CAN: Feature Co-Action for Click-Through Rate Prediction

Weijie Bian†, Kailun Wu†, Lejian Ren†, Qi Pi†, Yujing Zhang†, Can Xiao†
Xiang-Rong Sheng†, Yong-Nan Zhu†, Zhangming Chan†, Na Mou†, Xinchen Luo†
Shiming Xiang‡, Guorui Zhou†, Xiaoqiang Zhu†, Hongbo Deng†*
†Alibaba Group ‡Institute of Automation, Chinese Academy of Sciences
China
{weijie.bwj,kailun.wukailun,lejian.rlj,dhb167148}@alibaba-inc.com

ABSTRACT

Feature interaction has been recognized as an important problem in machine learning, which is also very essential for click-through rate (CTR) prediction tasks. In recent years, Deep Neural Networks (DNNs) can automatically learn implicit nonlinear interactions from original sparse features, and therefore have been widely used in industrial CTR prediction tasks. However, the implicit feature interactions learned in DNNs cannot fully retain the complete representation capacity of the original and empirical feature interactions (e.g., cartesian product) without loss. For example, a simple attempt to learn the combination of feature A and feature B <A, B> as the explicit cartesian product representation of new features can outperform previous implicit feature interaction models including factorization machine (FM)-based models and their variations. This indicates there is still a big gap between explicit and implicit feature interaction models. However, to learn all the explicit feature interaction (cartesian product) representations requires a very large sample size along with N times of original parameter space (where N is quite large in most industrial applications). In this paper, we propose a Co-Action Network (CAN) to approximate the explicit pairwise feature interactions without introducing too many additional parameters. More specifically, giving feature A and its associated feature B, their feature interaction is modeled by learning two sets of parameters: 1) the embedding of feature A, and 2) a Multi-Layer Perceptron (MLP) to represent feature B. The approximated feature interaction can be obtained by passing the embedding of feature A through the MLP network of feature B. We refer to such pairwise feature interaction as feature co-action, and such a Co-Action Network unit can provide a very powerful capacity to fitting complex feature interactions. In addition, FM can be viewed as a special case of the CAN unit when the MLP is a single layer with only one output. Experimental results on public and industrial datasets show that CAN outperforms state-of-the-art CTR models and the cartesian product method. Moreover, CAN has been deployed in the display advertisement system in Alibaba, obtaining 12% improvement on CTR and 8% on Revenue Per Mille (RPM), which is a great improvement to the business. The code for experiments in this paper is open-sourced[1].

[*]Hongbo Deng is the corresponding author (dhb167148@alibaba-inc.com).

CCS CONCEPTS

• **Information systems → Computational advertising**.

KEYWORDS

CTR Prediction; Neural Networks; Feature Interaction

ACM Reference Format:
Weijie Bian†, Kailun Wu†, Lejian Ren†, Qi Pi†, Yujing Zhang†, Can Xiao†, Xiang-Rong Sheng†, Yong-Nan Zhu†, Zhangming Chan†, Na Mou†, Xinchen Luo†, and Shiming Xiang‡, Guorui Zhou†, Xiaoqiang Zhu†, Hongbo Deng†. 2022. CAN: Feature Co-Action for Click-Through Rate Prediction. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining (WSDM '22), February 21–25, 2022, Tempe, AZ, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3488560.3498435

1 INTRODUCTION

With the growing complexity of machine learning models, especially those in recommendation systems, how to deal with abundant input features effectively and efficiently becomes a crucial problem. For online recommendation system in an industrial setting, models are often trained on billion-scale sparse features with one-hot encoding [3, 17, 28]. Each feature can also be seen as a unique ID, which is often mapped to a low-dimensional embedding before being fed into the model. A simple way to deal with the large-scale input is to consider each feature independently. Under this strategy, a DNN can be directly trained to estimate CTR based on the combination (e.g., concatenation) of features in which the feature interaction relies on the implicit modeling of fully connected layers.

However, features like "candidate item" and "user click history" in the recommendation system are highly relevant [27, 28], i.e., there exist co-occurrence information. Another typical example is the story of "Beer and Nappies". This kind of feature interaction is beneficial to better estimate CTR. As illustrated in Fig.1, if no feature interaction is considered, the effects of features A and B toward the final label (click or not in CTR prediction) are modeled independently, as shown by the blue lines. Feature interaction shown in the grey line explicitly bridges the correlations of feature pair (A, B) to the target label and brings more useful information for the learning objective. We will discuss it more deeply in Sec. 3.

To model feature interaction, the most simple way is to use cartesian product. Given two features A and B, once features A and B are selected, the co-occurrence <A, B> is treated as a new feature and fed into the model. As the co-occurrence information is provided

[1]https://github.com/CAN-Paper/Co-Action-Network

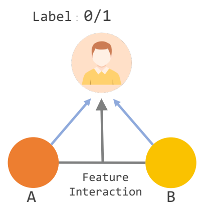

Figure 1: Illustration of feature interaction. Blue lines indicate the effects of two features A and B for the final label are modeled separately while feature interaction in the grey line bridges them together.

directly in the form of additional input, the training process gets easier. Although the cartesian product is simple and effective, it has some serious defects, such as huge parameter volume and feature space, poor generalization ability.

Different from cartesian product that introduces additional input information, several research efforts have been devoted to model feature interaction through careful combinations of input features. The typical examples are factorization based methods [12, 14, 16, 19, 23, 24] which emphasize both low- and/or high-order feature interactions by directly combining feature embeddings in latent vector space with various operators. As these methods consider feature interaction from model aspect and the operators are well-designed, they are usually more deployment-friendly compared with cartesian product. However, they still have some downsides that are nonnegligible. Regrettably, the shallow structures of some factorization based methods [9, 23] limit their representative power. More critically, the embeddings generated by their operators take the responsibility of representation learning and feature interaction modeling simultaneously which may hinder the training process. This kind of combination reduces the memory ability of feature interactions, thus degrading the model capacity.

