xDeepInt: a hybrid architecture for modeling the vector-wise and bit-wise feature interactions

Yachen Yan
yachen.yan@creditkarma.com
Credit Karma
San Francisco, California

Liubo Li
liubo.li@creditkarma.com
Credit Karma
San Francisco, California

ABSTRACT

Learning feature interactions is the key to success for the large-scale CTR prediction and recommendation. In practice, handcrafted feature engineering usually requires exhaustive searching. In order to reduce the high cost of human efforts in feature engineering, researchers propose several deep neural networks (DNN)-based approaches to learn the feature interactions in an end-to-end fashion. However, existing methods either do not learn both vector-wise interactions and bit-wise interactions simultaneously, or fail to combine them in a controllable manner. In this paper, we propose a new model, xDeepInt, based on a novel network architecture called polynomial interaction network (PIN) which learns higher-order vector-wise interactions recursively. By integrating subspace-crossing mechanism, we enable xDeepInt to balance the mixture of vector-wise and bit-wise feature interactions at a bounded order. Based on the network architecture, we customize a combined optimization strategy to conduct feature selection and interaction selection. We implement the proposed model and evaluate the model performance on three real-world datasets. Our experiment results demonstrate the efficacy and effectiveness of xDeepInt over state-of-the-art models. We open-source the TensorFlow implementation of xDeepInt: https://github.com/yanyachen/xDeepInt.

CCS CONCEPTS

• **Computing methodologies**; • **Machine learning**; • **Machine learning approaches**; • **Neural networks**;

KEYWORDS

CTR prediction, Recommendation System, Explicit Feature Interaction, Deep Neural Network

ACM Reference Format:
Yachen Yan and Liubo Li. 2020. xDeepInt: a hybrid architecture for modeling the vector-wise and bit-wise feature interactions. In *San Diego 2020: ACM SIGKDD Conference on Knowledge Discovery and Data Mining, August 22–27, 2020, San Diego, CA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

DLP-KDD 2020, August 24, 2020, San Diego, California, USA

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Click-through rate (CTR) prediction model [24] is an essential component for the large-scale recommendation system, online advertising and search ranking [7, 12, 19, 31]. In online marketplace scenario, accurately estimating CTR will enable the recommendation system to show users the items they prefer to view and explore, which has a direct impact on both short-term revenue and long-term user experience.

The input features (e.g., user id, item id, item category, site domain) of CTR prediction model are usually in a multi-field categorical format [32] and transformed via field-aware one-hot encoding and multi-hot encoding [35]. The representation of each field is a sparse binary vector. The corresponding cardinality of each field determines the dimension of the sparse vector. The concatenation of these sparse vectors naturally generates high-dimensional and sparse feature representations.

In CTR prediction model, exploring useful feature interactions plays a crucial role in improving model performance[7, 8, 16, 23, 29]. Traditionally, data scientists search and build hand-crafted feature interactions to enhance model performance based on domain knowledge. In practice, feature interactions of high-quality require expensive cost of time and human workload [12]. Furthermore, it is infeasible to manually extract all possible feature interactions given a large number of features and high cardinality [7]. Therefore, learning low-order and high-order feature interactions automatically and efficiently in a high-dimensional and sparse feature space becomes an essential problem for improving CTR prediction model performance, in both academic and industrial communities.

Deep learning models have achieved great success in recommender systems due to its great feature learning ability. Several deep learning architecture has been proposed from both academia and industry (e.g.,[7, 8, 13, 16, 21, 22, 25, 26, 29]). However, All the existing models utilize DNNs as building block for learning high-order implicit bit-wise feature interactions, without bounded order. When modeling explicit feature interactions, the exiting approaches only capture lower order explicit interactions efficiently. Learning higher order typically requires higher computational cost.

In this paper, we propose a efficient neural network-based model called xDeepInt to learn the combination of vector-wise and bit-wise multiplicative feature interactions explicitly. Motivated by polynomial regression, we design a novel Polynomial Interaction Network layers to capture bounded degree vector-wise interactions explicitly. In order to learn the bit-wise and vector-wise interactions simultaneously in a controllable manner, we combine PIN with a subspace-crossing mechanism, which gives a significant boost to our model performance and brings more flexibility. The degree

of bit-wise interactions grows with the number of subspace. In summary, we make the following contributions in this paper:

- We design a novel neural network architecture named xDeepInt that models the vector-wise interactions and bit-wise interactions explicitly and simultaneously, dispensing with jointly-trained DNN and nonlinear activation functions. The proposed model is lightweight. But it yields superior performance than many existing models with more complex structure.
- Motivated by higher-order polynomial logistic regression, we design a Polynomial-Interaction-Network (PIN) layer which learns higher-order explicit feature interactions recursively. The degrees of interactions are controlled by tuning the number of PIN layers. An analysis is conducted to demonstrate the polynomial approximation properties of PIN.
- We introduce a subspace-crossing mechanism for modeling bit-wise interactions across different fields inside PIN layer. The combination of PIN layer and the subspace-crossing mechanism allows us to control the the degree of bit-wise interactions. As the number of subspaces increases, our model can dynamically learn more fine-grained bit-wise feature interactions.
- We design an optimization strategy which is in harmony with the architecture of the proposed model. We apply Group Lasso FTRL to the embedding table, which shrinks the entire rows to zero and achieves the feature selection. To optimize weights in PIN layers, we apply FTRL directly. The sparsity in weights results in selection of feature interactions.
- We conduct a comprehensive experiment on three real-world datasets. The results demonstrate that xDeepInt outperforms existing state-of-the-art models under extreme high-dimensional and sparse settings. We also conduct a sensitivity analysis on hyper-parameter settings of xDeepInt and ablation study on integration of DNN.

