Enhancing Explicit and Implicit Feature Interactions via Information Sharing for Parallel Deep CTR Models

Bo Chen[1*], Yichao Wang[1*], Zhirong Liu[1], Ruiming Tang[1],
Wei Guo[1], Hongkun Zheng[2], Weiwei Yao[2], Muyu Zhang[2], Xiuqiang He[1]
[1]Huawei Noah's Ark Lab [2]Huawei Technologies Co Ltd
{chenbo116, wangyichao5, liuzhirong, tangruiming,
guowei67, zhenghongkun1, yaoweiwei4, zhangmuyu, hexiuqiang1}@huawei.com

ABSTRACT

Effectively modeling feature interactions is crucial for CTR prediction in industrial recommender systems. The state-of-the-art deep CTR models with parallel structure (*e.g.*, DCN) learn explicit and implicit feature interactions through independent parallel networks. However, these models suffer from trivial sharing issues, namely insufficient sharing in hidden layers and excessive sharing in network input, limiting the model's expressiveness and effectiveness. Therefore, to enhance information sharing between explicit and implicit feature interactions, we propose a novel deep CTR model EDCN. EDCN introduces two advanced modules, namely *bridge module* and *regulation module*, which work collaboratively to capture the layer-wise interactive signals and learn discriminative feature distributions for each hidden layer of the parallel networks. Furthermore, two modules are lightweight and model-agnostic, which can be generalized well to mainstream parallel deep CTR models. Extensive experiments and studies are conducted to demonstrate the effectiveness of EDCN on two public datasets and one industrial dataset. Moreover, the compatibility of two modules over various parallel-structured models is verified, and they have been deployed onto the online advertising platform in Huawei, where a one-month A/B test demonstrates the improvement over the base parallel-structured model by 7.30% and 4.85% in terms of CTR and eCPM, respectively.

KEYWORDS

Click-Through Rate Prediction, Neural Network, Feature Interactions

ACM Reference Format:
Bo Chen[1*], Yichao Wang[1*], Zhirong Liu[1], Ruiming Tang[1],, Wei Guo[1], Hongkun Zheng[2], Weiwei Yao[2], Muyu Zhang[2], Xiuqiang He[1]. 2021. Enhancing Explicit and Implicit Feature Interactions via Information Sharing for Parallel Deep CTR Models. In *Proceedings of the 3rd Workshop on Deep Learning Practice for High-Dimensional Sparse Data with KDD (DLP-KDD '21), August 15, 2021, Singapore.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

*Co-first authors with equal contributions. Ruiming Tang is the corresponding author.

DLP-KDD '21, August 15, 2021, Singapore

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Click-through rate (CTR) prediction is critically important for industrial recommender systems [3, 8] and online advertising [25], which estimates the probability of a user clicking on a recommended item and decides whether this item will be exposed to the user. The estimated CTR may influence the subsequent recommendation decision-makings, such as item ranking and ad placement. Consequently, the accuracy of CTR prediction is a crucial factor for the user experience and platform revenue [22, 35].

Effectively modeling feature interactions is one of the most commonly-used optimization approaches to improve the prediction accuracy of CTR models. Early researches focus on design and utilize beneficial combining features, such as FTRL [21], FM [26], FFM [13] and Higher-Order FM (HOFM) [1]. Besides, the industry also spends a lot of workforce on feature interaction mining. These models leverage feature interactions explored explicitly by either experts' experience or pre-defined formulas for CTR prediction. However, current large-scale recommender systems contain enormous raw features and high-order feature interaction patterns [5, 37], which makes manual feature engineering or pre-defined formulas hard to cover all the essential interaction patterns in the feature space, thus limiting the usage of such shallow models in industry.

Deep learning-based CTR models, emerging rapidly in recent years, have an aptitude for capturing informative feature interactions in an end-to-end manner, getting rid of the hindrance of manual feature engineering and pre-defined formula. Representative models, such as Wide & Deep [2], DeepFM [5], DCN [30], PIN [24], DIN [37] and DIEN [36], learn explicit and implicit feature interactions jointly and achieve significant performance boost. As observed in [31, 34], these deep CTR models can be divided into two families with respect to the way of combining the networks for modeling explicit and implicit feature interactions, namely ***parallel structure*** and ***stacked structure*** (shown in Figure 1). Models with stacked structure combine the two networks, where one for explicit bounded-degree feature interactions and the other for implicit feature interactions, in a successive style, such as PIN, DIN and DIEN. On the other side, parallel structure models jointly train the two networks in a parallel manner, such as DeepFM, DCN and xDeepFM (shown in Figure 2).

In this paper, we focus on optimizing the models with ***parallel structure*** by enhancing the explicit and implicit feature interactions via information sharing. To make our presentation simple and straightforward, we take Deep & Cross Network (DCN [30])

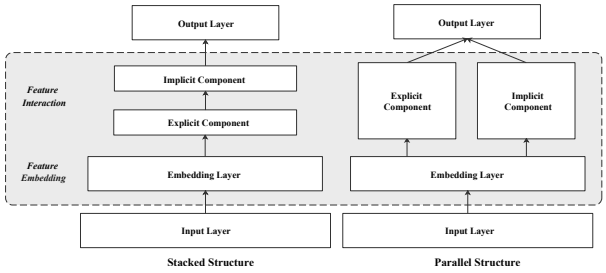


Figure 1: The backbone architecture comparison between stacked structure and parallel structure deep learning-based CTR models.

for illustration, which is a representative parallel-structured model with a good balance between model performance and efficiency. Nevertheless, the trivial sharing strategy performed between the two parallel networks limits its expressiveness and effectiveness, elaborated as follows.

Insufficient sharing in hidden layers. DCN performs cross network (explicit interaction) and deep network (implicit interaction) parallelly and independently, and the learned latent representations do not fuse until the last layer. We refer such fusion pattern as *late fusion*. With late fusion, explicit and implicit feature interaction networks do not share information in the intermediate hidden layers, which weakens the interactive signals between each other and may easily lead to skewed gradients during the backward propagation [9]. To summarize, such insufficient sharing strategy in hidden layers impedes the learning procedure of effective feature interactions in these parallel structure models.

Excessive sharing in network input. The cross network and deep network in DCN share the same embedding layer as input, meaning that all the input features are fed into the parallel networks indiscriminately. However, as pointed out in [14], different features are suitable for different interaction functions. Therefore, excessive sharing the network inputs and feeding all the features indiscriminately to the parallel networks may not be a reasonable choice and result in sub-optimal performance.