To solve these issues, we propose **C**o-**A**ction **N**etwork (**CAN**) that can capture feature interaction and utilize the mutual and common information of different feature pairs effectively. Specifically, for each feature pair, the embeddings of one side (induction feature) are used to construct an MLP applied to the other side (feed feature) in the co-action unit. This paradigm of feature interaction modeling reduces the additional parameter from $O(N^2 \times D)$ to $O(N \times (D' + D))$ (N is the number of features, D and D' are the dimensions of parameter used in co-action unit, $D < D' \ll N$) compared with cartesian product. Further, CAN provides more powerful fitting ability and better non-linearity using MLP in the co-action unit compared with traditional factorization based methods. Besides, CAN differentiates the parameter space for representation learning and feature interaction modeling to avoid mutual interference in training. Multi-order enhancement and multi-level independence are further utilized to enrich the expression ability in CAN.

The main contributions of this work are summarized as follows:

- We investigate the effectiveness of cartesian product and point out the potential of implicit feature interaction modeling. Inspired by the independent coding of cartesian product, we design a new feature interaction paradigm that has comparable performance compared with cartesian product yet has much less resource consumption.
- We propose Co-Action Network (CAN) to model feature interaction among raw features at the input stage. Each feature ID in CAN will be distributed to an individual micro-MLP to model the interaction with other features. In this way, CAN improves the expressive ability of modeling feature interaction under limited parameters. The individual micro-MLP has a more powerful expressive ability than the ordinary operators utilized in factorization based methods.
- We conduct extensive experiments on both public and industrial datasets. The consistent superiority validates the effectiveness of CAN in modeling feature interaction compared with other state-of-the-art competitors. The deployment of CAN brings 12% CTR and 8% RPM lift in Alibaba's display advertisement system.

2 RELATED WORK

Several research efforts have been devoted to model feature interactions in CTR prediction. These methods can be divided into several categories: aggregation, graph and factorization based methods. We give a brief introduction and discussion as follows.

Deep CTR prediction models generally follow an Embedding & MLP paradigm. Large-scale sparse input features, or IDs, are first mapped to low dimensional embedding vectors and then aggregated into fixed-length vectors in a group-wise manner. The finally concatenated vector is fed as the input to a multi-layer perceptron (MLP). A series of research work focuses on learning how to aggregate features to obtain discriminative representation for CTR prediction. DIN and DIEN [27, 28] uses attention to activate historical behaviors locally w.r.t. the given target item and successfully captures the diversity characteristics of user interest. MIND [11] uses sufficient multi-vectors to capture complicated patterns lying in the user and items. Moreover, inspired by the success of self-attention architecture in the tasks of sequence learning [2, 21], the transformer is introduced in Feng et al. [4] for feature aggregation. MIMN [13] proposes a memory-based architecture to aggregate features and tackle the challenge of long-term user interest modeling. These aggregation based methods only take feature interaction as the weight of each user action to represent user interest.

Graph based methods like Graph Neural Networks (GNNs) [5, 8] conduct feature propagation for each node, where the neighborhood information is aggregated. [1, 10, 22] propose a spectral graph-based extension of convolutional networks with a self-attention mechanism to graphs for feature propagation. There are also some works [18, 20, 26] that exploit meta-path between different nodes for embedding learning. Although graph-based methods achieve great success on graph data, feature interaction is modeled only by one-dimensional weight indicating the strength of connectives, resulting in insufficient expression of the interaction.

Factorization Machines (FM) [16] is a representative method in the age of shallow models. In FM, feature interaction is modeled as the inner product of latent vectors of features. However, FM uses the same latent vectors in different types of inter-field interactions. DeepFM [6] imposes a factorization machine as "wide" module in Wide&Deep [3] with no need of constructing cartesian product feature manually. In Product-based Neural Network (PNN) [14, 15], a

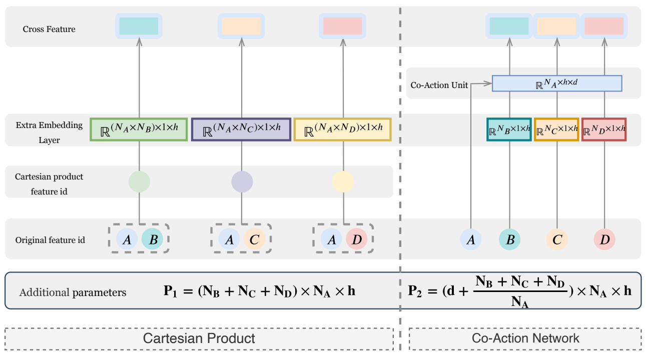
Cross Feature
Co-Action Unit
$\mathbb{R}^{N_A \times h \times d}$
Extra Embedding Layer
$\mathbb{R}^{(N_A \times N_B) \times 1 \times h}$
$\mathbb{R}^{(N_A \times N_C) \times 1 \times h}$
$\mathbb{R}^{(N_A \times N_D) \times 1 \times h}$
$\mathbb{R}^{N_B \times 1 \times h}$
$\mathbb{R}^{N_C \times 1 \times h}$
$\mathbb{R}^{N_D \times 1 \times h}$
Cartesian product feature id
Original feature id
A B
A C
A D
A B C D
Additional parameters
$\mathbf{P_1} = (\mathbf{N_B} + \mathbf{N_C} + \mathbf{N_D}) \times \mathbf{N_A} \times \mathbf{h}$
$\mathbf{P_2} = (\mathbf{d} + \frac{\mathbf{N_B} + \mathbf{N_C} + \mathbf{N_D}}{\mathbf{N_A}}) \times \mathbf{N_A} \times \mathbf{h}$
Cartesian Product
Co-Action Network

Figure 2: Illustration of the evolution from cartesian product to our feature co-action network, where A, B, C and D denote four kinds of the feature. $\mathbf{N_A}, \mathbf{N_B}, \mathbf{N_C}$ and $\mathbf{N_D}$ indicate the number of feature A, B, C and D, respectively. h is the dimension of the feature embedding and d is the dimension of the output from co-action unit. In this figure, there we use feature A to interact with other three features.

product layer is introduced to capture feature interactions between inter-field categories. Deep & Cross Network (DCN) [23] applies feature crossing at each layer. Operation-aware Neural Networks (ONN) [25] learns feature interaction through some different operations. Although the above approaches achieve some performance gain compared with plain DNN, the embedding of each ID takes the responsibility of representation learning and interaction modeling simultaneously and the mutual interferences between them might hurt the performance.

