Directed Acyclic Graph Factorization Machines for CTR Prediction via Knowledge Distillation

Zhen Tian
Gaoling School of Artificial Intelligence, Renmin University of China
Beijing, China
chenyuwuxinn@gamil.com

Ting Bai*‡
Beijing University of Posts and Telecommunications
Beijing, China
baiting@bupt.edu.cn

Zibin Zhang
Zhiyuan Xu
Weixin Open Platform, Tencent
Guangzhou, China
bingoozhang@tecent.com
zhiyuanxu@tecent.com

Kangyi Lin
Weixin Open Platform, Tencent
Guangzhou, China
plancklin@tecent.com

Ji-Rong Wen†
Gaoling School of Artificial Intelligence, Renmin University of China
Beijing, China
jrwen@ruc.edu.cn

Wayne Xin Zhao†
Gaoling School of Artificial Intelligence, Renmin University of China
Beijing, China
batmanfly@gmail.com

ABSTRACT

With the growth of high-dimensional sparse data in web-scale recommender systems, the computational cost to learn high-order feature interaction in CTR prediction task largely increases, which limits the use of high-order interaction models in real industrial applications. Some recent knowledge distillation based methods transfer knowledge from complex teacher models to shallow student models for accelerating the online model inference. However, they suffer from the degradation of model accuracy in knowledge distillation process. It is challenging to balance the efficiency and effectiveness of the shallow student models. To address this problem, we propose a **D**irected **A**cyclic **G**raph **F**actorization **M**achine (**KD-DAGFM**) to learn the high-order feature interactions from existing complex interaction models for CTR prediction via **K**nowledge **D**istillation. The proposed lightweight student model **DAGFM** can learn arbitrary explicit feature interactions from teacher networks, which achieves approximately lossless performance and is proved by a dynamic programming algorithm. Besides, an improved general model KD-DAGFM+ is shown to be effective in distilling both explicit and implicit feature interactions from any complex teacher model. Extensive experiments are conducted on four real-world datasets, including a large-scale industrial dataset from WeChat platform with billions of feature dimensions. KD-DAGFM achieves the best performance with less than 21.5% FLOPs of the state-of-the-art method on both online and offline experiments, showing the superiority of DAGFM to deal with the industrial scale data in CTR prediction task[1].

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Graph Factorization Machine; Knowledge Distillation; CTR Prediction; Recommender Systems

ACM Reference Format:
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 (WSDM '23), February 27-March 3, 2023, Singapore, Singapore.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3539597.3570384

1 INTRODUCTION

Click-Through Rate (CTR) prediction, which aims to predict the probability of a user clicking on an item, is a very critical task in recommender systems. The key to achieve good performance in CTR prediction is learning the effective high-order feature interactions, which has attracted great attention in recent years [3, 7, 13, 23]. One of the biggest challenges is the high computational cost to model the high-order interactions of raw features, because the number of feature combinations increases exponentially when the number of raw features increases. In real-world applications, raw features are usually highly sparse with millions of dimensions. For example, identifier feature such as the ID of user/item is very sparse after being encoded as an one-hot vector; so are the multi-filed vectors built from upstream tasks such as visual information. It is very time-consuming to calculate the high-order feature interactions on such sparse features with millions of dimensions, and there is a

* Ting Bai (baiting@bupt.edu.cn) is the corresponding author.
‡ Also with Beijing Key Laboratory of Intelligent Telecommunications Software and Multimedia.
† Also with Beijing Key Laboratory of Big Data Management and Analysis Methods.

WSDM '23, February 27-March 3, 2023, Singapore, Singapore

ACM ISBN 978-1-4503-9407-9/23/02…$15.00
https://doi.org/10.1145/3539597.3570384

[1] Our implementation code is available at: https://github.com/RUCAIBox/DAGFM

high risk of over-fitting problem for CTR prediction in real industry recommender systems.

Hence, a lightweight recommendation algorithm for CTR prediction is needed to simplify the online inference process, which enables it to avoid the explosion of computational costs in real industry recommender systems. Some efforts have been made to solve this problem [31, 35, 37]. They utilize Knowledge Distillation (KD) technique to transfer knowledge from complex teacher models to shallow student models with reduced learning parameters, so as to speed up the model inference for dealing with the real-time massive data in recommender systems. However, the acceleration of KD model with reduced learning parameters is at the expense of the degradation of the model accuracy, and it is inevitable for KD model to balance the effectiveness and efficiency [35]. The KD method [37] achieves better performance than its teacher model, but at cost of an undiminished amount of parameters of the student model, which may limit its adoption in online inference process.

To keep the low model complexity and meanwhile achieve better performance of KD student model, we propose a lightweight **K**nowledge **D**istillation based **D**irected **A**cyclic **G**raph **F**actorization **M**achine (**KD-DAGFM**) to predict the CTR in recommender systems. Our aim is to design a shadow student model with fewer learning parameters, but maintain the capability to distill the high-order feature interactions from existing complex interaction models. To achieve the approximately lossless knowledge distillation, we carefully design the student model: Directed Acyclic Graph Factorization Machine (**DAGFM**) to learn arbitrary explicit feature interactions. Specifically, each node in Directed Acyclic Graph (DAG) represents a feature field, and different nodes are connected by directed edges without cycles. The interaction of nodes, which represents the interaction of features, are modeled by the learnable weights of edges in DAG. With the specific interaction learning functions (*i.e.,* inner, outer and kernel), the feature interactions on DAG can be translated into explicit arbitrary-order interactions, which is demonstrated by a dynamic programming algorithm. Except for the explicit feature interactions, to distill the knowledge from implicit feature interaction models (*e.g.,* AutoInt [25] and FiBiNet [10]), we further propose an improved model KD-DAGFM+, in which an Multi-Layer Perceptron (MLP) component is integrated into the student model DAGFM. Experiments show that KD-DAGFM+ has the capability to distill both explicit and implicit feature interactions from any complex teacher models. The contributions are summarized as follows:

- We propose a lightweight distillation model KD-DAGFM to learn high-order explicit interactions from different complex teacher models, which could greatly reduce the computational costs (*i.e.,* FLOPs) by at least 10 times, making it possible to be applied into large-scale industry recommender systems.
- We design a directed acyclic graph neural network – DAGFM as the student model and use three different interaction learning functions (*i.e.,* inner, outer and kernel functions) in the propagation process to capture arbitrary-order explicit feature interactions, which achieves approximately lossless performance of the student model and is demonstrated by a dynamic programming algorithm.
- KD-DAGFM achieves the best performance with less than 21.5% FLOPs of the state-of-the-art method on both online and offline experiments, showing the superiority of KD-DAGFM to deal with the large scale industry data in CTR prediction task.