To solve the above-mentioned trivial sharing issues existed widely in various parallel structure models, we propose a new deep CTR model, named ***E**nhanced **D**eep & **C**ross **N**etwork* (EDCN) based on DCN. Specifically, two novel modules, namely ***bridge module*** and ***regulation module***, are introduced to tackle the insufficient sharing in hidden layers and excessive sharing in network input, respectively. On the one hand, bridge module performs *dense fusion* by building connections between cross and deep networks, so as to capture the layer-wise interactive signals between parallel networks and enhance the feature interactions. On the other hand, regulation module is designed to learn discriminative feature distributions for different networks by a field-wise gating network in a *soft selection* manner. Moreover, regulation module is also able to work jointly with bridge module to further learn reasonable inputs for each hidden layer, making two parallel networks learn explicit and implicit feature interactions collaboratively. These two modules are lightweight and model-agnostic, which can be generalized well to various parallel-structured deep CTR models for significantly improving performance with low time and space complexity.

To summarize, our contributions are as follows:

- We analyze the trivial sharing issues existed in parallel structure models, namely insufficient sharing in hidden layers and excessive sharing in network input, and propose a novel deep CTR model EDCN to enhance the information sharing between explicit and implicit feature interactions.
- In EDCN, bridge module is proposed to capture the layer-wise interactive signals between parallel networks, while regulation module is designed to learn discriminative feature distributions for each hidden layer of the two networks.
- Our proposed bridge module and regulation module are lightweight and model-agnostic, which can be generalized well to mainstream parallel deep CTR models for boosting performance.
- Extensive experiments are conducted on two public datasets and one industrial dataset to demonstrate the superiority of EDCN over the SOTA baselines. Moreover, the compatibility of bridge module and regulation module over various parallel-structured CTR models is also verified. A one-month online A/B test in a Huawei advertising platform shows that two modules improve the base model by 7.30% and 4.85% in terms of CTR and eCPM.

2 RELATED WORK

2.1 Feature Interactions

Most existing deep CTR models follow a **feature embedding & feature interaction** paradigm. According to different combinations of explicit and implicit components in the feature interaction, deep CTR models can be classified into two families: *stacked structure* and *parallel structure*, whose backbone architecture comparison is shown in Figure 1. Models with stacked structure first perform explicit feature interaction modeling over the embedding layer and then stack an implicit component (normally DNN) to extract high-level feature interactions successively. Representative models include PNN [23], NFM [7], OENN [6], DIN [37] and DIEN [36]. PNN, NFM and OENN leverage the product operation (inner product, outer product or hadamard product) to model explicit feature interaction. Besides, DIN and DIEN utilize attention operation [29] to capture explicit feature interactive signals. After performing explicitly feature interaction learning, stacked structure models use a DNN to further capture high-order implicit feature interactions, enabling strong modeling capacity.

On the contrary, models with parallel structure leverage two parallel networks to capture explicit and implicit feature interactive signals respectively, and finally fuse the information in the output layer. Classic members in this category are DeepFM [5], AutoFIS [18], DCN [30], DCN-V2 [31], xDeepFM [17] and AutoInt [28]. In these models, implicit feature interactions are extracted via a DNN model over the embedding layer, which is highly efficient and ubiquitous. While for modeling explicit feature interactions, DeepFM and AutoFIS adopt a FM [26] structure to perform pairwise interaction learning. DCN and its extended version DCN-V2 use a cross network to apply feature crossing at each layer explicitly. Moreover, xDeepFM employs a Compressed Interaction Network (CIN) to capture more complicated feature interactions of bounded orders, and AutoInt leverages a multi-head self-attention

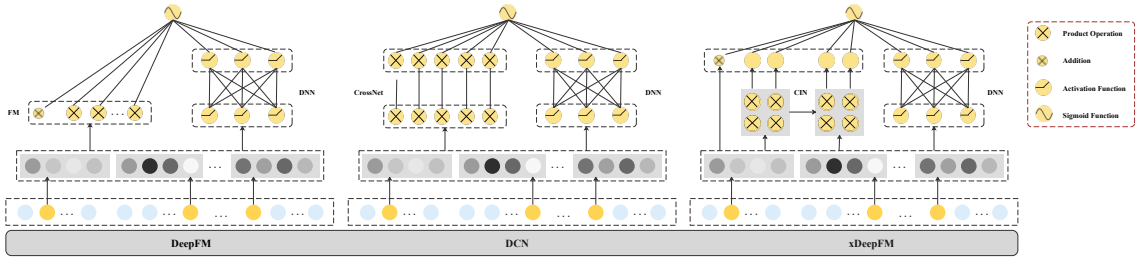


Figure 2: The architecture of three deep CTR models with parallel structure: DeepFM, DCN, xDeepFM

network [29] to model different order feature interactions explicitly. Three representative models are presented in Figure 2.

2.2 Discriminative Input

Generally speaking, most parallel-structured deep models share the unique network input (normally the embedding layer) for different sub-networks, such as DeepFM [5], DCN [30], AutoInt [28]. However, different sub-networks in parallel-structured models capture different interaction patterns while different features are intuitively suitable for different interaction functions, which is pointed out in [14]. Relevant to our consideration, SENET[10] is designed to learn feature importance by performing Squeeze, Excitation and Re-Weight steps over the original representations and FiBiNET [12] leverages the SENET for extracting informative features. GateNet [11] constructs bit-wise and vector-wise gate to select salient latent information for embedding layers and hidden layers. Similar idea is proposed in multi-task learning, such as MMOE[20], which utilizes a sparse gating network to automatically balance task specific information over the shared knowledge. Moreover, multi-view learning [19, 32] which expresses data from different views to learn unified item or user representations, is similar to discriminative input here to some extend. In our work, we design a field-wise gating network to generate discriminative feature distributions for different sub-networks in a soft-selection manner.

3 PRELIMINARY

3.1 CTR prediction

CTR prediction is a supervised binary classification task [3, 27]. Suppose a dataset for training CTR models consists of Q instances $(\mathbf{x}, y)$, where $y \in \{0, 1\}$ is the label indicating user's click behaviors and $\mathbf{x}$ is a multi-fields data record. A CTR prediction algorithm is devoted to approximate the probability $Pr(y = 1|\mathbf{x})$ for each instance with input $\mathbf{x}$.

