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

Guorui Zhou, Weijie Bian*
Kailun Wu, Lejian Ren, Qi Pi, Yujing Zhang, Can Xiao, Xiang-Rong Sheng, Na Mou, Xinchen Luo, Chi Zhang, Xianjie Qiao, Shiming Xiang, Kun Gai, Xiaoqiang Zhu, Jian Xu
Alibaba Group
Beijing, China

ABSTRACT

Inspired by the success of deep learning, recent industrial Click-Through Rate (CTR) prediction models have made the transition from traditional shallow approaches to deep approaches. Deep Neural Networks (DNNs) are known for its ability to learn non-linear interactions from raw feature automatically, however, the non-linear feature interaction is learned in an implicit manner. The non-linear interaction may be hard to capture and explicitly model the *co-action* of raw feature is beneficial for CTR prediction. *Co-action* refers to the collective effects of features toward final prediction. In this paper, we argue that current CTR models do not fully explore the potential of feature co-action. We conduct experiments and show that the effect of feature co-action is underestimated seriously. Motivated by our observation, we propose feature Co-Action Network (CAN) to explore the potential of feature co-action. The proposed model can efficiently and effectively capture the feature co-action, which improves the model performance while reduce the storage and computation consumption. Experiment results on public and industrial datasets show that CAN outperforms state-of-the-art CTR models by a large margin. Up to now, CAN has been deployed in the Alibaba display advertisement system, obtaining averaging 12% improvement on CTR and 8% on RPM.

KEYWORDS

CTR Prediction, Neural Networks, Feature Crosses

1 INTRODUCTION

With the growing complexity of machine learning models, especially models in recommender system, how to deal with abundant input features effectively and efficiently becomes a crucial problem. For online recommender in industrial setting, models are often trained on billion-scale binarized sparse features with one-hot encoding [3, 22]. Each feature can also be seen as a unique ID, which is mapped to a low dimensional embedding firstly and then fed into the model. A simple way to deal with the large-scale input is to consider each feature independent. Under this assumption, there are no connections between features so that a generalized linear models can be directly trained to estimate the click-through rate based on the combination (e.g., concatenation) of features.

However, features like "recommended item" and "user click history" in recommender system are highly relevant [21, 22], i.e., there exists feature collective effects toward final prediction target, such as click-through rate, namely feature **co-action**. For example, a female user that has "bathing suit" in her clicked history is likely to click a recommended "goggle" due to the co-action of "bathing suit" and "goggle". Feature co-action can be considered as modeling the sub-graph of a set of raw features. If the sub-graph only consist of two features, then modeling feature co-action is equivalent to modeling the edges between two IDs. The effect of the co-action explains how a set of features correlates with the optimization target. As shown in Fig.1, feature co-action explicitly bridges the feature pair $[A, B]$ to the target label.

Several research efforts have been devoted to model feature co-action in recent years. These methods can be divided into three categories. Aggregation based methods [5, 10, 11, 21, 22] focus on learning how to aggregate the historical behaviour sequence of user to obtain discriminative representation for the CTR prediction. These methods utilize the feature co-action to model the weight of each user action in historical behaviour sequence. The weighted user behavior sequence is then sum-pooled to represent user interest. Graph based methods [6, 9, 15] regard the features as nodes, which are connected as a directed or undirected graph. In this case, the feature co-action serves as an edge weight for information propagation along edges. Different from aggregation and graph based methods in which the feature co-action are modeled as the weight, the combinatorial embedding methods [12, 14, 19] models the feature co-action by explicitly combining feature embeddings.

Although previous methods have led to improvement on CTR prediction in different way, they still have some downsides. Aggregation based methods and graph based methods model the feature co-action only by the edge weights, however, the edges are only used for information aggregation but not information augmentation. Combinatorial embedding methods, on the other hand, combining embeddings of two features to model feature co-action. For example, PNN [12] performs inner or outer product of two features to augment the input. One major downside of combinatorial embedding methods is that the embeddings take the responsibility of both representation learning and co-action modeling. The representation learning and co-action modeling may conflict with each other thus bound the performance.

In this paper, we stress the importance of feature co-action modeling and argue state-of-the-art methods underestimate the importance of co-action seriously. These methods fail to capture the feature co-action due to the limited expressive power. The importance of capturing feature co-action to augment the input is that it can reduce the difficulty for the model to learning and capture the co-action. Suppose there exists an optimal function $F_*(A, B)$ that models the co-action between feature A and B, the learning

*Guorui Zhou and Weijie Bian contributed equally to this research. Corresponding author is Guorui Zhou. {guorui.xgr, weijie.bwj, kailun.wukailun, lejian.rlj}@alibaba-inc.com

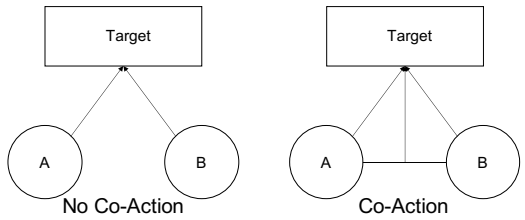

Figure 1: A schematic illustration of feature co-action.

difficulty can be substantially alleviated by explicitly providing $F_*(A, B)$ in the input stage.

