Towards Deeper, Lighter and Interpretable Cross Network for CTR Prediction

Fangye Wang* School of Computer Science Fudan University, Shanghai, China fywang18@fudan.edu.cn

Tun Lu*†

School of Computer Science Fudan University, Shanghai, China lutun@fudan.edu.cn Hansu Gu Seattle, United States hansug@acm.org

Peng Zhang* School of Computer Science Fudan University, Shanghai, China zhangpeng_@fudan.edu.cn Dongsheng Li Microsoft Research Asia Shanghai, China dongsli@microsoft.com

> Ning Gu* l of Computer Science

School of Computer Science Fudan University, Shanghai, China ninggu@fudan.edu.cn

ABSTRACT

Click Through Rate (CTR) prediction plays an essential role in recommender systems and online advertising. It is crucial to effectively model feature interactions to improve the prediction performance of CTR models. However, existing methods face three significant challenges. First, while most methods can automatically capture high-order feature interactions, their performance tends to diminish as the order of feature interactions increases. Second, existing methods lack the ability to provide convincing interpretations of the prediction results, especially for high-order feature interactions, which limits the trustworthiness of their predictions. Third, many methods suffer from the presence of redundant parameters, particularly in the embedding layer. This paper proposes a novel method called Gated Deep Cross Network (GDCN) and a Field-level Dimension Optimization (FDO) approach to address these challenges. As the core structure of GDCN, Gated Cross Network (GCN) captures explicit high-order feature interactions and dynamically filters important interactions with an information gate in each order. Additionally, we use the FDO approach to learn condensed dimensions for each field based on their importance. Comprehensive experiments on five datasets demonstrate the effectiveness, superiority and interpretability of GDCN. Moreover, we verify the effectiveness of FDO in learning various dimensions and reducing model parameters. The code is available on https://github.com/anonctr/GDCN.

CCS CONCEPTS

• Information systems → Recommender systems.

KEYWORDS

Cross Network, Information Gate, Feature Crossing, CTR Prediction

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

CIKM '23, October 21–25, 2023, Birmingham, United Kingdom.

© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM. ACM ISBN 979-8-4007-0124-5/23/10...\$15.00 https://doi.org/10.1145/3583780.3615089

ACM Reference Format:

Fangye Wang, Hansu Gu, Dongsheng Li, Tun Lu, Peng Zhang, and Ning Gu. 2023. Towards Deeper, Lighter and Interpretable Cross Network for CTR Prediction. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23), October 21–25, 2023, Birmingham, United Kingdom.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3583780.3615089

1 INTRODUCTION

Click-through Rate (CTR) prediction is an important component of recommender systems and online advertising [9, 12]. It aims to estimate the probability of a user clicking on a recommended item or an advertisement on a web page. Accurate CTR prediction can bring significant revenue gains and also improved user satisfaction [9, 57, 72]. Most methods typically consist of three layers [53, 67, 74]: feature embedding, feature interaction, and prediction. To improve the accuracy of CTR prediction, many methods have been proposed that focus on designing effective feature interaction architectures. However, previous works such as Logistic Regression (LR)[44], and FM-based methods [5, 24, 43] can only model low- or fixed-order feature interactions. As web-scale recommendation systems have become more complex, there is a growing demand for methods to capture high-order feature interactions. Therefore, more recent methods [10, 17, 30, 47, 55, 56, 62] enable joint modeling of both explicit and implicit high-order feature interactions and achieve significant performance improvements. While these methods have made great progress, they still have three major limitations.

First, the effectiveness of these methods tends to decrease as the order of feature interactions increases. In general, the maximum degree of interactions that can be captured is determined by the depth of feature interactions. As the interaction layers go deeper, the number of interactions increases exponentially, which enables the model to generate more high-order interactions. However, not all interactions are helpful, which also brings in many unnecessary interactions, resulting in decreased performance and increased computational complexity. Many existing state-of-the-art (SOTA) works [5, 10, 12, 16, 19, 28, 30, 42, 47, 55, 56, 71] have confirmed through hyper-parameter analysis that their performance deteriorates when the interaction order exceeds a certain depth, usually three orders. Therefore, it is crucial to make improvements and ensure that the high-order interactions have a positive impact rather than introducing more noise and lead to sub-optimal performance.

 $^{^*}$ Also with Shanghai Key Laboratory of Data Science, Shanghai Institute of Intelligent Electronics & Systems, China..

[†]Corresponding author.

Second, the lack of interpretability in existing methods limits the trustworthiness of their predictions and recommendations. Most methods [16, 19, 42, 45, 66] suffer from low interpretability due to the implicit feature interactions through DNNs or the assignment of equal weights to all feature interactions [8, 47]. Although a few methods [8, 29, 47] have attempted to provide explanations through attention scores learned by the Self-attention mechanism [51], this approach tends to fuse all the features information, making it difficult to distinguish which interactions are essential, especially for high-order crosses. Hence, it is vital to develop methods that can provide a persuasive interpretation from both the model and instance perspectives, enabling more reliable and trustworthy results.

Third, most existing models contain massive redundant parameters, particularly in the embedding layer. Many methods [8, 30, 35, 43, 47, 60, 71] rely on feature-wise interaction structures, which assume equal embedding dimensions for all fields. However, some fields only require a relatively short dimension considering their information capacity. Consequently, these models produce massive redundant parameters in the embedding layer. But directly reducing the embedding dimension leads to a decrease in model performance [19, 20, 47, 49]. Meanwhile, most methods [50, 56, 60, 65] only focus on reducing non-embedding parameters, and the impact on overall parameter reduction is not significant compared to the embedding parameters. Although DCN [55] and DCN-V2 [56] assign varying dimensions to each field using a rule-of-thumb formula which computes dimension only based on feature numbers, they overlook the importance of each field and often fail to reduce model parameters. Hence, we aim to assign field-specific and condensed dimensions to each field, considering their inherent importance and effectively reducing the embedding parameters.

This paper presents a model called Gated Deep Cross Network (GDCN) and a approach called Field-level Dimension Optimization (FDO) to address the above-mentioned limitations. Building upon the elegant and efficient design of DCN-V2 [56], GDCN further offers improved performance in both low-order and high-order interactions, and also shows great interpretability at both model and instance perspectives. GDCN models explicit feature interactions through a proposed Gated Cross Network (GCN) and then integrates with a DNN to learn implicit feature interactions. GCN consists of two core components: feature crossing and information gate. The feature crossing component captures explicit interactions within a bounded degree, while the information gate selectively amplifies important cross features with high importance and mitigates the influence of unimportant ones at each cross order. Additionally, considering their respective importance, the FDO approach can allocate condensed and independent dimensions to each field.

The core contributions of this work are summarized as follows:

- We introduce a novel method GDCN to learn both explicit and implicit feature interactions through GCN and DNN. GCN designs an information gate to dynamically filter the next-order cross features and effectively control the information flow. Compared to existing methods, GDCN demonstrates improved performance and stability in capturing deeper high-order interactions.
- We develop the FDO approach to assign condensed dimensions to each field, considering their inherent importance. By employing FDO, GCN achieves comparable performance with only 23%

- of the original model parameters and outperforms existing SOTA models with smaller model size and faster training speed.
- Comprehensive experiments show great effectiveness and generalization of GDCN on five datasets. Moreover, our methods provide remarkable interpretability at the model and instance levels, enhancing our understanding of the model predictions.

2 RELATED WORK

2.1 CTR Prediction

Modeling informative feature interactions has been widely studied in the field to improve the performance of CTR models. Traditional methods such as LR [44], and FM-based methods [5, 24, 43] model low-order feature interactions. Recent deep learning-based methods have made significant progress by capturing high-order feature interactions. These works can be divided into two categories: stacked and parallel models, based on how they integrate explicit and implicit interaction networks [6, 53].

Stacked Structure. Stacked models first employ an interaction network to capture explicit interactions on top of the embedding layer and then use a network such as DNN to further model implicit interactions. Representative explicit structures include inner product and outer product (e.g., PNN [42], ONN [61]), Hadamard product (e.g., FM [43], FFM [39]), cross network (e.g., CN [55], CN-V2 [56], XCrossNet [63]), Bi-Interaction (e.g., NFM [19]), attention operation (e.g., AFM [59], AutoInt [47], DCAP [8], and DIEN [72]). After capturing explicit interactions, DNN is used to model deeper implicit interactions based on the explicit interactions output.

Parallel Structure. Parallel models jointly capture explicit and implicit interaction information with two parallel networks. Representative parallel models include WDL [9], DeepFM [16], DCN [55], DCN-V2 [56], xDeepFM [30] and AFN+ [10]. Among these models, DNN is the most commonly used and efficient network to capture implicit interactions over the embedding layer. The main difference among these models lies in how they model explicit interactions. WDL adopts LR as the wide part to enhance memorization ability. DeepFM utilizes FM [43] operation to capture pair-wise interactions adaptively. DCN and DCN-V2 propose two kinds of cross networks(i.e., CN and CN-V2) to extract bounded-degree feature interactions automatically. xDeepFM designs a CIN structure to capture complex feature interactions of bounded orders. Some other models jointly train three or more parallel networks to achieve better performance, e.g., FED [70], NON [33], MaskNet [57].

Despite these models can capture high-order feature crosses, they experience a decline in performance as the cross layers go higher. And they lack interpretability in identifying important crosses at both model and instance levels. Moreover, most models assign equal dimensions for all fields, leading to massive redundancy parameters. This paper aims to address these issues with our proposed methods.