2 PRELIMINARY

We first give a preliminary introduction to the CTR prediction task, then introduce the explicit high-order feature interaction and the interaction learning function in CTR prediction task.

2.1 CTR Prediction

Click-Through Rate (CTR) prediction aims to predict the probability of the user clicking on an item. Specifically, given the m feature fields, we use $\boldsymbol{x} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_m\}$ to represent the feature of users and items, where $\boldsymbol{x}_i$ is the feature representation of the i-th feature. Label $y \in \{0, 1\}$ represents each item is click or not, and is predicted based on the input feature $\boldsymbol{x}$. The key to achieve good performance in CTR prediction task is learning the effective high-order feature interactions, which is a fundamental problem and has attracted great attention in research areas.

2.2 High-order Feature Interactions

Given the input feature $\boldsymbol{x} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_m\}$, the embedding features of $\boldsymbol{x}$ is represented as $\{\boldsymbol{e}_1, \boldsymbol{e}_2, .., \boldsymbol{e}_m\}$, where m is the number of feature fields, the explicit feature interactions can be formulated as:

$$\hat{y} = \sum_{t=2}^{m} \sum_{j_1<j_2<...<j_t} \phi(\boldsymbol{e}_{j_1}, \boldsymbol{e}_{j_2}, ..., \boldsymbol{e}_{j_t}), \tag{1}$$

where ϕ denotes the feature interaction learning function.

Take the 2-order feature interactions as example, formulated as:

$$\hat{y} = \sum_{i=1}^{m} \sum_{j=i+1}^{m} \phi(\boldsymbol{e}_i, \boldsymbol{e}_j). \tag{2}$$

2.3 Interaction Learning Function

The feature interaction learning function defines the way to compute the interactions among features. Different interaction models utilize different interaction learning functions.

2.3.1 Basic Inner Interaction Learning Function. The most commonly used interaction learning function in existing studies, *i.e.,* FM [22], IM [33], HOFM [1], is the basic inner interaction learning function, and it can be formulated as:

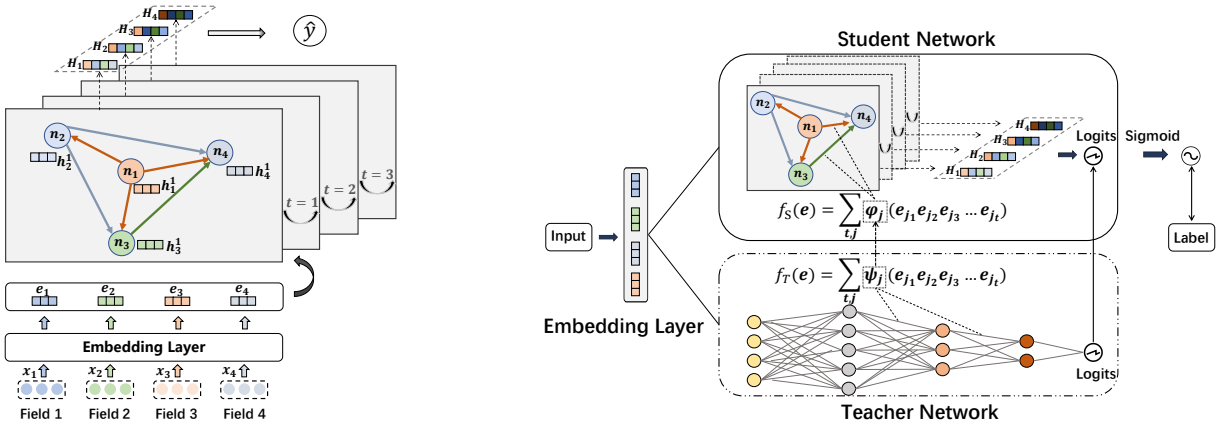
$$\phi(\boldsymbol{e}_i, \boldsymbol{e}_j) = \boldsymbol{e}_i \odot \boldsymbol{e}_j, \tag{3}$$

where $\odot$ is the element-wise product of two vectors.

2.3.2 Weighted Inner Interaction Learning Function. The above methods with basic inner function can not capture the distinct field information of features, leading to gradient coupled issue [21]. To address this problem, many field-aware variants such as FwFM [19] and FvFM [26] use weight parameters to model the feature interactions of different fields. The inner interaction learning function used in them can be formulated as:

$$\phi(\boldsymbol{e}_i, \boldsymbol{e}_j) = \boldsymbol{w}_{i,j} \odot \boldsymbol{e}_i \odot \boldsymbol{e}_j, \tag{4}$$

where $\boldsymbol{w}_{i,j}$ is a weight vector between the fields i and j.



(a) The architecture of DAGFM. (b) The architecture of KD-DAGFM.

Figure 1: Components and architecture of our proposed KD-DAGFM.

2.3.3 Kernel Interaction Learning Function. By extending the dimension of weight vector, recent studies [21, 26] propose a more powerful interaction learning function, *i.e.,* kernel interaction learning function, defined as:

$$\phi(\boldsymbol{e}_i, \boldsymbol{e}_j) = \boldsymbol{e}_i \boldsymbol{W}_{i,j} \odot \boldsymbol{e}_j, \tag{5}$$

where $\boldsymbol{W}_{i,j}$ is the learning weight matrix.

Despite the great improvements by incorporating field weights, the model complexity has largely increased. Enumerating all high-order feature interactions with distinct field weights is an NP-hard problem. Most existing high-order feature interaction models [16, 28] improve the model capability by increasing the parameters, but at the expense of high computational costs, making it impractical to be applied into large-scale industry recommendation scenarios.

3 METHODOLOGY

To improve model performance with low computational costs, we propose a Knowledge Distillation based Directed Acyclic Graph Factorization Machine (KD-DAGFM), in which a lightweight student model Directed Acyclic Graph Factorization Machine (DAGFM) is designed to learn the high-order feature interactions from existing complex teacher models.

3.1 Directed Acyclic Graph Factorization Machine (DAGFM)

To effectively and efficiently extract knowledge from complex high-order interaction models, we design a lightweight student model, named **D**irected **A**cyclic **G**raph **F**actorization **M**achine (**DAGFM**). The architecture of DAGFM is shown in Fig. 1 (a), we first introduce the construction of Directed Acyclic Graph (DAG) and the information interaction process, then introduce our proposed efficient outer interaction learning function.