To validate our hypothesis that current approaches fail to fully capture the feature co-action, we revisit state-of-the-art methods and design experiments to show that a simple way exploring the potential of feature co-action can boost the performance. For example, if feature A and B are selected, then the co-occurrence of A and B is treated as a new feature and fed to the model. We refer to this baseline as **cartesian product** model. Although cartesian product is the most direct method to do co-action modeling, it has some serious defects, such as huge parameter quantity, totally independent embedding with low feature frequency learning. However, it is surprising that according to some preliminary experiments in this paper, we discover that most of the state-of-the-art combinatorial embedding methods are totally beat by cartesian product. We speculate that situation may due to poor expressiveness and the inability to learn the embedding balancing representation and co-action modeling of these methods.

To this end, we propose feature **C**o-**A**ction **N**etwork (**CAN**) that can capture the feature co-action at the input stage and utilize the mutual and common information of different feature pairs effectively. Instead of directly parameterizing the cartesian product embeddings, CAN parameterizes the embeddings generation network. The re-parameterization reduces the additional parameter from $O(N^2 \times D)$ to $O(N \times T)$ (N is the number of features and D / T are the dimension of parameters with D,T $\ll$ N and D < T) while achieving better performance. Specifically, CAN differentiate the embedding space for representation learning and co-action modeling, where the embedding generation network are derived from the co-action embedding space. In this manner, CAN enriches its expressive power and alleviates the conflict between representation learning and co-action learning. Compared with cartesian product model, CAN reduces the storage and computation consumption significantly thanks to the improved utilization of parameters.

The main contributions of this work are summarized as follows:

- We stress the importance of feature co-action modeling, which is underestimated seriously by state-of-the-art methods. Specifically, we revisit existing methods that models feature co-action. The empirical results show that these methods can not catch the performance of cartesian product baseline. This reveals that current CTR models do not fully explore the potential of raw feature co-action.
- Motivated by our observation, we propose an lightweight model, Co-Action Network (CAN), to model the co-action among raw features. The proposed model can efficiently and effectively capture the feature co-action, improving the model performance while reducing the storage and computation consumption.
- We conduct extensive experiments on both public dataset and industrial environment. The consistent superiority validates the efficacy of CAN. Up till now, CAN has been deployed in the Alibaba display advertisement system. The deployment of CAN brings an averaging 12% CTR and 8% RPM lift.
- We present techniques for deploying CAN in industrial environment. The idea of CAN to exploit feature co-action and our lessons learned generalize to other setups and are thus of interest to both researchers and industrial practitioners.

2 RELATED WORK

Several research efforts have been devoted to model feature co-action for CTR prediction. These methods can be divided into three categories: aggregation based methods, graph based methods and combinatorial embedding methods. We give a brief introduction in the following subsections.

2.1 Aggregation Based Methods

Deep click-through rate prediction models generally follow a Embedding & MLP paradigm. In these methods, large scale sparse input features, or IDs, are first mapped into low dimensional embedding vectors and then aggregated into fixed-length vectors in a group-wise manner. The finally concatenated vector is fed as input to a multi-layer perceptron (MLP). A series of works focus on learning how to aggregate features to obtain discriminative representation for the CTR prediction. Different neural architectures such as CNN, RNN, Transformer and Capsule are utilized to aggregate features. DIN [22] is one of the representative work that employs the attention mechanism for feature aggregation. It uses attention to activate historical behaviors locally w.r.t. the given target item, and successfully captures the diversity characteristic of user interest. DIEN [21] further proposes an auxiliary loss to capture latent interest from historical behaviors. Additionally, DIEN integrates attention mechanism with GRU to model dynamic evolution of user interest for feature aggregation. MIND [10] argues that a single vector might be insufficient to capture complicated pattern lying in the user and items. Capsule network and dynamic routing mechanism are introduced in MIND to learn multiple representation to aggregate raw features. Moreover, inspired by the success of the self-attention architecture in the tasks of sequence to sequence learning [17], Transformer is introduced in [5] for feature aggregation. MIMN [11] proposes a memory-based architecture to aggregate features and tackle the challenge of long-term user interest modeling.

2.2 Graph Based Methods

A graph contains nodes and edges, where ID features can be represented by node embeddings and feature co-action can be modeled along edges. Graph based methods like Graph Neural Networks (GNNs) [6] conduct feature propagation for each node, where the neighborhood information are aggregated. The feature co-action is modeled as edge weights, which is used for feature propagation

that smoothing the node embedding locally along the edges. [2] first propose a spectral graph-based extension of convolutional networks to graphs for feature propagation. GCN [9] further simplifies graph convolutions by stacking layers of first-order Chebyshev polynomial filters with a redefined propagation matrix. In GCNs, the edges are predefined and edge weights are one-dimensional real values. The weights are used to aggregating neighborhood information to model feature co-action. [18] proposes graph attention networks (GAT) that learns to assign different edge weights at each intermediate layer. GAT also model feature co-action by edge weights, however, weights in GAT are a function of nodes due to the attention mechanism. The attention mechanism makes GAT more effective to model feature co-action. There are also some work [15, 16, 20] that exploit meta-path between different nodes for embedding learning. Although graph-based methods achieve great success on graph structured data, the feature co-action is modeled only by one-dimensional weight indicating the strength of connectives. The expressive power may be insufficient to model feature co-action.