2.2 Gating Mechanism in CTR Prediction

The gating mechanism has been widely adopted in various well-known methods, such as LSTM [21], GRU [11], MMoE [34]. Generally, gates enable the selection of essential features or control information flow by assigning different levels of importance to different features or sub-networks. Specifically, Multi-gate Mixture-of-Experts (MMoE) [34] adopts several gating networks to weight the

importance of several task-specific objectives. The idea of MMoE is applied in DCN-V2 [56] and DynInt [60] to reduce non-embedding parameters for better cost-efficiency through a mixture of experts or weight matrix decomposition. IFM [64], DIFM [32], and FiBiNet [23] propose different weight learning networks to recalibrate the importance of feature embeddings. Other works [13, 22, 52, 57, 58] propose different structures to select salient information from both feature embedding and intermediate representations in bit-level. In this paper, GDCN designs the information gate to identify important cross features in each cross layer, particularly for higher-order cross features. This enables GDCN to mitigate the noise introduced by exponential high-order interactions and provides dynamic interpretability to identify critical interactions for each instance.

3 PROPOSED ARCHITECTURE

Inspired by DCN-V2 [56], we develop GDCN, which consists of an embedding layer, gated cross network (GCN) and deep network (DNN). The embedding layer transforms the high-dimensional sparse input into low-dimensional dense representations. The GCN is designed to capture explicit feature interactions, with an information gate to identify the important cross features. Then, a DNN is integrated to model implicit feature crosses. In essence, GDCN is a generalization of DCN-V2, inheriting the excellent expressiveness of DCN-V2 with a simple and elegant formula for easy deployment. However, GDCN introduces a key difference by incorporating information gates, which adaptively filter the cross features in each order instead of uniformly aggregating all features. This enables GDCN to truly utilize deeper high-order cross-information without experiencing performance degradation and empowers GDCN with dynamic interpretability for each instance. The architecture of GDCN is depicted in Figure 1, showing two structures that combine the GCN and DNN networks: (a) GDCN-S and (b) GDCN-P.

3.1 Embedding Layer

In the CTR prediction task, the input features (e.g., categorical and numerical features) are typically high-dimensional and sparse [36, 47, 68]. Input instances are usually multi-field tabular data records [18, 54, 68], which contain F different fields and T features. Each instance is represented by a field-aware one-hot vector [36, 47, 53]. The embedding layer transforms the sparse high-dimensional features into a dense low-dimensional embedding matrix $E = [e_1; ...; e_F]$. Most CTR models [16, 23, 36, 38, 42, 47] require embedding dimension to be the same to accommodate specific interaction operations. However, GDCN allows arbitrary embedding dimensions, and the output of the embedding layer is represented by the concatenated vector $\mathbf{c_0} = [\mathbf{e_1} \parallel ... \parallel \mathbf{e_F}] \in \mathbb{R}^D$.

3.2 Gated Cross Network (GCN)

As the core structure of GDCN, the GCN aims to model explicit bounded-degree feature crosses with information gate. The $(l+1)^{th}$ gated cross layer of the GCN is represented by:

$$\mathbf{c}_{l+1} = \underbrace{\mathbf{c}_0 \odot \left(\mathbf{W}_l^{(c)} \times \mathbf{c}_l + \mathbf{b}_l \right)}_{Feature\ Crossing} \odot \underbrace{\sigma \left(\mathbf{W}_l^{(g)} \times \mathbf{c}_l \right)}_{Information\ Gate} + \mathbf{c}_l, \tag{1}$$

where \mathbf{c}_0 is the base input from the embedding layer, which contains the 1st-order features; $\mathbf{c}_I \in \mathbb{R}^D$ are the output features from the

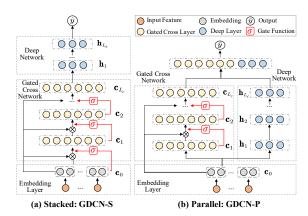


Figure 1: Architecture of the GDCN-S and GDCN-P. \otimes is the cross operation(a.k.a, the gated cross layer) in Equation 1.

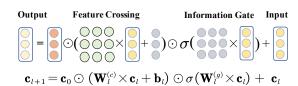


Figure 2: Visualization of the gated cross layer.⊙ is element-wise/Hadamard product, and × is matrix multiplication.

previous l^{th} gated cross layer and used as input to the current $(l+1)^{th}$ layer, and $\mathbf{c}_{l+1} \in \mathbb{R}^D$ are the output; $\mathbf{W}_l^{(c)}$, $\mathbf{W}_l^{(g)} \in \mathbb{R}^{D \times D}$ and $\mathbf{b}_l \in \mathbb{R}^D$ are the two learnable matrices and the bias vector, respectively. Figure 2 visualizes the process of the gated cross layer.

In each gated cross layer, there are two core components: the feature crossing and the information gate as shown in Equ.(1) and Figure 2. The feature crossing component calculates the interaction between the 1st-order feature c_0 and the $(l+1)^{th}$ order features c_l in bit-level. It then outputs the next polynomial order interaction that contains all $(l+2)^{th}$ order cross features. The matrix $\mathbf{W}_l^{(c)}$, known as the cross matrix, indicates the inherent importance among various fields in the $(l+1)^{th}$ order. However, not all $(l+2)^{th}$ order features have a positive impact on the prediction. As cross depth increases, the cross features exhibit exponential growth, introducing cross noise that can lead to sub-optimal performance. To address this, the information gate component is introduced to act as a soft gate that adaptively learns the importance of $(l+2)^{th}$ order features. The gate values are obtained by applying the sigmoid function $\sigma(\cdot)$ to the result of matrix multiplication between the gate matrix $\mathbf{W}_{r}^{(g)}$ and the input c_I . They are then element-wise multiplied with the output of the feature crossing component that contains unselected $(l+2)^{th}$ order cross features. This process amplifies important features and mitigates the impact of unimportant features. As the number of cross layers increases, the information gate at each cross layer filters the next-order cross features and effectively controls the information flow. Finally, the ultimate output cross vector \mathbf{c}_{l+1} is generated by adding the input c_l to the result of the feature crossing and information gate, thus containing all the feature interactions from the 1st order to the $(l+2)^{th}$ order.

3.3 Deep Neural Network (DNN)

The objective of the DNN is to model implicit feature interactions. Each deep layer of DNN is represented by $\mathbf{h}_{l+1} = f(\mathbf{W}_l\mathbf{h}_l + \mathbf{b}_l)$, where $\mathbf{W}_l \in \mathbb{R}^{n_{l+1} \times n_l}$, $\mathbf{b}_l \in \mathbb{R}^{n_{l+1}}$ are the weight matrix and bias vector in l^{th} deep layer; and $\mathbf{h}_{l+1} \in \mathbb{R}^{n_{l+1}}$, $\mathbf{h}_l \in \mathbb{R}^{n_l}$ are the output and input. $f(\cdot)$ is the activation function, which is usually ReLU.

3.4 Combine GCN and DNN

Existing works adopt two main structures to integrate explicit and implicit interaction information: stacked and parallel. We also have two versions of GDCN by combining the GCN and DNN.

Figure 1(a) shows the **stacked** structure: GDCN-S. The embedding vector \mathbf{c}_0 is fed into the GCN and output \mathbf{c}_{L_c} , followed by a DNN to generate the final cross vector $\mathbf{c}_{final} = \mathbf{h}_{L_d}$. L_c and L_d are the depth of gated cross layer and deep network, respectively.

Figure 1(b) shows the **parallel** structure: GDCN-P. Vector \mathbf{c}_0 is parallel fed into GCN and DNN. Their outputs (i.e., \mathbf{c}_{L_c} and \mathbf{h}_{L_d}) are concatenated to obtain the final cross vector $\mathbf{c}_{final} = [\mathbf{c}_{L_c} \parallel \mathbf{h}_{L_d}]$.

Prediction and Training. Finally, we calculate the prediction click probability \hat{y}_i by a standard logistic regression function: $\hat{y}_i = \sigma(\mathbf{w}_{logit}\mathbf{c}_{final})$, where \mathbf{w}_{logit} is the weight vector and $\sigma(z) = 1/(1+e^{-z})$.

The loss function is the widely used binary cross-entropy loss (a.k.a, LogLoss) [31, 56, 74]:

$$\mathcal{L}_{ctr} = -\frac{1}{N} \sum_{i=1}^{N} (y_i \log (\hat{y}_i) + (1 - y_i) \log (1 - \hat{y}_i)), \qquad (2)$$

where \hat{y}_i and y_i are the predicted and the ground-truth click probability, respectively. N is the number of all training instances.

3.5 Relationship with DCN-V2

GDCN is a generalization of DCN-V2. When the information gate is omitted, or all gate values are set to 1, GDCN falls back to DCN-V2 [56]. In DCN-V2, the cross layer (i.e., CN-V2) treats all cross features equally and directly aggregates them to the next order without considering the varying importance of different cross features. However, GDCN introduces GCN, incorporating an information gate at each gated cross layer. It adaptively learns bit-wise gate values to all cross features, enabling fine-grained control over the importance of each cross feature. Notably, both GDCN and DCN-V2 are capable of modeling both bit-wise and vector-wise feature crosses, as demonstrated in DCN-V2.

Although both GDCN and DCN-V2 use the gate mechanism, their purpose and design principles differ. DCN-V2 introduces the idea of MMoE [34, 46] to decompose the cross matrix into multiple smaller subspaces or "experts." A gating function then combines these experts. This approach primarily reduces non-embedding parameters in its cross matrices while maintaining performance. Differently, GDCN leverages the gating mechanism to choose important cross features adaptively and truly utilize deeper cross features without declining performance. It offers dynamic instance-based interpretability, allowing for a better understanding and analysis of the model's decision-making process. To further enhance the cost-efficiency of GDCN, Section 4 proposes a field-level dimension optimization approach to reduce embedding parameters directly.