2 RELATED WORK

Deep learning based models have been applied for CTR prediction problem in the industry since deep neural networks become dominant in learning the useful feature representation of the mixed-type input data and fitting model in an end-to-end fashion[31]. This merit can reduce efforts in hand-crafted feature design and automatically learn the feature interactions.

2.1 Modeling Implicit Interaction

Most of the DNN-based methods map the high-dimensional sparse categorical features and continuous features onto a low dimensional latent space in the initial step. Without designing specific model architecture, DNN-based method learns the high-order implicit feature interactions by feeding the stacked embedded feature vectors into a deep feed-forward neural network.

Deep Crossing Network [25] utilizes residual layers in the feed-forward structure to learn higher-order interactions with improved stability. Some hybrid network architectures, including Wide & Deep Network (WDL) [7], Product-based Neural Network (PNN) [21, 22], Deep & Cross Network (DCN) [29], Deep Factorization Machine (DeepFM) [8] and eXtreme Deep Factorization Machine (xDeepFM) [16] employ feed-forward neural network as their deep component to learn higher-order implicit interactions. The complement of the implicit higher-order interaction improves the performance of the network that only models the explicit interactions [2].

However, this type of approach detects all feature interactions at the bit-wise level [16] implicitly, without efficiency. And the degree of the interactions are not bounded.

2.2 Modeling Explicit Interaction

Deep & Cross Network (DCN) [29] explores the feature interactions at the bit-wise level in an explicit fashion. Specifically, each cross layer of DCN constructs all cross terms to exploits the bit-wise interactions. The number of recursive cross layers controls the degree of bit-wise feature interactions.

Some recent models explicitly learn the vector-wise feature interactions using a specific form of the vector product. Deep Factorization Machine (DeepFM) [8] combines factorization machine layer and feed-forward neural network through joint learning feature embedding. Factorization machine layer models the pairwise vector-wise interaction between feature i and feature j by the inner product of $\langle x_i, x_j \rangle = \sum_{t=1}^{k} x_{it} x_{jt}$. Then, the vector-wise interactions are concatenated with the output units of the feed-forward neural network. Product Neural Network (PNN) [21, 22] introduces the inner product layer and the outer product layer to learn explicit vector-wise interactions and bit-wise interactions respectively. xDeepFM [16] learns the explicit vector-wise interaction by using Compressed Interaction Network (CIN) which has an RNN-like architecture and learns all possible vector-wise interactions using Hadamard product. The convolutional filters and the pooling mechanism are used to extract information. FiBiNET [13] utilizes Squeeze-and-Excitation network to dynamically learn the importance of features and models the feature interactions via bilinear function.

In the recent research of the sequencing model, the architecture of the Transformer [27] has been widely used to understand the associations between relevant features. With different layers of the multi-head self-attentive neural networks, AutoInt [26] can learn different orders of feature combinations of input features. Residual connections [9, 28] are added to carry through different degrees of feature interaction.

The aforementioned approaches learn explicit feature interactions by using outer product, kernel product or multi-head self-attention, which require expensive computational cost.

3 MODEL

In this section, we give an overview of the architecture of xDeepInt. First, we introduce input and embedding layer, which map continuous features and high-dimensional categorical features onto a dense vector. Second, we present the Polynomial Interaction Network(PIN) which utilizes iterative interaction layers with residual connections [9, 28] to explicitly learn the vector-wise interactions. Third, we implement the subspace-crossing mechanism to model bit-wise interaction. The number subspaces controls the degree of mixture of bit-wise and vector-wise interactions.

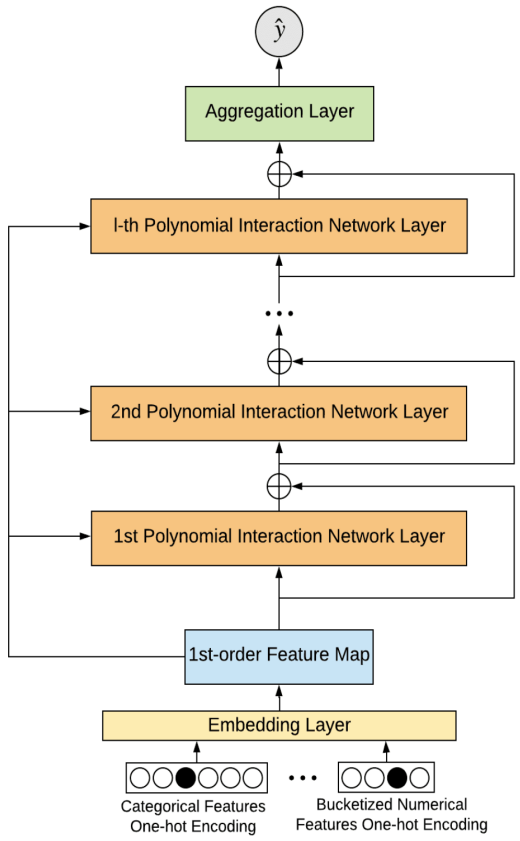

Figure 1: The architecture of unrolled Polynomial Interaction Network with residual connections

3.1 Embedding Layer

In large-scale recommendation system, inputs include both continuous features and categorical features. Categorical features are often directly encoded by one-hot encoding, which results in an excessively high-dimensional and sparse feature space. Suppose we have F fields. In our feature preprocessing step, we bucketize all the continuous features to equal frequency bins, then embed the bucketized continuous features and categorical features to same latent space R^K,

$$\mathbf{x}_f^e = \mathbf{x}_f^o \mathbf{V}_f,$$

where $\mathbf{x}_f^e = [x_{f,1}, x_{f,2}, \cdots, x_{f,K}]$, $x_{f,k}$ is the k-th bit of the f-th field of the embedding feature map, $\mathbf{V}_f$ is an embedding matrix for field f, and $\mathbf{x}_f^o$ is a one-hot vector. Lastly, we stack F embedding vectors and obtain an F-by-K input feature map X_0:

$$X_0 = \begin{bmatrix} \mathbf{x}_1^e \\ \mathbf{x}_2^e \\ \vdots \\ \mathbf{x}_F^e \end{bmatrix}$$