2.3 Combinatorial Embedding Methods

Combinatorial embedding methods measure feature co-action in terms of combinatorial embeddings. Factorization Machines (FM) [14] is a representative method in the age of shallow models. In FM, the feature co-action 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, which may cause the *coupled gradient* issue and degrades the model capacity [13]. The *coupled gradient* issue is caused by using the same latent vectors in different types of inter-field interactions, where two supposedly independent features are updated in the same direction during the gradient update process. Besides, the representative power of FM is limited by its shallow nature. Inspired by the success of deep learning, recent CTR prediction model has made the transition from traditional shallow approaches to modern deep approaches. DNNs are powerful to model non-linear interaction at bit-wise level, however, the feature co-action is learned in an implicit fashion. Many works have shown that model feature co-action explicitly by combining feature embeddings is beneficial to CTR prediction. Wide&Deep [3] manually designed cartesian product features as the input of the "wide" module, which is a generalized linear model. The "wide" module is combined with a deep neural network to predict the final score for CTR prediction. DeepFM [7] imposes a factorization machines as "wide" module in Wide&Deep with no need of constructing cartesian product feature manually. Qu et al. [12] proposes Product-based Neural Network (PNN), which introduces a product layer to capture feature co-action between inter-field categories. The output of the product layer is fed as input to the following DNN for the final prediction. Deep & Cross Network (DCN) [19] applies feature crossing at each layer. Although these methods achieve remarkable performance gain compared with plain DNN, they still have some limitation. Specifically, the embedding of each ID takes the responsibility of representation learning and co-action modeling simultaneously. The mutual interferences between representation learning and co-action modeling might hurt the performance. Consequently, the restriction of combinatorial embedding does not make full use of the power of feature co-action.

3 REVISITING FEATURE CO-ACTION FOR CTR PREDICTION

In this section, we first give a brief introduction about feature co-action in CTR prediction. Then we revisit state-of-the-art methods that models feature co-action. In the advertisement system, the CTR $\hat{y}$ of an user u clicking on an ad m is calculated via:

$$\hat{y} = \text{DNN}(E(u_1), \ldots, E(u_i), E(m_1), \ldots, E(m_j)),$$

where $\{u_1, \ldots, u_i\}$ is the set of user features including browsing history, click history, user profile feature, etc, and $\{m_1, \ldots, m_i\}$ is the set of item feature. The $E(\cdot) \in \mathbb{R}^d$ means the embedding which map the sparse IDs into learnable dense vector as the inputs of the DNN. Besides these unary terms, some works model the features interaction as an additional input of the DNN:

$$\hat{y} = \text{DNN}(E(u_1), \ldots, E(u_i), E(m_1), \ldots, E(m_j), \{F(m_i, u_j)\}_{i,j}),$$

where the $F(\{m_i, u_j\}_{i,j}) \in \mathbb{R}^d$ represents the feature interaction between item feature m_i and user feature u_i. The incorporation of feature interaction improves the prediction results, which demonstrates that the combinations of the features from different groups provide additional information. The intuitive reason is that in the CTR prediction task, some features combinations have stronger relationship with label than the isolated features themselves. Taking user click behavior as an example, there is a strong relationship between user click history and the target item that the user may click on due to the existence of users' interests. Therefore, the combination of user click history and target item is a effective co-occurrence feature for the CTR prediction. We call this kind of feature interaction that has strong relationship with label as the feature co-action.

Carefully revisiting previous DNN-based methods, it could be found that some deep neural networks can capture the interaction among specific features even if they do not use the combination features as input. For instance, DIN and DIEN uses attention mechanism to capture the interaction between user behavior features and items. However, the weakness of these methods lie in that they are limited to the feature interaction at the user's interest sequence, and all of them deal with the embedding vector of features while regular embedded vectors in low dimensional space, which often lose a lot of original information.

The most direct implement method is to learn an embedding vector for each combination feature directly, e.g., cartesian product. However, there are some serious defects. The first one is the parameter explosion issue. For instance, two features with size M and N do the cartesian product. The parameter space of cartesian product set will expand from $O(N + M)$ to $O(M \times N)$ comparing with the original parameter space, which will bring great burden to the online system. In addition, there are no information sharing between two combinations that contains same feature, which also limits the representation ability of cartesian product.

Some works try to use special network structures to model features interaction. However, most of these structures interact with

each other without any difference between the representations of feature groups[4, 7].

4 CO-ACTION NETWORK

In order to utilize the feature co-action without being constrained by the limitations of cartesian product and other previous works. We propose a Co-Action Network (CAN) to efficiently capture the inter-field interaction. According to above analysis, previous works do not fully explore the potential of feature co-action. Motivated by the independent coding of feature combination in cartesian product, the CAN introduces a pluggable module, co-action unit. The co-action unit focuses on expanding the parameter space and effectively applying the parameter to model the feature co-action. Specifically, the co-action unit sufficiently leverages the parameter of one side to construct a Multi-Layer Perceptron (MLP) applied to the other side. This kind of feature cross paradigm bring more flexibility to the model. On one hand, increasing the parameter dimension means expanding the MLP parameters and layers. On the other hand, comparing to the cartesian product that shares no information between different feature combination with same feature, the co-action unit improves the utilization of parameters since the MLPs are directly derived from the feature embeddings. Moreover, to incorporate high order information in the model, we introduce multi orders enhancement which explicitly constructs a polynomial input for the co-action unit. The multi orders information promote the model non-linearity and help to better estimate the feature co-action. Besides, multi-level independence including embedding independence, combination independence and order independence are proposed to guarantees the learning independence of co-action by broadening the parameter space.