3 REVISITING FEATURE INTERACTION FOR CTR PREDICTION

In the advertisement system, the predicted CTR $\hat{y}$ of a user u clicking on an ad m is calculated via:

$$\hat{y} = \mathrm{DNN}\big(E(u_1), \cdots, E(u_I), E(m_1), \cdots, E(m_J)\big) \tag{1}$$

where $U = \{u_1, \ldots, u_I\}$ is the set of user features including browsing and click history, user profile feature, etc., and $M = \{m_1, \ldots, m_J\}$ is the set of item features. User and item features are usually unique IDs. $E(\cdot) \in \mathbb{R}^d$ means the embedding with size d which maps the sparse IDs into learnable dense vectors as the inputs of DNNs. Besides these unary terms, previous works model feature interaction as binary terms:

$$\begin{aligned}\hat{y} =& \mathrm{DNN}\big(E(u_1), \cdots, E(u_I), E(m_1), \cdots, E(m_J), \\ & \{F(u_i, m_j)\}_{i \in [1, \cdots, I], j \in [1, \cdots, J]}\big)\end{aligned} \tag{2}$$

where $F(u_i, m_j) \in \mathbb{R}^d$ represents the interaction between user feature u_i and item feature m_j. The model can benefit from feature interaction due to the existence of feature co-occurrence, as shown in the example of "Beer and Nappies" in the previous section. Therefore, how to model the feature interaction effectively is crucial for improving performance.

After carefully revisiting previous methods, it can be found that they either take feature interaction as the weights or learn the correlation implicitly with other objectives simultaneously that may produce unsatisfactory results. The most direct way to learn feature interaction is to treat feature combinations as new features and learn an embedding vector for each feature combination directly, e.g., cartesian product. Cartesian product provides independent parameter space and thus has enough flexibility to learn co-action information to improve the ability of prediction.

However, there are some serious defects. The first one is the parameter explosion issue. The parameter space of cartesian product of two features with size N will expand from $O(N \times D)$ to $O(N^2 \times D)$, where D is the dimension of embeddings, which will bring great burden to the online system. In addition, as cartesian product regards <A, B> and <A, C> as totally different features, there is no information sharing between combinations, which also limits the representation ability.

Taking into account both the advantages of cartesian product and the serving efficiency of calculation, we introduce a new way to model feature interaction. As illustrated in Fig.2(a), for each feature pair, its cartesian product produces a new feature and corresponding embedding. Since different feature pairs may share the same feature, there exists an implicit similarity between any two feature pairs, which is ignored by cartesian product. If the implicit similarity can be effectively dealt with, the feature interactions among these pairs could be modeled more effectively and efficiently with a smaller parameter scale than cartesian product. In this paper, inspired by the independent coding of cartesian product, we firstly differentiate the parameters for embedding and feature interaction so that mutual interference is avoided. Considering that DNNs have the powerful fitting ability, we design a co-action unit that parameterizes the feature embeddings in the form of a micro-network. As different feature pairs can share the same micro-network, the similarity information is learned and stored naturally in this micro-network as illustrated in Fig.2(b).

4 CO-ACTION NETWORK

In this section, we propose Co-Action Network (CAN) to efficiently capture feature interaction which firstly introduces a pluggable module, co-action unit. The unit differentiates the parameters for embedding and feature interaction learning. Specifically, it is made up of two sides of information from raw features, i.e., induction and feed side. The induction side is used to construct a micro-MLP while the feed side provides the input for it. Moreover, to promote more non-linearity and deeply excavate feature interaction, multi-order enhancement and multi-level independence are introduced.

4.1 Architecture Overview

The whole architecture of CAN is shown in Fig.3. The features U and M of a user and the target item are fed into CAN in two manners. In the first manner, they are encoded using embedding layer to dense vectors $\{E(u_1), \cdots, E(u_I)\}$ and $\{E(m_1), \cdots, E(m_J)\}$ and further concatenated as e_{item} and e_{user}, respectively. In the second manner, we select a subset U_{feed} and $M_{induction}$ from U and M to model feature interactions $\{F(u_i, m_j)\}_{u_i \in U_{feed}, m_j \in M_{induction}}$ using our proposed co-action unit. The detailed explanation and implementation of co-action unit will be elaborated in the next sub-section. The formulation of CAN is:

