Towards Unifying Feature Interaction Models for Click-Through Rate Prediction

Yu Kang[1], Junwei Pan[2], Jipeng Jin[1], Shudong Huang[2], Xiaofeng Gao[1] (✉), and Lei Xiao[2]

[1] Shanghai Key Laboratory of Scalable Computing and Systems,
School of Computer Science, Shanghai Jiao Tong University, China
`{jerryykang,jinjipeng}@sjtu.edu.cn`, `gao-xf@cs.sjtu.edu.cn`
[2] Tencent, Shenzhen, China
`{jonaspan,ericdhuang,shawn}@tencent.com`

Abstract. Modeling feature interactions plays a crucial role in accurately predicting Click-Through Rates (CTR) in advertising systems. To capture the intricate interaction patterns, many existing models employ matrix-factorization techniques to represent features as lower-dimensional embedding vectors, enabling the modeling of interactions as products between these embeddings. In this paper, we propose a general framework called *IPA* to systematically unify these matrix-factorization-based models. Our framework comprises three key components: the *Interaction Function*, which facilitates feature interaction; the *Layer Pooling*, which constructs higher-level interaction layers; and the *Layer Aggregator*, which combines the outputs of all interaction layers to serve as input for the subsequent classifier. We demonstrate that most existing models can be categorized within our framework by making specific choices for these three components. Through extensive experiments and a Dimensional Collapse analysis, we evaluate the performance of these choices. Furthermore, by leveraging the most powerful components within our framework, we introduce a novel model PFL that achieves competitive results compared to state-of-the-art CTR models. PFL gets significant GMV lift during online A/B test in Tencent's advertising platform, and has been deployed as the production model in several primary scenarios.

Keywords: CTR Prediction · Factorization Machine · Recommender System.

This work was supported by the National Key R&D Program of China [2024YFF0617700] and the National Natural Science Foundation of China [U23A20309, 62272302, 62372296]. The authors are grateful to the support of Yiming Liu and Xinyi Zhou for their contributions in preliminary works. They would also thank Shifeng Wen, Zijun Liu and Xun Liu for their invaluable contributions of online evaluation, and Weihua Zhu for supporting the collaboration between SJTU and Tencent upon which this work is built. X. Gao is the corresponding author.

1 Introduction

Online advertising has become a billion-dollar business nowadays, with an annual revenue of 225 billion US dollars between 2022 and 2023 (increasing 7.3% YoY) [7]. One of the core problems in this field is to deliver the right ads to the right audiences in a given context. Accurately predicting the click-through rate (CTR) is crucial to solving this problem and has attracted significant attention over the past decade [5, 15, 17, 19, 24, 25, 27–29, 31, 35, 37, 38, 40, 46, 52].

CTR prediction commonly involves the handling of multi-field categorical data [28, 47], where all features are categorical and sparse. The primary challenge lies in effectively capturing interactions between these features, which is particularly difficult due to the extreme sparsity of feature co-occurrence. To address this challenge, numerous approaches [17, 19, 24, 28, 31, 35, 37, 40] have been proposed to explicitly model feature interactions, using matrix factorization techniques or hybrid methods that combine explicit modeling with implicit method of deep neural networks (DNNs).

Explicit interaction models, particularly the 2nd-order ones, have well-defined definitions and close-form formulations. These models originated from classic Matrix Factorization (MF) [20, 34], followed by FM [5, 31], FFM [19], FwFM [28], and FmFM [37].

In contrast, models like xDeepFM [24] and DCN V2 [40] employ a parameter or weight matrix to model higher-order interactions, going beyond the 2nd-order interactions captured by the aforementioned models. However, it is worth noting that while some attempts have been made to discuss the connections between explicit interaction models [11, 21, 40], these efforts have only covered a limited number of models. As a result, the differences and relationships between various explicit interaction models remain less well-understood.

Thus, we propose a simple framework, *i.e.*, IPA, to unify existing low- and high-order explicit interaction models for systematic comparisons between them. The name of IPA corresponds to its three components: the *Interaction Function* which captures the interaction between two terms (or features), the *Layer Pooling* which constructs explicit interaction layers based on the prior layers and raw feature embeddings, and the *Layer Aggregator* which takes all layers as input, and outputs a representation for the classifier. By combining these three components within the IPA framework, we can represent various CTR models, from 2nd-order interactions models like FM [5, 31], FwFM [28], FmFM [37] to high-order interaction models like xDeepFM [24] and CIN [40].

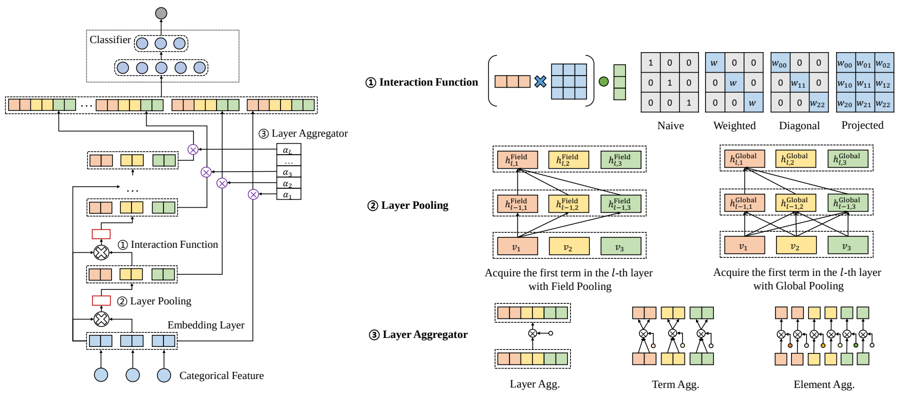
Furthermore, the IPA framework enables more granular analysis of structural differences between these CTR models in their choices of the three components, providing a better guidance for model design. For validation, we conduct extensive experiments over public datasets and production dataset, comparing between existing CTR models as well as models derived from our framework. These experiments not only identify the key factors behind model performance, supporting our derived models to be as robust as SOTA models, but also connect component choices and performance from a novel perspective: the phenomenon of Dimensional Collapse.

