DCAP: Deep Cross Attentional Product Network for User Response Prediction

Zekai Chen

Department of Computer Science George Washington University Washington, DC, USA zech_chan@gwu.edu

Xiao Zhang School of Computer Science and Technology Shandong University, China xiaozhang@sdu.edu.cn Fangtian Zhong Department of Computer Science George Washington University Washington, DC, USA squareky_zhong@gwu.edu

Robert Pless Department of Computer Science George Washington University Washington, DC, USA pless@gwu.edu

Zhumin Chen

School of Computer Science and Technology Shandong University, China chenzhumin@sdu.edu.cn

Xiuzhen Cheng Department of Computer Science George Washington University Washington, DC, USA cheng@gwu.edu

ABSTRACT

User response prediction, which aims to predict the probability that a user will provide a predefined positive response in a given context such as clicking on an ad or purchasing an item, is crucial to many industrial applications such as online advertising, recommender systems, and search ranking. For these tasks and many other machine learning tasks, an indispensable part of success is feature engineering, where cross features are a significant type of feature transformations. However, due to the high dimensionality and super sparsity of the data collected in these tasks, handcrafting cross features is inevitably time expensive. Prior studies in predicting user response leveraged the feature interactions by enhancing feature vectors with products of features to model second-order or high-order cross features, either explicitly or implicitly. However, these existing methods can be hindered by not learning sufficient cross features due to model architecture limitations or modeling all high-order feature interactions with equal weights. Different features should contribute differently to the prediction, and not all cross features are with the same prediction power.

This work aims to fill this gap by proposing a novel architecture Deep Cross Attentional Product Network (DCAP), which keeps cross network's benefits in modeling high-order feature interactions explicitly at the vector-wise level. By computing the inner product or outer product between attentional feature embeddings and original input embeddings as each layer's output, we can model cross features with a higher degree of order as the network's depth increases. We concatenate all the outputs from each layer, which further helps the model capture much information on cross features of different orders. Beyond that, it can differentiate the importance of different cross features in each network layer inspired by the multi-head attention mechanism and Product Neural Network

CIKM '21, November 1-5, 2021, Virtual Event, QLD, Australia

© 2021 Association for Computing Machinery.

ACM ISBN 978-1-4503-8446-9/21/11...\$15.00

https://doi.org/10.1145/3459637.3482246

(PNN), allowing practitioners to perform a more in-depth analysis of user behaviors. Additionally, our proposed model can be easily implemented and train in parallel. We conduct comprehensive experiments on three real-world datasets. The results have robustly demonstrated that our proposed model DCAP achieves superior prediction performance compared with the state-of-the-art models. Public codes are available at https://github.com/zachstarkk/DCAP.

CCS CONCEPTS

• Information systems \rightarrow Data mining; Computational advertising; • Computing methodologies \rightarrow Machine learning algorithms.

KEYWORDS

user response prediction; recommender system; self-attention; cross feature modeling

ACM Reference Format:

Zekai Chen, Fangtian Zhong, Zhumin Chen, Xiao Zhang, Robert Pless, and Xiuzhen Cheng. 2021. DCAP: Deep Cross Attentional Product Network for User Response Prediction. In Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, QLD, Australia. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3459637.3482246

1 INTRODUCTION

With the continuous and rapid growth of online service platforms, user response prediction (URP) has played an increasingly important role as the central problem of many online applications, such as online advertising [10, 14, 21], recommender systems [6, 15], and web search [29]. In online advertising, quantifying user intent allows advertisers or social platforms to target ads' right users. It leads to the judicious use of multi-billion marketing dollars and also renders a pleasant user experience. In recommender systems, correctly predicting the *rating* or *preference* a user would respond to an item can also create a delightful user experience while driving incremental revenue.

As illustrated in Figure 1, based on the user level features collected within a particular historical window of any specific task, the predictive system can estimate in advance how likely a user will provide a predefined positive response, e.g., clicks an ad (also

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

known as click-through rate, CTR), likes a post, purchases an item, etc., in a given context [23].

As Figure 1 also reflects, user-level data collected from these tasks are usually in a multi-field categorical form, for example, [Country=USA, Gender=Male, \cdots] which is normally transformed into binary features via one-hot encoding. When performing machine learning on solving such prediction tasks, an indispensable part of success is applying feature interactions across raw features, which has been emphasized in related literature [3, 6, 11, 36]. For instance, young male users may have a higher chance of clicking ads toward video games, which indicates that the cross feature regarding *age* and *gender* can be a significant predictor for estimating whether users will click on some certain types of ads. However, the binarized user data leads to a high-dimensional and sparse feature space (e.g., the well-known CTR prediction dataset Criteo¹ has a feature dimension of over one million with sparsity over 99% [30]) where handcrafting powerful cross features are inevitably time expensive, and results can rarely be generalized to unseen high-order feature interactions [7].

Some pioneering work was then proposed to overcome the limitation by leveraging feature interactions in an automation fashion. For example, Factorization Machines (FMs) [28] was proposed to model second-order cross features by parameterizing a cross feature's weight as the constituent features' inner product of the embedding vectors. Other recent developments [11, 13, 18, 36] have also augmented FMs with deep neural networks (DNNs) to model more expressive high-order feature interactions, which have gained promising results. However, we argue that two remaining challenges can potentially hinder these approaches. First, for models relying on DNNs to achieve the capability of modeling high-order cross features (e.g., DeepFM [11], NFM [13], etc.), an obvious drawback is that deep neural networks only learn feature interactions in an implicit way, which have been shown inefficient in learning multiplicative feature interactions [2, 30]. It also significantly reduces the model's interpretability. Second, for models able to explicitly model high-order feature interactions (e.g., xDeepFM [18], etc.), they are limited by assigning equal weights to factorized embeddings. In most real-business scenarios, it is undeniable that different predictors might have different predicting power, and not all the features contain useful signals for making the forecasting. Note that Attentional Factorization Machine (AFM) [12] produces an attention-based pooling layer to differentiate the cross feature importance such that the influence of less useful feature interactions can be compromised by assigning lower weights. Unfortunately, AFM only aims to model second-order feature interactions, which lacks the capability of capturing high-order feature information.