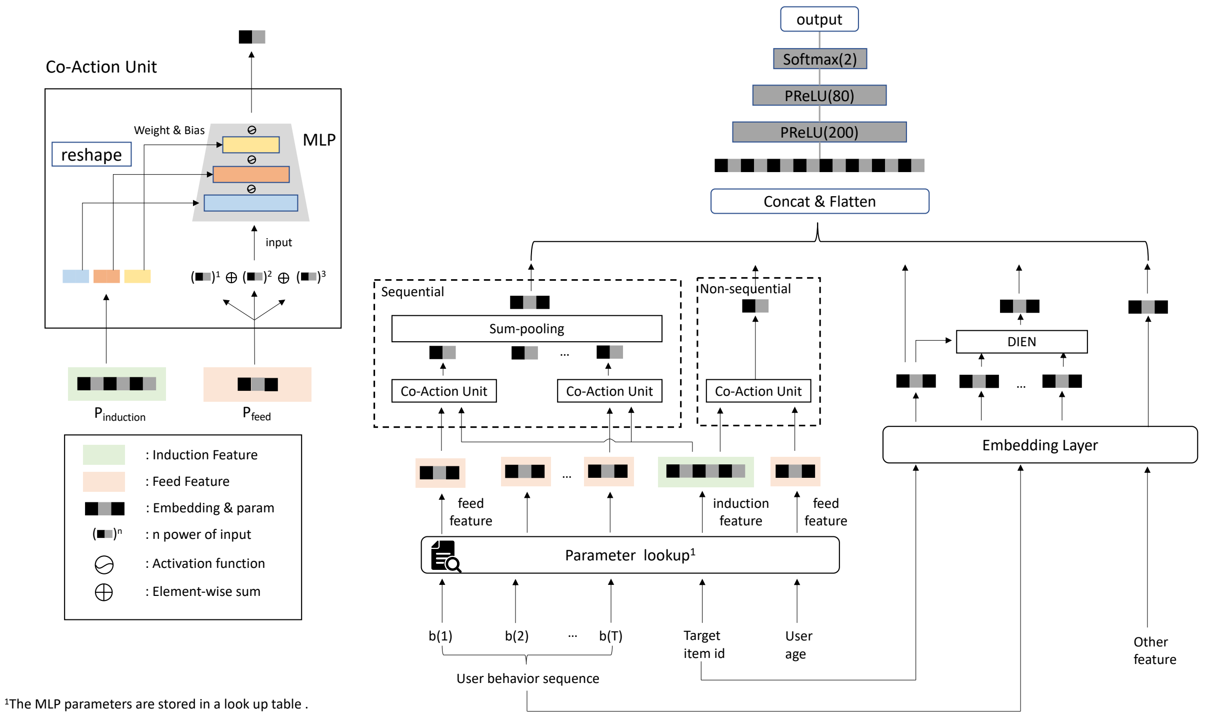

Figure 3: The overall framework of our Co-Action Network. Given target item and user features, the embedding layer encodes the sparse features into dense embeddings. Some selected features are divided into two sides $P_{induction}$ and P_{feed} that are components of co-action unit. $P_{induction}$ parameterizes the micro MLP and P_{feed} serves as the input. The output of co-action unit, together with the common feature embeddings, are used to make final CTR prediction.

$$\hat{y} = \text{DNN}\big(e_{item}, e_{user}, \{F(u_i, m_j)\}_{u_i \in U_{feed}, m_j \in M_{induction}} |\Theta\big) \quad (3)$$

where Θ denotes the parameters in the model and $\hat{y} \in [0,1]$ is the predicted probability of the click behavior. The ground truth click information is denoted as $y \in \{0,1\}$. We finally minimize the cross-entropy loss function between the prediction $\hat{y}$ and label y:

$$\min_{\Theta} -y\log(\hat{y}) - (1-y)\log(1-\hat{y}) \quad (4)$$

4.2 Co-Action Unit

Generally speaking, the co-action unit is an independent MLP for each feature pair, namely micro-MLP with the weight, bias and input of MLP provided by the feature pair. For a specific user feature ID $u_{o'} \in U_{feed}$, we use parameter lookup to obtain learnable parameters $P_{induction} \in \mathbb{R}^{D'}$ while item feature ID $m_o \in M_{induction}$ for $P_{feed} \in \mathbb{R}^{D}$ $(D < D')$. Next, $P_{induction}$ is reshaped and split into the weight matrix and bias vector for the micro-MLP. This process can be formularized as:

$$\Big\|_{i=0}^{L-1} (w_i \| b_i) = P_{induction} \quad (5)$$

$$\sum_{i=0}^{L-1} (|w_i| + |b_i|) = |P_{induction}| = D' \quad (6)$$

where w_i and b_i denote the weight and bias of i-th layer of micro-MLP, $\|$ indicates the concatenation operation, L determines the depth of micro-MLP, $|\cdot|$ gets the size of the variables. A visual illustration of this process is shown in the left part of Fig.3.

P_{feed} is then fed into the micro-MLP and the feature interaction is realized via the concatenation of the output of each layer:

$$h_0 = P_{feed} \quad (7)$$

$$h_i = \sigma(w_{i-1} \otimes h_{i-1} + b_{i-1}), \quad i = 1, 2, \cdots, L \quad (8)$$

$$F(u_{o'}, m_o) = H(P_{induction}, P_{feed}) = \Big\|_{i=1}^{L} h_i \quad (9)$$

where $\otimes$ denotes the matrix multiplication, σ indicates the activation function, H denotes the co-action unit with vector inputs $P_{induction}$ and P_{feed} instead of original symbol F whose inputs are features $u_{o'}$ and m_o.

For sequential features like user behavior history $P_{seq} = \{P_{b(t)}\}_{t=1}^{T}$, the co-action unit is applied to each click behavior followed by a sum-pooling over the sequences:

$$H(P_{induction}, P_{seq}) = H(P_{induction}, \sum_{t=1}^{T} P_{b(t)}) \quad (10)$$

In our implementation, $P_{induction}$ gets the information from item features while P_{feed} is from user features. However, P_{feed} can also serve as the parameter of micro-MLP and vice versa for

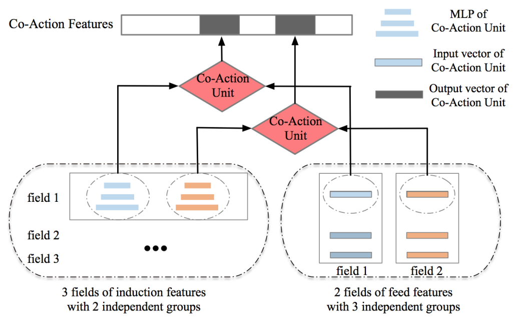
Co-Action Features
MLP of Co-Action Unit
Input vector of Co-Action Unit
Output vector of Co-Action Unit
Co-Action Unit
Co-Action Unit
field 1
field 2
field 3
field 1
field 2
3 fields of induction features with 2 independent groups
2 fields of feed features with 3 independent groups

Figure 4: Illustration of combination independence.

$P_{induction}$. Empirically, in the advertisement system, the candidate items are a small part of all items so that their number is less than that of the items in user click history. Hence we choose $P_{induction}$ as the micro-MLP parameter to reduce the total parameter, which makes the learning procedure easier and more stable.