To better predict user's behaviors under complex real-world environments, web-scale industrial recommender systems collect a large number of features, including user profiles (*e.g.*, gender, age), item attributes (*e.g.*, name, category) as well as contextual information (*e.g.*, weekday, location) to build a training dataset. For numerical features (*e.g.*, bidding price, usage count), common-used approaches are discretization, including soft discretization like AutoDis [4] and hard discretization via transforming numerical features to categorical features, such as logarithm discretization [13] and tree-based discretization [8]. Therefore, multi-field categorical record can be transformed into a high-dimensional sparse features via field-aware one-hot encoding [8]. For example, an instance (`Gender`=Male, `Age`=18, `Category`=Electronics, ..., `Weekday`=Monday) can be represented as:

$$\mathbf{x} = \underbrace{[0, 1]}_{\text{Gender}}\ \underbrace{[0, 1, 0, \ldots, 0]}_{\text{Age}}\ \underbrace{[0, 0, 1, \ldots, 0]}_{\text{Category}} \ldots \underbrace{[1, 0, 0, \ldots, 0]}_{\text{Weekday}} \tag{1}$$

For deep learning-based CTR models following feature embedding & feature interaction paradigm, an embedding layer is applied to transform each sparse vector into a low-dimensional dense vector, which is habitually called feature embedding. For the i-th categorical field, the feature embedding can be obtained by embedding look-up operation:

$$\mathbf{e}_i = \mathbf{E}_i \cdot \mathbf{x}_i, \tag{2}$$

where $\mathbf{E}_i \in \mathbb{R}^{v_i \times k}$ is the embedding matrix, $\mathbf{x}_i$ is the one-hot vector, v_i is the vocabulary size and k is embedding size. Therefore, the result of embedding layer is represented as:

$$\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_f], \tag{3}$$

where f denotes the number of fields.

Then, the feature embedding is fed into stacked or parallel structure models for capturing explicit/implicit feature interactions and the predictive CTR score is obtained via a discriminative function $\hat{y} = f_{CTR}(\mathbf{E})$.

3.2 Deep & Cross Network (DCN)

DCN is a representative parallel structure CTR model, which feeds the embedding $\mathbf{E}$ into two separate sub-network, *i.e.*, cross network and deep network for explicit and implicit feature interaction modeling. Specifically, the cross layers and deep layers in these two networks are respectively denoted as:

$$\begin{aligned} \mathbf{x}_{l+1} &= \mathbf{x}_0 \mathbf{x}_l^{\mathsf{T}} \mathbf{w}_l^c + \mathbf{b}_l^c + \mathbf{x}_l, \\ \mathbf{h}_{l+1} &= \mathtt{ReLU}(\mathbf{w}_l^{d\mathsf{T}} \mathbf{h}_l + \mathbf{b}_l^d), \end{aligned} \tag{4}$$

where $\mathbf{x}_l$ and $\mathbf{h}_l$ are the outputs of the l-th cross and deep layer, $\mathbf{w}_l^c$, $\mathbf{b}_l^c$, $\mathbf{w}_l^{\mathrm{d}}$ and $\mathbf{b}_l^d$ are the weight and bias parameters of the l-th cross and deep layer. Note that the inputs for both cross and deep network are identical, *i.e.*, $\mathbf{x}_0 = \mathbf{h}_0 = \mathbf{E}$.

Finally, the results of two networks are fused in the last layer (L-th layer) and then sent into the output layer for prediction, which can be represented as:

$$\hat{y}_{DCN} = \mathtt{Sigmoid}(\mathbf{w}^{\mathsf{T}}[\mathbf{x}_L, \mathbf{h}_L] + \mathbf{b}), \tag{5}$$

where the activation function is `Sigmoid`, $[\cdot, \cdot]$ is the concatenation operation, $\mathbf{w}$ and $\mathbf{b}$ are the weight and bias parameters, respectively.

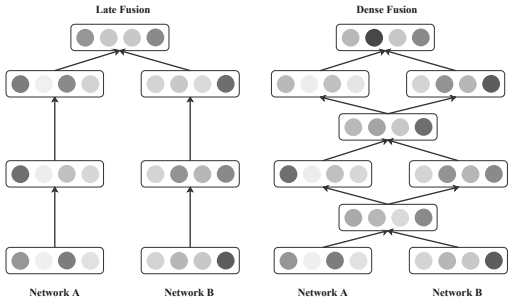


Figure 3: Two different fusion strategies.

However, trivial sharing strategy limits the expressiveness and effectiveness of DCN. On the one hand, the cross network and deep network are independent and the learned representations do not fuse until the last layer. On the other hand, both cross and deep networks take the same embedding $\mathbf{E}$ as input and ignore the emphasis on feature selection. These two defects impede the learning procedure of effective feature interactions in DCN, resulting in sub-optimal performance. To overcome these issues, we propose an Enhanced Deep & Cross Network (EDCN) with two modules, *i.e.*, bridge module and regulation module, which will be introduced in the next Section.

4 ENHANCED DEEP & CROSS NETWORK (EDCN)

In this section, we elaborate on the details of EDCN together with the training procedure. The architecture of EDCN is presented in Figure 4, including two core modules in comparison with the original DCN, *i.e.*, *bridge module* and *regulation module*.

4.1 Bridge Module

The existing parallel deep CTR models learn explicit and implicit feature interactions separately via two parallel sub-networks respectively, such as DeepFM [5], DCN [30], xDeepFM [17] and AutoInt [28]. Two networks are performed separately and independently, meaning no information fusion until the last layer, which is called *late fusion*. The late fusion strategy fails to capture the correlation between two parallel networks in the intermediate layers, weakening the interactive signals between explicit and implicit feature interaction. Moreover, the lengthy updating progress in each sub-networks may lead to skewed gradients during the backward propagation [9], thus hindering the learning procedure of both networks.

To overcome this limitation, we introduce a *dense fusion* strategy, which is implemented by our proposed *bridge module*, to capture the layer-wise interactive signals between two parallel networks. Unlike the late fusion that only performs information sharing in the last layer of sub-networks, dense fusion shares the intermediate information at each layer, taking advantage of the multi-level interactive signals and mitigating the gradient issue. Comparison between late fusion and dense fusion is demonstrated in Figure 3.

