FINT: FIELD-AWARE INTERACTION NEURAL NETWORK FOR CLICK-THROUGH RATE PREDICTION

Zhishan Zhao[1], *Sen Yang* [2,3], *Guohui Liu*[1], *Dawei Feng*[2,3], *Kele Xu*[2,3*]

[1] iQIYI Inc., China
[2] National Key Lab of Parallel and Distributed Processing, Changsha, China
[3] National University of Defense Technology, Changsha, China
zhishan777@gmail.com, kelele.xu@gmail.com

ABSTRACT

As a critical component for online advertising and marketing, click-through rate (CTR) prediction has drawn lots of attention from both industry and academia. Recently, deep learning has become the mainstream methodological choice for CTR. Despite sustainable efforts have been made, existing approaches still pose several challenges. On the one hand, high-order interaction between the features is under-explored. On the other hand, high-order interactions may neglect the semantic information from the low-order fields. In this paper, we proposed a novel prediction method, named FINT, that employs the Field-aware INTeraction layer which explicitly captures high-order feature interactions while retaining the low-order field information. To empirically investigate the effectiveness and robustness of the FINT, we perform extensive experiments on the three realistic databases: KDD2012, Criteo and Avazu. The obtained results demonstrate that the FINT can significantly improve the performance compared to the existing methods, without increasing the amount of computation required. Moreover, the proposed method brought about 2.72% increase to the advertising revenue of iQIYI, a big online video app through A/B testing. To better promote the research in CTR field, we released our code as well as reference implementation at: https://github.com/zhishan01/FINT.

Index Terms— Click-through Rate, Field-Aware Interaction, High-order Feature Interactions.

1. INTRODUCTION

Click-through rate (CTR) prediction aims to forecast the probability that a user will click a particular recommended item or an advertisement on a web page [1]. The applications of CTR seem to be evident in several different fields, such as recommendation systems, online advertising and product search. During the last decades, CTR has drawn dramatic interest, due to its important roles in both the academic and industry. Unlike other data types, such as images and texts, data used in CTR are usually of high sparsity and large scale. Making an accurate and robust prediction is still far from being solved. In the early years, logistic regression (LR) and factorization machines (FM) were widely explored. Recently, deep learning-based approaches have been the mainstream methodological choice for CTR, such as, Wide&Deep [2], DeepFM [3], DCN [4] and xDeepFM [5]. Based on previous studies, how to fully utilize both the low- and high-order feature interactions simultaneously can bring extra performance improvements, compared to the cases of considering either alone.

Existing high-order feature interaction based approaches still confront with a significant challenge. The field-level semantic information is lost during the high-order feature interaction. Consequently, the subsequent deep modules cannot fully employ the explicit features. In this paper, we propose a novel framework to capture high-order feature interactions while preserving the low-order field semantic information by introducing a field-aware interaction layer in the model. Specifically, a unique feature interaction vector is maintained for each field, which contains interaction information of the field and each other fields in any order within K and allow the subsequent DNN model to better explore the non-linear high-order relationship between fields. We employed this method to improve the CTR prediction accuracy on an online recommendation service, a system that recommends ads to users at iQIYI, which is one of the most popular online video apps in China. Extensive experiments on both public datasets and online A/B testing demonstrate the effectiveness and robustness of the proposed method.

The remainder of this paper is structured as follows. In Section 2, we elaborately describe the proposed methodology. In Section 3, we perform comprehensive experiments on three widely used databases. Finally, in Section 4, we draw conclusions.

*Corresponding author.

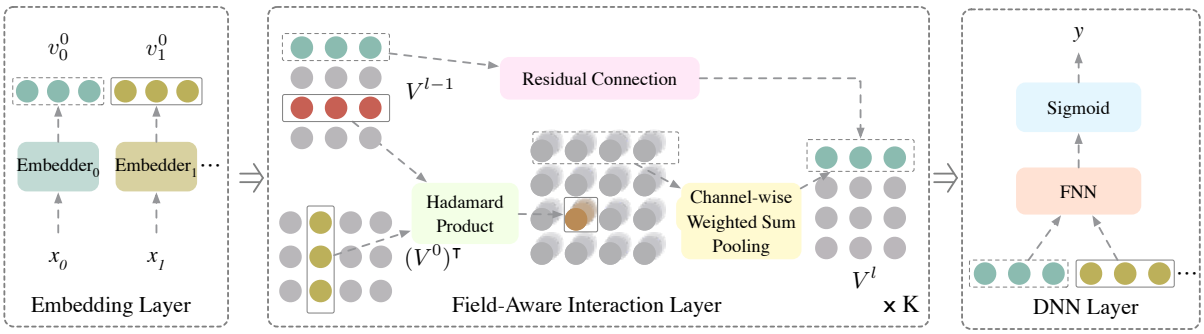

Fig. 1. The architecture of FINT, which consists of: embedding layer, Field-Aware Interaction Layer, and DNN layer.