Note that the number of micro-MLP layers depends on the difficulty of learning. Empirically, a larger feature size usually requires deeper MLP. In fact, FM[6, 16] can also be regarded as a special case of CAN when the micro-MLP is one layer $1 \times D$ matrix without bias and activation function.

The proposed co-action unit has at least three advantages compared with other methods. First, different from previous works that use the same latent vectors in different types of inter-field interactions, the co-action unit utilizes the computational ability of micro-MLP and couples two component features $P_{induction}$ and P_{feed} dynamically rather than a fixed model, which provides more capacity to guarantee the disentangled update of two field features. Second, a smaller scale of the parameters needs to learn. For instance, considering two features with both N IDs, the parameter scale of their cartesian product should be $O(N^2 \times D)$, where D is the dimension of embeddings. However, by using co-action unit, this scale will decrease to $O(N \times (D' + D))$, where D' is the dimension of the $P_{induction}$ in co-action unit. Fewer parameters are not only conducive to learning but also can effectively reduce the burden of the online system. Third, the co-action unit has a better generalization to new feature combinations compared with cartesian product. Given a new feature combination, the co-action unit still works as long as their embeddings of two sides are trained before.

4.3 Multi-order Enhancement

The aforementioned feature is formed upon the first-order features. However, feature interaction can be estimated over high-orders. Although the co-action unit can implicitly learn the high-order feature interaction considering the non-linearity of micro-MLP, the learning process is supposed to be difficult because of the sparsity of feature interaction. To this end, we explicitly introduce multi-order information to obtain a polynomial input. This is achieved by applying micro-MLP to different orders of P_{feed}:

$$H_{\text{Multi-order}}(P_{induction}, P_{feed}) = \sum_{c=1}^{C} H\left(P_{induction}, (P_{feed})^c\right) \tag{11}$$

where C is the number of orders. We utilize Tanh to avoid the numerical problem caused by high-order terms. The multi-order enhancement effectively promotes the model's non-linear fitting ability without bringing additional computational and storage costs.

4.4 Multi-level Independence

Learning independence is one of the main concerns for feature interaction modeling. To ensure learning independence, we propose a three-level strategy from different aspects.

The first level, parameter independence, which is necessary. As mentioned in Sec.4.2, our approach disentangles the update of representation learning and feature interaction modeling. The parameter independence is the basis of our CAN.

The second level, combination independence, which is recommended. The feature interaction grows linearly as the number of feature combinations increases. Empirically, the target item features like "item_id" and "category_id" are selected as the induction side while the user features are for the feed side. Since one induction side micro-MLP can be combined with several feed sides and vice versa, our approach can easily enlarge the expression capability of the model exponentially. We illustrated this idea in Fig.4. Formally, if the induction and feed sides have Q and S groups respectively, the combinations of feature interaction should satisfy:

$$|P_{induction}| = \sum_{s=1}^{S} \sum_{i=0}^{L_s-1} \left(|w_i(s)| + |b_i(s)|\right) \tag{12}$$

$$|P_{feed}| = \sum_{q=1}^{Q} |x(q)| \tag{13}$$

where $|x(q)|$ is the input dimension of the q-th micro-MLP. In the forward pass, the feed feature is divided into several parts to fulfill each micro-MLP.

The third level, orders independence, which is optional. To further improve the flexibility of feature interaction modeling in multi-order inputs, our approach makes different induction side embeddings for different orders. However, the dimension of these embeddings correspondingly increases C times similar to Eq.12.

The multi-level independence helps the feature interaction modeling but at the same time, brings additional memory access and computations. There is a trade-off between the independence level and deployment cost. Empirically, the higher independence level the model uses, the more training data it needs. In our real system, three levels of independence are used yet only parameter independence is used in public datasets due to the lack of training samples.

5 EXPERIMENTS

In this section, we present the empirical studies in detail. In Sec.5.1, we first introduce general experimental settings. The results and discussions are elaborated in Sec.5.2. Sec.5.3 evaluates the effect of each component via the ablation studies.

5.1 Experimental Settings

Dataset. We conduct experiments on three publicly accessible datasets for CTR prediction task: Amazon, Taobao and Avazu, whose statistics are summarized in Tab.1.

Table 1: Statistics of datasets used in this paper.

Dataset	Training	Validation	Feature Size
Amazon (book)	1086120	121216	912642
Taobao	691456	296192	5159463
Avazu	36387240	403793	6763060

- **Amazon dataset**[2] contains product reviews and metadata from Amazon. Among 24 categories of products, we select the Books subset in our experiment. Following previous works [13, 27, 28], we randomly select products not rated by a specific user as negative samples and create corresponding user behavior sequence.
- **Taobao dataset**[3] is a set of user behaviors from Taobao's recommender system. The dataset contains about 1 million users whose behaviors include click, purchase, adding items to the shopping cart, etc. The click behaviors for each user are taken and sorted according to the timestamp to construct the behavior sequence.
- **Avazu dataset**[4] is a mobile ad dataset including 11 days (10 days for training and 1 day for test) real industrial data provided by Avazu. Different from Amazon and Taobao datasets, we model the feature interaction using discrete features as this dataset contains various data fields, which is suitable to verify the effect of (non-)sequence to feature interaction modeling. During training, data of the 10th day is regarded as the validation set.

Competitors.

To verify the effectiveness of our approach, we compare CAN with state-of-the-art CTR prediction models focusing on feature interaction modeling.

- **Cartesian product** is the multiplication of two sets to form the set of all ordered pairs. The former of the produced pair belongs to the first set and the latter is from the second set.
- **DeepFM** [6] based on DNN and adopts a product layer to combine the power of factorization machines for recommendation.
- **xDeepFM** [12] aims to generate feature interactions in an explicit fashion and at the vector-wise level using the proposed Compressed Interaction Network (CIN).
- **FFM & DeepFFM** [9] is a variant of Factorization Machines (FMs) with field aware that can classify large sparse data. DeepFFM appends a DNN term to incorporate high order combination information implicitly.
- **PNN** [14] uses a product layer followed by a fully connected layer to explore high-order feature interactions.
- **NCF** [7] presents a neural network architecture to model collaborative filtering between the latent vectors.
- **ONN** [25] proposes Operation-aware Neural Networks which learns different representations for different operations.
- **DIEN** [27] designs an interest extractor layer to capture user interests from user behavior sequence. An interest evolving layer is further used to model the interest evolving process.