4 FIELD-LEVEL DIMENSION OPTIMIZATION

The embedding dimension typically determines the ability to encode information [48, 69]. However, assigning the same dimension to all fields ignores information capacity in different fields. For example, the number of features in fields like "gender" and "item id" range from O(2) to $O(10^6)$. DCN-V2 and DCN employ a rule-of-thumb formula [2, 56] to assign independent dimensions for each field based on its feature number, i.e., (feature number)^{0.25}. This is a priori method but ignores the true importance of each field. Inspired by FmFM [48], we use a posteriori Field-level Dimension Optimization (FDO) method that learns independent dimensions for each field based on its intrinsic importance in a specific dataset.

To begin, we train a full model with a fixed field dimension of 16, as suggested by previous works [48, 74, 75]. This process allows us to generate an informative embedding table for each field. Next, we employ PCA [3] to calculate a set of singular values for each field's embedding table, arranged in descending order of magnitude. By evaluating the information utilization (i.e., information ratio), we can determine the optimal dimension by identifying the $argmin_k$ singular values that contribute most significantly to the overall information summation. This step enables us to select a suitable condensed dimension for each field. Lastly, we train a new model with the learned field dimensions from the previous step. In practice, we only need to learn a set of field dimensions once based on a full model and reuse it for subsequent model refreshes.

Table 1 presents the optimized dimension for each field with 80% and 95% information ratio. When retaining a 95% ratio, the field dimension ranges from 2 to 15. Decreasing the information ratio results in a reduction in the dimension of each field. Fields with enormous features sometimes require a higher dimension, as observed in fields {#23, #24}. However, this is not always the case; for instance, fields {#16, #25} exhibit smaller dimensions. In section 5.5, we present experimental evidence showing that the dimension of a field is strongly correlated with its importance in the prediction process rather than its feature number. Moreover, by preserving over 80% information ratio, we can obtain lighter GCN models, which slightly surpass the performance of the GCN model with full embedding dimensions and exceed other SOTA models. We also conduct a more comprehensive analysis of FDO to understand the connection between field dimension and its inherent importance.

Parameter analysis. Let $\mathbb{E}=[\mathrm{E}_1,\mathrm{E}_2,...,\mathrm{E}_F]$ represents the embeddings of all features, where E_f corresponds to the subset of feature representations for the $f^{th}(1 \leq f \leq F)$ field. The number of features in the f^{th} field is denoted as $|\mathrm{E}_f|$, and the total number of features of a dataset is $T=\sum_{f=1}^F |\mathrm{E}_f|$. Similarly, let $\mathrm{d}=[d_1,d_2,..,d_F]$ represents the different dimensions for each field, where d_f is the dimension of f^{th} field. For an input instance, the arithmetic average dimension is calculated as $\overline{K}=(\sum_{f=1}^F d_f)/F$, and the output dimension of the embedding layer is determined as $D=F\overline{K}$. Considering all features, the weighted average dimension is denoted as $\overline{D}=(\sum_{f=1}^F d_f |\mathrm{E}_f|)/T$. The total number of embedding parameters is $\mathcal{P}_e=\sum_{f=1}^F d_f |\mathrm{E}_f|=T\overline{D}$. The number of features T is typically massive in a web-scale dataset. For example, in the well-known Criteo dataset, the original feature number is over 30 million with sparsity over 99.99% [47, 57], with embedding

Table 1: Optimized dimension of each field in Criteo dataset (Refer to section 5.1), 95% and 80% information ratio.

Field	Feature	Emb	Dim	Field	Feature	Emb	Dim		
ID	Number	95%	80%	ID	Number	95%	80%		
#1	49	5	2	#22	4	2	2		
#2	101	13	8	#23	42,646	15	10		
#3	126	7	3	#24	5,178	14	10		
#4	45	4	2	#25	192,773	2	2		
#5	223	13	7	#26	3,175	13	9		
#6	118	9	4	#27	27	4	3		
#7	84	8	3	#28	11,422	13	9		
#8	76	5	2	#29	181,075	5	4		
#9	95	9	3	#30	11	5	4		
#10	9	4	2	#31	4,654	12	8		
#11	30	3	2	#32	2,032	11	7		
#12	40	4	2	#33	5	3	2		
#13	75	4	2	#34	189,657	5	2		
#14	1,458	12	8	#35	18	5	4		
#15	555	12	5	#36	16	7	4		
#16	193,949	4	3	#37	59,697	11	8		
#17	138,801	8	5	#38	86	5	3		
#18	306	10	6	#39	45,571	10	6		
#19	19	6	4						
#20	11,970	14	10	Weig	thted Ave. \overline{D}	5.92	3.98		
#21	634	11	6	Arithmetic Ave. \overline{K} 7.87 4					

parameters occupying the most portion of the model parameters. Hence, \overline{D} determines the number of embedding parameters, while \overline{K} mainly affects the number of non-embedding parameters, e.g., the cross matrix $\mathbf{W}^{(c)} \in \mathbb{R}^{F\overline{K} \times F\overline{K}}$ in DCN-V2 and GCN.

By adopting the FDO approach, we can refine the feature dimension by shrinking unnecessary dimensions for some fields to reduce redundant embedding parameters. When using a fixed dimension of 16, the embedding parameters are 16T. However, after applying FDO with 95% information ratio, the embedding parameters decrease to 5.92T, which accounts for only 37% of the original embedding parameters. If we calculate the field dimension based on the formula (i.e., $d_f = |\mathbf{E}_f|^{0.25}$), the weighted average dimension \overline{D} becomes 18.66, resulting in embedding parameters of 18.66T, which is larger than 16T. This formula assigns a larger dimension to the field with massive features, overlooking the specific importance of each field. In contrast, FDO is a posteriori method that learns fieldlevel dimensions based on specific information extracted from the trained embedding table. As the field dimension decreases, the arithmetic average dimension \overline{K} also decreases accordingly (e.g., from 16 to 7.87). Thus the non-embedding parameters in the GCN network, i.e., cross matrix $\mathbf{W}^{(c)} \in \mathbb{R}^{F\overline{K} \times F\overline{K}}$ and gate matrix $\mathbf{W}^{(g)} \in \mathbb{R}^{F\overline{K} \times F\overline{K}}$ are also reduced naturally.

5 EXPERIMENTAL ANALYSIS

5.1 Experiment Setup

Datasets. We choose five widely-used datasets to evaluate our proposed methods with other CTR models, i.e., **Criteo** [25, 53, 75] **Avazu** [26, 36, 47] **Malware** [37, 57] **Frappe** [4, 10, 59] **ML-tag** [10, 15, 59] The statistics of these datasets are shown in Table 2 and detailed descriptions can be found in the given references.

Data preparation. Firstly, we randomly split each dataset into training(80%), validation (10%) and testing (10%) datasets. Secondly, in Criteo and Avazu, we remove infrequent features in a certain field appearing less than *threshold* times and treat them as a dummy feature "*<unkonwn>*". The *threshold* is set to {10, 5} for Criteo and

Table 2: Dataset statistics. K means thousand.

Datasets	Positive	#Training	#Validation	#Testing	#Features	#Fields
Criteo	26%	36,672K	4,584K	4,584K	1,086K	39
Avazu	17%	32,343K	4,043K	4,043K	1,544K	23
Malware	50%	7,137K	892K	892K	976K	81
Frappe	33%	231K	29K	29K	5K	10
ML-tag	33%	1,605K	201K	201K	90K	3

Avazu, respectively. Finally, in the Criteo dataset, we normalize numerical values by transforming a value z to $\lfloor log^2(z) \rfloor$ when x > 2, which is adopted by the winner of the Criteo competition [1].

Evaluation Metrics. We adopt **AUC** (Area Under ROC) and **Logloss** (Cross Entropy) to assess the performance of all models on the testing set. A higher AUC or a lower Logloss (LL) at 0.001-level can be considered a significant improvement for the CTR prediction task, which is the common consensus in existing works [23, 30, 53, 56, 73]. Additionally, ΔAUC and ΔLL are calculated to show the averaged performance improvement compared to a given benchmark over five datasets in Table 3 and 4. Also, Rel.Imp denotes the relative improvements compared with the best baseline.

Comparison methods. To evaluate our methods, we compare them with four classes of representative methods. 1) First-order method, e.g., LR [44]; 2) FM-based methods that model second-order cross features, including FM [43], FwFM [39], DIFM [32], and FmFM [48]; 3) Methods that capture high-order cross features, including CrossNet(CN) [55], CIN [30], AutoInt [47], AFN [10], CN-V2 [56], IPNN [42], OPNN [42], FINT [71], FiBiNET [23] and SerMaskNet [57]. 4) representative ensemble/parallel methods, including WDL [9], DeepFM [16], DCN [55], xDeepFM [30], AutoInt+ [47], AFN+ [10], DCN-V2 [56], NON [33], FED [70], and ParaMaskNet [57]. We do not show the results of some methods, e.g., CCPM [14], GBDT [7], FFM [24], HoFM [5], AFM [59], NFM [19], as many works [10, 56, 57] have surpassed them.

Implementation details. We implement all models with Pytorch [40] and refer to existing works [8, 74, 75]. We use Adam [27] optimizer to optimize all models, and the default learning rate is 0.001. We utilize the Reduce-LR-On-Plateau scheduler during the training process to reduce the learning rate by 10 when the performance stops improving in 3 consecutive epochs. We apply early stopping with the patience of 5 on the validation set to avoid overfitting. The batch size is set to 4096. The embedding dimension for all datasets is 16. Following previous works [6, 10, 16, 23, 36, 47], we adopt the same structures (i.e., 3 layers, 400-400-400) for models that involve DNN for a fair comparison. Unless otherwise specified, all activation functions are ReLU, and the dropout rate is 0.5. For our proposed GCN, GDCN-S and GDCN-P, the default number of gated cross layer is 3 without specifically mentioned. For other baselines, we refer to two benchmark works (i.e., BARS [74] and FuxiCTR [75]) and their original literature to finetune their hyper-parameters.