2. APPROACH

We take CTR as a binary classification task to predict whether a user will click a given item. Specifically, we denote the model input as $X = \{x_0, \cdots, x_{M-1}\}$ which contains M features. X includes not only user features but also item features and context features. Such features could be categorical, such as age and gender, or continuous, such as item price. The gold target of CTR model is a scalar y ($y = 1$ means the user will click the given item, otherwise $y = 0$).

2.1. FINT

Figure 1 shows the overall architecture of our proposed model FINT. It mainly contains three modules: embedding layer, field-aware interaction layer, and DNN layer (dense neural network). The first one aims to embed features into dense vectors. The second one maintains a unique interaction vector for each field, it captures high-order field-aware feature interactions while retaining the low-order field information. The last module targets to exploit non-linear high order feature interaction and predict the user's clicking behavior.

Embedding Layer: The FINT first embeds each feature $x_i \in X$ into a dense vector v_i^0 of dimension D as its initial representation. If the field is multivalent, the sum of feature embedding is used as the field embedding. Here, "0" in the subscript denotes the initial one. We denote the output of embedding layer, i.e., the initial representation of X, as:

$$V_0 = [v_0^0, v_1^0, \cdots, v_{M-1}^0]^\intercal. \tag{1}$$

Field-aware Interaction Layer: This layer aims to promote field-aware interaction between features to explore more possible combinations. Field-awareness means maintaining a unique interaction vector for each field, making further non-linear interaction among fields possible. We stack K field-aware interaction layers in the FINT to achieve high-order feature interaction. As shown in Figure 1, to achieve field-aware interaction, each interaction layer conducts two steps, computing Hadamard product and channel-wise weighted sum pooling. The first step takes the initial representation V^0 and representation of the last step V^{l-1} as input and computes the Hadamard product between all pairs of $v_i^{l-1} \in V^{l-1}$ and $v_j^0 \in V^0$. Then channel-wise weighted sum pooling is conducted along features with the learnable parameter $W^l \in \mathbb{R}^{M \times M}$. To avoid training collapse, we also add residual connection [6] in each field-aware interaction layer. Specifically, representation v_i^l is computed as:

$$v_i^l = \sum_{j=0}^{M-1} w_{i,j}^l \times (v_i^{l-1} \odot v_j^0) + u_i^l v_i^{l-1}, \tag{2}$$

where, $w_{i,j}^l$ is the element of W^l, u_i^l is a scalar, $\odot$ denotes Hadamard product. Equation 2 indicates v_i^l contains combination information of field i and other fields in any order within l. If we consider all representations, Equation 2 can be re-organized in the matrix format:

$$V^l = V^{l-1} \odot (W^l \otimes V^0) + U^l \times V^{l-1}, \tag{3}$$

in which $U^l = [u_0^l, \cdots, u_{M-1}^l]^\intercal$ is a learnable vector parameter for residual connection, $\otimes$ indicates matrix multiplication.

DNN Layer: As the last module of FINT, the DNN layer is to explore non-linear interactions and predict the clicking probability according to the final representations. Concretely, the DNN layer works as:

$$\hat{y} = (\sigma \circ \text{FFN})(v_0^K \| \cdots \| v_{M-1}^K), \tag{4}$$

where $\|$ indicates vector concatenation, $\circ$ indicates function composition, σ is the sigmoid function, FFN is a feed-forward network that contains multiple fully-connection layers with hidden size D_F and active function RELU. Such an architecture allows the DNN layer to explore high-order non-linear feature interaction in the semantic space. In the training stage, we use binary cross entropy as the training loss for FINT. In the inference stage, we take the output $\hat{y}$ as the probability of a user clicking the given item.