3.1.1 The Construction of DAG. As shown in Fig. 1(a), given the input features $\boldsymbol{x} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_m\}$, where m is the number of total feature fields, and $\boldsymbol{x}_i$ is the feature representation of the i-th feature. The embedding layer projects the input features from a high-dimensional sparse space to a lower-dimensional dense space, *i.e.,* $\boldsymbol{e}_i = \boldsymbol{x}_i \boldsymbol{W}_i$, where $\boldsymbol{W}_i \in \mathbb{R}^{|\boldsymbol{x}_i| \times d}$ and d is the embedding size.

The directed acyclic graph (DAG) is defined as $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where each node $n_i \in \mathcal{N}$ corresponds to a feature field and $|\mathcal{N}| = m$. The initial state vector of each node n_i is set to $\boldsymbol{e}_i$, *i.e.,* $\boldsymbol{h}_i^1 = \boldsymbol{e}_i$. The edge from node n_i to n_j $(j > i)$ is directed, which controls the interaction weight between two features from different fields.

3.1.2 Learning Feature Interactions on DAG. The interactions among features can be modeled as the information propagation process on DAG in a recurrent fashion. At each propagation layer, each node will aggregate the state from its neighbors and combine it with the initial state $\boldsymbol{h}_i^1$ of itself to update the state vector. Formally, the state vector of node n_i at propagation layer t can be represented as:

$$\boldsymbol{h}_i^{t+1} = \sum_{j \in \mathcal{N}(i) \cup \{i\}} \phi(\boldsymbol{h}_j^t, \boldsymbol{h}_i^1), \tag{6}$$

where $\mathcal{N}(i) = \{j | n_j \rightarrow n_i \in \mathcal{E}\}$, and ϕ is the interaction learning function.

To obtain the information of high-order feature interactions for CTR prediction, we first apply sum pooling of the hidden state vector for each node as $p_i^t = \sum_{k=1}^{d} h_{i,k}^t$ at each interaction step, where $h_{i,k}^t$ denotes the k-th element of $\boldsymbol{h}_i^t$. Then we concatenate all node statesp at layer t as $\boldsymbol{p}^t = [p_1^t, p_2^t, ..., p_m^t]$, and the different-order interactions of features on DAG can be represented as $\boldsymbol{p} = [\boldsymbol{p}^1; \boldsymbol{p}^2; ...; \boldsymbol{p}^l]$, where the number of propagation layers is $l - 1$. Finally, the prediction function for CTR prediction can be formulated as:

$$\hat{y} = \sigma(\boldsymbol{p}\boldsymbol{w}^\top + b), \tag{7}$$

where σ is the sigmoid activation function, $\boldsymbol{w}$ is the transition vector and b is the bias.

3.1.3 Outer Interaction Learning Function. The feature interaction learning function ϕ of DAGFM (see in Eq. 6) is flexible, we can utilize the most commonly used learning functions, *i.e.,* the inner interaction learning function (see Eq. 4) or the kernel interaction learning function with more powerful model capability (see Eq. 5), to learn the high-order feature interactions.

To reduce the computational complexity of kernel interaction learning function, we propose a simplified **outer interaction learning function**, in which the learning weight matrix in kernel interaction learning function (see Eq. 5) can be decomposed as:

$$\boldsymbol{W}_{j,i}^{t} = (\boldsymbol{p}_{j,i}^{t})^{\top} \boldsymbol{q}_{j,i}^{t}, \tag{8}$$

where $\boldsymbol{p}_{j,i}^{t} \in \mathbb{R}^{d}$ and $\boldsymbol{q}_{j,i}^{t} \in \mathbb{R}^{d}$ are the decomposed vectors.

Then the outer interaction learning function can be defined as:

$$\phi(\boldsymbol{h}_j^t, \boldsymbol{h}_i^1) = \boldsymbol{h}_j^t \boldsymbol{W}_{j,i}^t \odot \boldsymbol{h}_i^1 = (\boldsymbol{h}_j^t (\boldsymbol{p}_{j,i}^t)^{\top}) \cdot (\boldsymbol{q}_{j,i}^t \odot \boldsymbol{h}_i^1). \tag{9}$$

Our proposed outer interaction learning function is decomposed from kernel interaction learning function (see Eq. 5) and has the same complexity with inner interaction learning function (*i.e.,* $O(d)$), which is much less than kernel interaction learning function (*i.e.,* $O(d^2)$), making it more efficient to capture the feature interactions.

3.2 Arbitrary-Order Interaction Learning in DAGFM

In this section, we demonstrate that the feature interactions learned in DAGFM align with different unique weighted paths in a dynamic programming (DP) algorithm, showing the ability of DAGFM to learn arbitrary-order feature interactions.

We define the suffix of a feature interaction as the feature with the largest subscript. For example, for the interactions with feature $\boldsymbol{e}_2\boldsymbol{e}_3$ and $\boldsymbol{e}_3\boldsymbol{e}_4\boldsymbol{e}_5$, the suffix are $\boldsymbol{e}_3$ and $\boldsymbol{e}_5$ respectively. Let S_i^t denote the set of all t-th order feature interactions suffixed with feature $\boldsymbol{e}_i$. Take 2-order feature interactions suffixed with $\boldsymbol{e}_3$ as example: $S_3^2 = \{\boldsymbol{e}_1\boldsymbol{e}_3, \boldsymbol{e}_2\boldsymbol{e}_3, \boldsymbol{e}_3\boldsymbol{e}_3\}$. The 1-order feature interaction suffixed with $\boldsymbol{e}_i$ is itself, *i.e.,* $S_i^1 = \{\boldsymbol{e}_i\}$.

In dynamic programming algorithm, the state variable $\boldsymbol{h}_i^t$ is defined as the sum of elements in S_i^t, *i.e.,* $\boldsymbol{h}_i^t = \sum_{j \in S_i^t} \boldsymbol{j}$. In fact, all $t+1$-th order interactions suffixed with $\boldsymbol{e}_i$ can be expressed as the interaction between $\boldsymbol{e}_i$ and t-th order interactions suffixed with $\boldsymbol{e}_j$ ($j \leq i$). Therefore, the state transfer equation of $\boldsymbol{h}_i^t$ in DP algorithm can be formulated as:

$$\boldsymbol{h}_i^{t+1} = \sum_{j=1}^{i} \boldsymbol{h}_j^t \odot \boldsymbol{e}_i = \sum_{j=1}^{i} \boldsymbol{h}_j^t \odot \boldsymbol{h}_i^1. \tag{10}$$

In DAGFM, the feature interactions are propagated based on a directed acyclic graph, which means the neighbors $\mathcal{N}(i)$ of node n_i refer to the directly linked nodes n_j, where $j < i$. Hence, the state vector of node n_i in graph propagation process in Eq. 6 is equivalent to the transferred state in DP algorithm (see Eq. 10).

$$\boldsymbol{h}_i^{t+1} = \sum_{j \in \mathcal{N}(i) \cup \{i\}} \phi(\boldsymbol{h}_j^t, \boldsymbol{h}_i^1) = \sum_{j=1}^{i} \boldsymbol{h}_j^t \odot \boldsymbol{h}_i^1, \tag{11}$$

here the interaction learning function ϕ is equal to the basic inner interaction learning function (see in Eq. 3).