To overcome these challenges, we propose a novel architecture *Deep Cross Attentional Product* network (DCAP) that aims to explicitly learn high-order cross features with discriminated importance at a vector-wise level. Our approach is inspired by the *cross network* [33] architecture and utilizes the multi-head self-attention mechanism proposed in Transformer [32] which is now the *defacto* state-of-the-art model in broad NLP domains [4, 8]. To be specific, we first form the attentional feature embeddings in the first network layer using the self-attention mechanism. We then compute the

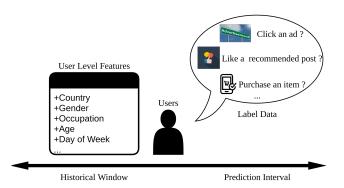


Figure 1: An visualization of illustrating user response prediction tasks.

inner product or outer product between attentional feature embeddings and original input embeddings as each layer's output. By increasing the network's depth, we further generate the cross attentional features across each layer using the last layer's outputs and the original inputs. As the depth of the network increases, the degree of feature interactions grows up correspondingly. Significantly, this attention mechanism greatly enhances our model's interpretability and transparency, allowing practitioners to perform a more in-depth analysis of user behaviors. Inspired by PNN [26], for each layer, we represent all cross features in product fashion. To summarize, we make the following contributions:

- We propose a novel Deep Cross Attentional Product Network (DCAP) that explicitly models high-order feature interactions with discriminated importance at a vector-wise level in an automation fashion. It can efficiently capture highly nonlinear feature interactions of bounded degree concerning the network's depth, requiring no handcrafted feature engineering or large-scale searching at a tremendous computational cost.
- We concatenate outputs from each layer such that the network outputs consist of all cross features with the degree of order ranging from 1 to the highest, which contains richer predictor information.
- We conduct comprehensive experiments on three real-world datasets, and the results have demonstrated that our proposed DCAP achieves superior performance compared with the state-of-the-art models.

The remainder of this paper is organized as follows. The related work is first reviewed in Section 2. We further provide some preliminary knowledge in Section 3, which is necessary for understanding our model architecture and the inspirations of our design. We then describe our proposed architecture DCAP in Section 3. We conduct experiments and report the results in Section 4. For simplicity, we take the click-through rate (CTR) estimation in online advertising as working examples to explore our model's potential effectiveness as it is one of the most critical applications of user response prediction. Finally, we conclude our work in Section 5.

2 RELATED WORK

The problem of user response prediction in machine learning is essentially a binary classification problem with prediction likelihood

¹http://labs.criteo.com/2014/02/kaggle-display-advertising-challenge-dataset/

as the training objective [23, 26]. It is widely acknowledged that modeling feature interactions are crucial for a good performance [6, 11, 18, 33].

2.1 Individual Models

Rendle *et al.* [28] proposed the pioneering work Factorization Machines (FMs) to capture at most second-order feature interactions. It has been demonstrated effective in both recommender systems [12, 25] and CTR prediction [11, 18]. With the success of FMs, different variants of FMs have been proposed. For example, Juan *et al.* [14] proposed Field-aware Factorization Machine (FFM) to model finegrained interactions across features within different fields. Cheng *et al.* [5] proposed GBFM and He *et al.* [12] introduced AFM to capture the importance of different second-order feature interactions. However, all these approaches focused on modeling low-order feature interactions, which are not sufficient for the model to capture high-order information. Blondel *et al.* [3] proposed a high-order FMs algorithm (HOFMs) to generalize the capability of FMs for modeling any cross feature at any degree of order. However, it is limited by the polynomial model complexity.

2.2 Ensemble Models

As deep neural networks (DNNs) demonstrated the capability to explore high-order hidden patterns, more ensemble models were then proposed to integrate with DNNs as encounter parts to boost the prediction performance. Cheng et al. [6] developed Wide & Deep for online recommendation at Google combining linear model and DNN, which achieves a remarkable performance in APP recommendation. He et al. [13] introduced NFM, Guo et al. [11] proposed DeepFM, and Zhang et al. [35] proposed FNFM to further combine FMs with DNNs to gain the capability of modeling implicit highorder feature interactions. Qu et al. [26] proposed Product Neural Network (PNN) that utilizes the inner product and outer product of features to model the inter-field feature interactions followed by DNNs. Wang et al. [33] devised a cross network (DCN) architecture that jointly learn both explicit and implicit high-order feature interactions in an automation fashion. It can learn feature interactions efficiently; however, all interactions come in a bit-wise fashion, bringing more implicitness. More importantly, it is inefficient in learning multiplicative feature interactions [2]. Lian et al. [18] combined compressed interaction network with a multi-layer perceptron (xDeepFM) to learn high-order feature interactions explicitly at a vector-wise level. Additionally, Cheng et al. [7] devised AFN with a logarithmic transformation layer that converts each feature's power in a feature combination into the coefficient to be learned. Despite the promising results, these methods can be hindered due to the low interpretability caused by DNNs' implicitness or assigning equal weights to all the feature interactions without discrimination.

Moreover, AutoFIS [20] can automatically identify important feature interactions for factorization models with computational cost just equivalent to training the target model to convergence. Song *et al.* [30] adopted the multi-head attention mechanism (MHA) [32] for modeling feature interactions due to its superiority over modeling attentional pairwise feature correlations. Feng *et al.* [9] also directly applied this self-attention mechanism in the Deep Session Interest Network (DSIN) for CTR prediction. However, these

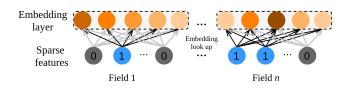


Figure 2: An example of the embedding layer with an embedding dimension of 5.

models' performance can be limited in modeling high-order complicated feature interactions by merely stacking the self-attention blocks. Thus, we propose a model that can fully utilize the MHA mechanism to model explicit high-order feature interactions with discriminated importance.

3 METHODOLOGIES

Inspired by the Deep&Cross Network [29], we design this Deep Cross Attentional Product Network (DCAP) with the following considerations: (1) high-order explicit feature interactions at vectorwise level; (2) differentiating the significance of feature interactions with any degree of order; (3) easy parallelization for computation efficiency. This section presents the architecture of our proposed model and analyzes both time and space complexity. We present a comprehensive description of how to learn a low-dimensional dense feature embedding that models high-order cross features with discriminated importance.

3.1 Sparse Input and Embedding Layer

In web-scale recommender systems or CTR prediction problems, the input features are usually sparse with tremendous dimension and present no clear spatial or temporal correlation. One common approach is transforming the multi-field categorical data into a high-dimensional sparse feature space via field-aware one-hot encoding. An embedding layer is developed upon the raw feature input to compress it to a low dimensional, dense real-value vector representation through a shared latent space. Expressly, we represent each categorical feature with a low-dimensional vector as following:

$$\mathbf{x}_i = \mathbf{V}_i \mathbf{e}_i \tag{1}$$

where V_i is an embedding matrix for field *i*, and e_i is a one-hot vector. If the field is multivalent, we take the field embedding as the sum of feature embedding. The output of the embedding layer is then a wide concatenated vector as:

$$\mathbf{x} = [\mathbf{x}_1, \cdots, \mathbf{x}_n] \tag{2}$$

where *n* denotes the number of feature fields, and $\mathbf{x}_i \in \mathcal{R}^d$ denotes the embedding of one field. The embedding layer is illustrated in Figure 2. We apply the input and embedding layer as the first stage of our model, which converts a set of sparse representations of input features into dense vectors as shown in the left panel of Figure 3.