Time Complexity: Equation 3 shows that each feature interaction layer can be efficiently computed in $\mathcal{O}(M^2 D)$. For the DNN layer, the vector-matrix multiplication is the

main operation which can be done in $\mathcal{O}(MDD_F + D_F^2)$. Since there are K field-aware interaction layers(K is usually small), the overall time complexity of FINT is $\mathcal{O}(KM^2D + MDD_F + D_F^2)$. Although the feature interaction layer complexity is inferior to the one of traditional machine learning based FM [7] and NFM [8], which is $\mathcal{O}(KMD)$, it surpasses a variety of deep learning based peers, such as xDeepFM [3,5] which is $\mathcal{O}(KM^2DT)$, where T is the number of multiple pooling operations. Moreover, as FINT conducts most operations on matrixes and requires no sequence operations, it can achieve a higher GPU acceleration ratio.

2.2. Relationship with Other Models

FINT shares a similar paradigm with several factorization based models [5, 7–9], as they explore feature interaction in the vector space and exploit pooling operations to reduce dimension and promote further classification. NFM [10] has shown the advantage of integrating linear feature combination and nonlinear high-order feature combination. Therefore, FINT also exploits nonlinear feature interaction through the DNN layer. On the other hand, the field-aware interaction layer makes FINT distinguishable from others. Unlike with previous approaches that cast all feature representations into a single scalar or vector during feature interaction [5, 7, 8], the field-aware interaction layer maintains a vector for each field, retaining their boundaries, to allows the following DNN module to further mine nonlinear interaction. AutoINT [9] is the most related work to FINT in paradigm because it also retains the feature boundary in linear feature interaction and exploits nonlinear high-order feature interaction. However, it is based on the Transformer model [11] and uses the self-attention mechanism to learn feature weights, while FINT uses the Hadamard product, which is a more general and effective method in recommendation systems.

3. EXPERIMENTS

In this section, we aim to provide the experimental results of FINT, from the perspective of efficiency and effectiveness. The prototype of FINT is implemented by TensorFlow 1.14.0 and runs with a Nvidia Tesla P40 GPU. Two metrics Logloss and AUC (Area Under the ROC Curve) are used in our experimental studies. For the training of the FINT model, we set the learning rate as 1e-3 and Adam optimizer is employed in our experiments. The batch size is set 1024, while the embedding size is set as 16. We use 3 field-aware interaction layers. For the DNN layers, the number of hidden layers is set as [300, 300, 300]. We replace the features that appear less than 10 times as "unknown". Numerical features can be normalized with $z^* = log(z + 1) + 1$. The settings of the experimental part are basically kept the same as AutoINT.

3.1. Dataset and Baselines

We evaluate the proposed method on three publicly available datasets including the KDD12 [1], Criteo [2], and Avazu [3]. Criteo and Avazu contain chronologically ordered click-through records from Criteo and Avazu which are two online advertising companies. For the Avazu and Cretio dataset, we randomly split the dataset into training (80%), validation (10%), and test (10%) sets. While for the KDD12, we follow the official public and private split. We implemented 9 widely used CTR prediction approaches, and compared them in the experiment. Below is a brief introduction of these models:

LR, we employ the LR only with basic features as our first baseline.

FM, we employ the original FM model, which has demonstrated its effectiveness in many CTR prediction tasks.

NFM, which aims to encode all feature interactions, through a multi-layer neural network coupled with a bit-wise bi-interaction pooling layer.

PNN, which applies a product layer and multiple fully connected layers to explore the high-order feature interactions.

Wide & Deep, which aims to model low- and high-order feature interactions simultaneously.

DeepFM, which explores the integration of the of FM and deep neural networks (DNN). Through the modeling of low-order feature interactions like FM and models high-order feature interactions like DNN.

AutoInt, which employs a multi-head self-attentive neural network as the core module and can automatically learn the high-order interactions of input features.

DCN, which makes use of the deep cross network and takes the outer product of concatenated feature embeddings to explicitly model feature interaction.

xDeepFM, which has a compressed interaction network to model vector-wise feature interactions for CTR prediction.

3.2. Performance Evaluation

As can be observed from Table 1: the proposed FINT model can provide better performance compared to other models on Criteo, KDD2012, and Avazu on both metrics. The FINT obtains different improvements on three datasets. For example, on the Criteo dataset, FINT surpasses the previous best model (PNN) over 0.08% point on AUC, which is already a considerable improvement in the CTR task. On the Avazu dataset, the FINT achieves a comparable AUC with XDeepFM, only superior with 0.01% point. On the other hand, it outperforms XDeepFM on the Logloss metric. The improvement on the KDD2012 dataset is similar to the one of Criteo.