4.1 Architecture Overview

The whole architecture of CAN is shown in Fig.2. The features of user and target item are fed into the CAN in two manners. In the first manner, all features of user x_{user} and target item x_{item} are encoded as dense vector using embedding layer, which are then concatenated as e_{item} and e_{user}, respectively. In the second manner, part of features from x_{user} and x_{item} are selected and mapped into parameter P_{user} and P_{item} for the co-action unit. The operator of co-action unit is defined as $H(P_{user}, P_{item})$, which play a role of MLP with its parameter taken from P_{item} and its input taken from P_{user}. The detail implementation of co-action unit is elaborated in Sec.4.2.

The final structure of Co-Action Network is formulated as:

$$\hat{y} = \text{DNN}(e_{item}, e_{user}, H(x_{user}, x_{item}, \Theta_{CAN}), \Theta_{DNN}), \quad (1)$$

where $\hat{y}$ is the predicted probability of the click behavior, Θ_{CAN} is the parameters set of the lookup table for co-action unit, and Θ_{DNN} is the parameter set of the DNN. The ground truth is denote as $y \in \{0, 1\}$ and we finally minimize the cross-entropy loss function between the $\hat{y}$ and label y:

$$\min_{\Theta_{CAN}, \Theta_e, \Theta_{DNN}} -y\log(\hat{y}) - (1-y)\log(1-\hat{y}), \quad (2)$$

where Θ_e is the parameter set of feature embedding.

4.2 Co-Action Unit

The detail structures of co-action unit is shown in the left part of Fig.2. The $P_{item} \in \mathbb{R}^{M \times T}$ serves as the weight and bias of each layer in MLP_{can} and the $P_{user} \in \mathbb{R}^{M \times D}$ is fed into the MLP_{can} to output the co-action H, where M denotes the numbers of unique ID, D and T are the dimensions of vector, $D < T$. In fact, the P_{user} can also serve as MLP_{can} parameter and vice versa for P_{item}. Empirically, in the advertisement system, the candidate items are a small part of all items so that its number is less than the items in user click history. Hence we choose P_{item} as the MLP_{can} parameter. The dimension of P_{user} is just the same as input dimension of the MLP_{can} while the P_{item} has a higher dimension since it's the container of weights and biases. In the following sections, we denote P_{item} and P_{user} as the parameters of specific item feature ID and user feature ID for simplicity, where $P_{user} \in \mathbb{R}^D$ and $P_{item} \in \mathbb{R}^T$. The P_{item} is reshaped and split into the weight matrix and bias vector of all MLP_{can} layers. This process can be formulized as:

$$P_{item} = \text{concatenate}(\{\text{flatten}(w^{(i)}), b^i\}_{i=0,\dots,K-1}), \quad (3)$$

$$|P_{item}| = \sum_{i=1}^{K} |w^{(i)}| + |b^{(i)}|, \quad (4)$$

where $w^{(i)}$ and $b^{(i)}$ denote the weight and bias of i-th layer of MLP_{can}, respectively and $|\cdot|$ means the dimension of a matrix or vector. Next, the feature co-action is calculated via:

$$h^{(0)} = P_{user}, \quad (5)$$

$$h^{(i)} = \sigma(w^{(i-1)} \otimes h^{(i-1)} + b^{(i-1)}), \quad (6)$$

$$H(P_{user}, P_{item}) = h^{(K)}, i \in 1, 2, ..., K-1, \quad (7)$$

where $\otimes$ and σ denotes the matrix multiplication and activation function, H is the features co-action defined previously. For sequence features like user click history, the co-action unit is applied to each item followed by a sum-pool over the sequence.

Our proposed co-action unit could achieve at least three advantages compared with other methods. First, different from previous works that using the same latent vectors in different types of inter-field interactions, the co-action unit utilizes the calculate ability of DNNs and couple two component features by dynamical parameter and input but not a fixed model, which provides more capacity to guarantee the update of two field features and avoid couple gradients. Second, less scale of the learnable parameters. The final aim of co-action learning is to learn excellent representation vector of every co-action features. However, directly learning the embedding of the carstein product of the component features needs to learn quite large-scale parameters. For instance, consider two features with both number of N IDs. If we learning the co-action representation by learning the embeddings of their cartesian product, the parameters scale 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 $O(N \times T)$, where the T is the dimension of the parameter of co-action unit and far less than N. Less parameters are not only conducive to learning, but also can effectively reduce the burden of online system. Third, the co-action unit has better generalization to new feature combinations comparing with other previous works. Given a new feature combination, the co-action unit still works as long as embeddings of two sides are trained before.