3.2 Polynomial Interaction Network

Consider a l-th order polynomial with f variables of the following form:

$$\Pi_{j=1}^{l}(\sum_{i=1}^{f} a_{ij}x_i + b_j). \tag{1}$$

This polynomial contains all possible multiplicative combinations of x_i's with order less than or equal to l and has an iterative form:

$$F^{(l-1)}(x_1, \cdots, x_f)(\sum_{i=1}^{f} a_{il}x_i + b_l) \tag{2}$$

where $F^{(l-1)}(x_1, \cdots, x_f) = \Pi_{j=1}^{l-1}(\sum_{i=1}^{f} a_{ij}x_i + b_j)$. Motivated by the iterative form, we propose polynomial interaction network defined by the following formula:

$$\begin{aligned} X_l &= f(W_{l-1}, X_{l-1}, X_0) + X_{l-1} \\ &= X_{l-1} \circ (W_{l-1}X_0) + X_{l-1} \\ &= X_{l-1} \circ [W_{l-1}X_0 + \mathbf{1}] \end{aligned} \tag{3}$$

where $\circ$ denotes the Hadamard product. For instance, $[a_{i,j}]_{m\times n} \circ [b_{i,j}]_{m\times n} = [a_{i,j}b_{i,j}]_{m\times n}$. $W_{l-1} \in R^{F\times F}$ and $\mathbf{1} \in R^{F\times K}$ with all entries are equal to one. $X_{l-1}, X_l \in R^{F\times K}$ are the output matrices of (l-1)-th and l-th interaction layer. Like (1), the l-th PIN layer's output is the weighted sum of all vector-wise feature interactions of order less than or equal to l.

The architecture of the polynomial interaction network is motivated by the following aspects.

First, the polynomial interaction network has a recursive structure. The outputs of the current layer are built upon the previous layer's outputs and the first order feature map, ensuring that higher-order feature interactions are based on lower-order feature interactions from previous layers.

Second, we use the Hadamard product to model the explicit vector-wise interaction, which brings us more flexibility in crossing the bits of each dimension in shared latent space and preserves more information of each degree of feature interactions.

Third, we build a field aggregation layer $Agg^{(l)}(X) = W_lX$ which combines the feature map at the vector-wise level using a linear transformation W_l. Each vector of the field aggregation feature map can be viewed as a combinatorial feature vector constructed by the weighted sum of the input feature map. Then we take the Hadamard product of the output of the previous layer and field aggregation feature map for this layer. This operation allows us to explore all possible l-th order polynomial feature interactions based on existing (l-1)-th order feature interactions.

Last, we utilize residual connections [9, 28] in the polynomial interaction network, allowing a different degree of vector-wise polynomial feature interactions to be combined, including the first feature map. Since the polynomial interaction layer's outputs contain all degree of feature interactions, the skipped connection enable next polynomial interaction layer to focus on searching useful higher-order feature interactions while complementing lower-order feature interactions. As the number of layer increases, the degree of the polynomial feature interactions increases. The recurrent architecture of the proposed polynomial interaction network enables to bound the degree of polynomial feature interactions.

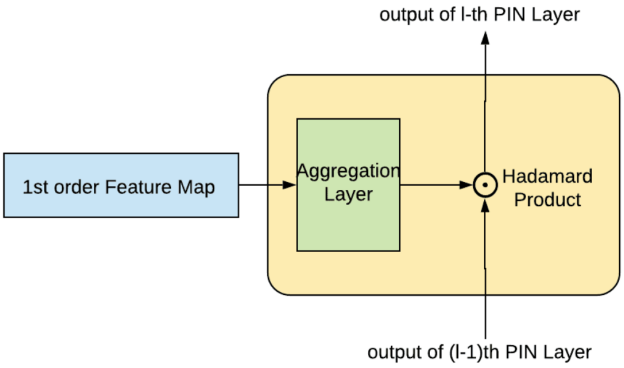

Figure 2: Details of PIN layer

The aggregation layer is defined by $Agg^{(l)}(X_0) = W_l X_0$. The PIN takes the Hadamard product of the aggregated 1st-order feature map and the output of the previous PIN layer to generate the higher order vector-wise interaction.

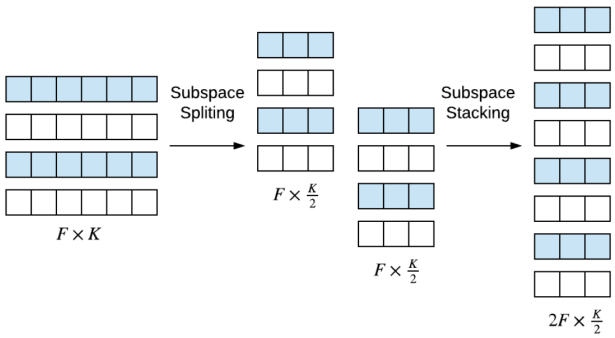

Figure 3: Subspace-crossing mechanism

3.3 Subspace-crossing Mechanism

The Polynomial Interaction Network (PIN) models the vector-wise interactions. However, PIN does not learn the bit-wise interaction in the shared latent embedding space. In order to cross the bits of different embedding dimensions, we propose the subspace-crossing mechanism which allows xDeepInt to learn the bit-wise interactions. Suppose we split the embedding space into h sub-spaces, the input feature map X_0 is then represented by h sub-matrices as follow:

$$X_0 = [X_{0,1}, X_{0,2}, \cdots, X_{0,h}] \tag{4}$$

where $X_{0,i} \in R^{F \times K/h}$ and $i = 1, 2 \cdots, h$. Next, we stack all sub-matrices at the field dimension and construct a stacked input feature map $X_0^{'} \in R^{(F*h) \times (K/h)}$.