More specifically, in EDCN, a bridge module is presented after each pair of cross layer and deep layer to capture effective layer-wise interactions. Formally, suppose the outputs of the l-th cross layer and l-th deep layer are represented as $\mathbf{x}_l$ and $\mathbf{h}_l$, the bridge module

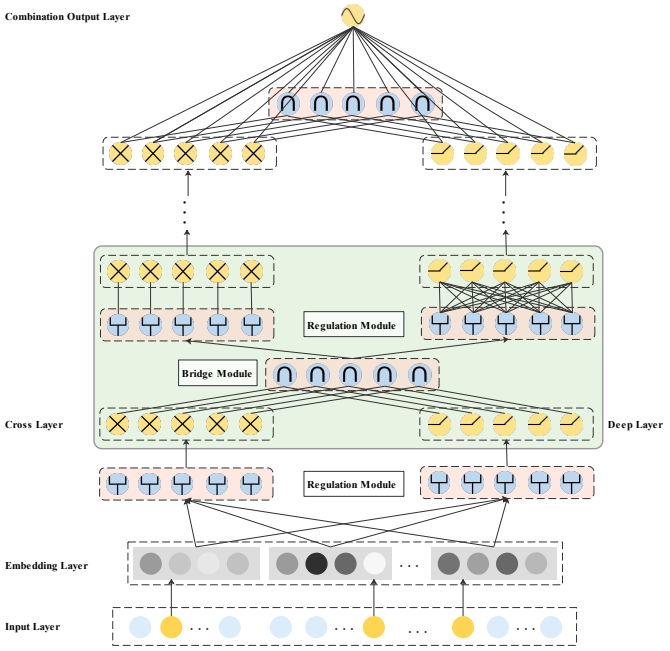


Figure 4: The architecture of EDCN.

can be formulated as $\mathbf{f}_l = f(\mathbf{x}_l, \mathbf{h}_l)$, where $f(\cdot)$ is a pre-defined interaction function that takes two vector as input and outputs a vector with the same dimension. Particularly, we empirically compare the following four interaction functions $f(\cdot) : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^d$.

- *Pointwise Addition* computes the element-wise sum of the input vectors. It has no parameters and is formulated as $\mathbf{f}_l = \mathbf{x}_l \oplus \mathbf{h}_l$.
- *Hadamard Product* takes their element-wise product and is denoted as $\mathbf{f}_l = \mathbf{x}_l \otimes \mathbf{h}_l$. It is also parameter-free.
- *Concatenation* concatenates the input vectors and passes to a feed-forward layer with $\mathtt{ReLU}$ activation function to keep the dimension of the output vector as d. This function is formulated as $\mathbf{f}_l = \mathtt{ReLU}(\mathbf{w}_l{}^\mathsf{T}[\mathbf{x}_l, \mathbf{h}_l] + \mathbf{b}_l)$, where $\mathbf{w}_l$ and $\mathbf{b}_l$ are the weight and bias parameters in bridge module of the l-th layer, respectively.
- *Attention Pooling* leverages a self-attention network [29] to measure the importance of the two input vectors and performs attentive pooling accordingly. This interaction function is denoted as $\mathbf{f}_l = a_l^x \mathbf{x}_l \oplus a_l^h \mathbf{h}_l$, where a_l^x and a_l^h are the attention weights in the l-th layer. Weight a_l^x is obtained by $\mathtt{Softmax}(\mathbf{p}_l{}^\mathsf{T}\mathtt{ReLU}(\mathbf{w}_l^\mathsf{T}\mathbf{x}_l + \mathbf{b}_l))$ where $\mathbf{p}_l$ is the transform weight parameter and a_l^h can be obtained in a similar way.

To summarize, the bridge module serves as a bridge connecting the explicit and implicit modeling networks, strengthening the interactive signals across networks and avoiding skewed gradients during backward propagation. Empirical comparisons of the above four functions are presented in the ablation study.

4.2 Regulation Module

Deep CTR models with parallel structure exploit explicit and implicit features simultaneously based on the shared embeddings. Explicit feature interactions are usually modeled with pre-defined interaction functions for efficiently exploring bounded-degree interaction (*e.g.*, cross network in DCN), while implicit feature interactions are mostly learned via fully connected layers. Intuitively, different features are suitable for different interaction functions, as observed in [14]. As a consequence, there is a need to carefully select different features for the two parallel networks, instead of feeding all the features equally to these two networks as DCN does.

Inspired by the gating mechanism used in MMoE [20], we propose a *regulation module*, implemented by a field-wise gating network to ***soft-select*** discriminative feature distributions for each parallel networks. Concretely, an efficient field-wise gating unit $\mathbf{G}^b = [\mathbf{g}_1^b, \mathbf{g}_2^b, \cdots, \mathbf{g}_f^b]$ is utilized, where $b \in B$ is for parallel network b ($B = \{cross, deep\}$ in DCN) and the real-valued scalar $\mathbf{g}_i^b$ denotes the gating weight for the i-th field. To obtain discriminative feature distributions, `Softmax` activation function is performed on $\mathbf{G}^b$ and obtained $\widehat{\mathbf{G}}^b = [\widehat{\mathbf{g}}_1^b, \widehat{\mathbf{g}}_2^b, \cdots, \widehat{\mathbf{g}}_f^b]$ by:

$$\widehat{\mathbf{g}}_i^b = \frac{e^{\frac{1}{\tau}\mathbf{g}_i^b}}{\sum_{j=1}^{f} e^{\frac{1}{\tau}\mathbf{g}_j^b}}, \tag{6}$$

where τ is the temperature coefficient to control distribution, scalar $\widehat{\mathbf{g}}_i^b$ represents the gating score of the i-th field in network b. Therefore, the regulated representation $\mathbf{E}^b$ for network b is derived as:

$$\mathbf{E}^b = \widehat{\mathbf{G}}^b \odot \mathbf{E} = [\widehat{\mathbf{g}}_1^b \mathbf{e}_1, \widehat{\mathbf{g}}_2^b \mathbf{e}_2, \cdots, \widehat{\mathbf{g}}_f^b \mathbf{e}_f]. \tag{7}$$

Besides the shared embedding layer, we also perform regulation module together with each bridge module, as shown in Figure 4. Note that the cross network in DCN is actually a linear transformation of $\mathbf{x}_0$, as demonstrated in xDeepFM [17]. In other words, *field* information still exists in the fused representation of the bridge module. Therefore, the regulation module after the bridge module works in the same principle as the one for the shared embedding layer.

4.3 Combination Output Layer

After stacking L layers, the outputs are concatenated and fed into a standard logits layer for prediction. Assume the outputs of the L-th cross layer, deep layer and bridge module are $\mathbf{x}_L$, $\mathbf{h}_L$ and $\mathbf{f}_L$, the result of EDCN is represented as:

$$\hat{y} = \texttt{Sigmoid}(\mathbf{w}^\top [\mathbf{x}_L, \mathbf{h}_L, \mathbf{f}_L] + \mathbf{b}), \tag{8}$$

where $\mathbf{w}$ and $\mathbf{b}$ are the the weight and bias parameters.

The loss function is the widely-used LogLoss with a regularization term, as:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i) + \lambda \|\Theta\|_2, \tag{9}$$

where y_i and $\hat{y}_i$ are the ground truth label and estimated value of the i-th instance, respectively. N is the total number of training instances, λ is the L_2 regularization weight and Θ is the set of model parameters.