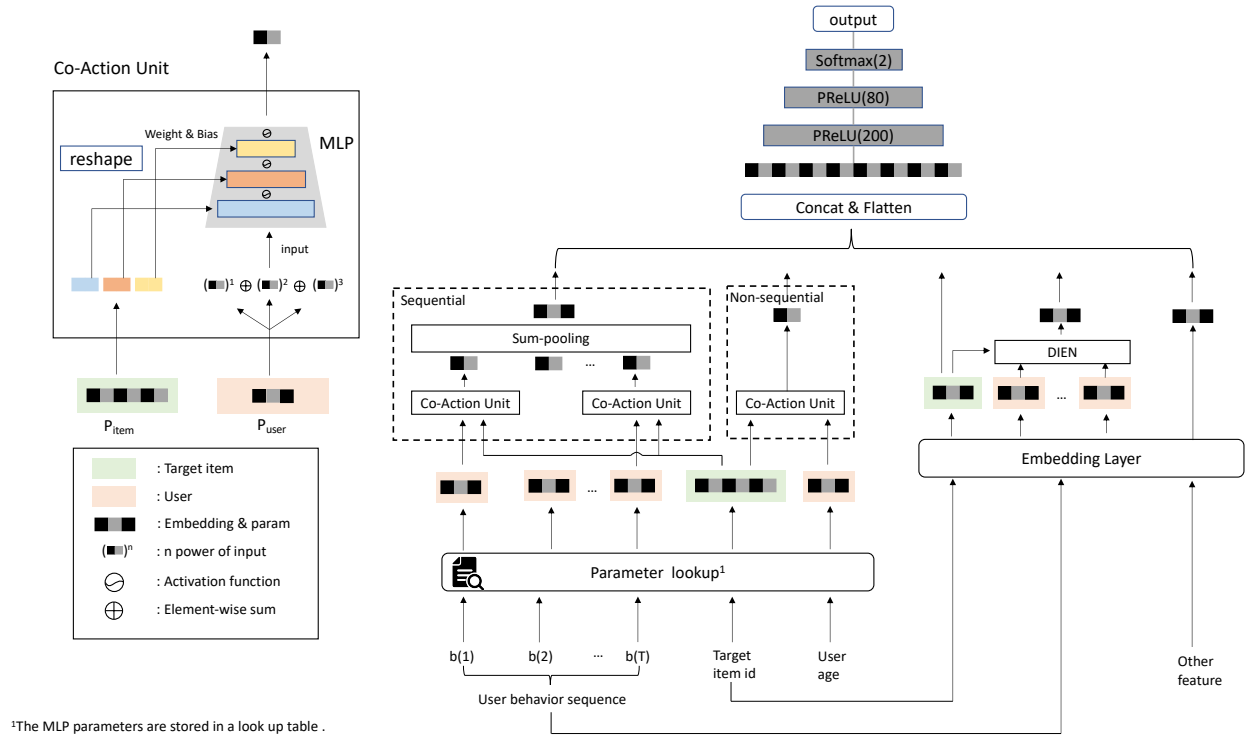

Figure 2: The overall framework of the Co-Action Network. Given target item and user features, the embedding layer encodes the sparse features into dense embeddings. Meanwhile, some features are selected for co-action modeling. Each item feature corresponds to a Multi Layer Perceptron (MLP) through MLP table look up while user features are regarded as input of these MLPs. The output feature co-actions, together with the common feature embeddings, are used to make final CTRs prediction. This figure is best viewed in color.

4.3 Multi-order Enhancement

The aforementioned feature co-action is basically 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, the learning process is suppose to be lengthy. To this end, we explicitly introduce multi-order information in the co-action unit to obtain a polynomial input. This is achieved by applying MLP_{can} to different orders of P_{user}:

$$H(P_{user}, P_{item}) = \sum_{c=1}^{C} MLP_{can}((P_{user})^c) \tag{8}$$

$$\approx MLP_{can}(\sum_{c=1}^{C}(P_{user})^c), \tag{9}$$

where C is the number of orders. Note that SeLU is used as activation function when $C = 1$. Otherwise, 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 for co-action modeling without bringing additional computational and storage cost.

4.4 Multi-level Independence

The learning independence is one of the main concerns for co-action modeling. To ensure the learning independence, we propose a three-level strategy from different aspects according to the importance:

First level, parameter independence, which is necessary. As mentioned in Sec.4.1, our approach differentiate the parameters for representation learning and co-action modeling. The parameter independence is the basis of our CAN.

Second level, combinations independence, which is recommended. The feature co-action grows linearly as the number of feature combinations increases. Empirically, the target item features "like item_id" and "category_id" are selected as the weight-side embeddings while the user features are for the input-side. Since a weight-side embedding can be combined with several input-side and vice versa, our approach enlarges their dimension exponentially. Suppose there are M weight-side and N input-side embeddings, we expand the dimension of weight-side embeddings N times and M times for

input-side:

$$|P_{item}| = (\sum_{i=1}^{K} |w^{(i)}| + |b^{(i)}|) \times N \tag{10}$$

$$|P_{user}| = |x| \times M, \tag{11}$$

where $|x|$ is the input dimension of the MLP_{can}. In the forward pass, these embeddings are divided into several parts to fulfill the MLP operations.

Third level, orders independence, which is optional. To further improve the flexibility of co-action modeling in multi-order inputs, our approach makes different weight-side embeddings for different orders. The dimension of weight-side embeddings correspondingly increases $orders$ times similar to Eq.10. Note that as the MLP_{can} share no parameters in different order terms, the approximation in Eq.8 is not feasible.