$$X_0^{'} = \begin{bmatrix} X_{0,1} \\ X_{0,2} \\ \vdots \\ X_{0,h} \end{bmatrix}, \tag{5}$$

where $X_{0,j} \in R^{F \times (K/h)}$ and h denotes the number of sub-spaces. By splitting the embedding vector of each field to h sub-vectors and stacking them together, we can align bits of different embedding dimension and create the vector-wise interactions on stacked sub-embeddings. Accordingly, we feed $X_0^{'}$ into the Polynomial Interaction Network (PIN):

$$\begin{aligned} X_1^{'} &= X_0^{'} \circ [W_0^{'} X_0^{'} + \mathbf{1}] \\ &\vdots \\ X_l^{'} &= X_{l-1}^{'} \circ [W_{l-1}^{'} X_0^{'} + \mathbf{1}] \end{aligned} \tag{6}$$

where $W_l^{'} \in R^{(F*h) \times (F*h)}$, $\mathbf{1} \in R^{(F*h) \times (K/h)}$ and $X_l^{'} \in R^{(F*h) \times (K/h)}$.

The field aggregation of feature map and the multiplicative interactions building by Hadamard product are both of the vector-wise level in vanilla PIN layers. The subspace-crossing mechanism enhanced PIN takes the h aligned subspaces as input, so that encouraging the PIN to capture the explicit bit-wise interaction by crossing features of the difference subspaces. The number of subspaces h controls the complexity of bit-wise interactions. Larger h helps the model to learn more complex feature interactions.

3.4 Output Layer

The output of Polynomial Interaction Network is a feature map that consists of different degree of feature interactions, including raw input feature map reserved by residual connections and higher-order feature interactions learned by PIN. For the final prediction, we merely use formula as follows:

$$\hat{y} = \sigma\big((W_{out} X_l + b\mathbf{1}^T)\mathbf{1}\big) \tag{7}$$

where σ is the sigmoid function, $W_{out} \in R^{1 \times F}$ is a feature map aggregation vector that linearly combines all the features in the feature map, $\mathbf{1} \in R^K$ and $b \in R$ is the bias.

3.5 Optimization and Regularization

For optimization, we use Group Lasso Follow The Regularized Leader (G-FTRL) [20] as the optimizer for the embedding layers for feature selection, and Follow The Regularized Leader (FTRL) [19] as the optimizer for the PIN layers for interaction selection.

Group lasso FTRL regularizes the entire embedding vector of insignificant features in each field to exactly zero, which essentially conducts feature selection and brings more training efficiency for industrial settings. The group lasso regularization is applied prior to the subspace splitting mechanism such that feature selection is consistent between each subspaces.

FTRL regularizes the single elements of weight kernel in PIN layers to exactly zero, which excludes insignificant feature interactions and regularizes the complexity of the model.

This optimization strategy takes advantages of the properties of the different optimizers and achieves row-wise sparsity and element-wise sparsity at embedding table and weight kernel respectively. Therefore, it improves generalization ability and efficiency for both training and serving. It also plays an important role in model compression.

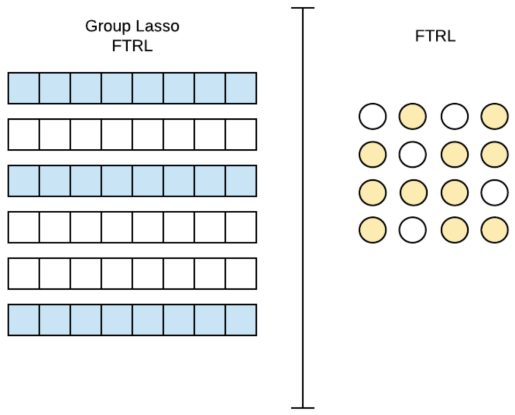

Figure 4: Group Lasso FTRL v.s. FTRL

The Group Lasso FTRL regularizes the embedding table with group-wise sparsity. FTRL regularizes the weight kernels of PIN layer with element-wise sparsity.

3.6 Training

The loss function we use is Log Loss,

$$\text{LogLoss} = -\frac{1}{N}\sum_{i=1}^{N}(y_i log(\hat{y}_i) + (1-y_i)log(1-\hat{y}_i)), \quad (8)$$

where y_i is the true label and $\hat{y}_i$ is the estimated click through rate. N is the total number of training examples.

3.7 Difference Between PIN and DCN

PIN and DCN both have an iterative form. However, the two network architectures are quite different in extracting feature interactions.

$$\begin{aligned} x_l &= f_{DCN}(x_{l-1}, w_{l-1}, b_{l-1}) \\ &= x_0 x_{l-1}^T w_{l-1} + b_{l-1} + x_{l-1} \end{aligned} \quad (9)$$

For DCN, feature map are flattened and concatenated as a single vector. All higher order bit-wise interactions are firstly constructed by the term $x_0 x_{l-1}^T$, and then aggregated by a linear regression for next layer. This structure results in a special format of the output. As discussed in [16], the output the DCN layer is a scalar multiple of x_0. [16] also pointed out the downsides: 1) the output of DCN is in a special form, with each hidden layer is a scalar multiple of x_0 and thus limits expressive power; 2) interactions only come in a bit-wise fashion.