4.4 Discussion

4.4.1 Complexity Analysis. In this subsection, we analyze the time and space complexity of the bridge module and regulation module. Denote the embedding size as k and we take Hadamard Product as the interaction function in the bridge module as it achieves the best performance in the experiments which will be shown in Section 5.5.1. The time and space complexity of a single bridge module is $O(k)$ and $O(1)$ respectively because of the parameter-free element-wise product. Moreover, the time complexity for a single regulation module is also $O(k)$ due to the multiplication operation in Eq.(7); while its space complexity is $O(1)$ as the parameters in $\mathbf{G}^b$ are field-wise. The above analysis indicates that these two modules are lightweight and the empirical study about model efficiency will be elaborated in Section 5.6.

4.4.2 Compatibility Analysis. EDCN proposes two core modules, namely bridge module and regulation module, which can be applied seamlessly to mainstream models with parallel structure, such as DeepFM, DCN, xDeepFM, AutoInt and *etc.* The bridge module captures the layer-wise interaction between different parallel networks, strengthening the interactive signals across networks; while the regulation module can discriminate feature distributions for different parallel networks. These two modules are generalized well to the parallel-structured models, which is demonstrated by the compatibility study elaborated in Section 5.3. Specifically, to cope with the setting that implicit and explicit networks have different layers, we leverage the last layer output of the sub-network with fewer layers to perform bridge operations with another one repetitively.

5 OFFLINE EXPERIMENTS

5.1 Offline Experimental Setting

5.1.1 Dataset and Evaluation Protocols. To evaluate the effectiveness of our proposed EDCN and the compatibility of two core modules, we conduct extensive offline experiments on two popular benchmarks (*i.e.*, *Avazu*[1], *Criteo*[2]) and one industrial dataset. The statistics of all the three datasets are summarized in Table 1.

- Avazu dataset contains 23 fields spanning from user/device features to ad attributes. We randomly split the dataset into training, validation and test with ratio 8:1:1.
- Criteo dataset consists of 26 categorical fields and 13 numerical fields, where day 1-7 are used for training, day 8 and day 9 for validation and test, respectively. We follow the same data processing procedure in PNN [23] by performing negative down-sampling to keep the positive ratio close to 50% and converting numerical features into categorical ones.
- The industrial dataset contains 9 consecutive days of click logs sampled from the Huawei advertising platform. The feature set of this dataset is comprised of 44 categorical features and 41 numerical features, which are discretized via a variety of hybrid manually-designed rules. We set day 1-7 as training set, day 8 and day 9 as validation and test set respectively.

To evaluate the performance, we leverage the most commonly-used offline evaluation metrics in CTR prediction, namely **AUC**

[1] http://www.kaggle.com/c/avazu-ctr-prediction
[2] http://labs.criteo.com/downloads/download-terabyte-click-logs/

Table 1: Statistics of evaluation datasets.

Dataset	#Feature Fields	#Instances ($\times 10^7$)	Positive Ratio
Avazu	23	3.64	17%
Criteo	39	9.69	50%
Industrial	85	8.75	3.3%

Table 2: The overall performance comparison. Boldface denotes the highest score and underline indicates the best result of the baselines. $\star$ represents significance level p-value < 0.05 of comparing EDCN with the best baseline.

Model	Avazu		Criteo		Industrial	
	AUC	LogLoss	AUC	LogLoss	AUC	LogLoss
FNN	0.7738	0.3841	0.7941	0.5482	0.7261	0.1368
Wide&Deep	0.7745	0.3836	0.7952	0.5470	0.7255	0.1369
DeepFM	0.7747	0.3833	0.7955	0.5467	0.7262	0.1369
DCN	0.7751	0.3835	0.7963	0.5459	0.7263	0.1368
DCN-V2	0.7755	0.3822	0.7983	0.5435	0.7272	0.1368
xDeepFM	0.7754	0.3825	0.7968	0.5455	0.7275	0.1367
AutoInt	0.7756	0.3821	0.7983	0.5437	<u>0.7282</u>	<u>0.1365</u>
PNN	<u>0.7759</u>	<u>0.3820</u>	<u>0.7985</u>	<u>0.5434</u>	0.7269	0.1366
EDCN	**0.7793***	**0.3803***	**0.8001***	**0.5415***	**0.7310***	**0.1361***
Rel Impr.	0.44%	0.45%	0.20%	0.35%	0.38%	0.29%

and **LogLoss**. It is noticeable that a slightly higher AUC (↑) or lower Logloss (↓) at ***0.001****-level* is regarded significant for the CTR prediction task, which has been pointed out in existing works [2, 5, 28, 30]. All the experiments are repeated 5 times to get the average performance. The two-tailed unpaired t-test is performed to detect a significant difference between EDCN and the best baseline.

5.1.2 Baselines and Implementation Details. To demonstrate the effectiveness of EDCN, we compare the performance with representative deep CTR models, including FNN [33], Wide & Deep [2], DeepFM [5], DCN [30], DCN-V2 [31], xDeepFM [17], AutoInt [28] and PNN [24].

All the models are implemented on TensorFlow, and we optimize all the models with mini-batch Adam [15], where the learning rate is searched from $[10^{-5}, 10^{-4}, \ldots, 10^{-2}]$ and the batch size is fixed at 2,000. Besides, the embedding size is set to 40. The hidden layers of deep network are fix to 400-400-400 by default and Batch Normalization is applied. The weight of L_2 regularization [16] is tuned in $[10^{-5}, 10^{-4}, \ldots, 10^{-3}]$ and dropout rate is searched from $[0.1, 0.2, \ldots, 0.9]$. Specifically, the network structure for modeling explicit feature interactions in DCN, DCN-V2 and xDeepFM (namely, CrossNet and CIN) are set to 3 layers. The head and attention factor of the multi-head attention in AutoInt is set to 2 and 20, respectively. By default, Hadamard Product is chosen as interaction function in bridge module. $\widetilde{\mathbf{g}}_f^b$ in regulation module is initialized as 1.0 to ensure equal weight for each field at start.

5.2 Performance Comparison

Table 2 presents the model performance on all the three datasets, from which we have the following observations:

- EDCN outperforms all the SOTA baselines over three datasets in terms of both AUC and LogLoss by a significant margin, which demonstrates the superior performance of EDCN in CTR prediction task.
- In comparison with the original DCN model, EDCN improves AUC on the three datasets by 0.54%, 0.48% and 0.65%, respectively. We argue that this significant improvement attributes to the following reasons: (1) Bridge module based on the dense fusion strategy contributes to strengthening the layer-wise interaction and supervision across the parallel networks. Though the network width of the deep layers is extended to be the same as the embedding layer, the improved performance is not achieved by the additional hidden neurons (as the average AUC over three datasets achieved by the extended-width DCN improves only less than 0.03%). (2) Regulation module regulates the shared embeddings and fused information to pass discriminative feature distributions into different sub-networks, which helps sub-networks to soft-select suitable features.