The co-action independence help the co-action modeling but at the same time, bring additional memory access and computational costs. There is trade-off between the independence level and deployment costs. Emperically, the higher independence level the model use, the more training data the model need. In our advertisement system, three levels of independence are used yet only embedding independence is used in public dataset due to the lack of training samples.

5 EXPERIMENTS

In this section, we present the experiments in details. In Sec.5.1, we first introduce the used datasets including Amazon dataset, Taobao dataset and Avazu dataset followed by the previous methods and the implementation details. The results and discussion are elaborated in Sec.5.2. Sec.5.3 elaborates the ablation studies. Model universality and generalization are presented in Sec.5.4. Experimental results of industrial data and deployment optimizations are shown in Sec.5.5. Both the public datasets and experimental codes are made available[1].

5.1 Experimental Settings

Dataset. We experiment with three publicly accessible datasets for ctr prediction task: Amazon, Taobao and Avazu. The characteristics of these datasets are listed as follow:

- **Amazon dataset**[2] contains product reviews and metadata from Amazon.Among 24 categories of products, we select the Books subset which contains 75053 users, 358367 items and 1583 categories. As this Amazon dataset is not originally a CTRs prediction dataset, the negative samples are not provided. Following previous works [11, 21, 22], we randomly select products not rated by a specific user as negative sample for this user and create corresponding user behavior sequence (click and non-click). The maximum sequence length is limited to 100.
- **Taobao dataset**[3] is a collection of user behaviors from Taobao's recommender system. The dataset contains about 1 million users who have behaviors including click, purchase, adding item to shopping cart and item favoring. The click behaviors for each user are taken and sorted according to the timestamp to construct the user behavior sequence. The maximum sequence length is limited to 200.
- **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. For Amazon and Taobao datasets, we model the feature co-action based on the user behavior sequences. On the contrary, for the Avazu dataset, we model the feature co-action using the discrete features as the avazu dataset contains various data fields, which is suitable to verify the effect of sequence / non-sequence to feature co-action modeling. During training, the 10th day is regarded as validation set.

The datasets statistics are summarized in Tab.1

Table 1: The datasets used in this paper.

dataset	training	validation	feature size
Amazon (book)	135040	14976	450000
Taobao	691456	296192	5159463
Avazu	36387240	403793	6763060

Baselines. In this paper, we use DIEN as the base model of the CAN. Note that any other model is allowed since the co-action unit is a pluggable module. To verify the effectiveness of our approach, we compare the CAN with current approaches focusing on feature interaction. For fair comparison, DIEN is used as the basis of these approaches.

- **DIEN** [21] designs an interest extractor layer to capture user interests from user behavior sequence. An interest evolving layer is further used to model interest evolving process.
- **Cartesian product** is the multiplication of two sets to form the set of all ordered pairs. The first element of the ordered pair belong to first set and second pair belong the second set.
- **PNN** [12] uses a product layer followed by a fully connected layers to explore high-order feature interactions.
- **NCF** [8] presents a neural network architecture to learn latent features of user and item, which are used to model collaborative filtering using neural networks.
- **DeepFM** [7] is a new neural network architecture adopts product layer combines the power of factorization machines for recommendation and deep learning.

Implementation details. We implement the CAN using Tensorflow [1]. For the P_{item}, eight-layer MLP is used with weight dimension set to 4×4, which results in a dimension of $(4 * 4 + 4) \times 8 = 160$ (bias included). The order of P_{user} is set to 2. The model is trained from scratch and the model parameters is initialized with a Gaussian distribution (with a mean of 0 and standard deviation of 0.01). We use Adam to optimize the training with the batch size set to 128 and learning rate set to 0.001. The three-layers MLP with $200 \times 100 \times 2$ is used for final CTR prediction. The common used metric AUC is used to evaluate the model performance.

[1] https://github.com/CAN-Paper/Co-Action-Network
[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: Comparison with other approaches on Amazon book and Taobao datasets.

Model	Amazon (mean ± std)	Taobao (mean ± std)
DIEN	0.7518 ± 0.0004	0.9028 ± 0.0016
DIEN+Cartesian	0.7608 ± 0.0005	0.9091 ± 0.0012
PNN	0.7589 ± 0.0002	0.9072 ± 0.0014
NCF	0.7536 ± 0.0005	0.9064 ± 0.0023
DeepFM	0.7549 ± 0.0007	0.9049 ± 0.0011
CAN	**0.7690 ± 0.0011**	**0.9095 ± 0.0017**
CAN+Cartesian	**0.7692 ± 0.0008**	**0.9163 ± 0.0013**

Table 3: Ablation studies on Amazon book dataset.