For a fair comparison, DNN is used as the base model (**CAN-DNN**) so that the differences of these methods (except DIEN) lie on the feature interaction modeling. Meanwhile, we present additional experiments based on DIEN (**CAN-DIEN**), which is a state-of-the-art method focusing on user interest, to evaluate the promotion of CAN on sequence-oriented modeling.

Implementation details. We implement CAN using Tensorflow. Specifically, for $P_{induction}$, a two-layer MLP is used with input/output dimension set to 16/8 and 8/4. The order of P_{feed} is set to 3. The model parameters are initialized with a Gaussian distribution (with a mean of 0 and standard deviation of 0.01). We use Adam to optimize the training procedure with the batch size set to 128 and the learning rate set to 0.001. A three-layer MLP with layers $200 \times 80 \times 2$ is used for final CTR prediction. The commonly used metric AUC is taken to evaluate the model performance. Note that all experiments are conducted 5 times independently from random train and validation partitions.

5.2 Results

Overall Performance. We report the performance of our proposed CAN and baselines on both three datasets in Tab.3 and Tab.4.

Tab.3 shows the experimental results on Amazon(book) and Taobao dataset. As mentioned in Tab.3, CAN-DNN outperforms other state-of-the-art approaches on both datasets and improves AUC by 3.86% and 4.23%, respectively, compared with the base model DNN. Meanwhile, CAN-DNN surpasses other feature interaction approaches with a large margin, outperforming the strongest feature interaction (non-sequatial) baseline ONN in both two datasets. Therefore, it validates the effectiveness of our approach on interaction modeling. Since the two datasets contain rich user sequential data, sequence-oriented approaches (which are more suitable for real industrial systems) like DIEN performs better than DNN. Thus, we also use DIEN as the base model to evaluate the effects of CAN (CAN-DIEN). The results show that DIEN can still benefit from CAN with 1.46% and 1.36% improvement on AUC, respectively.

Tab.4 shows the experimental results on Avazu dataset. Although CAN is designed mainly for real industrial data that contain a lot of behavior sequences, it's still capable of non-sequential input. The Avazu dataset contains 24 data fields among which we select 9 fields to construct 16 kinds of feature combinations. Empirical results show that CAN outperforms all the other approaches.

Compare with Cartesian Product. Firstly, it's worth noting that, as a pure representation learning method, cartesian product method could achieve better performance compared with other embedding combination methods. It indicates that although those combination methods could extract some information of feature interaction, there is still a large gap to direct combination encoding, i.e., cartesian product. As shown in Tab.2, CAN achieves comparable results with the cartesian product by only 1/6 parameters.

Meanwhile, we found there has a strong overlap between CAN and cartesian product in feature interaction, for example, CAN (+3.86%) and cartesian product (+3.28%) only brings 5.42% lift on Amazon(book) dataset. The experimental results on Taobao dataset and Avazu dataset show a similar result and this reveals that CAN can effectively model the feature co-occurrence.

Parameter Number Analysis. Fig.5 shows the parameter number and corresponding test AUC of different methods on Amazon book dataset. In CAN (Small) model, we set the dimensions of tensor in co-action unit and extra embedding layers to a small value to

[2]http://jmcauley.ucsd.edu/data/amazon/
[3]https://tianchi.aliyun.com/dataset/dataDetail?dataId=649
[4]https://www.kaggle.com/c/avazu-ctr-prediction

Table 2: The AUC performance of Cartesian Product and CAN based on DNN in Amazon (book), Taobao and Avazu datasets.

Model \ Dataset	Amazon (book)		Taobao		Avazu	
	AUC (mean ± std)	Parameter	AUC (mean ± std)	Parameter	AUC (mean ± std)	Parameter
DNN	0.7640 ± 0.0007	1.0x	0.8470 ± 0.0011	1.0x	0.7624 ± 0.0008	1.0x
+ Cartesian	0.7891 ± 0.0007 (3.29% ↑)	17.0x	0.8863 ± 0.0012 (4.64% ↑)	16.5x	0.8041 ± 0.0014 (5.47% ↑)	21.0x
+ CAN	0.7935 ± 0.0007 (3.86% ↑)	3.3x	0.8828 ± 0.0016 (4.23% ↑)	2.6x	0.8037 ± 0.0013 (5.42% ↑)	3.3x
+ CAN + Cartesian	0.8054 ± 0.0007 (5.42% ↑)	18.8x	0.8967 ± 0.0017 (5.87% ↑)	15.3x	0.8120 ± 0.0014 (6.51% ↑)	23.4x

Table 3: The AUC performance on Amazon (book) and Taobao datasets (sequential).

Model \ Dataset	Amazon (book)	Taobao
	AUC (mean ± std)	AUC (mean ± std)
FFM	0.7523 ± 0.0004	0.7918 ± 0.0016
DNN	0.7640 ± 0.0007	0.8470 ± 0.0011
DeepFM	0.7682 ± 0.0005	0.8500 ± 0.0012
DeepFFM	0.7711 ± 0.0004	0.8545 ± 0.0016
xDeepFM	0.7697 ± 0.0005	0.8573 ± 0.0012
PNN	0.7801 ± 0.0002	0.8649 ± 0.0014
NCF	0.7820 ± 0.0005	0.8717 ± 0.0023
ONN	0.7851 ± 0.0007	0.8752 ± 0.0011
DIEN	0.8346 ± 0.0007	0.9262 ± 0.0011
CAN-DNN	**0.7935 ± 0.0007**	**0.8828 ± 0.0016**
CAN-DIEN	**0.8468 ± 0.0008**	**0.9388 ± 0.0013**

Table 4: Results on Avazu dataset (non-sequential).