From Eq. 10 and Eq. 11, we can see that each feature interaction in DAGFM can align with a unique path in DP graph (see in Fig. 2): each k-order feature interaction corresponds to a unique path with length $k-1$ from the first layer. By summing all the state vectors in DP algorithm, all the paths can be traversed, indicating all the feature interactions can be modeled in DAGFM.

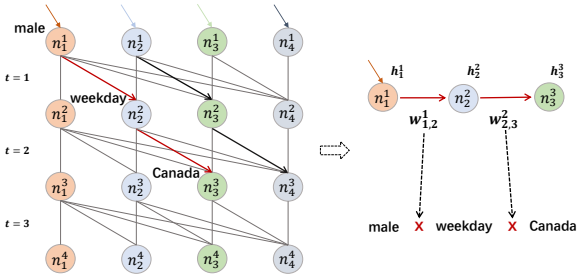


Figure 2: The propagation graph of DAGFM. Each k-order feature interaction corresponds to a unique path with length $k-1$ in the dynamic programming algorithm.

For graph propagation processes with inner (see Eq. 4) and kernel (see Eq. 5) interaction learning functions, they can be equivalent to Eq. 11 by assigning the weights $\boldsymbol{w}_{j,i}^t$ to be all-ones vector and identity matrix respectively. Besides, the weights can also be learned during training process, which enables DAGFM to model all types of feature interactions with different semantic information. For example, let nodes n_1, n_2, n_3, n_4 represent `male`, `weekday`, `Canada`, `sunny` at first layer respectively. Feature interaction `male` × `weekday` × `Canada` corresponds to the path $n_1^1 \rightarrow n_2^2 \rightarrow n_3^3$ marked red color with the learning weights $\boldsymbol{w}_{1,2}^1 \odot \boldsymbol{w}_{2,3}^2$. The edge weight $\boldsymbol{w}_{i,j}^t$ controls the strength of the interaction between i-th feature and j-th feature at t-th order interactions. In the above example, $\boldsymbol{w}_{1,2}^1$ controls the semantic strength of `male` × `weekday` in `male` × `weekday` × `Canada`.

3.3 Knowledge Distillation Process

The overall distillation framework of KD-DAGFM is shown in Fig. 1(b). In knowledge distillation process, the student network DAGFM learns the knowledge of feature interactions from teacher networks. To ensure the feature space in student and teacher networks be the same, we share the embedding layer of teacher network with student network. The loss function in knowledge distillation can be formulated as:

$$\mathcal{L}_{KD} = \frac{1}{N} \sum_{i=1}^{N} \Big(T(\boldsymbol{x}_i, \boldsymbol{\psi}) - S(\boldsymbol{x}_i, \boldsymbol{\varphi}) \Big)^2, \tag{12}$$

where N is the total number of training instances, T and S denote the learning function, $\boldsymbol{\psi}$ and $\boldsymbol{\varphi}$ are the parameters in teacher and student networks respectively.

Besides, we adopt cross entropy loss function for CTR prediction, which could be formulated as:

$$\mathcal{L}_{CTR} = -\frac{1}{N} \sum_{i=1}^{N} \Big(y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \Big), \tag{13}$$

where y_i is the true label ,$\hat{y}_i$ is predicted result.

The final optimization objective of the knowledge distillation model KD-DAGFM is:

$$\mathcal{L} = \alpha \mathcal{L}_{KD} + \beta \mathcal{L}_{CTR}, \tag{14}$$

where α, β are hyper-parameters to balance the weights of two loss functions.

After knowledge distillation, the student model can well learn the useful feature interactions from the teacher network. However,

Table 1: The statistics of datasets.

Dataset	# Features	# Fields	# Instances
Criteo	1.3M	39	45M
Avazu	1.5M	23	40M
MovieLens-1M	13k	7	740K
WeChat	2.9M	264	40.9M

since the feature embedding of student is inherited from teacher which is not trained during distillation, it may result in the underfitting problem and make sub-optimal performance of the student model. To alleviate this problem, we fine-tune the student model for the final CTR prediction.

4 EXPERIMENTS

We conduct experiments to demonstrate the effect of our proposed knowledge distillation model KD-DAGFM. Four real world datasets are used in our experiments: Criteo[2], Avazu[3], MovieLens-1M[4] and WeChat. Criteo is the most popular CTR prediction benchmark dataset contains user logs over a period of 7 days. Avazu contains user logs over a period of 7 days, which was used in the Avazu CTR prediction competition. MovieLens-1M is the most popular dataset for recommendation systems research. WeChat is collected on WeChat platform, which contains more than one hundred million daily active users. Table 1 summarizes the dataset statistics information.

We first verify the effectiveness of our proposed student model DAGFM and then compare the model performance of KD-DAGFM with the state-of-the-art interaction models for CTR prediction task.

4.1 Effectiveness Analysis of the Student Model DAGFM

In this section, we conduct experiments to demonstrate that our proposed student model DAGFM can achieve the approximately lossless performance of the complex teacher models.

4.1.1 The Teacher Networks. We choose the most commonly used complex models with competitive performance in CTR prediction task, *i.e.,* xDeepFM [16] and DCNV2 [28]. Due to the fact that DAGFM is an explicit model and its distillation from deep implicit component may incorporate noise to degrade model performance, for fair comparison and speed up the knowledge distillation process, we only use the explicit feature interaction component **CIN** and **CrossNet** as the teacher models in xDeepFM and DCNV2 respectively. (results with different types of teachers, including both implicit and explicit teacher networks, are shown in Sec. 4.3.4.)