The main contribution of the paper can be summarized as:

– We propose a general framework IPA for feature interaction models, consisting of the Interaction Function, the Layer Pooling and the Layer Aggregator. The framework supports structural analysis between models.
– We derive novel models by combinations of IPA components. Experiment results demonstrate the effectiveness of these new models on both public datasets and online testing, providing essential findings for model designing.
– We present a novel Dimensional Collapse perspective to understand the evolution of Interaction Functions and analyze the potential to learn data orders.

2 The IPA Framework

In this section, we present the three modules in our framework: the Interaction Functions, Layer Pooling and Layer Aggregator. The structure of framework is illustrated in Fig. 1 below:



(a) The IPA framework. (b) Illustration of IPA components.

Fig. 1. Illustration of the IPA framework and common choices of its three components.

2.1 Interaction Function

Starting with a set of embedding vectors, the very first thing to determine is the way to model interactions between these embedding vectors of different feature fields. As the first IPA component, the Interaction Function extracts information from explicit interaction of these embedding vectors and returns the result in vector form. The input accepts both raw embeddings directly obtained from features and calculated interaction terms, enabling explicit feature interaction of any order.

Table 1. Formulation of Interaction Functions where diag() denotes diagonal matrix.

Type	Notation	Matrix	Example
Naive	$\boldsymbol{W}^{\mathrm{N}}$	$\boldsymbol{I}$	FM [5, 31], HOFM [6]
Weighted	$\boldsymbol{W}^{\mathrm{W}}$	$\mathrm{diag}(w, \ldots, w)$	FwFM [28], xDeepFM [24]
Diagonal	$\boldsymbol{W}^{\mathrm{D}}$	$\mathrm{diag}(w_1, \ldots, w_K)$	FvFM [37]
Projected	$\boldsymbol{W}^{\mathrm{P}}$	full matrix	FmFM [37], DCNV2 [40]

Definition 1. ***Interaction Function:*** *For two input embeddings $\boldsymbol{t}_i, \boldsymbol{t}_j \in R^K$, the Interaction Function f is defined as a function mapping $\boldsymbol{t}_i, \boldsymbol{t}_j$ to the interaction term of two these two embeddings:*

$$\boldsymbol{t}_{i,j} = f(\boldsymbol{t}_i, \boldsymbol{t}_j) \in R^K \tag{1}$$

In our framework, illustrated in the middle part of Fig. 1(a), the Interaction Function serves as the basic unit of feature interaction operations. Utilizing the interaction terms generated by the Interaction Function, the whole CTR model is able to model interaction patterns throughout the training progress.

From our observations, the Interaction Function in many existing models can be formulated by an interaction matrix $\boldsymbol{W} \in \mathcal{R}^{K \times K}$:

$$f(\boldsymbol{t}_i, \boldsymbol{t}_j, \boldsymbol{W}) = (\boldsymbol{t}_i^\top \boldsymbol{W}) \odot \boldsymbol{t}_j^\top \tag{2}$$

We find most of them taking one of the four forms of $\boldsymbol{W}_{i,j}$ in Tab. 1 and illustrated in the first part of Fig. 1(b), where blue elements are trainable.

Dimensional Collapse of Interactions Recent work [15] reveals that feature interaction on recommendation models leads to the Dimensional Collapse of embeddings, that is, the embeddings may only be able to span a low-dimensional space. Inspired by [18], which proves that the projection matrix alleviates the Dimensional Collapse in contrastive learning, we study if the projection matrix in the Interaction Functions can achieve the same goal. Refer to Sec. 3.3 for details.

2.2 Layer Pooling

Existing works [24, 35, 41, 40] tend to build layers to capture explicit high-order interactions. However, the skyrocketing number of combinations ($\binom{M}{l}$) requires interaction terms to be explored systematically with tractable complexity.

A widely-employed method of traversal is to generate interaction terms layer by layer corresponding to their order, starting from the first layer which is simply a concatenation of embeddings of all fields. The embedding $\boldsymbol{t}_n$ of field n is defined as either the embedding of the active feature for a one-hot encoding field, or a pooling over embeddings for all active features for a multi-hot field. Formally,

$$\boldsymbol{h}_1 = [\boldsymbol{t}_1, \ldots, \boldsymbol{t}_n, \ldots, \boldsymbol{t}_M] \tag{3}$$

Definition 2. ***Layer Pooling:*** *For the raw embeddings $h_1 = [\boldsymbol{t_1, ..., t_M}]$ and the layer of all $(l-1)$-order terms $h_{(l-1)} = [\boldsymbol{t_{l-1,1}, ..., t_{l-1,M}}]$, then the Layer Pooling is defined as the way to generate interaction terms of order l from all interactions between terms of h_1 and $h_{(l-1)}$, that is:*

$$\boldsymbol{t}_{l,n} = \sum\nolimits_{m=1}^{M} \sum\nolimits_{n=1}^{M} \alpha_{m,n} f(\boldsymbol{t}_n, \boldsymbol{t}_{l-1,m}) \tag{4}$$

$\alpha_{m,n}$ are either 0 or 1 indicating presence of corresponding terms.

Illustrated in the middle bottom part of Fig. 1(a), to capture higher-order interactions, an IPA model relies on its Layer Pooling to systematically generate and combine new interaction terms, forming interaction layers of higher order. Furthermore, to control the number of interactions, two widely employed methods build layers with a fixed number of terms, where each term is a pooling of interactions between terms from prior layers and raw feature embeddings.

Field-wise Layer Pooling (Field Pooling) This Field Pooling will pool all the interactions that correspond to a specific field. Specifically, the l-th layer consists of M terms, with the n-th term $\boldsymbol{t}_{l,n}$ defined as a pooling over all interactions between the n-th term in the first layer, *i.e.*, the embedding of the n-th field $\boldsymbol{t}_n$, and all terms in the $(l-1)$-th layer, *i.e.*, $\boldsymbol{t}_{l-1,m}$.

$$\boldsymbol{h}_l^{\mathrm{F}} = [\boldsymbol{t}_{l,1}^{\mathrm{F}}, \ldots, \boldsymbol{t}_{l,n}^{\mathrm{F}}, \ldots, \boldsymbol{t}_{l,M}^{\mathrm{F}}], \quad \boldsymbol{t}_{l,n}^{\mathrm{Field}} \quad = \sum\nolimits_{m=1}^{M} f(\boldsymbol{t}_n, \boldsymbol{t}_{l-1,m}^{\mathrm{F}}, \boldsymbol{W}) \tag{5}$$

DCNV2 employs Field Pooling and Projected Product to build up layers, *i.e.*, $\boldsymbol{h}_{l,n}^{\mathrm{CrossNet}} = \sum_{m=1}^{M} f(\boldsymbol{t}_n, \boldsymbol{t}_{l-1,m}, \boldsymbol{W}_{l,n,m}^{F})$.