PIN constructs vector-wise feature interaction using Hadamard product, which preserve the information at vector-wise level. In order to allow different fields to cross at vector level, PIN firstly aggregates the input feature map by a linear transformation $W_{l-1}X_0$ for each iterative PIN layer and build interactions by term $X_{l-1} \circ W_{l-1}X_0$. Accordingly, PIN keeps the vector-wise structure of feature interactions and does not limit the output to a scalar multiple of X_0. Moreover, each PIN layer is directly connected with input feature map X_0, which improves model trainability. We also prove PIN's polynomial approximation property in later section.

3.8 xDeepInt Analysis

In this section, we analyze polynomial approximation property of the proposed xDeepInt model. We consider an xDeepInt model with l PIN layers, a subspace crossing mechanism with h subspaces and F input feature with the same embedding size K.

3.8.1 Polynomial Approximation. In order to understand how PIN exploits the vector-wise interactions, we examine the polynomial approximation properties of PIN. Let $\mathbf{X}^{(0)} \in R^{F\times K}$ be the embedded feature vector with $\mathbf{x}_i = [x_{i1}, \cdots, x_{iK}]$ being the i-th row. $\mathbf{x}_i^{(l)} = [x_{i1}^{(l)}, \cdots, x_{iK}^{(l)}]$ is the i-th row of the output of l-th layer. Then, $x_{ik}^{(l)}$ has the following explicit form:

$$x_{ik}^{(l)} = x_{ik}^{(0)} \prod_{r=0}^{l-1}(\sum_{i=1}^{F} w_{ij}^{(r)} x_{jk}^{(0)} + 1), \text{for } k = 1, \cdots, K \quad (10)$$

where $W^{(r)} = [w_{ij}^{(r)}]_{1\le i,j\le F}$ is the weight matrix at r-th PIN layer. The product $\prod_{r=0}^{l-1}(\sum_{i=1}^{F} w_{ij}^{(r)} x_{jk}^{(0)} + 1)$ is the weighted sum of all possible crossed terms of the embbeded input at the k-th bit having order less than or equal to $l-1$. Thus, $x_{ik}^{(l)}$ is the weighted sum of all crossed terms that contains $x_{ik}^{(0)}$ and has the order less than or equal to l.

For the bit-wise interaction modeled by subspace-crossing mechanism, we consider the case where the number of subspaces equals to the embedding size K. In this extreme case, each row of $W_0' X_0'$ is a weighted sum of all bits in all fields. This design allows the combination of embedded features at different bits. To be more explicit, we consider the stacked input feature map with $h = K$

$$X_0' = \begin{bmatrix} X_{0,1} \\ X_{0,2} \\ \vdots \\ X_{0,K} \end{bmatrix}, \quad (11)$$

where $X_{0,i} = \begin{bmatrix} x_{i,1}^{(0)} \\ x_{i,2}^{(0)} \\ \vdots \\ x_{i,F}^{(0)} \end{bmatrix} \in R^F$ .The weight matrix W_0' is given by

$$W_0' = \begin{bmatrix} w_{1,1}^{(0)} & w_{1,2}^{(0)} & \cdots & w_{1,K}^{(0)} & w_{1,K+1}^{(0)} & \cdots & w_{1,FK}^{(0)} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ w_{FK,1}^{(0)} & w_{FK,2}^{(0)} & \cdots & w_{FK,K}^{(0)} & w_{FK,K+1}^{(0)} & \cdots & w_{FK,FK}^{(0)} \end{bmatrix} \in R^{FK\times FK}. \quad (12)$$

Thus, the k-th row of $W_0' X_0' + 1$ has the following form:

$$\sum_{i=1}^{F}\sum_{j=1}^{K} w_{k,(i-1)K+j}^{(0)} x_{i,j} + 1. \quad (13)$$

This is a linear combination of bits in all fields, which allows the PIN exploit all crossing of the feature map at bit-wise level. For example, the $[(i-1)K + j]$-th row of X_l' is given by

$$x_{i,j}^{(l)} = x_{i,j}^{(0)} \prod_{r=0}^{l-1}\sum_{i=1}^{F}\sum_{j=1}^{K} w_{k,(i-1)K+j}^{(r)} x_{i,j} + 1 \quad (14)$$

$x_{i,j}^{(l)}$ is the weighted sum of all crossed terms that contains $x_{i,j}^{(0)}$ and has the order less than or equal to l.

3.8.2 Time Complexity. The cost of computing feature maps $W_{l-1}X$ at l-th PIN layer is $O(hF^2K)$. For a L-layers xDeepInt model, the total cost of feature maps is $O(LhF^2K)$. The additional cost is from the Hadamard product and residual connection, which is $O(LFK)$. In practice, h is not too large. Hence, the total time complexity mainly relies on the number of fields F and embedding size K. For an L layers DNN with each layer has D_k hidden nodes, the time complexity is $O(FK \times D_1 \times D_2 + \sum_{k=2}^{L-1} D_{k-1} D_k D_{k+1})$. The time complexity of xDeepInt relies on the number of subspaces. Therefore, xDeepInt has higher time complexity than DNN when modelling higher degrees of bit-wise interactions.

3.8.3 Space Complexity. The embedding layer contains $\sum_{f=1}^{F} K \times C_f$ parameters, where C_f is the cardinality of f-th field. The output layer aggregates the feature map at the last PIN layer. Hence, the output layers require $F + 1$ parameters. The subspace crossing mechanism needs $h^2 \times F^2$ parameters at each PIN layer, which is the exact size of the weight matrix W_r' with $0 \leq r \leq l-1$. There are $(K/h) \times k' + h^2 \times F^2 \times l$ parameters in l PIN layers. Usually, we will pick a small h to control the model complexity and k' is comparable to K. Accordingly, the overall complexity of the xDeepInt is approximate $O(h^2 \times F^2 \times l)$, which is affected by the embedding size K heavily. A plain L layers DNN with each layer has D_k hidden nodes requires $FK \times D_1 + \sum_{k=2}^{L} D_{k-1} D_k$ parameters. The complexity mainly depends on the embedding size and the number of hidden nodes at each layer. To reduce the space complexity of xDeepInt, we can apply the method introduced in [16]. The space complexity of the model can be further reduced by exploiting a L-order decomposition and replace the weight matrix W_r' with two low-rank matrices.