Significance Test. To ensure a fair comparison, we run each method 10 times with random seeds on a single GPU (NVIDIA TITAN V) and report the average testing performance. We perform a two-tailed t-test [31, 52, 53] to detect the statistical significance between our method and the best baseline methods. The improvements over the best baselines are statistically significant with p<0.01 in all experiments, represented by ★ in Table 3 and Table 4.

Model	Datasets	Cr	iteo	Av	azu	Mal	ware	Fra	ppe	ML	-tag	ΔAUC	ΔLL
Class	Model	AUC	Logloss	AUC	Logloss	AUC	Logloss	AUC	Logloss	AUC	Logloss	1 1	\downarrow
First-order	LR	0.7937	0.4562	0.7573	0.3925	0.7107	0.6196	0.9376	0.2882	0.9339	0.2956	-3.62%	23.87%
	FM	0.8085	0.4433	0.7829	0.3785	0.7363	0.5982	0.9583	0.2336	0.9539	0.2523	-1.08%	11.33%
Second-	FwFM	0.8112	0.4408	0.7857	0.3772	0.7367	0.5980	0.9738	0.1851	0.9591	0.2307	-0.51%	3.30%
order	DIFM	0.8128	0.4395	0.7887	0.3748	0.7424	0.5926	0.9792	0.1880	0.9569	0.2316	-0.18%	3.37%
	FmFM	0.8122	0.4399	0.7874	0.3765	0.7433	0.5922	0.9762	0.1875	0.9589	0.2357	-0.22%	3.76%
	CN	0.8071	0.4442	0.7853	0.3784	0.7265	0.6049	0.9772	0.1876	0.9498	0.2582	-1.02%	6.45%
	CIN	0.8121	0.4398	0.7881	0.3754	0.7429	0.5924	0.9792	0.1850	0.9593	0.2490	-0.15%	4.55%
	AutoInt	0.8118	0.4399	0.7881	0.3750	0.7363	0.5976	0.9788	0.1671	0.9540	0.2575	-0.45%	3.25%
	AFN	0.8116	0.4401	0.7883	0.3752	0.7427	0.5924	0.9801	0.1674	0.9587	0.2305	-0.15%	0.78%
High-	CN-V2	0.8140	0.4380	0.7893	0.3745	0.7383	0.5959	0.9803	0.1710	0.9549	0.2529	-0.26%	3.16%
order	IPNN	0.8128	0.4390	0.7890	0.3747	0.7433	0.5918	0.9801	0.1724	0.9598	0.2344	-0.07%	1.64%
	OPNN	0.8135	0.4383	0.7892	0.3746	0.7436	0.5918	0.9804	0.1645	0.9595	0.2346	-0.04%	0.65%
	FiBiNet	0.8129	0.4389	0.7889	0.3746	0.7441	0.5914	0.9789	0.1777	0.9595	0.2352	-0.08%	2.33%
	FINT	0.8128	0.4390	0.7891	0.3747	0.7424	0.5924	0.9807	0.1631	0.9598	0.2356	-0.08%	0.62%
	SerMaskNet	0.8141	0.4379	0.7891	0.3746	0.7440	0.5920	0.9804	0.1628	0.9602	0.2297	-	-
	GCN	0.8154*	0.4367*	0.7903*	0.3742*	0.7445*	0.5908*	0.9820*	0.1634*	0.9619*	0.2250*	0.14%	-0.45%
Ours	GDCN-S	0.8158*	0.4364^{\star}	0.7905*	0.3739*	0.7456*	0.5899*	0.9838*	0.1470^{\star}	0.9645*	0.2230*	0.28%	-2.70%
	Rel.Imp	0.0017	-0.0015	0.0012	-0.0006	0.0015	-0.0015	0.0031	-0.0158	0.0043	-0.0067	-	-

Table 3: Overall performance comparison in the five datasets. $\triangle AUC$ and $\triangle LL$ indicate averaged performance boost compared with SerMaskNet. Bold scores are the best performance, and underlined scores are the best baseline performance.

5.2 Overall Performance

5.2.1 Comparison within stacked models. We compare GCN and GDCN-S with stacked baseline models, including first-order, second-order and high-order models. The overall performance is summarized in Table 3. We have the following observations:

First, in most cases, high-order models outperform first- and second-order models, demonstrating the effectiveness of learning complex high-order feature interactions. Notably, models such as OPNN, FiBiNet, FINT and SerMaskNet perform even better, which simultaneously capture explicit and implicit feature crosses with a stacked DNN. This confirms the rationale behind modeling both explicit and implicit high-order feature interactions.

Second, GCN consistently outperforms all stacked baseline models by considering only explicit polynomial feature interactions. GCN is a generalization of CN-V2, with the addition of an information gate to identify meaningful cross features. The performance of GCN validates that not all cross features are beneficial to the final prediction, and a large number of irrelevant interactions introduce unnecessary noise. By adaptively re-weighting cross features in each order, GCN achieves significant performance improvement over CN-V2. Moreover, it outperforms SerMaskNet by increasing the average ΔAUC by 0.14% and improving ΔLL by 0.45%.

Third, GDCN-S surpasses all stacked baselines and achieves the best performance. In GDCN-S, the stacked DNN further learn implicit interaction information over the GCN structure. As a result, GDCN-S outperforms GCN and achieves superior prediction accuracy compared to other stacked models, e.g., OPNN, FINT and SerMaskNet. Specifically, GDCN-S achieves an average improvement of 0.28% (ΔAUC) and 2.70% (ΔLL) compared to SerMaskNet.

5.2.2 Comparison within parallel models. Table 4 presents the performance of SOTA ensemble/parallel models. Each method incorporates parallel networks, such as FM and DNN in DeepFM, and CN-V2 and DNN in DCN-V2. Additionally, we compare these models with the vanilla DNN model and calculate the average improvement ΔAUC and ΔLL based on it. Our observations are as follows:

Firstly, integrating implicit and explicit feature interactions enhances predictive ability. While DNN solely models implicit feature interactions, parallel models that combine DNN with other networks to capture explicit feature interactions outperform individual networks. Notably, GCN shows the most substantial average performance improvement, as evident from the ΔAUC and ΔLL values of 0.46% and 5.57%, respectively. This highlights the superior power of GCN compared to other basic operation networks.

Secondly, GDCN-P outperforms all parallel and stacked models. When compared to the parallel models in Table 4, GDCN-P achieves superior performance, surpassing all the best baselines indicated with Rel.Imp of 0.0011 to 0.0053 on AUC and -0.0010 to -0.0129 on Logloss. Furthermore, when considering stacked models and using SerMaskNet from Table 3 as the benchmark, GDCN-P achieves an average improvement of 0.38% (ΔAUC) and 4.26% (ΔLL).

5.3 Deeper High-order Feature Crossing

5.3.1 Compare GCN to other models by changing cross depth. We compare GCN with five representative models that can capture high-order interactions, namely AFN [10], FINT [71], CN-V2 [56], DNN [66] and IPNN [42]. Figure 3 illustrates the testing AUC and Logloss as the cross depth increases on the Criteo dataset.

As the cross layers increase, the performance of the five compared models improves. However, their performance experiences a significant decline when the cross depth goes deeper (e.g., over 2, 3, or 4 cross layers). These models can capture deeper high-order explicit and implicit feature interactions functionally, but higher-order cross features also introduce unnecessary noise, which can result in overfitting and lead to a degradation in results. This issue is also observed in the hyper-parameter analysis of cross depth conducted in many SOTA works [5, 10, 12, 16, 30, 42, 47, 55, 56, 71].

In contrast, the performance of GCN continues to improve as the number of cross layers increases from 1 to 8. Specifically, GCN incorporates an information gate compared to the CN-V2 model. This information gate enables GCN to identify useful cross features

D	atasets	Cri	iteo	Av	azu	Mal	ware	Fra	ppe	ML	-tag	ΔAUC	ΔLL
Model	Networks	AUC	Logloss	1	\downarrow								
DNN	DNN	0.8139	0.4383	0.7886	0.3748	0.7424	0.5928	0.9794	0.1661	0.9598	0.2418	-	-
WDL	LR, DNN	0.8141	0.4379	0.7888	0.3747	0.7436	0.5915	0.9801	0.1652	0.9605	0.2411	0.07%	-0.23%
DeepFM	FM, DNN	0.8140	0.4378	0.7891	0.3745	0.7424	0.5925	0.9808	0.1613	0.9606	0.2440	0.05%	-0.42%
DCN	CN, DNN	0.8142	0.4378	0.7890	0.3745	0.7434	0.5917	0.9805	0.1618	0.9607	0.2283	0.08%	-1.69%
xDeepFM	CIN, DNN	0.8142	0.4377	0.7893	0.3744	0.7442	0.5911	0.9806	0.1638	0.9609	0.2328	0.12%	-1.10%
AutoInt+	AutoInt, DNN	0.8144	0.4376	0.7886	0.3745	0.7435	0.5917	0.9802	0.1654	0.9609	0.2343	0.07%	-0.76%
AFN+	AFN, DNN	0.8138	0.4384	0.7887	0.3748	0.7439	0.5912	0.9807	0.1620	0.9605	0.2295	0.07%	-1.53%
DCN-V2	CN-V2, DNN	0.8144	0.4375	0.7898	0.3743	0.7433	0.5920	0.9810	0.1539	0.9603	0.2284	0.11%	-2.67%
NON	LR, Att., DNN	0.8133	0.4390	0.7889	0.3747	0.7426	0.5925	0.9803	0.1564	0.9609	0.2279	0.03%	-2.26%
FED	Concat., Att., DNN	0.8141	0.4379	0.7891	0.3745	0.7436	0.5915	0.9811	0.1590	0.9610	0.2294	0.12%	-1.99%
ParaMaskNet	MaskBlocks, DNN	0.8142	0.4377	0.7895	0.3748	0.7435	0.5920	0.9805	0.1559	0.9608	0.2272	0.10%	-2.47%
GDCN-P	GCN, DNN	0.8161*	0.4360*	0.7909*	0.3733*	0.7462*	0.5893*	0.9852*	0.1410*	0.9663*	0.2144*	0.46%	-5.57%
I	Rel.Imp	0.0017	-0.0015	0.0011	-0.0010	0.0020	-0.0018	0.0041	-0.0129	0.0053	-0.0128	-	-

Table 4: Performance of parallel models, which integrates implicit feature interactions, i.e., DNN. We list the main parallel networks for each model. "Att." and "Concat." represents the attention and concatenation operation, respectively.