Global-wise Layer Pooling (Global Pooling) This Field Pooling will globally pool all the interactions, regardless of fields. Specifically, the l-th layer consists of H_l terms, with the n-th term defined as a pooling over all interactions between all terms in the first layer, and all terms in the $(l-1)$-th layer.

$$\boldsymbol{h}_l^{\mathrm{G}} = [\boldsymbol{t}_{l,1}^{\mathrm{G}}, \ldots, \boldsymbol{t}_{l,n}^{\mathrm{G}}, \ldots, \boldsymbol{t}_{l,H_l}^{\mathrm{G}}], \boldsymbol{t}_{l,n}^{\mathrm{G}} \quad = \sum\nolimits_{n=1}^{M} \sum\nolimits_{m=1}^{H_{l-1}} f(\boldsymbol{t}_n, \boldsymbol{t}_{l-1,m}^{\mathrm{G}}, \boldsymbol{W}) \tag{6}$$

Global Pooling introduces symmetric interactions in each layer, thus more redundant than Field Pooling. xDeepFM [24] employs AGT to construct layers, *i.e.*, $\boldsymbol{h}_{l,n}^{\mathrm{CIN}} = \sum_{n=1}^{M} \sum_{m=1}^{M} f(\boldsymbol{t}_n, \boldsymbol{t}_{l-1,m}, \boldsymbol{W}_{l,n,m}^{F})$. The second part of Fig. 1(b) illustrates the details on acquiring terms in the l-th layer with both pooling.

2.3 Layer Aggregator

After constructing layers, complexity of input features requires model to learn from interaction of all orders, combining terms from these layers into vector from for following classifiers. So the last component of our framework is defined:

Definition 3. ***Layer Aggregator:*** *Considering* $[h_1, ..., h_L]$ *as interaction layers of model, then the Layer Aggregator LA is defined as a function aggregating elements of layers into one output vector as input of classifier, that is:*

$$\boldsymbol{r} = LA([\boldsymbol{h}_1, ..., \boldsymbol{h}_L]) \tag{7}$$

Illustrated in the middle-top part of Fig. 1(a), the Layer Aggregator component takes the interaction layers as input, integrates terms within layers and then aggregates all the layers to form the final output of feature interaction module, serving as the input of following classifier in the model.

Here are some widely employed ways to aggregate layers:

- **Direct Agg.**: Directly link each layer, with No weights at all.
- **Layer Agg.**: Assign a Layer-wise weight α_l for each layer.
- **Term Agg.**: Assign a Term-wise weight for each term in each layer, *e.g.*, assign $\alpha_{l,n}$ for the n-th term of the l-th layer.
- **Element Agg.**: Assign an Element-wise weight for elements in each layer, *e.g.*, assign $\alpha_{l,n,k}$ for the k-th element in the n-th term of the l-th layer.

The third part of Fig. 1(b) illustrates how to assign weights to a term with Layer, Term and Element Agg., respectively. Tab. 2 summarizes the formulation and number of learnable parameters of different Layer Aggregators.

Table 2. Formulation of aggregators, where $\|\{\cdot\}$ denotes the concatenation function.

Abbr.	Weight	# Param.	Output
Direct	1	0	$\boldsymbol{r} = \sum_{l=1}^{L} \|_{n=1}^{M}\{\boldsymbol{t}_{l,n}\}$
Layer	α_l	L	$\boldsymbol{r} = \sum_{l=1}^{L} \alpha_l \cdot \|_{n=1}^{M}\{\boldsymbol{t}_{l,n}\}$
Term	$\alpha_{l,n}$	LM	$\boldsymbol{r} = \sum_{l=1}^{L} \|_{n=1}^{M}\{\alpha_{l,n} \cdot \boldsymbol{t}_{l,n}\}$
Element	$\alpha_{l,n,k}$	LMK	$\boldsymbol{r} = \sum_{l=1}^{L} \|_{n=1}^{M}\{\|_{k=1}^{K}\{\alpha_{l,n,k} \cdot \boldsymbol{t}_{l,n,k}\}\}$

2.4 The Framework

Now, combining the three components above, we propose our framework **IPA** as a modularized transformation:

- First of all, IPA reads in the embedding vectors as the 0-th layer and prepares the Interaction Function.
- Then, IPA constructs the interaction layers using its Interaction Function on the built layers guided by its Layer Pooling.
- Finally, IPA aggregates all the layers by its Layer Aggregator, creating output vector of desired length for following classifiers.

Generalizing the structure of feature interaction module by the three components, the IPA framework dives into the details of existing CTR models, exploring their similarity and differences on various aspects. We can compare how models make basic interactions, how they build up high-order interaction terms, and how they extract information from interaction layers. Besides, the framework can help us control variables when analyzing the effect of one component (like Interaction Function), by simply fixing the choices of other components.

2.5 Deriving New Models

While many existing CTR models can fit into IPA as a combination of three components, there are several new models that can be derived from unexplored combinations.

Specifically, we choose Projected Product to capture interactions between terms, employ Field Pooling to build up layers and employ Layer Agg. to assign layer-wise weight during layer aggregation.

We name it as PFL, which stands for Projected Product with Field Pooling and Layer Agg.. Comparing PFL to DCN V2, there are two main differences between them:

1. PFL employs Field Pooling without residual connections.
2. PFL employs Layer Agg. to aggregate layers instead of Direct Agg..

There are many other new models that can be derived in this way, and we present some of the most effective ones in Tab. 3.

3 Evaluation

In this section, we aim to answer several research questions related to the performance of components in the IPA framework.

- RQ1: How do the Interaction Functions perform? Does a more complicated Interaction Function lead to better performance in the real-world scenario?
- RQ2: Regarding Dimensional Collapse, how does Interaction Function influence the extent of collapse?
- RQ3: How do the various Layer Poolings perform? What can we infer from them about model design?
- RQ4: How do the various Layer Aggregators perform? What can we learn from choices of this component?
- RQ5: How do existing CTR models evolve so far? How does our framework offer a direction towards a better model?
- RQ6: Regarding the derived model PFL, to what extent has this model performed in industrial applications?