4 EXPERIMENTS

In this section, we focus on evaluating the effectiveness of our proposed models and answering the following questions:

- **Q1**: How does our proposed xDeepInt perform in CTR prediction problem? Is it effective and efficient under extreme high-dimensional and sparse data settings?
- **Q2**: How do different hyper-parameter settings influence the performance of xDeepInt?
- **Q3**: Will modeling implicit higher-order feature interactions further improve the performance of xDeepInt?

4.1 Experiment Setup

4.1.1 Datasets. We evaluate our proposed model on three public real-world datasets widely used for research.

1. Avazu.[1] Avazu dataset is from kaggle competition in 2015. Avazu provided 10 days of click-through data. We use 21 features in total for modeling. All the features in this dataset are categorical features.

2. Criteo.[2] Criteo dataset is from Kaggle competition in 2014. Criteo AI Lab officially released this dataset after, for academic use. This dataset contains 13 numerical features and 26 categorical features. We discretize all the numerical features to integers by transformation function $\lfloor Log\,(V^2) \rfloor$ and treat them as categorical features, which is conducted by winning team of Criteo competition.

3. iPinYou.[3] iPinYou dataset is from iPinYou Global RTB(Real-Time Bidding) Bidding Algorithm Competition in 2013. We follow the data processing steps of [33] and consider all 16 categorical features.

For all the datasets, we randomly split the examples into three parts: 70% is for training, 10% is for validation, and 20% is for testing. We also remove each categorical features' infrequent levels appearing less than 20 times to reduce sparsity issue. Note that we want to compare the effectiveness and efficiency on learning higher-order feature interactions automatically, so we do not do any feature engineering but only feature transformation, e.g., numerical feature bucketing and categorical feature frequency thresholding.

4.1.2 Evaluation Metrics. We consider AUC and LogLoss for evaluating the performance of the models.

LogLoss LogLoss is both our loss function and evaluation metric. It measures the average distance between predicted probability and true label of all the examples.

AUC Area Under the ROC Curve (AUC) measures the probability that a randomly chosen positive example ranked higher by the model than a randomly chosen negative example. AUC only considers the relative order between positive and negative examples. A higher AUC indicates better ranking performance.

4.1.3 Competing Models. We compare xDeepInt with following models: LR(logistic regression) [18, 19], FM(factorization machine) [23], DNN (plain multilayer perceptron), Wide & Deep [7], DeepCrossing [25], DCN (Deep & Cross Network) [29], PNN (with both inner product layer and outer product layer) [21, 22], DeepFM [8], xDeepFM [16], AutoInt [26] and FiBiNET [13]. Some of the models are state-of-the-art models for CTR prediction problem and are widely used in the industry.

4.1.4 Reproducibility. We implement all the models using Tensorflow [1]. The mini-batch size is 4096, and the embedding dimension is 16 for all the features. For optimization, we employ Adam [15] with learning rate set to 0.001 for all the neural network models, and we apply FTRL [18, 19] with learning rate as 0.01 for both LR and FM. For regularization, we choose L2 regularization with $\lambda = 0.0001$ for dense layer. Grid-search for each competing model's hyper-parameters is conducted on the validation dataset. The number of DNN, Cross, CIN, Interacting layers is from 1 to 4. The number of neurons ranges from 128 to 1024. All the models are trained with early stopping and are evaluated every 2000 training steps.

For the hyper-parameters search of xDeepInt, The number of recursive feature interaction layers is from 1 to 4. For the number of sub-spaces h, the searched values are 1, 2, 4, 8 and 16. Since our embedding size is 16, this range covers from complete vector-wise interaction to complete bit-wise interaction. We use G-FTRL optimizer for embedding table and FTRL for PIN layers with learning rate as 0.01.

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

4.2 Model Performance Comparison (Q1)

Table 1: Performance Comparison of Different Algorithms on Criteo, Avazu and iPinYou Dataset.

	Criteo		Avazu		iPinYou	
Model	AUC	LogLoss	AUC	LogLoss	AUC	LogLoss
LR	0.7924	0.4577	0.7533	0.3952	0.7692	0.005605
FM	0.8030	0.4487	0.7652	0.3889	0.7737	0.005576
DNN	0.8051	0.4461	0.7627	0.3895	0.7732	0.005749
Wide&Deep	0.8062	0.4451	0.7637	0.3889	0.7763	0.005589
DeepFM	0.8069	0.4445	0.7665	0.3879	0.7749	0.005609
DeepCrossing	0.8068	0.4456	0.7628	0.3891	0.7706	0.005657
DCN	0.8056	0.4457	0.7661	0.3880	0.7758	0.005682
PNN	0.8083	0.4433	0.7663	0.3882	0.7783	0.005584
xDeepFM	0.8077	0.4439	0.7668	0.3878	0.7772	0.005664
AutoInt	0.8053	0.4462	0.7650	0.3883	0.7732	0.005758
FiBiNET	0.8082	0.4439	0.7652	0.3886	0.7756	0.005679
xDeepInt	**0.8111**	**0.4408**	**0.7675**	**0.3872**	**0.7791**	**0.005565**

The overall performance of different models is listed in Table 1. We have the following observations in terms of model effectiveness:

- LR is generally worse than other algorithms, which indicates that learning higher-order feature interactions is essential for CTR model performance.
- FM brings the most significant boost in performance while we increase model complexity. This reveals the importance of learning explicit vector-wise feature interactions.
- Models that combining vector-wise and bit-wise interactions together consistently outperform other models. This phenomenon indicates that both types of feature interactions are essential to prediction performance and compensate each other.
- xDeepInt achieves the best prediction performance among all models. However, different datasets favor feature interactions of different degrees and bit-wise feature interactions of different complexity. The superior performance of our model could attribute to the fact that xDeepInt model the bounded degree of polynomial feature interactions by adjusting the depth of PIN and achieve different complexity of bit-wise feature interactions by changing the number of sub-spaces.

