Nicola Ferro, Fabio Crestani, Marie-Francine Moens, Josiane Mothe, Fabrizio Silvestri, Giorgio Maria Di Nunzio, Claudia Hauff, and Gianmaria Silvello (Eds.). Springer International Publishing, Cham, 45–57.

[22] XianXing Zhang, Yitong Zhou, Yiming Ma, Bee-Chung Chen, Liang Zhang, and Deepak Agarwal. 2016. GLMix: Generalized Linear Mixed Models For Large-Scale Response Prediction. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (San Francisco, California, USA) (KDD '16). Association for Computing Machinery, New York, NY, USA, 363–372. https://doi.org/10.1145/2939672.2939684

[23] Xiangyu Zhao, Xudong Zheng, Xiwang Yang, Xiaobing Liu, and Jiliang Tang. 2020. Jointly Learning to Recommend and Advertise. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (Virtual Event, CA, USA) (KDD '20). Association for Computing Machinery, New York, NY, USA, 3319–3327. https://doi.org/10.1145/3394486.3403384

[24] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep Interest Evolution Network for Click-Through Rate Prediction. Proceedings of the AAAI Conference on Artificial Intelligence 33, 01 (Jul. 2019), 5941–5948. https://doi.org/10.1609/aaai.v33i01.33015941

[25] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep Interest Network for Click-Through Rate Prediction. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (London, United Kingdom) (KDD '18). Association for Computing Machinery, New York, NY, USA, 1059–1068. https://doi.org/10.1145/3219819.3219823