3.2 Multi-head Self-attention

3.2.1 Motivations. One efficient way to enable feature interactions to contribute differently to the prediction is the attention mechanism. Attention-based models can be traced back to the Neural Machine Translation [1, 22], where Recurrent Neural Networks

(RNNs) applied the attention mechanism between encoder and decoder. However, the recurrent mechanism for temporal dependency modeling is not easy to parallelize. Transformer [32] extends the attention mechanism to a multi-head self-attention with scaled dot-production. It has been widely used in diverse NLP domains and many other tasks such as recommendation system [9], information retrieval [31], and computer vision [27]. The main idea of self-attention is a soft addressing process within pair-wise token representations in a sequence. Only a subset of tokens are worth more *attention* instead of assigning equal weights to all the features. For example, consider a sentiment classification problem with the texts "I feel so happy to buy a new laptop." The words besides "happy" are not indicative of the positive emotion exhibited in this sentence. Those interactions involving irrelevant features can be considered as noises that have less contribution to the prediction. In real-world applications, different predictor variables usually have different predictive power, and not all features contain a useful signal for estimating the target. The interactions with less useful features should be assigned a lower weight as they contribute less to the prediction. Here we extend this multi-head attention mechanism to model the dependencies across different feature fields due to its capability of differentiating the importance of feature interactions.

3.2.2 Formulations. Specifically, for a set of input feature embeddings $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n)^T \in \mathcal{R}^{n \times d}$ (with number of feature fields n and dimensionality d), where $\mathbf{x}_i \in \mathcal{R}^d$. The self-attention function firstly projects them into queries $\mathbf{Q} \in \mathcal{R}^{n \times d_q}$, keys $\mathbf{K} \in \mathcal{R}^{n \times d_k}$ and values $\mathbf{V} \in \mathcal{R}^{n \times d_v}$, with different, learned linear projections to d_q , d_k and d_v dimensions, respectively. Then a particular scaled dot-product attention was computed to obtain the weights on the values as:

Attention(Q, K, V) = softmax(
$$\frac{QK^T}{\sqrt{d_k}}$$
)V (3)

For any single head, this attention mechanism operates on the input embedding and computes a new embedding output $\mathbf{Z} = (\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n)^T$ of the same number of fields where $\mathbf{z}_i \in \mathcal{R}^{d_v}$.

Each output element, z_i , is computed as weighted sum of a linearly transformed input elements:

$$\mathbf{z}_{i} = \sum_{j=1}^{n} \alpha_{ij} \left(\mathbf{x}_{j} W^{V} \right) \tag{4}$$

Each weight coefficient, α_{ij} , is computed using a softmax function:

$$\alpha_{ij} = \frac{\exp e_{ij}}{\sum_{k=1}^{n} \exp e_{ik}}$$
(5)

And e_{ij} is computed by the attention function that essentially captures the correlations between queries and keys using this dotproduct so as to perform a soft-addressing process:

$$e_{ij} = \frac{\left(\mathbf{x}_i W^Q\right) \left(\mathbf{x}_j W^K\right)^T}{\sqrt{d_v}} \tag{6}$$

where $W^Q \in \mathcal{R}^{d \times d_q}, W^K \in \mathcal{R}^{d \times d_k}, W^V \in \mathcal{R}^{d \times d_v}$ are parameter matrices.

Multi-head attention allows the model to jointly attend to information from different representation subspaces of different fields.

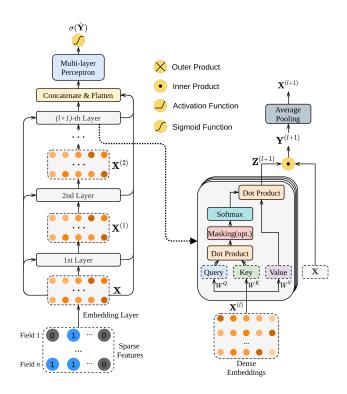


Figure 3: The overall architecture of our Deep Cross Attentional Network consists of multiple cross attentional product layers (left panel). Specifically, we illustrate the modeling process in detail for the (l+1)-th layer in the right panel.

The weighted sum of all concatenated heads is then computed as following:

$$MULTIHEAD(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = CONCAT(head_1, \cdots, head_h)W^O$$
(7)

in which, h is the number of total heads. Each head is defined as:

$$head_i = \text{ATTENTION}(\mathbf{X}W_i^Q, \mathbf{X}W_i^K, \mathbf{X}W_i^V)$$
(8)

where the projections are parameter matrices $W_i^Q \in \mathcal{R}^{d \times d_q}, W_i^K \in \mathcal{R}^{d \times d_k}, W_i^V \in \mathcal{R}^{d \times d_v}$ and $W^O \in \mathcal{R}^{hd_v \times d}$. In practice, we usually set $d_q = d_k = d_v$, and $d = hd_v$.

3.3 Deep Cross Attentional Product Network

In order to explore richer explicit high-order feature interactions by deepening the attention network, we design cross attention layers with each layer having the following functions:

$$\mathbf{Z}^{(l+1)} = \mathrm{MULTIHEAD}(\mathbf{Q}^{(l)}, \mathbf{K}^{(l)}, \mathbf{V}^{(l)})$$
(9)

$$\mathbf{P}^{(l+1)} = \mathbf{Z}^{(l+1)} \odot | \otimes \mathbf{X}$$
(10)

$$\mathbf{Y}^{(l+1)} = \sum_{d} \mathbf{P}^{(l+1)}$$
(11)

$$\mathbf{X}^{(l+1)} = \operatorname{AvgPooling}(\mathbf{P}^{(l+1)})$$
(12)

where $\mathbf{Q}^{(l)}$, $\mathbf{K}^{(l)}$ and $\mathbf{V}^{(l)}$ are projections of $\mathbf{X}^{(l)} \in \mathcal{R}^{n \times d}$ that are embedding vectors denoting the inputs for the (l + 1)-th layer. $\mathbf{P}^{(l+1)} \in \mathcal{R}^{n(n-1)/2 \times d}$ represents the production between the weighted

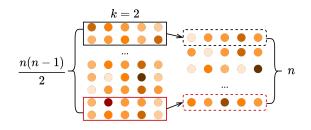


Figure 4: An example of the average pooling process with a kernel size as 2.

outputs of multi-head attention $\mathbb{Z}^{(l)} \in \mathbb{R}^{n \times d}$ and the original input embeddings $\mathbb{X} \in \mathbb{R}^{n \times d}$ using either inner product \odot or outer product \otimes . We further sum up the products along the embedding dimension to get $\mathbb{Y}^{(l+1)} \in \mathbb{R}^{n(n-1)/2}$ as the output (used for prediction) of the (l+1)-th layer. The average pooling layer then involves calculating the average for each patch of the feature interactions which means that each k (kernal size) cross features is down sampled to the average value of them. In output layer, we concatenate the flattened input embeddings with outputs from each layer followed by a fully connected neural network. A visualization of the general architecture is shown in Figure 3.