Model	AUC (mean ± std)
MLP layers=2, order=1	0.7656 ± 0.0008
MLP layers=2, order=2	0.7666 ± 0.0012
MLP layers=2, order=3	0.7669 ± 0.0020
MLP layers=2, order=4	0.7647 ± 0.0014
order=2, MLP layers=1	0.7645 ± 0.0007
order=2, MLP layers=2	0.7666 ± 0.0012
order=2, MLP layers=4	0.7688 ± 0.0013
order=2, MLP layers=8	0.7690 ± 0.0011
CAN w/o activation	0.7649 ± 0.0008
CAN w/ SeLU	0.7652 ± 0.0007
CAN w/ Tanh	**0.7690 ± 0.0011**

5.2 Results

Tab.2 shows the experiment results on Amazon and Taobao dataset. As can be seen, The CAN outperforms other state-of-the-art approaches on both datasets. Comparing to the base model DIEN, the CAN improve the AUC by 1.7% and 2.1%, respectively. Meanwhile, the CAN surpasses other co-action approaches with a large margin, which proves the effectiveness of our approach on co-action modeling. It's worth noting that, as the pure representation learning method, the cartesian product method could achieve better performance compared with other combining embedding methods like PNN, NCF, and DeepFM, which indicates that although the these combining embedding methods could extract some information of co-action features, they could really learn embedding with excellent representation and co-action. In contrast, CAN achieves much better results than the cartesian product and combining embedding methods, which means that the network based mechanism of CAN could learn co-action representation both the representational ability and the collaborative ability.

5.3 Ablation Study

To investigate the effect of each component, we conduct several ablation studies, which is shown in Tab.3.

Multi orders First, we evaluate the influences of multi orders. On the basis of 1st order term, 2nd, 3rd and 4th order terms are added, gradually. From 1st order to 2nd order, the AUC promotes a lot. Afterwards, as the order growing, the gap starts to shrink even cause negative effect. The multi orders have a marginal effect to the performance gain so that 2 or 3 power terms is proper in practical applications.

MLP depth. Second, we show the influences of MLP_{can} architecture to the co-action modeling. Specifically, we train models with different number of MLP layers, 1, 2, 4 and 8, respectively. The input and output dimension of MLP layers are the same. In general, deeper MLP leads to higher performance. However, when the number of layers exceed 4, there is no obvious AUC gain, i.e., 8-layers MLP only increase the AUC by 0.02%. The main reason is that the training samples are not enough for such a deep architecture.

Activation functions. Third, we compare the influences of different activation functions. As can be seen from the table, the nonlinearity improves the AUC by 0.03 0.41%. Under the order=2 setting, the Tanh shows more significant performance comparing to SeLU since the Tanh plays a role of normalizer to avoid numerical issues in high orders.

Table 4: Results of different approaches using 16 kinds of feature combinations (DNN excluded) on Avazu dataset. DNN is used as the basic model as the Avazu dataset does not contain sequential features.

Model	AUC (mean ± std)
DNN	0.7854 ± 0.0008
Cartesian product	0.8041 ± 0.0016
PNN	0.7871 ± 0.0011
NCF	0.7865 ± 0.0015
DeepFM	0.7862 ± 0.0014
CAN	**0.8037 ± 0.0017**
CAN+Cartesian	**0.8120 ± 0.0016**

5.4 Model Universality and Generalization

To validate the feature universality and generalization of CAN, we compare the CAN and other approaches from and generation two aspect: validating the co-action features with non-sequences components and predicting the samples with the unseen co-action features while training.

Universality Although the CAN is designed mainly for real industrial data which contain a lot of behaviour sequences, it's still capable for non-sequential input. The Avazu dataset contains 24 data fields among which we select 9 fields to construct 16 kinds of feature combination. As shown in Tab.4, the CAN outperforms most approaches and is comparable to the cartesian product.

Generalization In the real commercial scene, countless feature combinations arise on each day, which requires quickly response of CTR models. The generalization is quite important for practical application. To this end, we remove the samples that contain existing feature combinations from test set. In this way, we obtain a new test set whose feature combinations are brand new for a well trained model. Note that we only require the feature combination to be zero shot rather than all features. From Tab.5, the cartesian product is ineffective under this setting because it relies on well trained co-action embeddings which is not available under this

Table 5: Results of different approaches handling new feature combinations in Amazon dataset.

method	AUC (mean ± std)
DIEN	0.7028 ± 0.0013
DIEN+Cartesian	0.7040 ± 0.0013
NCF	0.7066 ± 0.0019
DeepFM	0.7073 ± 0.0012
CAN	**0.7132 ± 0.0017**

setting. On the contrary, the CAN still works well which shows excellent generalization ability to new feature combination comparing to other approaches. In real industrial environment, the feature combinations are extremely sparse that it's much easier to handle the new feature combinations using CAN so long as the P_{item} and P_{user} are well trained.

5.5 Results on Industrial Data

Online Serving and Challenges. At the very beginning, we deploy the Cartesian product model on our system that causes a lot of troubles. On one hand, the model size is expanding at an extremely fast rate even using IDs frequency filtering. On the other hand, the additional $M \times N$ IDs brings unacceptable number of embedding look up operations as well as system response latency. By contrast, the CAN is much more friendly in this aspect. In order to deploy the CAN on our advertisement system, we select 21 features including 6 ad features and 15 user feature to generate feature combinations so that extra 21 embeddings space are allocated due to co-action independence. The significantly increasing embeddings space still lead to heavy pressure of online serving. As the user features are mostly behaviour sequences with length of more than 100, additional memory access is required which cause response latency rise. Moreover, the computational costs of feature co-action grow linearly according to the number of feature combination which also bring considerable response latency to our system.