4.3 Hyper-Parameter Study (Q2)

In order to have deeper insights of the proposed model, we conduct experiments on three datasets and compare several variants of xDeepInt on different hyper-parameter settings.

4.3.1 Depth of Network. The depth of PIN determines the order of feature interactions. Table 2 illustrates the performance change with respect to the number of PIN layers. When the number of layers set to 0, our model is equivalent to logistic regression and no interactions are learned. The performance of xDeepInt achieves the best when the number of layers is about 3 or 4. In this experiment, we set the number of sub-spaces as 1, to disable the bit-wise feature interactions.

Table 2: Impact of hyper-parameters: number of layers

	#Layers	0	1	2	3	4	5
Criteo	AUC	0.7921	0.8038	0.8050	0.8057	**0.8063**	0.8061
	LogLoss	0.4580	0.4477	0.4466	0.4461	**0.4452**	0.4454
Avazu	AUC	0.7536	0.7654	0.7664	**0.7675**	0.7670	0.7662
	LogLoss	0.3951	0.3888	0.3879	**0.3872**	0.3875	0.3883
iPinYou	AUC	0.7690	0.7740	0.7775	**0.7791**	0.7783	0.7772
	LogLoss	0.005604	0.005576	0.005569	**0.005565**	0.005580	0.005571

4.3.2 Number of Sub-spaces. The subspace-crossing mechanism enables the proposed model to control the complexity of bit-wise interactions. Table 3 demonstrates that subspace-crossing mechanism boosts the performance. In this experiment, we set the number of PIN layers as 3, which is generally a good choice but not the best setting for each dataset.

Table 3: Impact of hyper-parameters: number of sub-spaces

	#Sub-spaces	1	2	4	8	16
Criteo	AUC	0.8072	0.8081	0.8089	0.8096	**0.8101**
	LogLoss	0.4445	0.4435	0.4425	0.4421	**0.4418**
Avazu	AUC	0.7660	0.7668	**0.7674**	0.7672	0.7668
	LogLoss	0.3880	0.3877	**0.3875**	0.3878	0.3879
iPinYou	AUC	0.7772	0.7783	**0.7788**	0.7787	0.7784
	LogLoss	0.005590	0.005583	**0.005568**	0.005572	0.005580

4.3.3 Activation Function. By default, we use linear activation function on neurons of PIN layers. We also would like to explore how different activation function of PIN affect the performance. Table 4 shows that the linear activation function is the most performant one for the PIN. We study the effect of activation function on Criteo dataset.

Table 4: Impact of hyper-parameters: activation function

	AUC	LogLoss
linear	**0.8111**	**0.4408**
tanh	0.8100	0.4418
sigmoid	0.8082	0.4434
softplus	0.8080	0.4436
swish	0.8100	0.4418
relu	0.8098	0.4419
leaky relu	0.8102	0.4415
elu	0.8099	0.4418
selu	0.8100	0.4418

4.3.4 Optimizer. We also build our model with Adam optimizer, same as all the competing models, to compare with our G-FTRL and FTRL combined optimization strategy. Table 5 shows that our G-FTRL and FTRL combined optimization strategy achieves better performance. Table 6 shows that our optimization strategy gets higher degree of feature sparse ratio (ratio of all zero embedding vectors in embedding table) and sparse ratio (ratio of zero weights in PIN layers), which results in lightweight model. One thing should be noted is that xDeepInt still achieves the best prediction performance among all models when using Adam optimizer, which demonstrates the effectiveness of xDeepInt architecture.

Table 5: Impact of hyper-parameters: optimizer

Dataset	Model	LogLoss	AUC
Criteo	G-FTRL/FTRL	**0.4408**	**0.8111**
	Adam	0.4415	0.8105
Avazu	G-FTRL/FTRL	**0.3872**	**0.7675**
	Adam	0.3873	0.7674
iPinYou	G-FTRL/FTRL	**0.005565**	**0.7791**
	Adam	0.005583	0.7784

Table 6: Analysis of model sparsity

Dataset	Feature Sparse ratio	Weight Sparse ratio
Criteo	0.6506	0.1030
Avazu	0.2193	0.0448
iPinYou	0.8274	0.0627

4.4 Ablation Study: Integrating Implicit Interactions (Q3)

In this section, we conduct ablation study comparing the performance of our proposed model with and without integrating implicit feature interactions.

Feed-forward neural network is widely used by various model architectures for learning implicit feature interactions. In this experiment, we jointly train xDeepInt with a three-layer feed-forward neural network and name the combined model as xDeepInt+ to compare with vanilla xDeepInt.

Table 7 compares vanilla xDeepInt and xDeepInt+. We observe that the jointly-trained feed-forward neural network does not boost the performance of vanilla xDeepInt. The reason is that vanilla xDeepInt model has already learned bit-wise interactions through the subspace-crossing mechanism. Thus, feed-forward neural network does not bring in additional predictive power.