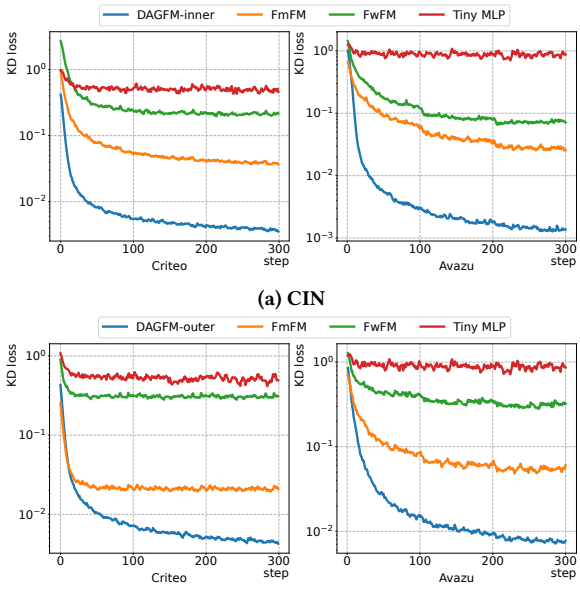
4.1.2 The Student Networks. Considering the high computational costs of deep methods may limit their adoption in large-scale recommendation scenarios, we use the most commonly used light models: **FwFM** [19], **FmFM** [26], and **tiny MLP** (*i.e.,* the MLP with only three layers: $128 \times 128 \times 128$) as the student models for comparison. For DAGFM, we adopt the best combination of teachers and interaction learning functions, *i.e.,* CIN with DAGFM-inner and CrossNet with DAGFM-outer (more detailed experimental results are analyzed in Sec. 4.3.2).

[2]http://labs.criteo.com/2014/02/kaggle-display-advertising-challenge-dataset
[3]http://www.kaggle.com/c/avazu-ctr-prediction
[4]https://grouplens.org/datasets/movielens

Table 2: Effectiveness comparisons of different student models. l is the depth, m is the number of feature fields, d is the embedding size, H is the dimension of hidden vectors.

Distillation	Criteo		Avazu		Order	Complexity
	AUC	Log Loss	AUC	Log Loss		
CIN	0.8109	0.4424	0.7816	0.3803	≥ 2	$O(mH^2dl)$
CIN → FwFM	0.8008	0.4511	0.7779	0.3823	2	$O(m^2d)$
CIN → FmFM	0.8091	0.4445	0.7809	0.3806	2	$O(m^2d^2)$
CIN → Tiny MLP	0.8098	0.4506	0.7794	0.3839	NA	$O(mdH + H^2l)$
CIN → DAGFM-inner	**0.8109**	**0.4425**	**0.7816**	**0.3803**	≥ 2	$O(m^2dl)$
CrossNet	0.8123	0.4398	0.7817	0.3805	≥ 2	$O(m^2d^2l)$
CrossNet → FwFM	0.7945	0.4559	0.7690	0.3874	2	$O(m^2d)$
CrossNet → FmFM	0.8108	0.4411	0.7800	0.3811	2	$O(m^2d^2)$
CrossNet → Tiny MLP	0.8102	0.4516	0.7795	0.3837	–	$O(mdH + H^2l)$
CrossNet → DAGFM-outer	**0.8122**	**0.4397**	**0.7815**	**0.3806**	≥ 2	$O(m^2dl)$

4.1.3 Implementation Details. The experimental settings of the knowledge distillation process is identical to the settings of KD-DAGFM in Section 4.2. The details of implementation are summarized in Section 4.2.2.



(a) CIN

(b) CrossNet

Figure 3: The loss curves in knowledge distillation process of different student models.

4.1.4 The Experimental Results of Different Student Models. The experimental results of different student models on Criteo and Avazu datasets are shown in Table 2, we can see that:

(1) FwFM and FmFM perform the worst due to the limited 2-order feature interaction capability, which can not well fit the high-order information of the teacher network.

(2) Tiny MLP learns high-order feature interactions with multi-layer neural networks in an implicit manner, and its capability is limited by the shallow model architecture.

Table 3: Performance comparisons. Note that a higher AUC or lower Logloss at 0.001-level is significant for CTR prediction.

Model	Criteo		Avazu		MovieLens-1M		WeChat				
	AUC	Log Loss	AUC	Log Loss	AUC	Log Loss	AUC	Log Loss	Params	FLOPs	Latency
FmFM	0.8112	0.4408	0.7744	0.3831	0.8864	0.3295	0.6593	0.2660	5.99M	9.44M	0.099 ms
FwFM	0.8104	0.4414	0.7741	0.3835	0.8815	0.3351	0.6702	0.2637	0.03M	1.11M	0.046 ms
xDeepFM	0.8122	0.4407	0.7821	0.3799	0.8913	0.3244	0.6712	0.2627	282.77M	3761.16M	0.588 ms
DCNV2	0.8127	0.4394	0.7838	0.3782	0.8946	0.3229	0.6683	0.2640	87.63M	87.63M	0.198 ms
FiBiNet	0.8126	0.4415	0.7837	0.3783	0.8860	0.3291	0.6681	**0.2449**	569.01M	587.76M	0.219 ms
AutoInt+	0.8126	0.4396	0.7832	0.3786	0.8937	0.3288	0.6774	0.2618	34.14M	64.92M	0.222 ms
FiGNN	0.8109	0.4412	0.7830	0.3799	0.8939	0.3232	0.6623	0.2641	9.91M	41.13M	0.323 ms
GraphFM	0.8070	0.4448	0.7792	0.3807	0.8890	0.3311	0.6532	0.2660	3.60M	1193.74M	0.192 ms
ECKD	0.8123	0.4422	0.7834	0.3838	0.8951	**0.3173**	0.6635	0.2672	25.44M	25.44M	0.108 ms
CIN (teacher)	0.8109	0.4424	0.7816	0.3803	0.8850	0.3320	0.6668	0.2636	231.96M	3710.57M	0.213 ms
DAGFM-inner (student)	0.8105	0.4413	0.7801	0.3805	0.8839	0.3339	0.6620	0.2651	1.75M	3.36M	0.068 ms
KD-DAGFM-inner	0.8109	0.4425	0.7816	0.3803	0.8849	0.3320	0.6668	0.2636	1.75M	3.36M	0.068 ms
KD-DAGFM$_{FT}$-inner	0.8121	0.4400	**0.7883**	**0.3760**	0.8880	0.3304	**0.6777**	0.2617	**1.75M**	**3.36M**	**0.068 ms**
CrossNet (teacher)	0.8123	0.4398	0.7817	0.3805	0.8907	0.3474	0.6681	0.2638	53.54M	53.54M	0.125 ms
DAGFM-outer (student)	0.8119	0.4401	0.7791	0.3810	0.8895	0.3361	0.6672	0.2646	3.42M	5.04M	0.086 ms
KD-DAGFM-outer	0.8122	0.4397	0.7815	0.3806	0.8904	0.3476	0.6680	0.2638	3.42M	5.04M	0.086 ms
KD-DAGFM$_{FT}$-outer	**0.8132**	**0.4390**	0.7864	0.3780	**0.8976**	0.3189	0.6748	0.2665	**3.42M**	**5.04M**	**0.086 ms**

(3) DAGFM achieves competitive performance compared with the teacher network, showing its capability to transfer the high-order feature interaction information.