3.3.1 Product Operation. We adopt the product operation to model the feature interactions as inspired by [26]. For inner product, we define the operation of two vectors $\mathbf{a} \in \mathcal{R}^d$ and $\mathbf{b} \in \mathcal{R}^d$ as following:

$$\mathbf{a} \odot \mathbf{b} \triangleq (\mathbf{a}_1 \mathbf{b}_1, \cdots, \mathbf{a}_i \mathbf{b}_i, \cdots, \mathbf{a}_n \mathbf{b}_n)$$
(13)

For outer product, the definition of the operation is as following:

$$\mathbf{a} \otimes \mathbf{b} \triangleq \left(\sum_{j} \mathbf{a}_{1} \mathbf{b}_{j}, \cdots, \sum_{j} \mathbf{a}_{i} \mathbf{b}_{j}, \cdots, \sum_{j} \mathbf{a}_{n} \mathbf{b}_{j}\right)$$
(14)

Now, if we apply inner product operation between $Z^{(l+1)}$ and X, then

$$\mathbf{Z}^{(l+1)} \odot \mathbf{X} = \{\mathbf{z}_i^{(l+1)} \odot \mathbf{x}_j\}_{(i,j) \in \mathcal{R}_x}$$
(15)

where $\mathcal{R}_x = \{(i, j)\}_{1 \le i, j \le n, j > i}$. Note that we only use the cross relationships between features from different fields once such that we expand *n* features to n(n - 1)/2 interacted cross features for each layer.

3.3.2 Adaptive Average Pooling. In fact, the production result $\mathbf{P}^{(l+1)}$ of the (l + 1)-th layer has a total number of $O(n^2)$ cross feature representations, and it will go quadratically as we further pass it to the multi-head attention in the next layer since the dot-product attention computation takes another $O(n^2)$ complexity, n is the number of input features. Average pooling is a common technique used in computer vision, especially in the pooling layer of convolutional neural network [19]. In our model, we extend the average pooling approach to avoid the quadratic complexity caused by the depth of network without losing too much information. The general 1D average pooling process is as following:

$$Output = \frac{1}{k} \sum_{m=0}^{k-1} Input \left(\text{STRIDE} \times l + m, \cdots \right)$$
(16)

where $output \in \mathcal{R}^{L_{out} \times d}$ and $input \in \mathcal{R}^{L_{in} \times d}$. Adaptive average pooling strategy can adaptively tune the kernel size k and stride steps to our desired cross feature dimension since one can easily

conclude that $L_{out} = \left\lfloor \frac{L_{in} + 2 \times \text{PADDING} - \text{KERNEL SIZE}}{\text{STRIDE}} + 1 \right\rfloor$. Figure 4 is a visualization of the average pooling process.

By applying the average pooling method, we compress down the cross features space from $O(n^2)$ to O(n) while each new feature is a combination of a subset of original cross features.

3.4 Combination with Bit-wise Feature Interactions

Upon the deep feature interactions, we concatenate all the outputs from each layer with the flattened input embedding followed by a multi-layer perceptron (also known as a fully connected neural network with multiple layers) to further explore the implicit bitwise interactions. Suppose a cross attentional product network with *L* layers, the final output would be computed as:

$$\widehat{\mathbf{Y}} = \mathrm{MLP}(\mathrm{CONCAT}[\mathbf{X}_{\mathrm{flatten}}, \mathbf{Y}^{(1)}, \cdots, \mathbf{Y}^{(L)}])$$
(17)

where the architecture of MLP can be referred to Section $\ref{eq:matrix}$. At the end, the output of the hidden layers is transformed to the final user clicking probabilities through sigmoid function $\sigma(\widehat{\mathbf{Y}}) = 1/(1+e^{-\widehat{\mathbf{Y}}})$.

For binary classifications, the loss function is the log loss:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} \mathbf{y}_i \log \hat{\mathbf{y}}_i + (1 - \mathbf{y}_i) \log (1 - \hat{\mathbf{y}}_i)$$
(18)

where N is the total number of training instances. The optimization process is to minimize the following objective function:

$$\mathcal{J} = \mathcal{L} + \lambda_* \|\Theta\| \tag{19}$$

where λ_* denotes the regularization term and $\|\Theta\|$ denotes the set of parameters, including these in both cross feature modeling layers and DNN part.

3.5 Model Analysis

In this section, we analyze the proposed DCAP to study the model complexity and potential effectiveness.

3.5.1 Model Effectiveness. First of all, we argue that this deep and cross structure is able to model high-order explicit feature interactions as the degree of cross features grows with layer depth.

THEOREM 3.1. Consider an l-layer deep cross attentional product network with the (m + 1)-th layer defined as Section 3.3. Let the input embedding be X, the output of attentional feature embeddings in (m + 1)-th layer be $Z^{(m+1)}$. Then, the highest degree of cross features captured in terms of original input embedding X is l + 1.

PROOF. As an example, for the first layer, as Eq. 15 demonstrates,

$$\mathbf{z}_{i}^{(1)} \odot \mathbf{x}_{k} = \sum_{j=1}^{n} \alpha_{ij} \left(\mathbf{x}_{j}^{(l)} W^{V} \right) \odot \mathbf{x}_{k}$$
$$= \sum_{j=1}^{n} \alpha_{ij} \left(\mathbf{x}_{j}^{(l)} W^{V} \odot \mathbf{x}_{k} \right)$$
$$= \mathbf{p}^{T} \sum_{i=1}^{n} \alpha_{ij} \left(\mathbf{x}_{j}^{(l)} \odot \mathbf{x}_{k} \right)$$
(20)

which models cross features ranging from x_1x_k to x_nx_k and the degree of cross feature in terms of input embedding x is 2. Note that

the average pooling process does not change the order of feature interactions since it can be viewed as multiplying an additional averaging coefficient β_{ij} to the cross feature representations. Suppose this statement holds for l = m. For (m + 1)-th layer, we have:

$$\mathbf{z}_{i}^{(m+1)} \odot \mathbf{x}_{k} = \mathbf{p}^{T} \sum_{j=1}^{n} \alpha_{ij} \left(\mathbf{x}_{j}^{(m)} \odot \mathbf{x}_{k} \right)$$
(21)

where $\mathbf{x}_{j}^{(m)} \odot \mathbf{x}_{k}$ can model cross features with degree of order (m + 1) since $\mathbf{x}_{j}^{(m)}$ represents the *m*-th order cross features. By induction hypothesis, one can easily prove that the highest degree in terms of original input embedding X for an *l*-layer cross attentional product network is l + 1.

3.5.2 Model Complexity. For computing multi-head attention, the total number of Mult-Adds is $O(2n^2d + 4nd^2)$. For production of features, it takes another $O(nd(n-1)/2) = O(n^2d)$ Mult-Adds. So the total computation complexity for each layer in terms of Mult-Adds is $O(3n^2d + 4nd^2)$. A cross attentional product network with l layers would take a total number of Mult-Adds as $O(3n^2dl + 4nd^2l)$. For the feed-forward network, the input size is d' = nd + ln(n-1)/2 such that the total number of Mult-Adds is $O(d'h_1 + h_1h_2 + h_2)$, where h_1, h_2 are the hidden size of each hidden layer, respectively. For space complexity, the attention part takes $O(4d^2)$ including three projection matrices and one final weighted sum layer. For DNN part, it contains weight matrices that are $O(d'h_1 + h_1h_2 + h_2)$. For a model with l layers, the total space complexity would be $O(4d^2l + d'h_1 + h_1h_2 + h_2)$, which is quadratic with respect to both number of fields n and embedding dimension d.