Solution. To tackle these problems, much effort are devoted to reduce the response latency. We simplify the model from three aspects:

- Sequence cut-off. The length of 16 user features range from 50 to 200. To reduce the memory access cost, we simply apply sequence cut-off to our user features, e.g., all user behaviour sequences of length 200 are reduce to 50. The most recent behaviours are kept. The sequence cut-off promote the QPS (Query Per Second) by 20% yet results in 0.1% decrease of AUC, which is acceptable.
- Combination reduction. 6 ad features and 15 user feature can obtain up to 90 feature combinations which is a heavy burden. Empirically, the combinations with ad feature and user feature of the same type can better model the feature co-occurrence. According to this principle, we keep the combinations like "item_id" and "item_click_history" and "category_id" and "category_click_history" and remove some irrelevant combinations. In this way, the number of combinations reduce from 90 to 48 which brings 30% QPS improvement.

Table 6: The CTR and RPM gains in real online advertising system.

	CTR	RPM
Scene1	+11.4%	+8.8%
Scene2	+12.5%	+7.5%

- Computational kernel optimization. The co-action computation refers to a time-consuming large matrix multiplication between P_{item} and P_{user} with shape of [Batch_size × K × dim_in × dim_out] × [batch_size × K × seq_len × dim_in], where K, seq_len, dim_in and dim_out denote the number of feature co-action, length of user behavior sequence, MLP input dimension and output dimension, respectively. In our case, The dim_in and dim_out are not commonly used shape so that such matrix multiplication are not well optimized by the BLAS (Basic Linear Algebra Subprograms). To solve this problem, the internal calculation logic is rewritten which brings 60% QPS lift. In addition, as this matrix multiplication is followed by a sum-pooling over the seq_len dimension, we further make a kernel fusion between matrix multiplication and sum-pooling. By doing so, the intermediate GPU memory writing of the matrix multiplication output is avoid, which brings another 47% QPS lift.

The series of optimizations make the CAN capable for online serving in main traffic stably. In our system, the CTR prediction step takes 12 ms using the CAN which can handle nearly 1K QPS per GPU. Tab.6 shows the improvement of CAN on CTR and RPM (Revenue Per Mille) in our online A / B test.

6 CONCLUSION

In this paper, we stress the importance of feature co-action modeling, which is underestimated by previous works. Inspired by cartesian product model, we propose a new feature cross paradigm using a specially designed network, Co-Action Network (CAN). The CAN disentangles the representation learning and co-action 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 co-action modeling. The experiments show that the CAN outperforms the previous works and has better generalization ability to new feature combinations. For now, the CAN is deployed in Alibaba display advertisement system and serving the main traffic.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/ Software available from tensorflow.org.

[2] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral Networks and Locally Connected Networks on Graphs. In *Procedding of the 2nd International Conference on Learning Representations*. Banff, AB, Canada.

[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 *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. ACM, 7–10.

[4] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *Advances in Neural Information Processing Systems 29*. Barcelona, Spain, 3837–3845.

[5] Yufei Feng, Fuyu Lv, Weichen Shen, Menghan Wang, Fei Sun, Yu Zhu, and Keping Yang. 2019. Deep Session Interest Network for Click-Through Rate Prediction. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. 2301–2307.

[6] Marco Gori, Gabriele Monfardini, and Franco Scarselli. 2005. A new model for learning in graph domains. In *Proceedings of the IEEE International Joint Conference on Neural Networks*, Vol. 2. IEEE, 729–734.

[7] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. Deepfm: a factorization-machine based neural network for ctr prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. Melbourne, Australia., 2782–2788.

[8] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*. 173–182.

[9] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations*. Toulon, France.

[10] 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 *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2615–2623.

[11] 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 *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1059–1068.

[12] 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 *Proceedings of the16th International Conference on Data Mining*. IEEE, 1149–1154.

[13] Yanru Qu, Bohui Fang, Weinan Zhang, Ruiming Tang, Minzhe Niu, Huifeng Guo, Yong Yu, and Xiuqiang He. 2019. Product-Based Neural Networks for User Response Prediction over Multi-Field Categorical Data. *ACM Transactions on Information Systems* 37, 1 (2019), 5:1–5:35.

[14] Steffen Rendle. 2010. Factorization machines. In *Proceedings of the 10th International Conference on Data Mining*. IEEE, 995–1000.

[15] Chuan Shi, Binbin Hu, Wayne Xin Zhao, and Philip S. Yu. 2019. Heterogeneous Information Network Embedding for Recommendation. *IEEE Transactions on Knowledge and Data Engineering* 31, 2 (2019), 357–370.

[16] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu. 2011. PathSim: Meta Path-Based Top-K Similarity Search in Heterogeneous Information Networks. *Proceedings of the VLDB Endowment* 4, 11 (2011), 992–1003.

[17] 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 *Advances in Neural Information Processing Systems 30*. 5998–6008.

[18] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *Procedding of the 6th International Conference on Learning Representations*. Vancouver, BC, Canada.

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

[20] Huan Zhao, Quanming Yao, Jianda Li, Yangqiu Song, and Dik Lun Lee. 2017. Meta-Graph Based Recommendation Fusion over Heterogeneous Information Networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 635–644.

[21] 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 *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*. Honolulu, Hawaii, USA, 5941–5948.

[22] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1059–1068.