[1] https://www.kaggle.com/c/kddcup2012-track2
[2] https://www.kaggle.com/c/criteo-display-ad-challenge
[3] https://www.kaggle.com/c/avazu-ctr-prediction

Table 1. Effectiveness comparison of different algorithms.

Model	Criteo		Avazu		KDD12	
	AUC	Logloss	AUC	Logloss	AUC	Logloss
LR	0.7846	0.4670	0.7616	0.3901	0.7352	0.1385
FM	0.7912	0.4627	0.7753	0.3826	0.7419	0.1383
NFM	0.7991	0.4541	0.7761	0.3820	0.7419	0.1378
PNN	0.8069	0.4473	0.7793	0.3802	0.7571	0.1357
DeepFM	0.8014	0.4524	0.7785	0.3806	0.7517	0.1366
Wide/ Deep	0.8042	0.4495	0.7776	0.3811	0.7509	0.1366
AutoInt	0.8053	0.4482	0.7770	0.3813	0.7613	0.1356
DCN	0.8053	0.4483	0.7777	0.3811	0.7546	0.1360
XDeepFM	0.8055	0.4484	**0.7796**	0.3801	0.7531	0.1360
FINT	**0.8077**	**0.4461**	0.7795	**0.3800**	**0.7618**	**0.1355**

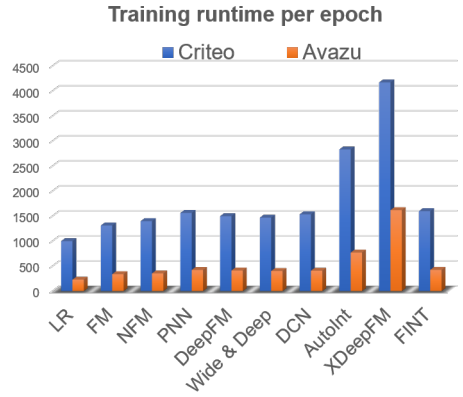

Fig. 2. Time comparison (in seconds).

3.3. Effectiveness Comparison

In Figure 2, we conducted a quantitative comparison on the runtime between FINT and seven state-of-the-art models with GPU implementations on Criteo and Avazu. In the Figure, the y-axis provides an average runtime per epoch over five training epochs after which all models start to converge observably. We keep the hardware settings identical to the aforementioned in the experiment setting session. From the figure, we observe that FINT displays an superior efficiency by spending the minimum time for each epoch among the ten models, while retaining best prediction performance. The main property of FINT enables the huge speedup: the Hadamard product operations across the features can reduce the problem scale (from exponential to linear).

3.4. Results from online A/B testing

Careful online A/B testing in the advertising display system was conducted. iQiYi, as a large video app, has more than 100 million users using it to watch videos every day. Ads are distributed in multiple locations of the app, including video pre-post ads(video general roll), startup screen ads when the app is opened(open screen), short video feed flow ads (infeed), and long video feed flow ads (semi-feed). During almost a month's testing, FINT trained with the proposed FINT contributes up to 2.92% CTR and 3.18% RPM (Revenue Per Mille) promotion (are shown in Table 2.) compared with the baseline models (Wide & Deep). Now FINT has been deployed online and serves the main traffic.

Table 2. A/B testing on different advertising positions.

position	revenue	click rate
overall	+2.72%	+2.92%
video general roll	+1.53%	+0.41%
open screen	+2.67%	+4.11%
infeed	+3.39%	+5.38%
semi-feed	+4.81%	+4.69%

4. CONCLUSION

In this paper, we proposed an efficient and effective CTR predictor named FINT, which aims to learn the high-order feature interactions by employing the Field-aware interaction layer, and captures high-order feature interactions without losing the field-level semantic information. We have conducted extensive experiments on public realistic datasets and A/B testing on large online systems. The obtained results suggest that FINT can learn effective high-order feature interactions, while running faster than state-of-the-art models, meaning a high efficiency on CTR prediction and achieves comparable or even better performances.

5. ACKNOWLEDGEMENT

This work is partially supported by the major Science and Technology Innovation 2030 "New Generation Artificial Intelligence" project 2020AAA0104803.

6. REFERENCES

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

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

[3] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He, "Deepfm: a factorization-machine based neural network for ctr prediction," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 2017, pp. 1725–1731.

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

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

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[7] Steffen Rendle, "Factorization machines," in *2010 IEEE International Conference on Data Mining*. IEEE, 2010, pp. 995–1000.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[9] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang, "Autoint: Automatic feature interaction learning via self-attentive neural networks," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 1161–1170.

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

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.