(4) As the student network, the model complexity of DAGFM (*i.e.,* m^2dl) is much lower than the teacher model CIN (*i.e.,* mH^2dl) and CrossNet (*i.e.,* m^2d^2l) (d is the size of embedding and H is the dimension of hidden vectors, which are much larger than the number of fields m). For the student networks, the complexity of DAGFM is much lower than FmFM and Tiny MLP and is comparable with FwFM.

We also show the distillation loss in training process. As shown in Fig. 3: the deviation of DAGFM from the teacher network is very small (10^{-3}), which is almost 1000 times smaller than it in the tiny MLP model, 30 times and 300 times smaller than it in FmFM and FwFM respectively, showing the approximately lossless distillation capability of DAGFM.

4.2 Experimental Results of KD-DAGFM

We conduct experiments to show the effectiveness of our proposed knowledge distillation model KD-DAGFM in CTR prediction task.

4.2.1 Compared Methods. We compare KD-DAGFM with state-of-the-art methods in CTR prediction task, including:

• FmFM [26]: it utilizes kernel product with matrix weights to capture the field information.

• FwFM [19]: it utilizes inner product with scalar weights to capture the field information.

• xDeepFM [16]: it utilizes inner product to generate multiple feature maps, which encode all pairwise interactions to model explicit interactions.

• DCNV2 [28]: it utilizes kernel product to model vector-wise feature interactions and achieves state-of-the-art performance.

• FiBiNet [10]: it uses SENet and bi-linear operation to improve pair-wise feature interactions.

• AutoInt [25]: it uses multi-head self-attentive mechanism to learn implicit feature interactions. AutoInt+ improves AutoInt by combining a feed-forward neural network.

• FiGNN [14]: it uses GRU components to model implicit feature interactions in a fully-connected graph.

• GraphFM [15]: it performs graph sampling and utilizes multi-head attentive mechanism to model implicit feature interactions in a two-stage manner.

• ECKD [37]: it is a knowledge distillation based approach, and utilizes large-scale MLP as student network to distill the knowledge from several state-of-the-art teachers.

The above approaches could cover different types in the area of CTR prediction. FmFM, FwFM, xDeepFM and DCNV2 model explicit feature interactions. The rest methods model implicit feature interactions. Specially, GraphFM and FiGNN are GNN based methods and ECKD is a knowledge distillation based approach. Our proposed KD-DAGFM is a knowledge distillation model based on the field-aware graph neural networks, and is able to capture the high-order explicit feature interactions.

4.2.2 Implementation Details. For each method, grid search is applied to find the optimal settings. All methods are implemented in Pytorch [20]. The dimension of feature embedding is 16. The learning rate is in [1e-3, 1e-4, 1e-5]. The L_2 penalty weight is 1e-5. The optimizer is Adam. The distillation loss weights α is in [1e-1, 1, 10] and β is in [10, 100, 1000]. The depth of CIN and CrossNet is in [2, 3, 4]. The layer size of CIN is in [100, 200, 400]. For DAGFM, the number of propagation layers is in [2, 3, 4]. For FwFM and FmFM, we adopt field-wise linear weight. For AutoInt+, the depth, number of head and attention size is 2, 2, 40 respectively. For ECKD, we choose AutoInt+, DCNV2 and xDeepFM as teachers and adopt "soft label+pre-train" scheme. For KD-DAGFM, we report the results with the simple CIN and CrossNet as teachers and the number of propagation layers of DAGFM is equal to the depth of teacher.

4.2.3 The Experimental Results of Different CTR Methods. We present the results of different models for CTR prediction task in Table 3. We have the following observations:

(1) Deep methods (*i.e.,* xDeepFM, DCNV2, AutoInt+) outperform the shallow explicit models (*i.e.,* FwFM, FmFM) on the public

datasets, which indicates the importance of modeling high-order feature interactions. Compared with the implicit methods (*i.e.,* AutoInt+, FiGNN, FiBiNet), explicit method DCNV2 achieves better performance on Criteo and MovieLens-1M datasets, which suggests the explicit feature interactions are important patterns for CTR prediction task.

(2) Most graph-based models (*i.e.,* FiGNN, DAGFM) except for GraphFM perform competitively on Criteo, Avazu and Movielens-1M datasets, which suggests that the performance of graph-based model is also influenced by the different aggregation strategies and graph topology. For the knowledge distillation model ECKD, by transferring knowledge from multiple teachers, it achieves better performance than graph-based methods, showing the effectiveness of knowledge distillation.

(3) Our proposed model KD-DAGFM achieves the best performance with knowledge distillation and fine-tuning. KD-DAGFM with knowledge distillation performs better than DAGFM, showing the effectiveness of learning high-order feature interaction from complex teacher models. The fine-tuning of KD-DAGFM improves the model performance due to the alleviation of the under-fitting problem of the student's feature embedding. Besides, different interaction learning functions in KD-DAGFM have different effects: DAGFM-inner outperforms DAGFM-outer on Avazu and WeChat datasets, and loses the advantages on Criteo and Movielens datasets.

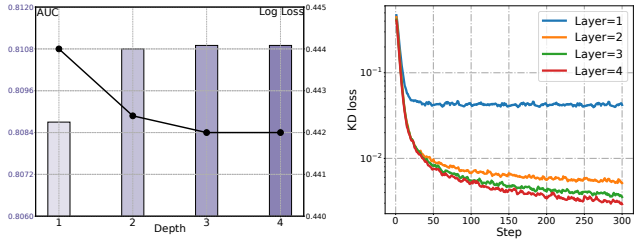
(4) Comparing the number of parameters and FLOPs, we can see that FwFM is the most efficient algorithm due to its simplicity. The DNN based methods are much more time-consuming due to their complicated feature interacting units, which makes them impractical in large-scale recommendation scenarios. In contrast, DAGFM is an efficient model, in which the amount of parameters is comparable with that in the shallow linear methods. It could greatly reduce the computational costs (i.e., FLOPs) by more than 10 times compared with the teacher model. Besides, the latency of DAGFM is much more smaller than the other models aside from the light model FwFM, making it possible to be applied into large-scale industry recommender systems.

To summarize, our proposed KD-DAGFM could effectively learn the high-order feature interactions. The fewer parameters make it much more efficient to be applied into large-scale recommendation scenarios.