3.1 Experiment Setup

Public Dataset We evaluate the models on two public datasets: Criteo-x1 and Avazu. Both are divided into 8:1:1 for training, validation, and test set.

- Criteo-x1 [2]. It is the most popular benchmark dataset for CTR prediction, consisting of 45 million click feedback records of display ads. The dataset includes 13 numerical feature fields and 26 categorical feature fields.
- Avazu [1]. It includes 10 days of Avazu click-through data, with 13 feature fields that have disclosed meanings and 9 anonymous categorical fields.

Synthetic Dataset We follow [40] to generate data with specified interaction orders. Let $\boldsymbol{x} = [x_1, \ldots, x_n]$ be a feature vector of length n, and the element $x_i \in \mathcal{R}$ denotes the i-th feature of $\boldsymbol{x}$. x_i is uniformly sampled from interval $[-1, 1]$. The monomial $x_1 \cdots x_O$ is called O-order cross-term, representing an O-order interaction between features. Given $\boldsymbol{x}$ of length n and the data order O, we generate a label as the sum of the cross-terms whose orders $\leq O$, each with an individual weight. Specifically, we define $\boldsymbol{I}_n = \{i | i \leq n, i \in \mathbf{N}^*\}$ as the set of all positive integers $\leq n$, and Ω_n^O as the set of all combinations of O distinct integers randomly sampled from $\boldsymbol{I}_n$.

The label y is generated by the equation

$$y = \sum_{i=1}^{O} \sum_{(\omega_1, \cdots, \omega_i) \in \Omega_n^i} w_{\omega_1, \cdots, \omega_i} \prod_{j=1}^{i} x_{\omega_j} + \epsilon \tag{8}$$

where ω_i is the i-th element in the combination, $w_{\omega_1, \cdots, \omega_i} \sim N(0, 1)$ is the individual weight of a specific interaction term, and $\epsilon \sim N(0, 0.1)$ models the label noise which is ignored in [40]. We define O, the order of a dataset, as hyperparameter controlling the maximum order of cross-terms.

Baseline Models We choose the following CTR models as baselines: FM, FwFM, FvFM, FmFM, DeepFM, FiBiNet, HOFM, xDeepFM and DCN V2. Most of them fit in our IPA framework.

Implementation Details Our implementation is based on a public PyTorch library for recommendation models[3]. We set the embedding size to 16, the dropout rate to 0.2, and launch early-stopping. We use the Adam optimizer with a learning rate of 0.001. The number of layers L of high-order models are set to 4 and 5 for Criteo-x1 and Avazu, based on the best performance. Our derived models using PGT have 10 terms in each layer. All models are trained on a NVIDIA V100 GPU with a batch size of 2048. We repeat all experiments 3 times and report the average AUC performance in Tab. 3. The best performance and the comparable performance are denoted in **bold** and <u>underlined</u> fonts, respectively.

[3] https://github.com/rixwew/pytorch-fm

Table 3. AUC of baselines and derived models on public datasets. O stands for orders.

Model	Interaction Function	Layer Pooling	Layer Aggregator	Criteo		Avazu	
				L	AUC	O	AUC
FM	naive	field/global	direct	2	0.8009(2e-4)	2	0.7758(1e-4)
DeepFM	naive	field/global	direct	2	0.8122(1e-4)	2	0.7899(5e-4)
HOFM	naive	field/global	direct	4	0.8040(3e-4)	5	0.7781(8e-4)
FwFM	weighted	field/global	direct	2	0.8095(2e-4)	2	0.7854(4e-4)
xDeepFM	weighted	global	term	4	0.8119(2e-4)	5	0.7897(7e-4)
FvFM	diagonal	field/global	direct	2	0.8103(2e-4)	2	0.7870(3e-4)
FmFM	projected	field/global	direct	2	0.8115(3e-4)	2	0.7882(5e-4)
FiBiNet	projected	field/global	direct	2	0.8113(2e-4)	2	0.7907(4e-4)
DCN V2	projected	field’	direct	4	0.8137(3e-4)	5	**0.7917(1e-4)**
WFL	weighted	field	layer	4	0.8124(2e-4)	5	0.7891(3e-4)
DFL	diagonal	field	layer	4	0.8123(1e-4)	5	0.7903(9e-4)
PFL	projected	field	layer	4	**0.8138(3e-4)**	5	0.7916(4e-4)
PFT	projected	field	term	4	**0.8138(3e-4)**	5	0.7904(4e-4)
PFE	projected	field	element	4	**0.8138(3e-4)**	5	0.7907(2e-4)
PFD	projected	field	direct	4	0.8131(4e-4)	5	0.7912(5e-4)

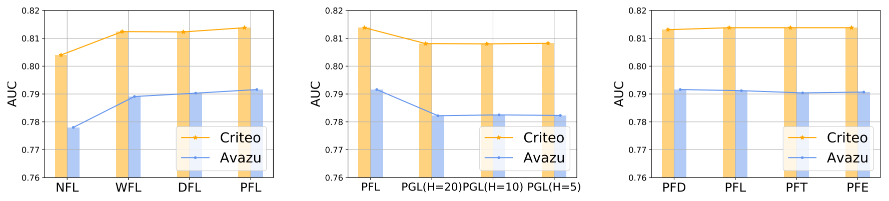
3.2 RQ1: Evaluation of Interaction Function

We compare the models with various Interaction Functions in several settings regarding the Layer Pooling, Layer Aggregator and classifier. First, among the simplest 2nd-order explicit interaction models (Field/Global Pooling, direct aggregator, no classifier), *i.e.*, FM, FwFM, FvFM and FmFM, the more complicated the Interaction Function, the better AUC on both datasets.

Then, we compare the derived IPA models using the same Layer Pooling (*i.e.*, Field Pooling) and Layer Aggregator (*i.e.*, Layer Agg.). The result is shown in Fig. 2(a). Across all models (*i.e.*, NFL, WFL, DFL and PFL), PFL achieves the best AUC, with a strong trend: models with more complex Interaction Functions (Projected $>$ Diagonal $>$ Weighted $>$ Naive) generally achieve better AUC. This indicates the importance of Interaction Function to model performance.