Table 3: Compatibility study of bridge module over parallel CTR models with multi-layers.

Model	Avazu		Criteo		Industrial	
	AUC	LogLoss	AUC	LogLoss	AUC	LogLoss
xDeepFM	0.7754	0.3825	0.7968	0.5455	0.7275	0.1367
$\text{xDeepFM}_{\text{Bridge}}$	0.7761	0.3824	0.7974	0.5445	0.7291	0.1364
DCN	0.7751	0.3835	0.7963	0.5459	0.7263	0.1368
$\text{DCN}_{\text{Bridge}}$	0.7775	0.3814	0.7997	0.5420	0.7300	0.1364
DCN-V2	0.7755	0.3822	0.7983	0.5435	0.7272	0.1368
$\text{DCN-V2}_{\text{Bridge}}$	0.7776	0.3811	0.7987	0.5431	0.7292	0.1364

5.3 Compatibility Analysis with Different Models

5.3.1 Compatibility Study of Bridge Module. To demonstrate the compatibility of our proposed bridge module, we introduce bridge module for each hidden layer in three popular deep parallel CTR models, namely xDeepFM, DCN and DCN-V2. A model $\mathcal{M}$ equipped with bridge module is represented as $\mathcal{M}_{\text{Bridge}}$. From Table 3, we can observe that bridge module improves the performance of deep parallel CTR models consistently. The average improvements over AUC metric are 0.13% for xDeepFM, 0.42% for DCN and 0.20% for DCN-V2 respectively, which demonstrates the effectiveness of the bridge module. The reason lies in that layer-wise bridge module leverages the dense fusion strategy to capture the interactive correlation signals and bring benefit to explicit and implicit feature modeling subsequently. Moreover, the gradient update is more balanced when backward propagation due to the introduction of multi-paths between different sub-networks, so that the gradient skew problem can be well eased.

5.3.2 Compatibility Study of Regulation Module. The proposed regulation module is also model-agnostic. In this section, we conduct extensive experiments to demonstrate its compatibility by applying regulation module to five state-of-the-art parallel CTR models: DeepFM, xDeepFM, AutoInt, DCN, DCN-V2, together with $\text{DCN}_{\text{Bridge}}$. All these models except $\text{DCN}_{\text{Bridge}}$ apply late fusion strategy, so that we only perform regulation module over the embedding layer. A model $\mathcal{M}$ equipped with regulation module is

Table 4: Compatibility study of regulation module over parallel CTR models.

Model	Avazu		Criteo		Industrial	
	AUC	LogLoss	AUC	LogLoss	AUC	LogLoss
DeepFM	0.7747	0.3833	0.7955	0.5467	0.7262	0.1369
$\text{DeepFM}_{\text{Regulation}}$	0.7756	0.3831	0.7968	0.5452	0.7280	0.1365
xDeepFM	0.7754	0.3825	0.7968	0.5455	0.7275	0.1367
$\text{xDeepFM}_{\text{Regulation}}$	0.7765	0.3817	0.7977	0.5443	0.7288	0.1365
AutoInt	0.7756	0.3821	0.7983	0.5437	0.7282	0.1365
$\text{AutoInt}_{\text{Regulation}}$	0.7763	0.3818	0.7986	0.5432	0.7294	0.1365
DCN	0.7751	0.3835	0.7963	0.5459	0.7263	0.1368
$\text{DCN}_{\text{Regulation}}$	0.7771	0.3817	0.7985	0.5433	0.7286	0.1365
DCN-V2	0.7755	0.3822	0.7983	0.5435	0.7272	0.1368
$\text{DCN-V2}_{\text{Regulation}}$	0.7761	0.3819	0.7985	0.5433	0.7289	0.1365
$\text{DCN}_{\text{Bridge}}$	0.7775	0.3814	0.7997	0.5420	0.7300	0.1364
$\text{DCN}_{\text{BridgeRegulate}}$	0.7793	0.3803	0.8001	0.5415	0.7310	0.1361

Note that model $\text{DCN}_{\text{BridgeRegulate}}$ = EDCN.

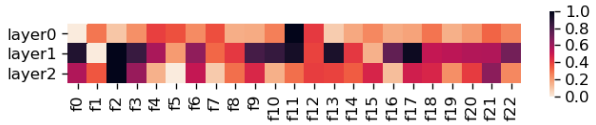
represented as $\mathcal{M}_{\text{Regulate}}$. Note that $\text{DCN}_{\text{BridgeRegulate}}$ is actually our proposed EDCN. Consistent improvement can be observed from Table 4. As stated earlier, regulation module discriminates feature distributions and passes them into different sub-networks where distinctive feature collections will be exploited respectively. From the perspective of back propagation, due to the shared embeddings in paralle-structured models, gradients from multiple sub-networks may conflict with each other to some extent. With regulation module, gradients are regulated before back-propagating to the embedding layer, alleviating the conflict of gradients.

5.4 Analyzing Regulation Module

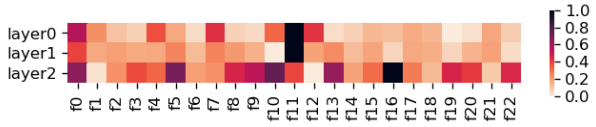
To vividly illustrate the feature regulation results of the regulation module, we visualize the weight distribution $\widehat{\mathrm{G}}^b$ (with min-max normalization to scale each element $\widehat{\mathrm{g}}_i^b$ into [0,1] for intuitive presentation) for each hidden layer of parallel networks in EDCN. The results over the Avazu dataset are shown in Figure 5. Note that darker color means more inclination of this field towards the corresponding feature interaction manner (*i.e.*, cross network and deep network in EDCN). From the heatmaps we can observe that feature distribution varies across different layers and different feature interaction manners. Specifically, `f11` has large weight in `layer0` and layer 1 for both cross and deep networks. `f2` is more discriminative in `layer2` of cross network while `f16` in `layer2` of deep network. Besides, heatmap in the cross network is more diverse, indicating that some fields are playing a more significant role than others in bounded-degree explicit feature interaction. On the contrary, most fields make a relatively similar contribution to the implicit feature interactions. Another observation is that as the number of layers increases, the preference of different sub-networks is more obvious. The feature distributions are relatively close in `layer0` while significantly different in `layer2`. Therefore, EDCN utilizes the regulation module to soft-select discriminative feature distributions, taking full advantage of the different features.