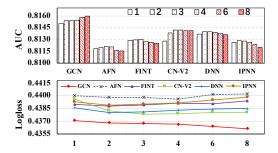


Figure 3: Comparison of different cross depth on six models.

in each order and accumulate only necessary information for final prediction. Therefore, GCN consistently outperforms CN-V2 and other models, even when the cross depth becomes deeper, verifying the design rationality of the information gate.

5.3.2 The performance of GCN and GDCN-P with deeper gated cross layers. We further increase the cross depth of GCN and GDCN-P on the Criteo and Malware datasets from 1 to 16. Notably, we kept the depth of the DNN fixed at 3 in GDCN-P to prevent overfitting caused by the DNN. Figure 4 presents the experimental results.

As the cross depth increases, the performance of both GCN and GDCN-P improves, and the trend observed in GDCN-P is consistent with that of GCN. GDCN-P consistently outperforms GCN after incorporating the DNN component, highlighting the importance of capturing implicit feature interactions using DNN. Furthermore, DNN enables GDCN-P to achieve the best results earlier. Specifically, the performance of GCN reaches a plateau when the depth exceeds 8 or 10, whereas the threshold for GDCN-P is 4 or 6. Although GCN and GDCN-P can select valuable cross features, the usefulness of high-order cross features decreases sharply as the cross depth increases. This phenomenon aligns with the common intuition that individuals are not typically influenced by too many features simultaneously when making a decision, such as clicking or purchasing an item. Notably, our models' performance remains stable instead of decreasing after surpassing the plateau threshold.

5.4 The Interpretability of GCN

Interpretability is vital in understanding the reasoning behind specific predictions and enhancing confidence in the prediction results.

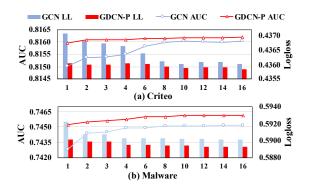


Figure 4: Performance with deeper gated cross layers.

The GCN offers both static and dynamic interpretations from the model and instance perspectives, facilitating a comprehensive understanding of the model's decision-making process.

5.4.1 Static model interpretability. Within GCN, the cross matrix $\mathbf{W}^{(c)}$ serves as an indicator of the relative importance of interactions among different fields. If each instance consists of F fields with the same field size d, the cross matrix can be represented in both a bit-wise and block-wise manner, as demonstrated in Equ.3.

$$\mathbf{W}^{(c)} = \begin{bmatrix} w_{1,1} & \cdots & w_{1,Fd} \\ \vdots & \ddots & \vdots \\ w_{Fd,1} & \cdots & w_{Fd,Fd} \end{bmatrix} = \begin{bmatrix} W_{1,1} & \cdots & W_{1,F} \\ \vdots & \ddots & \vdots \\ W_{F,1} & \cdots & W_{F,F} \end{bmatrix}$$
(3)

Each block-wise matrix $W_{i,j} \in \mathbb{R}^{d \times d}$ shows the importance of the 1st-order cross between the i-th and j-th fields. When FDO is applied to learn various field dimensions, d_i and d_j in block $W_{i,j} \in \mathbb{R}^{d_i \times d_j}$ are the different dimensions of the i-th and j-th fields. Additionally, as cross layers increase, the corresponding cross matrix can further indicate the inherent relation importance among multiple fields.

Figure 5(a) shows the heatmap of the block-wise cross matrix $\mathbf{W}_1^{(c)}$ in 1st cross layer learned by GCN on the Criteo dataset. Similar to DCN-V2 [56], each color box represents the *Frobenius norm* of the corresponding block-wise matrix $W_{i,j}$, indicating the importance of field crosses. Darker shades of red indicate stronger learned crosses, such as <#3, #2> and <#9, #6>. When applying the FDO approach to GCN, GCN-FDO still captures similar cross importance, as depicted

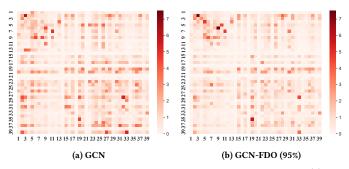


Figure 5: Visualization of the block-wise cross matrix $W_1^{(c)}$ in 1st gated cross layer learned by GCN and GCN-FDO (95%).

in Figure 5(b). The cosine similarity between the two matrices is 0.89, demonstrating a strong consistency in capturing the inherent field cross importance before and after the application of FDO.

5.4.2 Dynamic instance interpretability. Model-based interpretability, which captures static relationships among different fields, has limitations as the cross matrices remain fixed once a model is trained. However, GCN also offers dynamic interpretability through gate weights learned by the information gate, providing both bitwise and field-wise interpretations for each input instance.

We randomly select two instances from the Criteo dataset to visualize the learned gate weights and examine the gate values from the 1st to 3rd gated cross layer. Figure 6(a) presents the bit-wise gate weight vectors, with dimensions of $\mathbb{R}^{1\times(39*16)}$ in each layer, showing the importance of each bit-wise cross. Using the bit-wise gate vector, we derive the field-wise gate vectors by averaging the 16-bit values corresponding to each field. Figure 6(b) displays the field-wise gate weight vectors ($\mathbb{R}^{1\times39}$), indicating the importance of each specific feature. As the gate weights are calculated using the Sigmoid function in the gated cross layer, the red blocks (greater than 0.5) indicate important features, while the blue blocks (less than 0.5) indicate unimportant features.

Figures 6(a) and 6(b) reveal the importance of cross features at both the bit-level and field-level and how they vary across cross layers for each instance. Generally, lower-order feature crosses contain more significant features, while higher-order feature crosses contain less important features. In the 1st layer, numerous feature crosses are identified as important (red blocks), whereas in the 2nd and 3rd cross layers, most crosses are neutral (white blocks) or unimportant (blue blocks), particularly in the 3rd layer. This aligns with our intuition: as the cross layer increases, the number of important cross features significantly decreases, so we design the information gate to choose the important features adaptively. In contrast, most models fail to select useful feature crosses when modeling high-order crosses, resulting in a performance drop, as confirmed in Section 5.3. Moreover, from Figure 6(b), we can identify specific features that are important or unimportant, such as features {#20, #23, #28} being important and features {#6, #11, #30} being unimportant. We can also refer to the names of these specific important features from the input instance. Once we know which features are influential, we can interpret and even intervene in relevant features that contribute to the click probability for users.

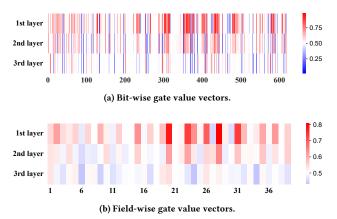


Figure 6: Visualization of the gate value vectors.

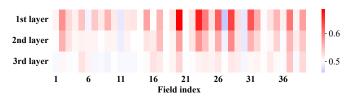


Figure 7: Visualization of average field-wise gate vectors on 1,000,000 instances.

Lastly, in Figure 7, we record and average the field-level gate vectors from 1,000,000 instances, indicating the average importance of each field, particularly in the 1st layer. For example, fields {#20, #23, #24} are significantly important, while fields {#11, #27, #30} are relatively less important. Furthermore, Figure 7 further verifies the significant decrease in the number of important cross features as the cross layer increases from a statistical perspective.

5.5 Comprehensive Analysis of FDO

5.5.1 Effectiveness analysis. We apply the FDO approach to GCN and assess its performance by keeping various information ratio from 50% to 98%. Table 5 shows the results, including the weight average dimension \overline{D} , arithmetic dimension \overline{K} and the total number of parameters (#Params). When maintaining 80% ratio, we only require 23% of the parameters, while the model's performance is on par with the full model. Moreover, when the information ratio is between 80% to 98%, the model performance is even better than the full model. This is because FDO refines the feature dimensions for each field, eliminating non-essential dimensional information and reducing redundancy during the cross process. However, when the information ratio is lower (i.e., less than 80%), the shorter dimension fails to adequately represent the features in the corresponding field, resulting in a significant decline in prediction performance. Lastly, we reduce the full model's dimension from 16 to 8, approximately equivalent to the weighted average dimension (7.56) with a 95% ratio. Although it directly decreases the model's parameters, it also leads to a drop in performance. Other studies [19, 20, 47, 49] have likewise confirmed the negative impact of reducing embedding size. In comparison, the FDO method can help reduce parameters number more directly while achieving comparable performance.

Table 5: The effectiveness of GCN with FDO approach. All results are statistically significant with p<0.001 compared to the full (16) model with fixed dimension. M is million.