4 EXPERIMENTS

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

- (**RQ1**) How does our proposed Deep Cross Attentional Product Network (DCAP) perform in high-order feature interactions learning compared to the other state-of-the-art methods?
- (**RQ2**) How do the different hyper-parameter settings affect model performance?
- (**RQ3**) Is the multi-head attention mechanism vital for our model, and how can it essentially be useful in prediction?

4.1 Experimental Setup

4.1.1 Datasets. As the click-through rate (CTR) prediction in online advertising is one of the most critical user response domains, we take it as our experimental examples to explore our model's potential effectiveness. We evaluate the proposed model on the following three public datasets: 1) **Criteo**² Display Ads dataset is a famous benchmark for ads click-through rate prediction. It has 45 million users' clicking records on displayed ads. It contains 13 numerical feature fields and 26 categorical feature fields where each category has a high cardinality. 2) **Avazu**³ This dataset contains users' mobile behaviors, including whether a user clicks a displayed mobile ad or not. It has 23 feature fields spanning from user/device

Table 1: Statistics of evaluation datasets.

Dataset	#Instances	#Fields	#Feature Dimension
Criteo	45,840,617	39	1,086,810
Avazu	14,426,917	23	767,250
MovieLens-1M	1,000,209	5	10,072

features to ad attributes. 3) **MovieLens-1M**⁴ This dataset contains users' ratings on movies. We treat samples with a rating of less than three as negative samples for binarization because a low score indicates that the user does not like the movie and will not click it for watching. We treat samples with a rating greater than three as positive samples. The statistics of the three datasets are summarized in Table 1.

4.1.2 Data Preparation. First, we set a frequency threshold for removing all infrequent features and treat them as a single feature "<unknown>", where threshold is set to {10, 4} for Criteo, Avazu, respectively. Second, for numerical features with a high variance that might influence the learning process, we normalize them by transforming a value z to $\log^2 z$ if z > 2, which is proposed by the winner of Criteo Competition⁵. Third, we randomly select 80% of all samples for training and consecutively split the rest into validation and test sets as 1:1.

4.2 Evaluation Metrics

We apply two famous metrics to evaluate the performance of our proposed approach and all other competitive models: **AUC** Area under ROC curve and **Logloss** (cross entropy). Note that **a slight increase in AUC or decrease in Logloss at 0.001-level** is known to be a significant improvement for the tasks such as CTR prediction [6, 11, 33].

4.3 Baselines

We compare DCAP with existing methods of three classes (also see Table 2): (a) The first-order approaches that model a linear combination of raw features; (b) FM-based methods that capture second-order feature interactions; (c) Advanced approaches that model high-order cross features.

- LR logistic regression models the linear relationship between features and targets.
- FM [28] factorization machine applies matrix factorization techniques to capture the second order feature interactions for prediction.
- **AFM** [12] attentional factorization machine assigns attention weights to feature interactions based on FM.
- **HOFM** [3] high-order factorization machine can model highorder feature interactions.
- NFM [13] neural factorization machine sums up pairwise Hadamard product of features followed by a fully connected neural network.
- PNN [26] product neural network models high-order feature interactions by computing pairwise inner (iPNN) or outer products (oPNN) of input features followed by a fully connected neural network.

²http://labs.criteo.com/2014/02/kaggle-display-advertising-challenge-dataset/ ³https://www.kaggle.com/c/avazu-ctr-prediction/data

⁴https://grouplens.org/datasets/movielens/

⁵https://www.csie.ntu.edu.tw/ r01922136/kaggle-2014-criteo.pdf

- Wide & Deep [6] wide & deep model integrates LR and DNN. Also, we omit the hand-crafted cross features for a fair comparison.
- **DCN** [33] deep & cross network models the explicit feature, by computing the outer product between original input embedding and corresponding output across layers.
- **DeepFM** [11] deep factorization machine is an ensemble between DNN and FM.
- xDeepFM [18] xDeepFM models both explicit and implicit high-order feature interactions by computing outer products of feature vectors at different orders. It also combines with DNN.
- AutoInt [30] auto feature interaction model applies multihead self-attention mechanism on modeling cross feature interactions.
- **AFN** [7] adaptive factorization network learns arbitraryorder cross features adaptively by building a logarithmic transformation layer that converts each feature's power in a feature combination into the coefficient to be learned.

4.4 Training Settings

We implement our approaches using PyTorch⁶. We apply Adam [16] with a learning rate of 0.001 and a weight decay of $1e^{-6}$ to prevent overfitting, and a mini-batch size of 4096 across all tasks. For the fair competition, we set the default architecture of the dense neural network with two hidden layers and 100 neurons per layer for all models that involve DNN. To avoid overfitting, we perform the early-stopping strategy based on AUC on the validation set. A dropout method is also applied across all models with a rate of 0.5 for the MovieLens-1M dataset and 0.2 for the other two datasets to prevent overfitting. The dimension of feature embeddings is set to 16 for all the models across all tasks consistently. More specifically, the number of layers in DCN set to 2. The maximum order in HOFM is set to 3. The attention embedding size of model AFM and AutoInt is 64. Additionally, the number of heads in AutoInt is set to 4. The default number of logarithmic neurons in AFN is set to 1500, 1200, 800 for Criteo, Avazu, and Movielens datasets. For our model DCAP, we set the maximum depth of network to 2 as a bounded order of feature interactions.

All the hyper-parameters are tuned on the validation set. We run the experiments for each empirical result as 20 independent trials on four NVIDIA Tesla P100 GPUs in parallel and report the average value with the standard deviation.

4.5 Comparative Performance (RQ1)

First, we want to know how our proposed model performs compared with various models involving first-order, second-order, and high-order feature interactions. Note that FM only models secondorder cross features explicitly while DNNs model high-order feature interactions implicitly. Models integrated with DNNs such as NFM, PNN, DeepFM, Wide&Deep, etc., are also capable of modeling high-order interactions. Additionally, models like DCN, xDeepFM, AutoInt, and AFN are all ensemble models that combine the effectiveness of jointly explicit and implicit learning, which means they can model high-order feature interactions both explicitly and implicitly. Our proposed DCAP is also such an ensemble model that cross attentional product explicitly models weighted high-order feature interactions followed by an implicit feed-forward neural network.

The results shown in Table 2 demonstrate that our method outperforms the other models across different datasets consistently. For the Criteo dataset, DCAP outperforms the second-best model oPNN by a significant 0.001 increase in AUC and 0.001 decreases in Logloss. It also has an 0.001 prediction performance increase in both AUC and Logloss on the Avazu dataset. For MovieLens-1M, the gap between DCAP and the second-best model is further widened. Compared to FM based models, DCAP has an almost 0.006 higher AUC score than DeepFM, which is generally considered a significant user response prediction benchmark. It also has a 0.0034 higher AUC score than xDeepFM, which also models high-order feature interactions explicitly yet without differentiating each cross feature's importance. Another interesting finding is that there is no theoretic guarantee of the superiority of high-order ensemble methods over the simple individual models since AFN performs worse than FM and AFM on both Avazu and MovieLens-1M datasets. Compared to the Criteo dataset, these two datasets contain fewer training instances and feature fields requiring minimal high-order feature interactions to make the predictions. As the AFN model learns arbitrary-order cross features, it can be hindered by overfitting the data due to the over-complicated feature interaction space. In this case, a simple model such as FM might achieve relatively better performance. Note that for the Avazu dataset, models such as DeepFM, xDeepFM, AutoInt also have worse performance than individual models like FM. These models share a typical architecture that combines the explicit cross features and implicit embeddings in an add fashion. However, for PNN and DCN, they stack explicit feature interaction outputs and input embeddings together, followed by the DNNs. As such, this concatenated way of combining outputs may have a superiority than sum up.