4.3 Experimental Analysis of KD-DAGFM

We analyze the factors that influence the model performance and conduct online A/B testing to show the superiority of KD-DAGFM in real industry recommender systems. Besides, an advanced knowledge distillation model KD-DAGFM+ is proposed, which improves its ability to be applied into a wider range of applications.

4.3.1 The Effect of Number of Propagation Layers. We explore the effects of the number of propagation layers, which implies the orders of feature interactions. We use CIN as the teacher network with 3 layers operation (4-order interactions), and the model performance on Criteo dataset is shown in Fig. 4. We can observe that as the number of layers increases, the model performance becomes better, showing the importance of learning the high-order feature interaction information for CTR prediction. The improvements of the model with 2 propagation layers (modeling 3-order interactions) are significant compared with the one layer model (modeling 2-order interactions), consequently becoming slight as the layers increase, showing the crucial of 2-order and 3-order interactions.



(a) Performance **(b) KD Loss**

Figure 4: The performance of KD-DAGFM with different number of propagation layers.

4.3.2 The Effect of Different Interaction Learning Functions. As the performance of knowledge distillation model is influenced by the interaction learning functions, we conduct experiments to explore the correlation between teacher models and the interaction learning functions in KD-DAGFM. As shown in Fig. 5, we can see that DAGFM with our proposed outer learning function can be well suitable for different teacher models (*i.e.,* CIN and CrossNet with inner and kernel interaction learning function respectively), while the DAGFM with inner function performs worse with teacher model CrossNet. Our proposed outer function is decomposed from kernel function, which can be degenerated into inner function, making it more effective to transfer knowledge from the teacher models.

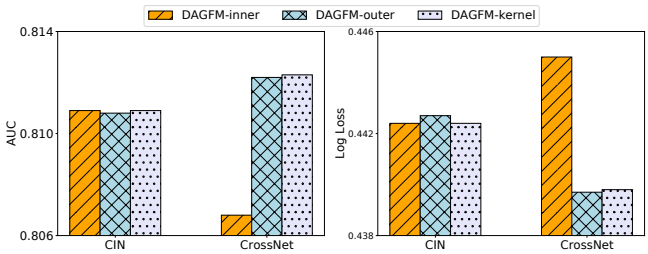


Figure 5: The performance of KD-DAGFM with different interaction learning functions.

4.3.3 Online A/B Testing. We conduct online experiments on a large scale industry recommender systems *i.e.,* WeChat Official Account Platform[5]. We calculate the gain of the corresponding evaluation metrics: # Click Users: the number of users had clicked the recommended articles; CTR: the Click Through Rate. The SOTA model is an extended version of DCNV2 which achieves the best performance in online service. Our model KD-DAGFM learns from the CrossNet (as teacher network) with outer interaction learning function. From Table 4, we can see that: compared with the SOTA method, KD-DAGFM gains in both the number of click users and the CTR. More importantly, KD-DAGFM outperforms the SOTA

[5]https://mp.weixin.qq.com/

Table 4: Online A/B test results on WeChat Official Account Platform. Higher ↑ is better for Click Users and CTR, while lower ↓ is better for FLOPs and Latency.

Method	#Click Users ↑	CTR ↑	FLOPs ↓	Latency ↓
SOTA Method	1,532,276	0.09789	70M	0.158 ms
KD-DAGFM	1,533,351	0.09847	15M	0.059 ms
Improvements	+0.07%	+0.59%	+78.5%	+62.7%

Table 5: Distillation performance of KD-DAGFM+.

Distillation	Criteo			Avazu		
	AUC	Log Loss	Latency	AUC	Log Loss	Latency
xDeepFM (teacher)	0.8122	0.4407	0.251 ms	0.7821	0.3799	0.067 ms
KD-DAGFM+	0.8122	0.4408	0.013 ms	0.7821	0.3801	0.011 ms
KD-DAGFM$_{FT}$+	0.8132	0.4388		0.7870	0.3776	
DCNV2 (teacher)	0.8127	0.4394	0.021 ms	0.7838	0.3782	0.013 ms
KD-DAGFM+	0.8126	0.4396	0.013 ms	0.7838	0.3784	0.007 ms
KD-DAGFM$_{FT}$+	0.8134	0.4387		0.7865	0.3775	
AutoInt+ (teacher)	0.8126	0.4396	0.571 ms	0.7832	0.3786	0.332 ms
KD-DAGFM+	0.8126	0.4396	0.013 ms	0.7831	0.3784	0.010 ms
KD-DAGFM$_{FT}$+	0.8137	0.4385		0.7875	0.3761	
FiBiNet (teacher)	0.8126	0.4415	0.124 ms	0.7837	0.3783	0.024 ms
KD-DAGFM+	0.8125	0.4418	0.009 ms	0.7836	0.3786	0.008 ms
KD-DAGFM$_{FT}$+	0.8131	0.4405		0.7875	0.3768	

method with only 21.5% computational costs and its inference speed is increased by nearly 3 times, showing the superiority of KD-DAGFM to deal with the web-scale data in real industry recommender systems.

4.3.4 A General Distillation Model KD-DAGFM+. Considering the teacher models are complex, *i.e.,* learning both explicit and implicit feature interactions, in order to effectively extract the knowledge from both explicit and implicit feature interactions, we propose an advanced distillation model KD-DAGFM+, in which an MLP component are attached after the student model DAGFM (the representations of all node states in DAG is concatenated to feed into the MLP). Four widely used methods *i.e.,* xDeepFM, DCNV2, AutoInt+ and FiBiNet, are chosen as teacher networks, and the results are shown in Table 5. We can observe that KD-DAGFM+ achieves approximately lossless distillation performance and effectively improves performance during fine-tuning. Besides, comparing with the complex teacher networks, the inference latency of KD-DAGFM+ is reduced greatly.

5 RELATED WORK

Feature Interactions Learning. Learning feature interactions is a fundamental problem in CTR prediction tasks. FM [22] is the most basic and widely used model, which assigns a learnable embedding vector to each feature to capture the second-order feature interactions. Besides, HOFM [1] captures the high-order feature interactions; FwFM [19] and FmFM [26] utilize inner product and kernel product to capture field interaction respectively. To better model high order feature interactions, lots of deep learning based methods learn feature interactions through a well-designed component and incorporate a deep neural network. For example, DeepFM [5] imposes a factorization machine as explicit component; xDeepFM [16] and DCNV2 [28] propose the Compressed Interaction Network (CIN) and Cross Network to model the explicit feature interactions. Besides, recent research [2, 11, 17, 18, 24, 34, 36] trend focuses on learning effective feature interactions via AutoML. However, these deep models have extremely high computational complexity, making them impractical be applied to real-time large-scale recommendation scenarios.