Ratio	\overline{D}	\overline{K}	#Params	Proportion	AUC	Logloss
Full (16)	16	16	19.73M	100%	0.8154	0.4367
98%	7.56	9.54	9.04M	45.8%	0.8157	0.4365
95%	5.92	7.87	7.00M	35.5%	0.8157	0.4365
90%	4.99	5.92	5.80M	29.4%	0.8156	0.4365
80%	3.98	4.85	4.54M	23.0%	0.8155	0.4366
70%	2.94	3.69	3.32M	16.8%	0.8152	0.4371
60%	2.41	2.90	2.69M	13.6%	0.8146	0.4374
50%	1.97	2.26	2.54M	11.1%	0.8137	0.4380
Full (8)	8	8	9.27M	47.9%	0.8151	0.4371

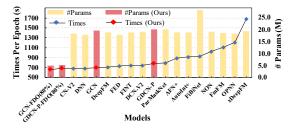


Figure 8: The comparison of model parameters (#Params) and the training time per epoch on Criteo dataset.

5.5.2 Memory and efficiency comparison. On the Criteo dataset, we have a total of 1.086M features after removing infrequent ones. When using a fixed dimension (K=16), the embedding parameters are around 17.4M. As shown in Figure 8, most existing models have parameters ranging from 18M to 20M, primarily due to the embedding parameters. By applying FDO, we can directly reduce the embedding parameters. Meanwhile, GCN-FDO and GDCN-P-FDO achieve comparable accuracy with only around 5M model parameters. We further compare the training time among existing SOTA models. The training time of the GCN and GDCN without dimension optimization is comparable to that of DNN and DCN-V2. After applying FDO, GCN-FDO and GDCN-P-FDO outperform all SOTA models with fewer parameters and faster training speed.

5.5.3 Compatibility analysis. We further apply FDO to DNN, CN-V2, DCN-V2 and FmFM, which do not require the same field dimensions. Additionally, we compare FDO with assigning fixed dimensions and the Formula method mentioned in Section 4 to assign various dimensions. Table 6 shows the results. First, the Formula method increases the model parameters and decreases its performance, as it only considers the number of features in each field, disregarding its importance during the training process. In contrast, FDO is a posteriori approach that learns field dimensions by incorporating important information in the trained embedding table. Second, applying FDO to CN-V2 and GCN yields better performance than the base models. Since CN-V2 and GCN mainly focus on explicit bit-level feature crosses, FDO refines the field dimensions by removing unnecessary embedding information. However, applying FDO to DNN, DCN-V2 and GDCN-P, which include a DNN network, leads to a slight performance decrease. Lastly, these results show that we can use FDO to obtain group field dimensions based on the SOTA model and reuse it as the default dimensions for other

Table 6: The comparison of different dimension assignment methods. We list the parameter number of all model variants.

Models	CN-V2	GCN	DNN	DCN-V2	GDCN-P	FmFM
Fixed	0.8140	0.8154	0.8141	0.8144	0.8161	0.8122
rixeu	(18.6M)	(19.7M)	(18.0M)	(19.1M)	(20.3M)	(18.7M)
Formula	0.8139	0.8154	0.8136	0.8140	0.8157	0.8121
romuna	(20.5M)	(20.7M)	(20.7M)	(20.9M)	(21.1M)	(21.4M)
FDO	0.8142	0.8157	0.8140	0.8143	0.8160	0.8122
(95%)	(9.2M)	(7.0M)	(12.2M)	(10.95M)	(7.6M)	(9.4M)
FDO	0.8142	0.8157	0.8138	0.8142	0.8160	0.8122
(GCN-95%)	(6.7M)	(7.0M)	(6.9M)	(7.2M)	(7.4M)	(6.5M)

models. In the last row of Table 6, we learn group field dimensions with 95% information ratio based on the GCN and apply them to the other five models. This approach achieves comparable performance and further reduces the number of parameters compared to using the models themselves to learn the dimensions.

5.5.4 The understanding of condensed field dimensions. The field dimensions learned through FDO indicate the importance of the corresponding fields. As observed in Figure 7, we can determine the average importance of each field, such as fields {#20, #23, #24 } being important, while fields {#11, #27, #30} are considered unimportant. Referring back to Table 1, fields {#20, #23, #24} indeed have longer dimensions after applying FDO with 95% information ratio. Conversely, fields {#11, #27, #30} have shorter dimensions. To further validate this observation, we compute the Pearson Correlation Coefficient between the averaged field importance in the 1st layer and the optimization field dimensions with 95% ratio. The correlation coefficient is 0.82 with a p-value less than 1*e-9, which confirms a significant correlation between the field dimensions and their respective importance. Therefore, we can roughly identify which fields are important directly from the field dimensions learned by FDO. Please note that the field dimension is not always associated with the number of features in the field. For example, fields {#16, #25} have the maximum feature number, but their field dimensions are short, and their importance is also insignificant. If the Formula method were used to assign field dimensions, fields {#16, #25} would have the longest dimensions. This comparison further highlights the rationality and superiority of the introduced FDO approach.

6 CONCLUSIONS

This paper proposes a novel method GDCN to model both explicit and implicit feature crosses. As the core structure, GCN utilizes the information gate to identify important cross features, avoiding model performance degradation in deeper high-order layers. Importantly, GCN shows great interpretability at both model and instance levels, helping us to understand the model predictions. The proposed FDO approach learns field-specific and condensed dimensions for different fields based on their inherent importance, assisting GCN and GDCN to achieve comparable performance with lighter model size. Extensive experiments verify the remarkable effectiveness and superiority of the GDCN model and FDO approach.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (NSFC) under Grants 62172106 and 61932007.

REFERENCES

- [1] 2014. Idiots' Approach for Display Advertising Challenge. https://www.csie.ntu.edu.tw/~r01922136/kaggle-2014-criteo.pdf
- [2] 2017. Introducing TensorFlow Feature Columns. https://developers.googleblog.com/2017/11/introducing-tensorflow-feature-columns.html
- [3] Hervé Abdi and Lynne J Williams. 2010. Principal component analysis. Wiley interdisciplinary reviews: computational statistics 2, 4 (2010), 433–459.
- [4] Linas Baltrunas, Karen Church, Alexandros Karatzoglou, and Nuria Oliver. 2015. Frappe: Understanding the usage and perception of mobile app recommendations in-the-wild. arXiv preprint arXiv:1505.03014 (2015).
- [5] Mathieu Blondel, Akinori Fujino, Naonori Ueda, and Masakazu Ishihata. 2016. Higher-order factorization machines. Advances in Neural Information Processing Systems 29 (2016).
- [6] Bo Chen, Yichao Wang, Zhirong Liu, Ruiming Tang, Wei Guo, Hongkun Zheng, Weiwei Yao, Muyu Zhang, and Xiuqiang He. 2021. Enhancing Explicit and Implicit Feature Interactions via Information Sharing for Parallel Deep CTR Models. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management. 3757–3766.
- [7] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. 785–794.
- [8] Zekai Chen, Fangtian Zhong, Zhumin Chen, Xiao Zhang, Robert Pless, and Xiuzhen Cheng. 2021. DCAP: Deep Cross Attentional Product Network for User Response Prediction. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management. 221–230.
- [9] 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.
- [10] Weiyu Cheng, Yanyan Shen, and Linpeng Huang. 2020. Adaptive factorization network: Learning adaptive-order feature interactions. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34. 3609–3616.
- [11] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014).
- [12] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In Proceedings of the 10th ACM conference on recommender systems. 191–198.
- [13] Hongliang Fei, Jingyuan Zhang, Xingxuan Zhou, Junhao Zhao, Xinyang Qi, and Ping Li. 2021. GemNN: gating-enhanced multi-task neural networks with feature interaction learning for CTR prediction. In Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval. 2166– 2171.
- [14] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *International Conference on Machine Learning*. PMLR, 1243–1252.
- [15] GroupLens. 2003. MovieLens Tag Dataset. https://grouplens.org/datasets/ movielens/.
- [16] 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. 1725–1731.
- [17] 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.
- [18] Wei Guo, Can Zhang, Zhicheng He, Jiarui Qin, Huifeng Guo, Bo Chen, Ruiming Tang, Xiuqiang He, and Rui Zhang. 2022. Miss: Multi-interest self-supervised learning framework for click-through rate prediction. In 2022 IEEE 38th International Conference on Data Engineering (ICDE). IEEE, 727–740.
- [19] 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.
- [20] 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.
- [21] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. Neural computation 9, 8 (1997), 1735–1780.
- [22] 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).
- [23] 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.
- [24] Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. 2016. Fieldaware factorization machines for CTR prediction. In Proceedings of the 10th ACM