Finding 1. Under the same setting of Layer Pooling, aggregation, and classifier, the more complicated the projection matrix (i.e., from identity, scaled identity, diagonal to full matrix) within the feature Interaction Function, the better the results regarding AUC.

One possible reason of such trend is that Projected Product learns a powerful matrix projection, *i.e.*, $\boldsymbol{W}^F$ for each field pair. In addition, the Projected Product has the highest number of trainable parameters, further improving its capability to fit training data, which connects closely with model performance.



(a) Interaction Functions. (b) Layer Poolings. (c) Layer Aggregators.

Fig. 2. Performance of various choices within each component in the IPA framework.

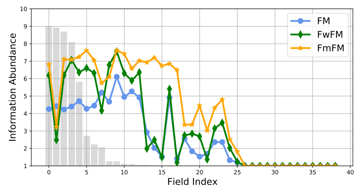
3.3 RQ2: Dimensional Collapse of Interaction Functions

To explore the connection between Dimensional Collapse and Interaction Functions, we analyze the singular value sums (SS) as well as the information abundance (IA) [15] of feature fields, where the latter one is defined as the singular value sum divided by the largest value. Different from [15], we evaluate these metrics on embeddings in the sample distribution rather than the unique feature ID distribution, as they reflects feature frequencies more accurately. All the following analysis is based on the Criteo-x1 dataset, since its features vary greatly in cardinality to better illustrate the Dimensional Collapse phenomenon.

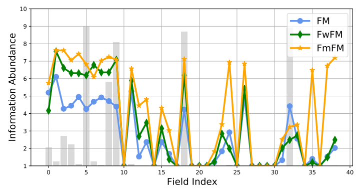
We run experiments on DCN V2 and three representational models: FM with Naive Product, FwFM with Weighted Product and FmFM with Projected Product, whose only difference lies in the Interaction Functions they take. In Fig. 3(a) and 3(c), we present the singular value sums of sample embeddings in the Criteo-x1 dataset, where feature fields are ordered by their cardinality and average pair importance from the FwFM model. Comparing the three representational models, we could observe that when the projection matrix of Interaction Function becomes more complex (FM's identity matrix to FwFM's scaled identity matrix to FmFM's full matrix), the singular values of the leftmost high-dimensional fields become higher and more balanced, obtaining larger SS and larger IA; this indicates that FmFM spans the largest space in its high-dimensional fields and thus experiencing less amount of Dimensional Collapse on these fields. In this aspect, a complex Interaction Function could serve as a buffer for the high-dimensional fields, alleviating Dimensional Collapse occurred during the interaction process with other low-dimensional fields.

Now, we want to connect this observation to the model performances. In Fig. 3(b) and 3(d), we derive field weights from the FwFM model and present the SSs as well as IAs ordered by these weights. Here FmFM still obtains the largest SS among the 10 most essential fields for prediction task, suggesting that models with a complex Interaction Function indeed learn more robust embeddings against Dimensional Collapse, which contributes to a better model performance.

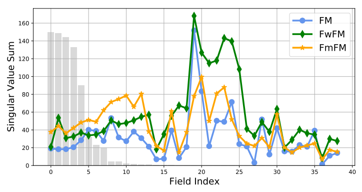
To further analyze the robustness of embeddings, we pick the representative fields of high cardinality and field importance, plotting their ordered singular



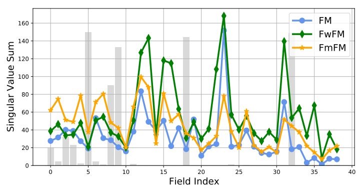
(a) IA ordered by cardinality



(b) IA ordered by importance



(c) SS ordered by cardinality



(d) SS ordered by importance

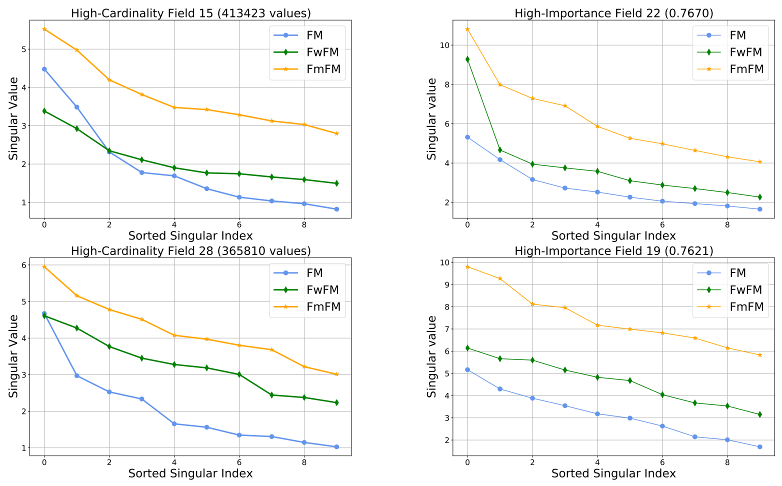
Fig. 3. Comparison of Dimensional Collapse for 2nd-order interaction models on the Criteo-x1 dataset. Both Singular Sum (SS) and Information Abundance (IA) are plotted for fields aligned in two ways. While fields in (a) and (c) are ordered by field cardinality, those in (b) and (d) are ordered by average feature pair importance in FwFM model.

values in Fig. 4. From the figure, FmFM possesses the highest and most steady level of singular values, for both high-dimensional and high-importance fields. This indicates that models with a more complex Interaction Function suffers less from Dimensional Collapse and learns a more robust set of embeddings.

Finding 2. Among 2nd-order interaction models, the more complicated projection matrix (i.e., from identity, scaled identity, diagonal to full matrix) within the Interaction Function, the more robust (less collapsed) the learned embeddings regarding both information abundance and singular value sum.

3.4 RQ3: Evaluation of Layer Pooling

To evaluate Layer Pooling, we compare the models with various Layer Poolings, but the same Interaction Function (*i.e.*, Weighted, Diagonal and Projected Product) and the same Layer Aggregator (*i.e.*, Layer Agg.) in Tab. 3. In addition, we tune the hyper-parameter H, *i.e.*, number of terms in each layer in global agg., by training the PGL model with $H = 5, 10, 20$, and present the results in Fig. 2(b). Field Pooling constantly outperforms Global Pooling in all comparisons, possibly due to that Global Pooling introduces too many redundant interactions, leading to optimization issue caused by co-linearity.