Table 7: Ablation study comparing the performance of xDeepInt with and without integrating DNN

Dataset	Model	LogLoss	AUC
Criteo	xDeepInt	**0.4408**	**0.8111**
	xDeepInt+	0.4412	0.8107
Avazu	xDeepInt	**0.3872**	**0.7675**
	xDeepInt+	0.3874	0.7673
iPinYou	xDeepInt	**0.005565**	**0.7791**
	xDeepInt+	0.005581	0.7787

5 CONCLUSION

In this paper, we design a novel network layer named polynomial interaction network (PIN), which learns the higher order vector-wise feature interactions on the embedding space. By incorporating PIN with the subspace-crossing mechanism, our proposed model xDeepInt learns bit-wise and vector-wise feature interactions of bounded degree simultaneously in controllable manner. We add residual connections to PIN layers, such that the output of each layer is an ensemble of the low-order and high-order interactions. The degree of interaction is controlled by the number of PIN layers, and the complexity of bit-wise interaction is controlled by the number of sub-spaces. Additionally, an optimization method is introduced to performs feature selection and interaction selection based on the network structure. Our experimental result demonstrates that the proposed xDeepInt outperforms existing state-of-art methods on real-world datasets. To our best knowledge, xDeepInt is the first neural network architecture that achieves state-of-art performance without integration of feed-forward neural network using non-linear activation functions.

We have multiple directions of future work. First, the proposed model only focuses on modeling fixed-length feature vectors. In order to model historical and sequential behavior in recommendation systems[34, 35], We are interested in making our model architecture applicable to variable-length feature vectors. Second, We would like to extend the application of polynomial interaction layers to more modeling scenarios and exploit PIN's potential on other problems. Third, the model is fully explainable when the subspace crossing mechanism is disable. The explainability of the model is another direction of future work.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. 265–283.

[2] Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Jia Li, Vince Gatto, and Ed H Chi. 2018. Latent cross: Making use of context in recurrent recommender systems. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 46–54.

[3] Mathieu Blondel, Akinori Fujino, Naonori Ueda, and Masakazu Ishihata. 2016. Higher-order factorization machines. In *Advances in Neural Information Processing Systems*. 3351–3359.

[4] Mathieu Blondel, Masakazu Ishihata, Akinori Fujino, and Naonori Ueda. 2016. Polynomial networks and factorization machines: New insights and efficient training algorithms. *arXiv preprint arXiv:1607.08810* (2016).

[5] Patrick PK Chan, Xian Hu, Lili Zhao, Daniel S Yeung, Dapeng Liu, and Lei Xiao. 2018. Convolutional Neural Networks based Click-Through Rate Prediction with Multiple Feature Sequences.. In *IJCAI*. 2007–2013.

[6] Chen Cheng, Fen Xia, Tong Zhang, Irwin King, and Michael R Lyu. 2014. Gradient boosting factorization machines. In *Proceedings of the 8th ACM Conference on Recommender systems*. ACM, 265–272.

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

[8] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247* (2017).

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[10] Xiangnan He and Tat-Seng Chua. 2017. Neural Factorization Machines for Sparse Predictive Analytics.(2017). (2017).

[11] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 173–182.

[12] Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, et al. 2014. Practical lessons from predicting clicks on ads at facebook. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*. ACM, 1–9.

[13] Tongwen Huang, Zhiqi Zhang, and Junlin Zhang. 2019. FiBiNET: Combining Feature Importance and Bilinear feature Interaction for Click-Through Rate Prediction. *arXiv preprint arXiv:1905.09433* (2019).

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

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

[16] 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 & Data Mining*. ACM, 1754–1763.

[17] Xiaoliang Ling, Weiwei Deng, Chen Gu, Hucheng Zhou, Cui Li, and Feng Sun. 2017. Model ensemble for click prediction in bing search ads. In *Proceedings of the 26th International Conference on World Wide Web Companion*. International World Wide Web Conferences Steering Committee, 689–698.

[18] H Brendan McMahan. 2011. Follow-the-regularized-leader and mirror descent: Equivalence theorems and l1 regularization. (2011).

[19] H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, 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*. ACM, 1222–1230.

[20] Xiuyan Ni, Yang Yu, Peng Wu, Youlin Li, Shaoliang Nie, Qichao Que, and Chao Chen. 2019. Feature Selection for Facebook Feed Ranking System via a Group-Sparsity-Regularized Training Algorithm. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2085–2088.

[21] Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. 2016. Product-based neural networks for user response prediction. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 1149–1154.

[22] Yanru Qu, Bohui Fang, Weinan Zhang, Ruiming Tang, Minzhe Niu, Huifeng Guo, Yong Yu, and Xiuqiang He. 2018. Product-Based Neural Networks for User Response Prediction over Multi-Field Categorical Data. *ACM Transactions on Information Systems (TOIS)* 37, 1 (2018), 5.

[23] Steffen Rendle. 2010. Factorization machines. In *2010 IEEE International Conference on Data Mining*. IEEE, 995–1000.

[24] 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*. ACM, 521–530.

[25] Ying Shan, T Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and JC Mao. 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*. ACM, 255–262.

[26] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. 2018. AutoInt: Automatic Feature Interaction Learning via Self-Attentive Neural Networks. *arXiv preprint arXiv:1810.11921* (2018).

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. 5998–6008.

[28] Andreas Veit, Michael J Wilber, and Serge Belongie. 2016. Residual networks behave like ensembles of relatively shallow networks. In *Advances in neural information processing systems*. 550–558.

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

[30] 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. *arXiv preprint arXiv:1708.04617* (2017).

[31] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)* 52, 1 (2019), 5.

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

[33] Weinan Zhang, Shuai Yuan, Jun Wang, and Xuehua Shen. 2014. Real-time bidding benchmarking with ipinyou dataset. *arXiv preprint arXiv:1407.7073* (2014).

[34] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2018. Deep Interest Evolution Network for Click-Through Rate Prediction. *arXiv preprint arXiv:1809.03672* (2018).

[35] 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 & Data Mining*. ACM, 1059–1068.

[36] Jie Zhu, Ying Shan, JC Mao, Dong Yu, Holakou Rahmanian, and Yi Zhang. 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*. ACM, 1703–1711.