Model	AUC (mean ± std)
FFM	0.7580 ± 0.0014
DNN	0.7624 ± 0.0008
DeepFM	0.7712 ± 0.0015
DeepFFM	0.7746 ± 0.0013
xDeepFM	0.7664 ± 0.0014
PNN	0.7871 ± 0.0011
NCF	0.7865 ± 0.0012
ONN	0.7902 ± 0.0014
CAN-DNN	**0.8037 ± 0.0013**

keep similar parameter number with other methods. As shown in Fig.5, CAN (Small) significantly outperforms other baselines which have similar number of parameters, such as ONN. It proves that the improvement of our CAN does not come from the amplified parameter number but from feature interaction modeling by co-action unit. Moreover, the performance can be further improved by increasing the parameter of CAN since the feature interaction modeling is closely related to the learning of micro-MLP.

5.3 Ablation Study

To investigate the effect of each component, we conduct several ablation studies, which are shown in Tab.5.

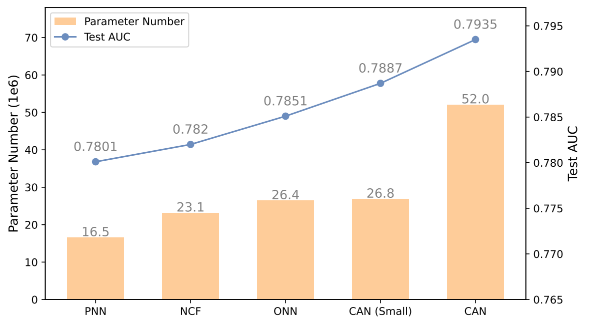

Figure 5: Parameter number (orange bar) and corresponding test AUC (blue line) of different methods.

MLP Layer. (mentioned in Sec. 4.2) First, we show the influences of MLP_{can} architecture to the feature interaction modeling. Specifically, we train models with a different number of MLP layers: 1 layer, 2 and 3 layers. The input/output dimension of each layer is set to 16/8, 8/4 and 4/4, respectively. In general, deeper MLP leads to higher performance. However, when the number of layers grows to 3, there is a decrease in AUC. The main reason may be that the network is not well trained as more complex network architecture usually requires more training data to get convergence.

Activation Function. Second, we study the influences of the activation function. As can be seen from Tab.5, the non-linearity improves AUC by 0.23%. The Tanh activation function plays the role of a normalizer to avoid numerical issues in high orders and helps the model to train stably.

Multi-order Enhancement (mentioned in Sec. 4.3). Third, we evaluate the influences of multi orders. Based on 1st order term, 2nd, 3rd and 4th order terms are added, gradually. From 1st order to 3rd order, AUC promotes a lot. Afterward, as the order grows, the gap starts to shrink even causing a negative effect. The multi orders have a marginal effect on the performance gain so that 2 or 3 power terms are proper in reality.

6 INDUSTRIAL EXPERIENCE

In this section, we share the industrial experiences of feature interaction modeling in our display advertising system.

Cartesian product is the most direct way in feature interaction modeling as discussed in previous sections. Nevertheless, cartesian product usually leads to heavy resource consumption. On one hand, the model size will expand at an extremely fast rate. The oversized model brings great challenges for storage and network transmission, which further affects the real-time update of the model. On the other

Table 5: Ablation studies on Amazon(book) dataset.

Components	AUC (mean ± std)
MLP in Co-Action Unit	
1 layer	0.7889 ± 0.0007
2 layers	0.7935 ± 0.0007
3 layers	0.7913 ± 0.0013
Activation Function	
w/o activation	0.7917 ± 0.0008
w/ Tanh	0.7935 ± 0.0007
Multi-order Enhancement	
order=1	0.7902 ± 0.0008
order=2	0.7921 ± 0.0012
order=3	0.7935 ± 0.0007
order=4	0.7934 ± 0.0014

hand, it increases the embedding look-up operations in application request as the features are increased at the input stage and this results in latency for system response.

Existing approaches are more friendly in industrial deployment. However, we also noticed that under the scale of the billions of data, the promotions were very limited compared with cartesian product. Meanwhile, simply increasing the parameter space like expanding the embedding size did not bring additional improvements.

CAN is designed as a new feature interaction modeling scheme under this situation. In our advertising system, 21 kinds of features including 6 ad features (e.g., ad_id, item_id, shop_id, etc.) and 15 kinds of user features (e.g., item_history, shop_history, etc.) are selected to construct feature interaction. We notice that CAN can achieve comparable performance compared with cartesian product with only one-tenth of model size.

As mentioned in Sec.4, given the 21 kinds of features, CAN allocates additional 21 embeddings due to feature interaction independence. As the user features are mostly behavior sequences with a length of more than 100, additional memory access is required which causes much response latency. Moreover, the computational costs of feature interaction grow linearly according to the number of feature combinations which also brings considerable response latency to our system. To exert all energy of CAN, many efforts are devoted to reducing the response latency. In the industrial deployment, we optimize CAN from three aspects:

- **Sequence cut-off.** The length of 16 kinds user features ranges from 50 to 200. We elegantly apply sequence cut-off to reduce memory cost, e.g., all user behavior sequences with length 200 are truncated at length 50. The most recent behaviors are kept. The sequence cut-off promotes the QPS (Query Per Second) by 20% yet results in a 0.1% decrease of AUC, which is acceptable.
- **Combination reduction.** 6 ad features and 15 user features can obtain up to 90 combinations which is a heavy burden. Empirically, the combinations of the same kind of ad and user feature can better model the co-occurrence. According to this principle, we keep the combinations like "item_id", "item_click_history", "category_id", "category_click_history" and remove some irrelevant combinations. In this way, the number of feature combinations reduces from 90 to 48 which brings 30% QPS improvement.
- **Computational kernel optimization.** The feature interaction computation refers to a time-consuming large matrix multiplication between $P_{induction}$ and P_{feed} with shape of [batch_size $\times$ M $\times$ dim_in $\times$ dim_out] $\times$ [batch_size $\times$ M $\times$ $T$$\times$ dim_in], where M, T, dim_in and dim_out denote the number of feature interaction, length of user behavior sequence, MLP input and output dimension, respectively. In our case, the dim_in and dim_out are not commonly used shape so that such matrix multiplication is not well optimized by the BLAS (Basic Linear Algebra Subprograms). To solve this, the internal calculation logic is rewritten which brings 60% QPS lift. Besides, we make a kernel fusion which combines several operations (such as Matmul and Tanh) as one to reduce the GPU memory I/O consumption. By doing so, the intermediate GPU memory writing of the matrix multiplication output is avoided, which brings another 47% QPS lift.