(a) Spectrum of high-cardinality fields (b) Spectrum of high-importance fields

Fig. 4. Field-wise singular value spectrum for 2nd-order interaction models on the Criteo-x1 dataset. The singular values of representative fields (high-order fields in (a) and high-importance fields in (b)) are ordered and displayed.

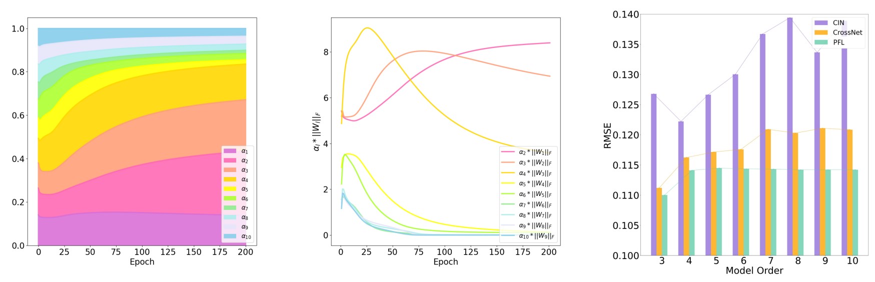
Finding 3. Under the same setting of Interaction Function and Layer Aggregator, models with Field Pooling outperform those with Global Pooling.

3.5 RQ4: Evaluation of Layer Aggregator

To compare the Layer Aggregator choices, we equip PFL with each of the four variants. Evaluations in Fig. 2(c) show comparable performance for these choices, and PFL itself gets comparable performance with DCN V2 in Tab. 3 with a fixed number of layers, *i.e.*, $L = 4$ in Criteo-x1 and $L = 5$ in Avazu. However, with fixed orders of public datasets, the flexibility of models remains unrevealed. A good model should effectively capture the intrinsic order of data and value interactions within such order to avoid overfitting, as Layer Aggregator does.

To explore such flexibility, we generate a synthetic dataset containing 1 million samples with $O = 4$, and then train a PFL model with $L = 10$. Please note that α_l in PFL used to explicitly model the importance of layer l might be absorbed by the order-wise $\boldsymbol{W}_l \in \mathcal{R}^{MK\times MK}$. Therefore, we analyze the effect of α_l and its joint influence with $\boldsymbol{W}_l$. We present their Frobenius norm, i.e., $\|\boldsymbol{W}_l\|_\mathrm{F}$ and $\alpha_l \cdot \|\boldsymbol{W}_{l-1}\|_\mathrm{F}$ in Fig. 5(a) and Fig. 5(b), respectively.

From the figures, we observe that through the training epochs, $\alpha_1 \sim \alpha_4$ corresponding to weights of the first four explicit interaction layers increase significantly, while weights of higher orders, *i.e.*, $\alpha_5 \sim \alpha_{10}$, decrease. Besides, when considering $\alpha_l \cdot \|\boldsymbol{W}_{l-1}\|_\mathrm{F}$ as a whole, the contribution of $\alpha_l \cdot \|\boldsymbol{W}_{l-1}\|_\mathrm{F}$ is small



(a) layer weights α_l (b) $\alpha_l \cdot \|\boldsymbol{W}_{l-1}\|_\mathrm{F}$ (c) RMSEs on O $= 3$

Fig. 5. Trends of α_l, $\alpha_l \cdot \|\boldsymbol{W}_{l-1}\|_\mathrm{F}$ and model performances in the training process. Our model learns low α_l and $\|\boldsymbol{W}_{l-1}\|_\mathrm{F}$ for extra layers (5-10), obtaining high-level and robust performance even when over-estimating data order.

and can be ignored when $l > 4$, indicating that PFL mainly learns from the most important layers and is capable of learning the order of the data.

Then, to establish connection of such capability to the choice of Layer Aggregator, we generated synthetic dataset of order $O = 3$, training CIN (xDeepFM), CrossNet (DCN V2) and PFL with layers in the range of $[O, 10]$. To make a clear analysis, we remove the MLP classifier from all models and only use the representation for prediction. The RMSE results are shown in Fig. 5(c). We do not include models with $l < 3$ as their performances are defective, which is trivial.

Regarding the three models, CIN performs badly in all settings: on the one hand, it performs the worst in all cases with a large margin; on the other hand, it cannot always achieve the best performance when the model order matches the data order. For example, while the data order is 3, CIN achieves the best performance with a model order of 4. Then, compared to CrossNet, our derived model PFL achieves the best performance when two orders match, and it manages to make a larger relative lift as more redundant layers are added to the model, while performance of CrossNet deteriorates. As PFL differs from DCN V2 mainly in its introduction of α_l, the performance gap indicates the success of layer-wise aggregator in filtering redundant layers in PFL.

Finding 4. Under the same setting of feature Interaction Function and Layer Pooling, models with layer-wise aggregator are capable of learning the order of data, thus outperforming those with direct connections.

3.6 RQ5: Derived models and Evolution of Existing Models

Based on the above evaluations, the Projected Product, Field Pooling and Layer Agg. outperform other choices for the three components. Combining these most powerful choices together, we get the derived model PFL achieving comparative performance with DCN V2 on both public datasets. Besides, many other derived models obtain decent performance. For example, WFL achieves comparative performance with xDeepFM, differing mainly in the Layer Pooling.

In a nutshell, through all these evaluations, we observe that CTR models evolve mainly through: 1) Employing more powerful Interaction Functions, from Naive Product (*e.g.*, FM, HOFM, DeepFM), to Weighted Product (*e.g.*, FwFM, PNN, xDeepFM) and Diagonal Product (*e.g.*, FvFM), and finally to Projected Product (*e.g.*, FmFM, FiBiNet, DCN V2 and PFL). 2) Employing more powerful Layer Poolings, from Global Pooling (*e.g.*, xDeepFM, WGL, DGL, PGL) to Field Pooling (DCN V2 and PFL). 3) Employing layer-wise Layer Aggregator instead of directly linking everything, in order to adapt to various orders of data.