Generally, models integrated with DNNs have a better performance over individuals, which further illustrates the effectiveness of implicit feature interactions. Among all the integrated models, the results of PNN and DCN are more impressive due to the consistent outstanding prediction performance on three datasets. We analyze the potential effectiveness of these models and compare them with our method in the following.

4.5.1 Comparison Between DCAP and DCN. Both DCAP and DCN adopt the cross network architecture to explicitly model high-order cross features. However, the differences are: (a) DCN can learn feature interactions very efficiently (with much less computation), but the output of each cross network layer is limited to a scalar multiple of original input embedding [18], which may lead to a suboptimal model performance; (b) DCAP has a higher requirement for computational resources while can learn more elaborate cross feature interactions with discriminated attention or weights; (c) all interactions in DCN come at a bit-wise fashion, which brings more implicitness; however, DCAP models high-feature interactions at a vector-wise level which can be followed by an implicit bit-wise interaction layer. Though they are both jointly explicit and implicit learners, DCAP owns better interpretability.

⁶https://pytorch.org/

Table 2: The overall performance of all models on Criteo, Avazu, and MovieLens-1M datasets. Best performance in boldface. We also mark the second-best model with an underline. We further analyze these results in Section 4.5.

Model Class Model	M - J - 1	Criteo		Avazu		MovieLens-1M	
	Model	AUC	Logloss	AUC	Logloss	AUC	Logloss
First-Order	LR	0.7943+/-0.0000	0.4560+/-0.0000	0.7608+/-0.0000	0.3916+/-0.0000	0.7918+/-0.0000	0.5406+/-0.0000
Second-Order FM AFM	FM	0.8040+/-0.0001	0.4478+/-0.0001	0.7816+/-0.0004	0.3826+/-0.0005	0.8004+/-0.0005	0.5391+/-0.0017
	AFM	0.8073+/-0.0001	0.4443 + -0.0001	0.7756+/-0.0008	0.3859+/-0.0017	0.7983+/-0.0018	0.5368+/-0.0021
High-Order DCN DeepFM	HOFM	0.8059+/-0.0001	0.4459+/-0.0001	0.7824+/-0.0002	0.3820+/-0.0006	0.7953+/-0.0008	0.5381+/-0.0011
	NFM	0.8061+/-0.0006	0.4455+/-0.0006	0.7809+/-0.0010	0.3892+/-0.0016	0.7896+/-0.0175	0.5512+/-0.0149
	iPNN	0.8118+/-0.0001	0.4400+/-0.0001	0.7847+/-0.0007	0.3772+/-0.0007	0.8039+/-0.0009	0.5292+/-0.0012
	oPNN	0.8132+/-0.0001	0.4387+/-0.0001	0.7851+/-0.0003	0.3766+/-0.0001	0.8046+/-0.0008	0.5289+/-0.0019
	Wide&Deep	0.8107+/-0.0001	0.4411+/-0.0001	0.7732+/-0.0006	0.3855+/-0.0003	0.7999+/-0.0012	0.5374+/-0.0009
	DCN	0.8112+/-0.0001	0.4408+/-0.0001	0.7833+/-0.0004	0.3784+/-0.0002	0.8051+/-0.0010	0.5271+/-0.0015
	DeepFM	0.8086+/-0.0001	0.4432+/-0.0000	0.7757+/-0.0005	0.3853+/-0.0005	0.8008+/-0.0011	0.5366+/-0.0017
	xDeepFM	0.8108+/-0.0002	0.4411+/-0.0002	0.7767+/-0.0007	0.3845+/-0.0004	0.8045+/-0.0009	0.5297+/-0.0013
	AutoInt	0.8108+/-0.0001	0.4411+/-0.0001	0.7760+/-0.0003	0.3846+/-0.0004	0.8047+/-0.0006	0.5289+/-0.0007
	AFN	0.8122+/-0.0000	0.4397+/-0.0001	0.7743+/-0.0012	0.3876+/-0.0010	0.7947+/-0.0021	0.5424+/-0.0025
	DCAP(ours)	0.8142+/-0.0001	0.4376+/-0.0001	0.7861+/-0.0003	0.3754+/-0.0002	0.8066+/-0.0012	0.5260+/-0.0013

Table 3: AUC scores of model with different number of layers and heads on MovieLens-1M dataset.

#Heads #Layers	1	2	4	8	16
1	0.8066	0.8073	0.8065	0.8072	0.8068
2	0.8074	0.8074	0.8073	0.8075	0.8061
3	0.8079	0.8068	0.8053	0.8072	0.8066
4	0.8070	0.8078	0.8073	0.8055	0.8081
5	0.8075	0.8071	0.8076	0.8078	0.8068

4.5.2 Comparison Between DCAP and PNN. The backbone of modeling cross features is inspired by PNN, which first applies inner product or outer product operations on modeling feature interactions. The downsides of Product Neural Network are: (a) PNN only models second-order explicit cross features while DNN implicitly captures the high-order cross features; DCAP can model high-order feature interactions both explicitly and implicitly; (b) similar to many other models, PNN is not capable of differentiating the importance of feature interactions. By deepening the cross product network combining with attention mechanism, we generalize the product network's capability to model high-order feature interactions explicitly.

4.6 Hyper-parameter Investigation (RQ2)

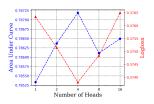
One intuitive question is that how the multi-head attention and depth of the network affect the model performance. In this section, we study hyper-parameters' impact, including (a) the number of heads in multi-head attention; and (b) the number of cross layers in the network. We conduct experiments via holding the identical settings for the DNN part while varying the cross attentional product network part settings. We show how the validation ACU and Logloss change by adding more layers and heads when setting the embedding size to 64. Table 3 and Table 4 show the changes in model performance in terms of AUC and Logloss on the MovieLens dataset, respectively.

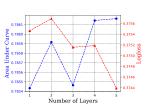
 Table 4: Logloss of model with different number of layers and heads on MovieLens-1M dataset.