- Conference on Recommender Systems. 43-50.
- [25] Kaggle. 2014. Criteo Display Advertising Challenge. https://www.kaggle.com/c/ criteo-display-ad-challenge.
- [26] Kaggle. 2015. Avazu Click-Through Rate Prediction. https://www.kaggle.com/c/avazu-ctr-prediction.
- [27] Diederik P Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In ICLR (Poster).
- [28] Lang Lang, Zhenlong Zhu, Xuanye Liu, Jianxin Zhao, Jixing Xu, and Minghui Shan. 2021. Architecture and operation adaptive network for online recommendations. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. 3139–3149.
- [29] Zeyu Li, Wei Cheng, Yang Chen, Haifeng Chen, and Wei Wang. 2020. Interpretable click-through rate prediction through hierarchical attention. In Proceedings of the 13th International Conference on Web Search and Data Mining. 313–321.
- [30] 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.
- [31] Bin Liu, Ruiming Tang, Yingzhi Chen, Jinkai Yu, Huifeng Guo, and Yuzhou Zhang. 2019. Feature generation by convolutional neural network for click-through rate prediction. In The World Wide Web Conference. 1119–1129.
- [32] Wantong Lu, Yantao Yu, Yongzhe Chang, Zhen Wang, Chenhui Li, and Bo Yuan. 2020. A Dual Input-aware Factorization Machine for CTR Prediction. In IJCAI. 3139–3145.
- [33] Yuanfei Luo, Hao Zhou, Wei-Wei Tu, Yuqiang Chen, Wenyuan Dai, and Qiang Yang. 2020. Network On Network for Tabular Data Classification in Real-world Applications. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. 2317–2326.
- [34] 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.
- [35] Kelong Mao, Jieming Zhu, Liangcai Su, Guohao Cai, Yuru Li, and Zhenhua Dong. 2023. FinalMLP: An Enhanced Two-Stream MLP Model for CTR Prediction. arXiv preprint arXiv:2304.00902 (2023).
- [36] Ze Meng, Jinnian Zhang, Yumeng Li, Jiancheng Li, Tanchao Zhu, and Lifeng Sun. 2021. A general method for automatic discovery of powerful interactions in click-through rate prediction. In Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval. 1298–1307.
- [37] Microsoft. 2019. Microsoft Malware Prediction. https://www.kaggle.com/c/microsoft-malware-prediction.
- [38] Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G Azzolini, et al. 2019. Deep learning recommendation model for personalization and recommendation systems. arXiv preprint arXiv:1906.00091 (2019)
- [39] Junwei Pan, Jian Xu, Alfonso Lobos Ruiz, Wenliang Zhao, Shengjun Pan, Yu Sun, and Quan Lu. 2018. Field-weighted factorization machines for click-through rate prediction in display advertising. In Proceedings of the 2018 World Wide Web Conference. 1349–1357.
- [40] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems 32 (2019).
- [41] Liang Qu, Yonghong Ye, Ningzhi Tang, Lixin Zhang, Yuhui Shi, and Hongzhi Yin. 2022. Single-shot embedding dimension search in recommender system. In Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval. 513–522.
- [42] 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.
- [43] Steffen Rendle. 2012. Factorization machines with libfm. ACM Transactions on Intelligent Systems and Technology (TIST) 3, 3 (2012), 1–22.
- [44] 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.
- [45] Ying Shan, T Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and JC Mao. 2016. Deep crossing: Web-scale modeling without manually crafted combinatorial features. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. 255–262.
- [46] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. arXiv preprint arXiv:1701.06538 (2017).
- [47] 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.

- [48] Yang Sun, Junwei Pan, Alex Zhang, and Aaron Flores. 2021. FM2: Field-matrixed factorization machines for recommender systems. In *Proceedings of the Web Conference* 2021. 2828–2837.
- [49] Yang Sun, Fajie Yuan, Min Yang, Guoao Wei, Zhou Zhao, and Duo Liu. 2020. A generic network compression framework for sequential recommender systems. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. 1299–1308.
- [50] Zhen Tian, Ting Bai, Zibin Zhang, Zhiyuan Xu, Kangyi Lin, Ji-Rong Wen, and Wayne Xin Zhao. 2023. Directed Acyclic Graph Factorization Machines for CTR Prediction via Knowledge Distillation. In Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining. 715–723.
- [51] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Advances in neural information processing systems. 5998–6008.
- [52] Fangye Wang, Hansu Gu, Dongsheng Li, Tun Lu, Peng Zhang, and Ning Gu. 2022. MCRF: Enhancing CTR Prediction Models via Multi-channel Feature Refinement Framework. In Database Systems for Advanced Applications: 27th International Conference, DASFAA 2022, Virtual Event, April 11–14, 2022, Proceedings, Part II. Springer, 359–374.
- [53] Fangye Wang, Yingxu Wang, Dongsheng Li, Hansu Gu, Tun Lu, Peng Zhang, and Ning Gu. 2022. Enhancing CTR prediction with context-aware feature representation learning. In Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval. 343–352.
- [54] Fangye Wang, Yingxu Wang, Dongsheng Li, Hansu Gu, Tun Lu, Peng Zhang, and Ning Gu. 2023. CL4CTR: A Contrastive Learning Framework for CTR Prediction. In Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining. 805–813.
- [55] 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.
- [56] Ruoxi Wang, Rakesh Shivanna, Derek Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed Chi. 2021. DCN V2: Improved Deep & Cross Network and Practical Lessons for Web-scale Learning to Rank Systems. In Proceedings of the Web Conference 2021. 1785–1797.
- [57] Zhiqiang Wang, Qingyun She, and Junlin Zhang. 2021. MaskNet: Introducing Feature-Wise Multiplication to CTR Ranking Models by Instance-Guided Mask. arXiv preprint arXiv:2102.07619 (2021).
- [58] Zhiqiang Wang, Qingyun She, Peng Tao Zhang, and Junlin Zhang. 2021. ContextNet: A Click-Through Rate Prediction Framework Using Contextual information to Refine Feature Embedding. arXiv preprint arXiv:2107.12025 (2021).
- [59] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. 2017. Attentional factorization machines: learning the weight of feature interactions via attention networks. In Proceedings of the 26th International Joint Conference on Artificial Intelligence. 3119–3125.
- [60] YaChen Yan and Liubo Li. 2023. DynInt: Dynamic Interaction Modeling for Large-scale Click-Through Rate Prediction. arXiv preprint arXiv:2301.08139 (2023).
- [61] Yi Yang, Baile Xu, Shaofeng Shen, Furao Shen, and Jian Zhao. 2020. Operation-aware Neural Networks for user response prediction. Neural Networks 121 (2020),

- 161-168
- [62] Feng Yu, Zhaocheng Liu, Qiang Liu, Haoli Zhang, Shu Wu, and Liang Wang. 2020. Deep interaction machine: A simple but effective model for high-order feature interactions. In Proceedings of the 29th ACM International Conference on Information & Knowledge Management. 2285–2288.
- [63] Runlong Yu, Yuyang Ye, Qi Liu, Zihan Wang, Chunfeng Yang, Yucheng Hu, and Enhong Chen. 2021. XCrossNet: Feature Structure-Oriented Learning for Click-Through Rate Prediction. In PAKDD (2). Springer, 436–447.
- [64] Yantao Yu, Zhen Wang, and Bo Yuan. 2019. An Input-aware Factorization Machine for Sparse Prediction. In IJCAI. 1466–1472.
- [65] Pengtao Zhang and Junlin Zhang. 2022. FiBiNet++: Improving FiBiNet by Greatly Reducing Model Size for CTR Prediction. arXiv preprint arXiv:2209.05016 (2022).
- [66] Weinan Zhang, Tianming Du, and Jun Wang. 2016. Deep Learning over Multifield Categorical Data: –A Case Study on User Response Prediction. In Advances in Information Retrieval: 38th European Conference on IR Research, ECIR 2016, Padua, Italy, March 20–23, 2016. Proceedings 38. Springer, 45–57.
- [67] Weinan Zhang, Jiarui Qin, Wei Guo, Ruiming Tang, and Xiuqiang He. 2021. Deep learning for click-through rate estimation. arXiv preprint arXiv:2104.10584 (2021).
- [68] Keke Zhao, Xing Zhao, Qi Cao, and Linjian Mo. 2022. A Non-sequential Approach to Deep User Interest Model for CTR Prediction. In Proceedings of the 2022 SIAM International Conference on Data Mining (SDM). SIAM, 531–539.
- [69] Xiangyu Zhao, Haochen Liu, Hui Liu, Jiliang Tang, Weiwei Guo, Jun Shi, Sida Wang, Huiji Gao, and Bo Long. 2021. Autodim: Field-aware embedding dimension searchin recommender systems. In Proceedings of the Web Conference 2021. 3015–3022
- [70] Zihao Zhao, Zhiwei Fang, Yong Li, Changping Peng, Yongjun Bao, and Weipeng Yan. 2020. Dimension relation modeling for click-through rate prediction. In Proceedings of the 29th ACM International Conference on Information & Knowledge Management. 2333–2336.
- [71] Zhishan Zhao, Sen Yang, Guohui Liu, Dawei Feng, and Kele Xu. 2021. FINT: Field-aware INTeraction Neural Network For CTR Prediction. arXiv preprint arXiv:2107.01999 (2021).
- [72] 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.
- [73] Guanghui Zhu, Feng Cheng, Defu Lian, Chunfeng Yuan, and Yihua Huang. 2022. NAS-CTR: Efficient Neural Architecture Search for Click-Through Rate Prediction. In Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval. 332–342.
- [74] Jieming Zhu, Quanyu Dai, Liangcai Su, Rong Ma, Jinyang Liu, Guohao Cai, Xi Xiao, and Rui Zhang. 2022. Bars: Towards open benchmarking for recommender systems. In Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval. 2912–2923.
- [75] Jieming Zhu, Jinyang Liu, Shuai Yang, Qi Zhang, and Xiuqiang He. 2021. Open benchmarking for click-through rate prediction. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management. 2759–2769.

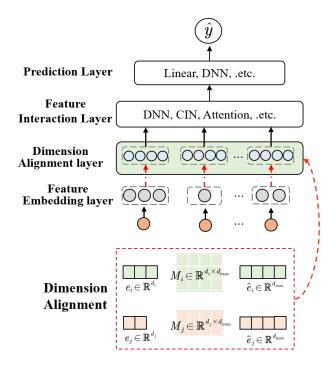


Figure 9: The structure of used generalization framework with dimension alignment layer.