3.7 RQ6: Online A/B Testing

We developed PFL in one of the world's largest advertising platforms. The production model employs Heterogeneous Experts with Multi-Embedding architecture [15, 29, 36]. We replace the IPNN expert in the production model with the PFL expert, which models the interactions between more than five hundred user-side, ad-side, and context-side features. Multiple embedding tables are learned for all features, each corresponding to one or several experts.

During the two-week 20% A/B testing, PFL demonstrated promising results, achieving 0.9%, 3.7%, 1.2%, and 2.7% GMV lift on several vital scenarios, including Moments pCTR, Content and Platform pCTR, and DSP pCTR. These improvements were statistically significant according to t-tests. PFL has been successfully deployed as the production model in the above-mentioned scenarios, leading to a revenue lift by hundreds of millions of dollars per year.

We also study the singular values of the baseline IPNN model and the PFL model. Specifically, we calculate a 95%-percentile dimension, that is, how many top singular values can cover 95% of the total singular values of each feature. As shown in Fig. 6, our proposed PFL gets a much higher 95%-percentile dimension on almost all features, especially on those with high cardinalities. This validates that PFL can mitigate the Dimensional Collapse.

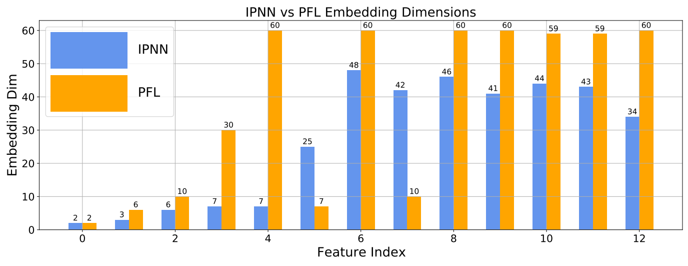


Fig. 6. Dimensional Collapse of IPNN v.s. PFL in our system.

4 Related Works

Many articles model the 2nd-order interactions after factorization machines. Early works use Logistic Regression (LR) [9, 26, 33] or Polynomial-2 (Poly2) [8] to learn interactions. Following matrix factorization [13, 20, 34], FM [5, 31] models interactions as dot product between embeddings. FFM [19] assigns field-wise embeddings for field-pair interactions. Based on FM, FwFM [28], FvFM [37] and FmFM [37] introduce field-pair wise weight, vector and matrix to better model interactions respectively. AFM [43] learns an attentive weight for each field pair.

Many recent articles propose to capture explicit high-order interactions [6, 22, 24, 25, 31, 35, 39–41, 52]. For example, [31] discusses a d-way FM to model d-order interactions in FM, and HOFM [6] presents an efficient algorithm to train it. Similarly, xDeepFM [24], DCN [41], DCN V2 [40] and DCN V3 [22] also employ variants of matrix factorization techniques to model high-order interactions, while AutoInt [35] resorts to a multi-head self-attentive network. Additionally, EulerNet [39] learns feature interactions in a complex vector space. MaskNet [42] introduces multiplicative operations block by block. FINAL [52] stacks multiple FINAL blocks to capture various interaction patterns. DCN V3 [22] uses subnetworks LCN and ECN to capture both low-order and high-order interactions.

Besides, it is also common to capture interactions both explicitly and implicitly (via deep neural networks). Wide & Deep [10], DeepFM [14] and ONN [45] combine an explicit interaction module (as the wide part) and multiple feed-forward layers (as the deep part) in parallel. NFM [16], IPNN [30], OPNN [30] and FiBiNet [17] employ an explicit interaction component and then add MLPs over it. FinalMLP [25] relies on MLPs and a two-stream structure, achieving amazing performance. Also, there has been long-standing discussion and debate on the role of MLPs in recommendations [3, 4, 12, 25, 32, 40, 50]. There are also works to benchmark and summarize existing CTR models [21, 23, 40, 44, 48, 49, 51, 53]. For example, AOANet [21] decomposes the models into projection, interaction, and fusion. DCN V2 [40] compares the explicit Interaction Functions.

5 Conclusion

In this paper, we present a unified framework, *i.e.*, IPA, for explicit interaction click-through rate models to systematically analyze and compare the existing CTR models in Recommender Systems. Further, we conduct extensive experiments on the effect of framework components and make several interesting discoveries about model design. Inspired by these analysis, we successfully derived a new model that achieves competitive performance with SOTA models and was successfully deployed to Tencent's advertising platform.

References

1. Avazu dataset. https://www.kaggle.com/competitions/avazu-ctr-prediction (2014)
2. Criteo dataset. https://www.kaggle.com/c/criteo-display-ad-challenge (2014)