5.5 Ablation Study

5.5.1 Bridge Module. To compare the performance of different interaction functions in the bridge module, we explore four interaction functions $f(\cdot)$, namely pointwise addition (EDCN-ADD), hadamard product (EDCN-PROD), concatenation (EDCN-CON) and attention pooling (EDCN-ATT). As observed from Figure 6, hadamard product outperforms the others significantly, which may due to the following reasons. On the one hand, hadamard product is parameter-free and therefore does not involve any additional learnable parameters, which is easier to train steadily than the methods with parameters (*e.g.*, concatenation and attention pooling). On the other hand, compared with pointwise addition, product operation is a better interaction modeling operation in recommendation models, as indicated in [5, 24, 26].

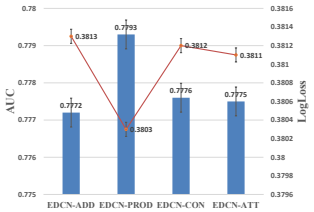


(a) Heatmap of $\widehat{\mathrm{G}}^{cross}$ *w.r.t* features in the cross network.

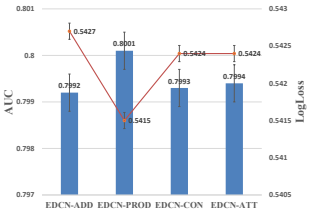


(b) Heatmap of $\widehat{\mathrm{G}}^{deep}$ *w.r.t* features in the deep network.

Figure 5: Heatmap of features in each layer of EDCN over Avazu dataset.



(a) AUC



(b) LogLoss

Figure 6: Ablation study on the bridge module.

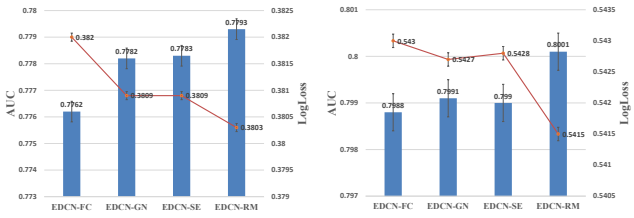
5.5.2 Regulation Module. To demonstrate the effectiveness of our proposed regulation module, we conduct experiments by replacing the regulation module (**RM**) with three methods.

FC. Fully connection is a commonly-used transformation method, extracting representation from the preceding layer.

SE. Squeeze-and-Excitation Network [10, 12] performs *Squeeze*, *Excitation* and *Re-Weight* steps to re-scale representation with informative features.

GN. GateNet [11] proposes feature embedding gate to select salient information from the feature-level.

Comparisons among these methods are presented in Figure 7. We can observe that simple FC obtains the worst results. Besides, GN and SE methods achieve improvement compared with the non-regulation (namely, $\text{DCN}_{\text{Bridge}}$) on Avazu while slight decrease on Criteo. Nevertheless, RM obtains significant improvement over the other competitors consistently, demonstrating the effectiveness of our proposed regulation module.



(a) AUC (b) LogLoss

Figure 7: Ablation study on the regulation module.

Table 5: Time and space complexity comparison on the Avazu dataset.

Model	Parameters ($\times 10^6$)	Relative ratio	Inference time (s)	Relative ratio
DCN	~ 9.17	-	~ 8.48	-
xDeepFM	~ 9.66	+5.3%	~ 44.18	+421.0%
AutoInt	~ 34.55	+276.8%	~ 69.89	+724.2%
EDCN	~ 11.03	+20.3%	~ 12.66	+49.3%

5.6 Model Complexity

In order to quantitatively analyze the space and time complexity of our proposed EDCN, we compare the model parameters and inference time (over the whole test set) of three representative deep CTR models with parallel structure. All the experiments are conducted on an NVIDIA Tesla P100-PCIE GPU with 16G memory. Table 5 reports the comparison results on Avazu dataset. We can observe that, compared with DCN, the increased model parameters and inference time of EDCN are acceptable, demonstrating that two modules introduced by EDCN are lightweight and feasible in practical industrial applications. Besides, the inference time of xDeepFM is much longer than that of EDCN while the increased model parameters of EDCN are much smaller than that of AutoInt.

6 ONLINE A/B TESTING

6.1 System Overview

In this section, we briefly describe the CTR prediction system, which consists of two core modules: *Online Inference* and *Offline Training*. The system architecture is depicted in Figure 8. The online inference module serves recommended ad lists to users, which is time-sensitive and must be done in several milliseconds. When a user arrives, a query with the user's attributes and contextual features is generated and send to the online service. The ad *Retrieval* stage is triggered and returns dozens of candidate ads that are popular or relevant to the user from a candidate pool with millions of ads. Then, an *Indexer* extracts the features of the target user, candidate ads as well as the context to splice and construct instances. Finally, a *Ranker* leverages the instances and a deep CTR model trained offline to predict scores $Pr(y|\mathbf{x})$ and a sorted ads list is organized according to some ranking function before presenting to the user. The offline training module periodically trains and updates the CTR models according to the newly generated user behavior logs. User's behaviors (with user consent) over the recommended ads list are recorded in the *Logs* and the data after filtering and cleaning are fed into *DataGen* for supporting online inference and offline training. During the offline training, the data after processing are fed into a

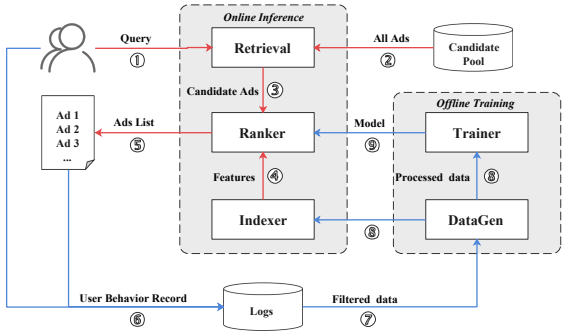


Figure 8: Overview of the CTR prediction system.

Trainer, which utilizes the historical data to train deep CTR models and push for online serving periodically.

6.2 Online Experimental Setting

The online A/B test is conducted for a month from March 10th to April 10th. The compared baseline is denoted as $\mathcal{M}_{base}$, which is a highly-optimized parallel-structured model. We deploy the bridge module and regulation module based on $\mathcal{M}_{base}$, named as $\mathcal{M}_{enhance}$. Both models are trained over the latest click logs, where an identical data process procedure is performed. Each model is deployed in a single cluster, where each node contains 16 core Xeon(R) Gold 6278C CPU (2.60GHz), 32GB RAM as well as 1 NVIDIA TESLA T4 GPU cards. For online serving, 5% of the users are randomly selected as the experimental group and are recommended ads by $\mathcal{M}_{enhance}$ while another 5% of the users are in the control group and receive results from $\mathcal{M}_{base}$.