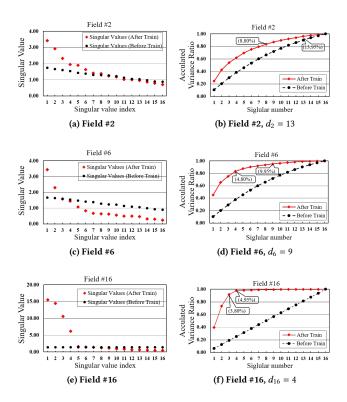


Figure 10: Schematic diagram of the FDO method for calculating feature dimensions based on the kept information ratio.

A THE GENERALIZATION FRAMEWORK OF FDO APPROACH

The proposed GDCN model can receive different dimensions as inputs and use the refined feature dimension information learned by FDO. In addition, methods such as DCN DCN-V2 can also accept different field dimensions as input. However, due to the limitations of the interaction structure, most methods can only accept the same feature representation dimension, such as xDeepFM, FGCNN, AutoInt, and FINT. Therefore, in order to make feature representations of different dimensions that can be used seamlessly in these base models, we use a generalization framework that contains a dimension alignment layer, as shown in Figure 9. This method is also used in [41, 69]. Specifically, for each field, we initialize an alignment matrix. For a specific dataset, all features in each field share a dimension alignment matrix, we initialize Falignment matrices $\mathbf{M} = \{\mathbf{M}_1, \mathbf{M}_2, ..., \mathbf{M}_F\}$, where $\mathbf{M}_f \in \mathbb{R}^{d_f \times d_{max}}$ and $d_{max} = max\{d_1, d_2, ... d_F\} \le D$. d_{max} is the maximum field dimension among all dimensions learned by the FDO approach. With the alignment matrix, we can get the aligned embedding $\hat{\mathbf{e}}_f = \mathbf{e}_f \mathbf{M}_f \in \mathbb{R}^{d_{max}}$. In this process, the dimension alignment layer generates a small number of learnable parameters:

$$\mathcal{P}_{align} = \sum_{i=1}^{F} d_{max} * d_f. \tag{4}$$

Take the Criteo dataset as an example, F=39, D=16 and $d_{max} \leq$ 16. Hence, the added parameters \mathcal{P}_{align} is less than 9984. Compared to the feature representation parameters that can be reduced, these parameters are negligible. After dimension alignment, most feature-based interactions can be performed. Based on the dimension alignment framework, other CTR models can also accept the refined dimensions learned by the FDO approach to reduce model parameters seamlessly and also improve training efficiency. Furthermore, subsequent experiments verify that applying the framework to basic model can improve their prediction performance.

B ADDED EXPERIMENTS

In addition to the experimental results that already given in Section 5, we further share several relevant and necessary experiments and corresponding results.

B.1 Dimension Understanding

Figure 10 illustrates the process of the FDO method to select the appropriate dimension based on the kept information ratio. We select three representative fields (i.e., fields #2, #6 and #16) for further analysis. Subplots 10 (a), (b), and (c) list the calculated singular values (sorted from largest to smallest) based on the feature embedding matrix of corresponding fields before and after model training. In those subplots, the black points indicate the calculated singular values based on the initial feature embedding matrix, and the red points indicate the calculated singular values based on the trained feature embedding matrix. The corresponding subplots (b), (e), and (f) plot the ratio of cumulative information based on calculated singular values, where the horizontal coordinate indicates the number of singular values, ranging from 1 to 16. Also, the figure lists the minimum number of singular values required to achieve the

specified information ratio. For example, (13, 95%) in subplot (b) indicates that a minimum of 13 singular values are required for the sum of their information variance to exceed 95% of the total information. Therefore, for field #2, the feature dimension that should be assigned while retaining 95% information ratio is 13. Furthermore, from table 1, we know the feature number contained in the fields {#2, #6, #16} are 101, 118, and 193,949, respectively. Based on the above information, the following observations can be observed:

Firstly, the singular values learned based on the initial feature embedding matrix do not provide valid information. As can be seen from subplots (a), (c) and (e), the difference in singular values in the same subplot is small at this point. Especially when the number of features contained in the field is large (subplot (e)), all the singular values are basically equal. From subplots (b), (d) and (f), it can be observed that the proportion of cumulative information calculated based on these singular values increases almost uniformly.

Secondly, the training-based feature embedding matrix can provide significant instruction for dimension selection. After model training, the importance of the feature field is implicitly reflected in the feature embedding matrix and can be expressed explicitly through the informative magnitude of the singular values. While keeping 95% information ratio, the calculated feature dimensions for the three feature fields are 13, 9 and 4, respectively. Comparing the red scatter plots in subfigures (a), (c), and (e), the smaller dimension required for the feature field corresponds to a larger singular value, and the gap between the maximum and minimum singular values will be larger. Larger singular values take up a larger proportion of the information, so subfigure (f) needs only 4 singular values to obtain 95% information ratio. For field #16, other dimensions contain very little information but occupy a large number of model parameters and even inversely affect the predictive performance.

B.2 Effectiveness Analysis

Table 7 is the supplement experiment of Section 5.5, which show the experiment results of the Malware dataset with different information ratio.

Similar to the Criteo dataset, when maintaining 80% information ratio, the model's performance is on par with the full model. For the Malware dataset, the FDO method is more effective in reducing the model parameters. Specifically, when the kept information ratio is between 80% to 98%, the corresponding model parameters are only 1.46M and 5.29M, only occupying 5.7% to 20.6% of the full

Table 7: The effectiveness of GCN with FDO approach on the Malware dataset.

Ratio	\overline{D}	\overline{K}	#Params	Proportion	AUC	Logloss
Full (16)	16	16	25.70M	100%	0.7445	0.5908
98%	4.88	3.64	5.29M	20.6%	0.7447	0.5904
95%	3.57	2.94	3.82M	14.9%	0.7445	0.5907
90%	2.40	2.38	2.56M	10.0%	0.7445	0.5908
80%	1.17	1.79	1.46M	5.7%	0.7442	0.5910
70%	1.11	1.52	1.18M	4.6%	0.7429	0.5920
60%	1.06	1.31	1.10M	4.3%	0.7419	0.5929
50%	1.01	1.12	1.04M	4.0%	0.7396	0.5947
Full (8)	8	8	10.33M	40.2%	0.7442	0.5910

model. Meanwhile, compared to basic stacked and parallel models (as shown in Table 3 and 4), GDCN-FDO (with 80% information ratio) enables to achieve better performance with only 1.46M model parameters.

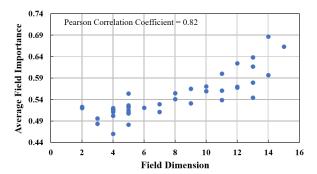


Figure 11: The visualization of the relationship between averaged field importance and field dimension.

B.3 Visualization of Relationship

In section 5.5.4, we confirm the significant positive correlation between the field dimensions and their respective importance. In Figure 11, we visualize the positive correlation between field dimensions and field importance through a scatter plot. It is clear to see that when the feature representation dimension is larger, the average importance of the feature is also more significant. In contrast, the dimension of a field is not directly related to the number of features it contains. Specifically, the Pearson Correlation Coefficient is -0.23 for these two groups of data. For example, fields {#16, #25} contain the most feature number but exhibit smaller dimensions. If an empirical formula is used to calculate feature dimensions, fields {#16, #25} will be assigned the longest dimension.

B.4 The Compatibility of the Generalization Framework

Table 8: The Compatibility of the Generalization Framework

Datasets		Criteo			Malware	
Models	AUC ↑	Logloss↓	#Params	AUC↑	Logloss ↓	#Params
FM	0.8085	0.4433	18.48M	0.7363	0.5982	16.60M
FM-FDO	0.8099	0.4418	7.52M	0.7368	0.5973	4.46M
Δ	0.0014	-0.0015	40.7%	0.0005	-0.0009	26.9%
IPNN	0.8128	0.4390	18.26M	0.7433	0.5918	17.76M
IPNN-FDO	0.8130	0.4388	7.29M	0.7436	0.5916	5.43M
Δ	0.0002	-0.0003	39.9%	0.0003	-0.0002	30.6%
FINT	0.8128	0.4390	19.05M	0.7424	0.5924	17.46M
FINT-FDO	0.8132	0.4387	8.84M	0.7430	0.5920	5.13M
Δ	0.0004	-0.0003	46.4%	0.0006	-0.0004	29.4%
AFN	0.8116	0.4401	17.78M	0.7427	0.5924	16.98M
AFN-FDO	0.8132	0.4386	7.91M	0.7429	0.5921	4.83M
Δ	0.0016	-0.0015	44.5%	0.0002	-0.0003	28.4%
CIN	0.8121	0.4398	17.93M	0.7424	0.5928	17.09M
CIN-FDO	0.8129	0.4388	6.98M	0.7430	0.5921	4.95M
Δ	0.0008	-0.0010	38.9%	0.0006	-0.0007	29.0%

Based on the proposed generalization framework, most deep CTR models can receive different feature dimensions as inputs. Table 8 lists the improvement in prediction performance and model parameters before and after applying the framework. The adopted field dimension is learned by the FDO approach with 95% kept information ratio.

Firstly, after applying the generalization framework, the prediction accuracy of the five base models significantly improves. For example, the FM and AFN in the Criteo dataset have an AUC boost of 0.0014 and 0.0016. Other models can also obtain a slight effect boost, with AUC boosts ranging from 0.0002 to 0.0006 and Logloss optimization ranging from 0.0003 to 0.0009.

Secondly, using different feature dimensions can significantly reduce the overall parameters of the underlying CTR prediction model. For the Criteo dataset, the model parameters are reduced to 39.9% to 46.4%; for the Malware dataset, the model parameters are reduced to 26.9% to 30.6%. More importantly, the performance of these models is maintained.

In summary, applying a dimensional alignment layer allows most CTR prediction models to receive embedded features of different dimensions, which reduces the overall parameters of the model while improving its predictive power.