Graph Neural Networks. In recent years, Graph Neural Network is widely used in recommender systems. Many GNN-based studies [8, 29, 30, 32] capture the collaborative signal existing in the user-item bipartite graph and treats the recommendation task as a link prediction problem. In the area of GNN based CTR predictions, DG-ENN [6] exploits the strengths of graph representation with two carefully designed learning strategies to refine the feature embeddings; Fi-GNN [14] learns implicit feature interactions by graph propagation on a fully-connected graph; GraphFM [15] proposes a two-stage aggregation strategy to model high-order feature interactions; PCF-GNN [12] proposes a pre-trained model to generate the explicit semantic cross features. Interestingly, a recent study [4] points out that GNNs are claimed to align with dynamic programming. In this work, we propose a lightweight and directed acyclic graph based model to learn the explicit high-order feature interactions.

Knowledge Distillation. Knowledge Distillation (KD) [9] is a method to extract knowledge from sophisticated models and compress it into a simple model, so that it can be deployed to practical applications. As KD has achieved great success in many research fields such as natural language processing [27], the KD based models [31, 35, 37] have attracted increasing attention in recommender systems. ECKD [37] is an ensemble learning approach via knowledge distillation to deliver the performance of MLP. It achieves better performance than its teacher at cost of an undiminished amount of parameters of its student. For model compression, the general defect of knowledge distillation is the degradation of model accuracy because the capacity gap between large teachers and small students always exists. The design of an effective student model is still a challenge.

6 CONCLUSION

In this paper, we propose a lightweight knowledge distillation model KD-DAGFM with a directed acyclic graph neural network for CTR predictions. The directed acyclic graph structure proposed in KD-DAGFM makes it possible to align with a DP algorithm, which improves its capability to learn arbitrary feature interactions from the complex teacher networks and makes it achieve approximately lossless model performance. The information propagation in the student model DAGFM with different interaction learning function could greatly reduce the computational resource costs. KD-DAGFM achieves the best performance on both online and offline experiments, showing the superiority of DAGFM to deal with the industrial scale data in CTR prediction task.

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China under Grant No. 62102038 and 62222215, and Weixin Open Platform under Grant No. S2021120.

REFERENCES

[1] Mathieu Blondel, Akinori Fujino, Naonori Ueda, and Masakazu Ishihata. 2016. Higher-order factorization machines. *Advances in Neural Information Processing Systems* 29 (2016).

[2] Yifan Chen, Pengjie Ren, Yang Wang, and Maarten de Rijke. 2019. Bayesian personalized feature interaction selection for factorization machines. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 665–674.

[3] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*. 7–10.

[4] Andrew Dudzik and Petar Veličković. 2022. Graph Neural Networks are Dynamic Programmers. *arXiv preprint arXiv:2203.15544* (2022).

[5] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247* (2017).

[6] Wei Guo, Rong Su, Renhao Tan, Huifeng Guo, Yingxue Zhang, Zhirong Liu, Ruiming Tang, and Xiuqiang He. 2021. Dual Graph enhanced Embedding Neural Network for CTR Prediction. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 496–504.

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

[8] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 639–648.

[9] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* 2, 7 (2015).

[10] 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.

[11] Farhan Khawar, Xu Hang, Ruiming Tang, Bin Liu, Zhenguo Li, and Xiuqiang He. 2020. Autofeature: Searching for feature interactions and their architectures for click-through rate prediction. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 625–634.

[12] Feng Li, Bencheng Yan, Qingqing Long, Pengjie Wang, Wei Lin, Jian Xu, and Bo Zheng. 2021. Explicit Semantic Cross Feature Learning via Pre-trained Graph Neural Networks for CTR Prediction. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2161–2165.

[13] 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.

[14] Zekun Li, Zeyu Cui, Shu Wu, Xiaoyu Zhang, and Liang Wang. 2019. Fi-gnn: Modeling feature interactions via graph neural networks for ctr prediction. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 539–548.

[15] Zekun Li, Shu Wu, Zeyu Cui, and Xiaoyu Zhang. 2021. GraphFM: Graph Factorization Machines for Feature Interaction Modeling. *arXiv preprint arXiv:2105.11866* (2021).

[16] 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.

[17] Bin Liu, Niannan Xue, Huifeng Guo, Ruiming Tang, Stefanos Zafeiriou, Xiuqiang He, and Zhenguo Li. 2020. AutoGroup: Automatic feature grouping for modelling explicit high-order feature interactions in CTR prediction. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 199–208.

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

[19] 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.

[20] 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).

[21] 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.

[22] Steffen Rendle. 2010. Factorization machines. In *2010 IEEE International conference on data mining*. IEEE, 995–1000.

[23] 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.

[24] Qingquan Song, Dehua Cheng, Hanning Zhou, Jiyan Yang, Yuandong Tian, and Xia Hu. 2020. Towards automated neural interaction discovery for click-through rate prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 945–955.

[25] 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.

[26] 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.

[27] Xu Tan, Yi Ren, Di He, Tao Qin, Zhou Zhao, and Tie-Yan Liu. 2019. Multilingual neural machine translation with knowledge distillation. *arXiv preprint arXiv:1902.10461* (2019).

[28] 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.

[29] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*. 165–174.

[30] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-supervised graph learning for recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 726–735.

[31] Chen Xu, Quan Li, Junfeng Ge, Jinyang Gao, Xiaoyong Yang, Changhua Pei, Fei Sun, Jian Wu, Hanxiao Sun, and Wenwu Ou. 2020. Privileged features distillation at Taobao recommendations. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2590–2598.

[32] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 974–983.

[33] 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.

[34] Pengyu Zhao, Kecheng Xiao, Yuanxing Zhang, Kaigui Bian, and Wei Yan. 2020. Amer: Automatic behavior modeling and interaction exploration in recommender system. *arXiv preprint arXiv:2006.05933* (2020).

[35] Guorui Zhou, Ying Fan, Runpeng Cui, Weijie Bian, Xiaoqiang Zhu, and Kun Gai. 2018. Rocket launching: A universal and efficient framework for training well-performing light net. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[36] Chenxu Zhu, Bo Chen, Weinan Zhang, Jincai Lai, Ruiming Tang, Xiuqiang He, Zhenguo Li, and Yong Yu. 2021. AIM: Automatic Interaction Machine for Click-Through Rate Prediction. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[37] Jieming Zhu, Jinyang Liu, Weiqi Li, Jincai Lai, Xiuqiang He, Liang Chen, and Zibin Zheng. 2020. Ensembled CTR prediction via knowledge distillation. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2941–2958.