3. Anelli, V.W., Bellogín, A., et al.: Reenvisioning the comparison between neural collaborative filtering and matrix factorization. In: RecSys. pp. 521–529 (2021)
4. Beutel, A., Covington, P., et al.: Latent cross: Making use of context in recurrent recommender systems. In: WSDM. pp. 46–54 (2018)
5. Blondel, M., Fujino, A., et al.: Convex factorization machines. In: ECML-PKDD. vol. 9285, pp. 19–35 (2015)
6. Blondel, M., Fujino, A., et al.: Higher-order factorization machines. NeurIPS pp. 3351–3359 (2016)
7. Bureau, I.A.: Iab/pwc internet advertising revenue report 2024 (2024), https://www.iab.com/insights/internet-advertising-revenue-report-2024/
8. Chang, Y.W., et al.: Training and testing low-degree polynomial data mappings via linear svm. Journal of machine learning research **11**(Apr), 1471–1490 (2010)
9. Chapelle, O., et al.: Simple and scalable response prediction for display advertising. TIST **5**(4), 61 (2015)
10. Cheng, H.T., Koc, L., et al.: Wide & deep learning for recommender systems. In: The 1st workshop on deep learning for recommender systems. pp. 7–10 (2016)
11. Cheng, Y., Xue, Y.: Looking at ctr prediction again: Is attention all you need? In: SIGIR. pp. 1279–1287 (2021)
12. Feng, N., Pan, J., et al.: Long-sequence recommendation models need decoupled embeddings. arXiv preprint arXiv:2410.02604 (2024)
13. Flanagan, A., Oyomno, W., et al.: Federated multi-view matrix factorization for personalized recommendations. In: ECML-PKDD. vol. 12458, pp. 324–347 (2020)
14. Guo, H., Tang, R., et al.: Deepfm: a factorization-machine based neural network for ctr prediction. In: IJCAI. pp. 1725–1731 (2017)
15. Guo, X., Pan, J., et al.: On the embedding collapse when scaling up recommendation models. arXiv preprint arXiv:2310.04400 (2023)
16. He, X., Chua, T.S.: Neural factorization machines for sparse predictive analytics. In: SIGIR. pp. 355–364 (2017)
17. Huang, T., Zhang, Z., et al.: Fibinet: combining feature importance and bilinear feature interaction for click-through rate prediction. In: RecSys. pp. 169–177 (2019)
18. Jing, L., Vincent, P., et al.: Understanding dimensional collapse in contrastive self-supervised learning. In: ICLR (2022)
19. Juan, Y., Zhuang, Y., et al.: Field-aware factorization machines for ctr prediction. In: RecSys. pp. 43–50 (2016)
20. Koren, Y., Bell, R., et al.: Matrix factorization techniques for recommender systems. Computer **42**(8), 30–37 (2009)
21. Lang, L., Zhu, Z., et al.: Architecture and operation adaptive network for online recommendations. In: SIGKDD. pp. 3139–3149 (2021)
22. Li, H., Zhang, Y., et al.: Dcnv3: Towards next generation deep cross network for ctr prediction. arXiv preprint arXiv:2407.13349 (2024)
23. Li, J.L., et al.: Decompose, then reconstruct: A framework of network structures for click-through rate prediction. In: ECML-PKDD. vol. 14169, pp. 422–437 (2023)
24. Lian, J., Zhou, X., et al.: xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In: SIGKDD. pp. 1754–1763 (2018)
25. Mao, K., Zhu, J., et al.: Finalmlp: An enhanced two-stream mlp model for ctr prediction. In: AAAI. pp. 4552–4560 (2023)
26. McMahan, H.B., Holt, G., et al.: Ad click prediction: a view from the trenches. In: SIGKDD. pp. 1222–1230 (2013)
27. Naumov, M., Mudigere, D., et al.: Deep learning recommendation model for personalization and recommendation systems. arXiv preprint arXiv:1906.00091 (2019)

28. Pan, J., Xu, J., et al.: Field-weighted factorization machines for click-through rate prediction in display advertising. In: WWW. pp. 1349–1357 (2018)
29. Pan, J., Xue, W., et al.: Ad recommendation in a collapsed and entangled world. In: SIGKDD. pp. 5566–5577 (2024)
30. Qu, Y., Cai, H., et al.: Product-based neural networks for user response prediction. In: ICDM. pp. 1149–1154 (2016)
31. Rendle, S.: Factorization machines. In: ICDM. pp. 995–1000 (2010)
32. Rendle, S., Krichene, W., et al.: Neural collaborative filtering vs. matrix factorization revisited. In: RecSys. pp. 240–248 (2020)
33. Richardson, M., Dominowska, E., et al.: Predicting clicks: estimating the click-through rate for new ads. In: WWW. pp. 521–530 (2007)
34. Salakhutdinov, R., Mnih, A.: Bayesian probabilistic matrix factorization using markov chain monte carlo. In: ICML. pp. 880–887 (2008)
35. Song, W., Shi, C., et al.: Autoint: Automatic feature interaction learning via self-attentive neural networks. In: CIKM. pp. 1161–1170 (2019)
36. Su, L., Pan, J., et al.: Stem: Unleashing the power of embeddings for multi-task recommendation. In: AAAI. pp. 9002–9010 (2024)
37. Sun, Y., Pan, J., et al.: Fm2: Field-matrixed factorization machines for recommender systems. In: WWW. pp. 2828–2837 (2021)
38. Tang, X., Qiao, Y., et al.: Optmsm: Optimizing multi-scenario modeling for click-through rate prediction. In: ECML-PKDD. vol. 14174, pp. 567–584 (2023)
39. Tian, Z., Bai, T., et al.: Eulernet: adaptive feature interaction learning via euler's formula for ctr prediction. In: SIGIR. pp. 1376–1385 (2023)
40. Wang, R., et al.: Dcn-v2: Improved deep & cross network and practical lessons for web-scale learning to rank systems. In: WWW. pp. 1785–1797 (2021)
41. Wang, R., Fu, B., et al.: Deep & cross network for ad click predictions. In: ADKDD. pp. 1–7 (2017)
42. Wang, Z., She, Q., et al.: Masknet: Introducing feature-wise multiplication to ctr ranking models by instance-guided mask. arXiv preprint arXiv:2102.07619 (2021)
43. Xiao, J., Ye, H., et al.: Attentional factorization machines: Learning the weight of feature interactions via attention networks. arXiv preprint arXiv:1708.04617 (2017)
44. Xu, L., Tian, Z., et al.: Towards a more user-friendly and easy-to-use benchmark library for recommender systems. In: SIGIR. pp. 2837–2847. (2023)
45. Yang, Y., Xu, B., et al.: Operation-aware neural networks for user response prediction. Neural networks **121**, 161–168 (2020)
46. Yue, Yunand Liu, Y., et al.: Adaptive optimizers with sparse group lasso for neural networks in ctr prediction. In: ECML-PKDD. vol. 12977, pp. 314–329 (2021)
47. Zhang, W., Du, T., et al.: Deep learning over multi-field categorical data: A case study on user response prediction. In: ECIR. pp. 45–57 (2016)
48. Zhao, W.X., Hou, Y., et al.: Recbole 2.0: Towards a more up-to-date recommendation library. In: CIKM. pp. 4722–4726 (2022)
49. Zhao, W.X., Mu, S., et al.: Recbole: Towards a unified, comprehensive and efficient framework for recommendation algorithms. In: CIKM. pp. 4653–4664 (2021)
50. Zhou, H., et al.: Temporal interest network for user response prediction. In: WWW. pp. 413–422 (2024)
51. Zhu, J., Dai, Q., et al.: Bars: Towards open benchmarking for recommender systems. In: SIGIR. pp. 2912–2923 (2022)
52. Zhu, J., Jia, Q., et al.: Final: Factorized interaction layer for ctr prediction. In: SIGIR. pp. 2006–2010 (2023)
53. Zhu, J., Liu, J., et al.: Fuxictr: An open benchmark for click-through rate prediction. arXiv preprint arXiv:2009.05794 (2020)