#Heads #Layers	1	2	4	8	16
1	0.5297	0.5286	0.5293	0.5296	0.5274
2	0.5282	0.5278	0.5282	0.5300	0.5303
3	0.5265	0.5296	0.5290	0.5270	0.5302
4	0.5278	0.5294	0.5276	0.5294	0.5266
5	0.5295	0.5278	0.5275	0.5266	0.5265

4.6.1 Number of Heads. The number of heads is a critical hyperparameter in multi-head attention. With each attention head potentially focusing on different parts of the input and revealing complementary information, this attention mechanism can express sophisticated functions beyond the simple weighted average. However, recent research [24] has discovered that not all heads are practically necessary since only a small percentage of heads will significantly impact the performance. Besides, many heads will have a higher chance of modeling a more sophisticated combination of input embedding and overfitting the data. We then hypothesize that the number of heads is not the more, the better. As such, it is worth investigating how the number of heads affects model performance in practice. Besides the experiment results listed in Table 3 and Table 4, we further visualize the model performance against the number of heads in Figure 5a and Figure 6a fixing other chosen randomly hyper-parameters. The performance generally ascends first and then descends as the number of heads increases. Specifically, when we have sixteen heads for the model, the performance drops down heavily, further demonstrating our conjecture. The model with eight heads has an astonishing AUC score, while the Logloss is not good on MovieLens dataset. For Avazu dataset, the model can achieve the best performance with a total of four heads. In all, we conclude that the model can achieve relatively good performance with at most four heads on both Avazu and MovieLens dataset.

4.6.2 Depth of Network. The depth of network controls the upper bound degree of cross features. The deeper the network is, the higher order of feature interactions are captured by the model.





against the depth of network.

performance

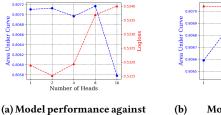
(a) Model performance against the number of heads.

the number of heads.

Figure 5: Results on the Avazu dataset.

Model

(b)



(b) Model performance against the depth of network.

Number of Laver

Figure 6: Results on the MovieLens dataset.

Meanwhile, a deeper network involves more computational complexity which may cause overfitting. To investigate the impact of number of layers, we increase the depth from 1 to 5 and report the AUC and Logloss results on Avazu and MovieLens datasets. We also visualize the model performance against number of layers in Figure 5b and Figure 6b. We observe that as the depth of network increases, the performance generally increases since the model captures more sophisticated high-order feature interactions. However, model complexity is linearly related to the number of layers and it could be a trade-off between prediction precision and computational requirements.

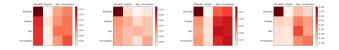


Figure 7: The attention pattern of each head on MovieLens dataset.

4.7 Is Multi-head Attention Really Necessary? (RQ3)

Another straightforward question is that is multi-head attention crucial for our model? Unlike the multi-head attention mechanism, Attentional Factorization Machine (AFM) generally applies a DNN on the inner product of feature embeddings followed by a softmax function to compute the attention scores, allowing different cross features contribute differently to the prediction.

Since multi-head attention is more computationally expensive, we further investigate whether this multi-head attention mechanism is necessary for user response prediction tasks. We conduct experiments on replacing the multi-head attention with this less complicated attention mechanism proposed by AFM and compare it with our model. Table 5 shows the results of comparison on two datasets, Avazu and MovieLens-1M. We let *w*/ represent the original multi-head attention mechanism used in our model, while *w*/o

 Table 5: The performance comparison between multi-head

 attention and AFM attention mechanism.

Datasets	Models	AUC	Losloss
	$DCAP_{w/}$	0.7861+/-0.0003	0.3754+/-0.0002
Avazu	$DCAP_{w/o}$	0.7830 + / - 0.0004	0.3818+/-0.0005
	$DCAP_{w/}$	0.8066+/-0.0012	0.5260+/-0.0013
MovieLens-1M	$DCAP_{w/o}$	0.8043+/-0.0015	0.5290+/-0.0015

stands for the attention mechanism used in AFM. As we can see, the original DCAP model consistently outperforms the model with the attention mechanism used in AFM, which further demonstrates the superiority of multi-head attention mechanism in modeling pairwise feature relevance.

To better understand the importance of multi-head attention, we also investigate the patterns learned by each attention *head*. We select out four feature fields *MovieID*, *Gender*, *Age* and *Occupation* from the MovieLens dataset and visualize the attention pattern of each attention *head* (four in all). As Figure 7 shows, the color darkness represents the degree of correlation or attention weights between pairwise feature fields. We observe that different heads attend to different parts of the feature fields as the first *head* concentrates more on the cross correlation between *MovieID* and other fields (left half). In contrast, the third *head* focuses on the cross dependencies between *Age*, *Occupation* and other features (right half). It is also obvious that *Gender* is the least attentional feature in predicting whether a user will watch a certain movie, which is reasonable to some extent.

The great interpretability of our model gives much more room for data engineers to analyze user behaviors further. More targeting strategies could be made based on these results to improve the user experience or expand the market.

5 CONCLUSION

This paper proposed a novel model Deep Cross Attentional Product Network (DCAP), which can learn high-order feature interactions explicitly at the vector-wise level. It also had the capability of differentiating the importance of all cross features of any degree of order. The core of our models consisted of three parts: (a) multihead attention for capturing pairwise feature relevance Product of features across layers for modeling high-order cross features; (b) concatenation of output from each layer for retaining information of all feature interactions at any order; (c) experimental results on three real-world datasets demonstrate the effectiveness and superiority of our model in predicting user response. Additionally, we demonstrated our model interpretability via visualizing the feature correlations of different attention heads. After incorporating DNNs for modeling implicit feature interactions, our model consistently achieved better AUC and Logloss scores than state-of-the-art models.

For future work, we will further explore the other attention variants with less computation complexity. Recent work such as Reformer [17] or Linformer [34] have provided a faster version of multi-head attention with only $O(n\sqrt{n})$ and O(n) complexity, respectively. Besides the attention blocks, we are also interested in redesigning computing feature production in a more efficient way such that the model can be trained on GPU clusters distributively.