The series of optimizations make CAN capable of online serving stably in the main traffic of our advertisement system. In the case of our practice, the CTR prediction step of CAN takes about 10 ms and the system can handle nearly 1.3K QPS per Tesla T4 GPU.

Table 6: The performance in real online advertising system.

Scene	CTR	RPM
Homepage Advertising	+11.4%	+8.8%
Post-purchase Advertising	+12.5%	+7.5%

6.1 Offline and Online Results

In all public datasets, CAN has an absolute increase of about 1.2% in AUC compared with DIEN as shown in Tab.3. Meanwhile, CAN also brings an additional 1.2% GAUC (Group AUC) lift in our 30-days industrial data compared with our online serving model. Tab.6 shows the online A/B test results of CAN on our two main scenes in feeds recommendation, Homepage Advertising and Post-purchase Advertising between Jul. 2020 and Oct. 2020. CAN achieves 11.4%/12.5% CTR and 8.8%/7.5% RPM promotions respectively, which are considerable in industrial practice.

7 CONCLUSION

In this paper, we stress the importance of feature interaction modeling, which is not fully explored by previous works. Inspired by cartesian product, we propose a new feature interaction paradigm using a specially designed network, Co-Action Network (CAN). The CAN disentangles the representation learning and feature interaction modeling via a flexible module, co-action unit. Moreover, multi-order enhancement and multi-level independence are introduced in co-action unit to further promote the ability of feature interaction modeling. The experiments show that the CAN outperforms the previous works. CAN has been deployed in the display advertisement system in Alibaba and served the main traffic. We believe this work has pushed feature interaction learning a step forward and multi-feature and lightweight interaction modeling will be further explored in the future.

REFERENCES

[1] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral Networks and Locally Connected Networks on Graphs. In *ICLR*.

[2] Zhangming Chan, Yuchi Zhang, Xiuying Chen, Shen Gao, Zhiqiang Zhang, Dongyan Zhao, and Rui Yan. 2020. Selection and Generation: Learning towards Multi-Product Advertisement Post Generation. In *EMNLP*. 3818–3829.

[3] 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 *DLRS*. 7–10.

[4] Yufei Feng, Fuyu Lv, Weichen Shen, Menghan Wang, Fei Sun, Yu Zhu, and Keping Yang. 2019. Deep Session Interest Network for Click-Through Rate Prediction. In *IJCAI*. 2301–2307.

[5] Marco Gori, Gabriele Monfardini, and Franco Scarselli. 2005. A new model for learning in graph domains. In *IJCNN*, Vol. 2. 729–734.

[6] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. Deepfm: a factorization-machine based neural network for ctr prediction. In *IJCAI*. 2782–2788.

[7] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*. 173–182.

[8] Wenpeng Hu, Zhangming Chan, Bing Liu, Dongyan Zhao, Jinwen Ma, and Rui Yan. 2019. GSN: A graph-structured network for multi-party dialogues. *IJCAI* (2019), 5010–5016.

[9] Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. 2016. Field-aware factorization machines for CTR prediction. In *RecSys*. 43–50.

[10] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

[11] Chao Li, Zhiyuan Liu, Mengmeng Wu, Yuchi Xu, Huan Zhao, Pipei Huang, Guoliang Kang, Qiwei Chen, Wei Li, and Dik Lun Lee. 2019. Multi-Interest Network with Dynamic Routing for Recommendation at Tmall. In *CIKM*. 2615–2623.

[12] 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 *SIGKDD*. 1754–1763.

[13] Qi Pi, Weijie Bian, Guorui Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Practice on Long Sequential User Behavior Modeling for Click-through Rate Prediction. In *SIGKDD*. 1059–1068.

[14] 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 *ICDM*. 1149–1154.

[15] 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), 1–35.

[16] Steffen Rendle. 2010. Factorization machines. In *ICDM*. 995–1000.

[17] Xiang-Rong Sheng, Liqin Zhao, Guorui Zhou, Xinyao Ding, Binding Dai, Qiang Luo, Siran Yang, Jingshan Lv, Chi Zhang, Hongbo Deng, and Xiaoqiang Zhu. 2021. One Model to Serve All: Star Topology Adaptive Recommender for Multi-Domain CTR Prediction. In *CIKM*. 4104–4113.

[18] Chuan Shi, Binbin Hu, Wayne Xin Zhao, and Philip S. Yu. 2019. Heterogeneous Information Network Embedding for Recommendation. *IEEE TKDE* 31, 2 (2019), 357–370.

[19] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. 2019. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *CIKM*. 1161–1170.

[20] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu. 2011. Path-Sim: Meta Path-Based Top-K Similarity Search in Heterogeneous Information Networks. *Proc. VLDB Endow.* 4, 11 (2011), 992–1003.

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NIPS*. 5998–6008.

[22] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.

[23] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & Cross Network for Ad Click Predictions. In *ADKDD'17*. 12:1–12:7.

[24] 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).

[25] Yi Yang, Baile Xu, Shaofeng Shen, Furao Shen, and Jian Zhao. 2020. Operation-aware Neural Networks for user response prediction. *Neural Networks* 121 (2020), 161–168.

[26] Huan Zhao, Quanming Yao, Jianda Li, Yangqiu Song, and Dik Lun Lee. 2017. Meta-Graph Based Recommendation Fusion over Heterogeneous Information Networks. In *SIGKDD*. 635–644.

[27] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep Interest Evolution Network for Click-Through Rate Prediction. In *AAAI*. 5941–5948.

[28] 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 *SIGKDD*. 1059–1068.