We choose two kinds of ad display scenarios in Huawei to evaluate our proposed bridge module and regulation module, where millions of daily active users interact with ads and tens of millions of user log events are generated.

- `Instant Access` scene is comprised of an *ad list*, which is at the top of the minus one screen and recommends a list of apps to users.
- `Video Page` scene is displayed by a *single ad card*, which is located below the Huawei Video.

Two commonly-used online evaluation metrics in online advertising, *i.e.*, CTR and eCPM, are used to evaluate the performance of different deployed models.

6.3 Online Results

The online result of consecutive 30 days shows significant improvement of our proposed modules over the baseline $\mathcal{M}_{base}$ whose results are shown in Table 6. We can observe that, $\mathcal{M}_{enhance}$ achieves 7.30% (2.42%) and 4.85% (1.71%) improvements with respect to CTR and eCPM respectively in the `Instant Access` (`Video Page`) scenario. Besides, the serving latency increased is also acceptable in the industry. The results show that the bridge module and regulation module bring significant performance improvement in terms of user experience and platform revenue with slight latency overload.

7 CONCLUSION

In this paper, we delve into the issue of trivial information sharing in parallel structure models (*e.g.*, DCN) and propose EDCN

Table 6: Online A/B testing results of $\mathcal{M}_{enhance}$ compared with the base model $\mathcal{M}_{base}$.

Scenario	CTR	eCPM	Latency
Instant Access	+7.30%	+4.85%	+10.8%
Video Page	+2.42%	+1.71%	+9.2%

for enhancing information sharing. EDCN performs a dense fusion strategy in the hidden layers and designs a bridge module to capture the layer-wise interactive signals between the deep and cross networks. Besides, a regulation module is proposed to learn discriminative feature distributions for different networks. Furthermore, the two proposed modules in EDCN are generalized well to mainstream CTR models with parallel structure. We conduct extensive experiments on two benchmark real-world datasets and an industrial dataset to demonstrate the effectiveness and compatibility of EDCN. Besides, a one-month online A/B test in the Huawei advertising platform shows that two modules improve the base model by 7.30% and 4.85% in terms of CTR and eCPM. Future work includes proposing some information sharing strategy into models with stacked structure.

REFERENCES

[1] Mathieu Blondel, Akinori Fujino, Naonori Ueda, and Masakazu Ishihata. 2016. Higher-order factorization machines. In *Proceedings of the 30th Conference on Neural Information Processing Systems*.

[2] 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*. 7–10.

[3] Thore Graepel, Joaquin Quinonero Candela, Thomas Borchert, and Ralf Herbrich. 2010. Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*. 13–20.

[4] Huifeng Guo, Bo Chen, Ruiming Tang, Weinan Zhang, Zhenguo Li, and Xiuqiang He. 2021. An Embedding Learning Framework for Numerical Features in CTR Prediction. In *Proceedings of the 27th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

[5] 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*.

[6] Wei Guo, Ruiming Tang, Huifeng Guo, Jianhua Han, Wen Yang, and Yuzhou Zhang. 2019. Order-aware embedding neural network for CTR prediction. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1121–1124.

[7] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. 355–364.

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

[9] Di Hu, Chengze Wang, Feiping Nie, and Xuelong Li. 2019. Dense multimodal fusion for hierarchically joint representation. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 3941–3945.

[10] Jie Hu, Li Shen, and Gang Sun. 2018. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7132–7141.

[11] Tongwen Huang, Qingyun She, Zhiqiang Wang, and Junlin Zhang. 2020. GateNet: Gating-Enhanced Deep Network for Click-Through Rate Prediction. *arXiv preprint arXiv:2007.03519* (2020).

[12] Tongwen Huang, Zhiqi Zhang, and Junlin Zhang. 2019. FiBiNET: combining feature importance and bilinear feature interaction for click-through rate prediction. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 169–177.

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

[14] Farhan Khawar, Xu Hang, Ruiming Tang, Bin Liu, Zhenguo Li, and Xiuqiang He. 2020. AutoFeature: Searching for Feature Interactions and Their Architectures for Click-through Rate Prediction. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 625–634.

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

[16] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[17] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1754–1763.

[18] Bin Liu, Chenxu Zhu, Guilin Li, Weinan Zhang, Jincai Lai, Ruiming Tang, Xiuqiang He, Zhenguo Li, and Yong Yu. 2020. Autofis: Automatic feature interaction selection in factorization models for click-through rate prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2636–2645.

[19] Chun-Ta Lu, Lifang He, Hao Ding, Bokai Cao, and Philip S Yu. 2018. Learning from multi-view multi-way data via structural factorization machines. In *Proceedings of the 2018 World Wide Web Conference*. 1593–1602.

[20] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. 2018. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1930–1939.

[21] H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, et al. 2013. Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1222–1230.

[22] Claudia Perlich, Brian Dalessandro, Rod Hook, Ori Stitelman, Troy Raeder, and Foster Provost. 2012. Bid optimizing and inventory scoring in targeted online advertising. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. 804–812.

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

[24] 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.

[25] Kan Ren, Weinan Zhang, Yifei Rong, Haifeng Zhang, Yong Yu, and Jun Wang. 2016. User response learning for directly optimizing campaign performance in display advertising. In *Proceedings of the 25th acm international on conference on information and knowledge management*. 679–688.

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

[27] Matthew Richardson, Ewa Dominowska, and Robert Ragno. 2007. Predicting clicks: estimating the click-through rate for new ads. In *Proceedings of the 16th international conference on World Wide Web*. 521–530.

[28] 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 *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 1161–1170.

[29] 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*. 6000–6010.

[30] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*. 1–7.

[31] Ruoxi Wang, Rakesh Shivanna, Derek Z Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed H Chi. 2020. DCN-M: Improved Deep & Cross Network for Feature Cross Learning in Web-scale Learning to Rank Systems. *arXiv preprint arXiv:2008.13535* (2020).

[32] C. Wu, F. Wu, M. An, J. Huang, Y. Huang, and X. Xie. 2019. Neural News Recommendation with Attentive Multi-View Learning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*.

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

[34] Weinan Zhang, Jiarui Qin, Wei Guo, Ruiming Tang, and Xiuqiang He. 2021. Deep Learning for Click-Through Rate Estimation. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence*.

[35] Weinan Zhang, Shuai Yuan, and Jun Wang. 2014. Optimal real-time bidding for display advertising. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1077–1086.

[36] 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 AAAI conference on artificial intelligence*, Vol. 33. 5941–5948.

[37] 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*. 1059–1068.