REFERENCES

- Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings (2015), 1-15. arXiv:1409.0473
- [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. WSDM 2018 - Proceedings of the 11th ACM International Conference on Web Search and Data Mining 2018-Febuary (2018), 46–54. https://doi.org/10.1145/3159652. 3159727
- [3] Mathieu Blondel, Akinori Fujino, Naonori Ueda, and Masakazu Ishihata. 2016. Higher-order factorization machines. Advances in Neural Information Processing Systems Nips (2016), 3359–3367. arXiv:1607.07195
- [4] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. (2020). arXiv:2005.14165 http://arxiv.org/abs/2005.14165
- [5] Chen Cheng, Fen Xia, Tong Zhang, Irwin King, and Michael R. Lyu. 2014. Gradient boosting factorization machines. RecSys 2014 - Proceedings of the 8th ACM Conference on Recommender Systems (2014), 265–272. https://doi.org/10.1145/ 2645710.2645730
- [6] 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. ACM International Conference Proceeding Series 15-Septemb (2016), 7–10. https://doi.org/10.1145/ 2988450.2988454
- [7] Weiyu Cheng, Yanyan Shen, and Linpeng Huang. 2019. Adaptive Factorization Network: Learning Adaptive-Order Feature Interactions. (2019). https://doi.org/ 10.1609/aaai.v34i04.5768 arXiv:1909.03276
- [8] Jacob Devlin, Ming Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference 1, Mlm (2019), 4171-4186. arXiv:1810.04805
- [9] 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. *IJCAI International Joint Conference on Artificial Intelligence* 2019-Augus (2019), 2301–2307. https://doi.org/10.24963/ijcai.2019/319 arXiv:arXiv:1905.06482v1
- [10] Kun Gai, Xiaoqiang Zhu, Han Li, Kai Liu, and Zhe Wang. 2017. Learning Piecewise Linear Models from Large Scale Data for Ad Click Prediction. (2017), 1–12. arXiv:1704.05194 http://arxiv.org/abs/1704.05194
- [11] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A factorization-machine based neural network for CTR prediction. *IJCAI International Joint Conference on Artificial Intelligence* 0 (2017), 1725–1731. https://doi.org/10.24963/ijcai.2017/239 arXiv:1703.04247
- [12] Xiangnan He. 2017. Attentional Factorization Machines. *Ijcai* (2017), 3119–3125. https://www.comp.nus.edu.sg/{-}xiangnan/papers/ijcai17-afm.pdf
- [13] Xiangnan He and Tat Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. SIGIR 2017 - Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (2017), 355–364. https://doi.org/10.1145/3077136.3080777 arXiv:1708.05027
- [14] Yuchin Juan, Yong Zhuang, Wei Sheng Chin, and Chih Jen Lin. 2016. Field-aware factorization machines for CTR prediction. *RecSys 2016 Proceedings of the 10th ACM Conference on Recommeder Systems* (2016), 43–50. https://doi.org/10.1145/2959100.2959134
- [15] Alexandros Karatzoglou and Balázs Hidasi. 2017. Deep learning for recommender systems. RecSys 2017 - Proceedings of the 11th ACM Conference on Recommender Systems 15-Septemb (2017), 396–397. https://doi.org/10.1145/3109859.3109933 arXiv:arXiv:1606.07792v1
- [16] Diederik P Kingma and Jimmy Lei Ba. 2015. 15iclr-ADAM. Iclr (2015), 1-15. arXiv:1412.6980v9 https://arxiv.org/pdf/1412.6980.pdf[%]22entiredocument
- [17] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The Efficient Transformer. (2020), 1–12. arXiv:2001.04451 http://arxiv.org/abs/2001.04451
- [18] Jianxun Lian, Zhongxia Chen, Xiaohuan Zhou, Xing Xie, Fuzheng Zhang, and Guangzhong Sun. 2018. xDeepFM: Combining explicit and implicit feature interactions for recommender systems. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2018), 1754–1763. https://doi.org/10.1145/3219819.3220023 arXiv:1803.05170
- [19] Min Lin, Qiang Chen, and Shuicheng Yan. 2014. Network in network. 2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings (2014), 1–10. arXiv:arXiv:1312.4400v3

- [20] 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 KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020, Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash (Eds.). ACM, 2636–2645. https: //doi.org/10.1145/3394486.3403314
- [21] Qiang Liu, Feng Yu, Shu Wu, and Liang Wang. 2015. A convolutional click prediction model. International Conference on Information and Knowledge Management, Proceedings 19-23-Oct- (2015), 1743–1746. https://doi.org/10.1145/ 2806416.2806603
- [22] Minh Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. Conference Proceedings - EMNLP 2015: Conference on Empirical Methods in Natural Language Processing (2015), 1412–1421. https://doi.org/10.18653/v1/d15-1166 arXiv:1508.04025
- [23] Aditya Krishna Menon, Krishna Prasad Chitrapura, Sachin Garg, Deepak Agarwal, and Nagaraj Kota. 2011. Response prediction using collaborative filtering with hierarchies and side-information. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2011), 141–149. https: //doi.org/10.1145/2020408.2020436
- [24] Paul Michel, Omer Levy, and Graham Neubig. 2019. Are Sixteen Heads Really Better than One? NeurIPS (2019), 1–13. arXiv:1905.10650 http://arxiv.org/abs/ 1905.10650
- [25] Junwei Pan, Jian Xu, Alfonso Lobos Ruiz, Wenliang Zhao, Shengjun Pan, Yu Sun, and Quan Lu. 2018. Field-weighted factorization machines for click-through rate prediction in display advertising. *The Web Conference 2018 - Proceedings of the World Wide Web Conference*, WWW 2018 (2018), 1349–1357. https://doi.org/10. 1145/3178876.3186040 arXiv:1806.03514
- [26] Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. 2017. Product-based neural networks for user response prediction. *Proceedings - IEEE International Conference on Data Mining, ICDM* (2017), 1149–1154. https://doi.org/10.1109/ICDM.2016.57 arXiv:1611.00144
- [27] Prajit Ramachandran, Irwan Bello, Niki Parmar, Anselm Levskaya, Ashish Vaswani, and Jonathon Shlens. 2019. Stand-alone self-attention in vision models. Advances in Neural Information Processing Systems 32, NeurIPS (2019), 1–13.
- [28] Steffen Rendle. 2010. Factorization machines. Proceedings IEEE International Conference on Data Mining, ICDM (2010), 995–1000. https://doi.org/10.1109/ ICDM.2010.127
- [29] Ying Shan, T. Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and J. C. Mao. 2016. Deep crossing: Web-scale modeling without manually crafted combinatorial features. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining 13-17-August-2016 (2016), 255–262. https://doi.org/ 10.1145/2939672.2939704
- [30] Weiping Song, Zhijian Duan, Yewen Xu, Chence Shi, Ming Zhang, Zhiping Xiao, and Jian Tang. 2019. Autoint: Automatic feature interaction learning via selfattentive neural networks. *International Conference on Information and Knowledge Management, Proceedings* (2019), 1161–1170. https://doi.org/10.1145/3357384. 3357925 arXiv:1810.11921
- [31] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Multi-Cast Attention Networks for Retrieval-based Question Answering and Response Prediction. (2018). arXiv:1806.00778 http://arxiv.org/abs/1806.00778
- [32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. Advances in Neural Information Processing Systems 2017-Decem, Nips (2017), 5999-6009. arXiv:arXiv:1706.03762v5
- [33] Ruoxi Wang, Gang Fu, Bin Fu, and Mingliang Wang. 2017. Deep & cross network for ad click predictions. 2017 AdKDD and TargetAd - In conjunction with the 23rd ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2017 (2017). https://doi.org/10.1145/3124749.3124754 arXiv:arXiv:1708.05123v1
- [34] Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-Attention with Linear Complexity. 2048, 2019 (2020). arXiv:2006.04768 http://arxiv.org/abs/2006.04768
- [35] Li Zhang, Weichen Shen, Jianhang Huang, Shijian Li, and Gang Pan. 2019. Field-Aware Neural Factorization Machine for Click-Through Rate Prediction. *IEEE Access* 7 (2019), 75032–75040. https://doi.org/10.1109/ACCESS.2019.2921026 arXiv:1902.09096
- [36] Weinan Zhang, Tianming Du, and Jun Wang. 2016. Deep learning over Multi-Field categorical Data - A case study on user response prediction. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 9626 (2016), 45–57. https://doi.org/10.1007/978-3-319-30671-1_4 arXiv:arXiv:1601.02376v1