FiBiNet++: Reducing Model Size by Low Rank Feature Interaction Layer for CTR Prediction

Pengtao Zhang
Sina Weibo
zpt1986@126.com

Zheng Zheng
Brandeis University
zhengzheng@brandeis.edu

Junlin Zhang
Sina Weibo
junlin6@staff.weibo.com

ABSTRACT

Click-Through Rate (CTR) estimation has become one of the most fundamental tasks in many real-world applications and various deep models have been proposed. Some research has proved that FiBiNet is one of the best performance models and outperforms all other models on Avazu dataset. However, the large model size of FiBiNet hinders its wider application. In this paper, we propose a novel FiBiNet++ model to redesign FiBiNet's model structure, which greatly reduces model size while further improves its performance. One of the primary techniques involves our proposed "Low Rank Layer" focused on feature interaction, which serves as a crucial driver of achieving a superior compression ratio for models. Extensive experiments on three public datasets show that FiBiNet++ effectively reduces non-embedding model parameters of FiBiNet by 12x to 16x on three datasets. On the other hand, FiBiNet++ leads to significant performance improvements compared to state-of-the-art CTR methods, including FiBiNet. The source code is in https://github.com/recommendation-algorithm/FiBiNet.

CCS CONCEPTS

• **Information systems → Recommender systems**.

KEYWORDS

Recommender System; Click-Through Rate

ACM Reference Format:
Pengtao Zhang, Zheng Zheng, and Junlin Zhang. 2023. FiBiNet++: Reducing Model Size by Low Rank Feature Interaction Layer for CTR Prediction. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23), October 21–25, 2023, Birmingham, United Kingdom.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3583780.3615242

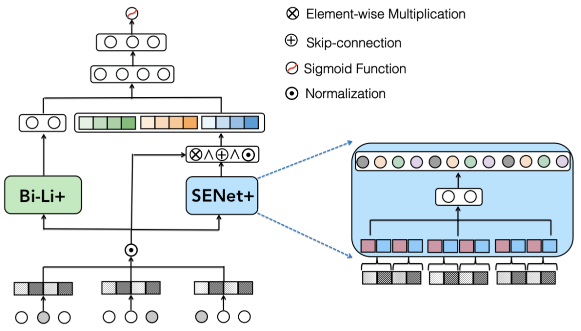


Figure 1: The Framework of FiBiNet++

1 INTRODUCTION

Click-through rate (CTR) prediction plays important role in personalized advertising and recommender systems[4, 5, 7, 11, 13]. In recent years, a series of deep CTR models have been proposed to resolve this problem such as Wide & Deep Learning[2], DeepFM[6], DCN[17], xDeepFM [12], AutoInt[16], DCN v2[18] and FiBiNet[9]. Specifically, Wide & Deep Learning[2] jointly trains wide linear models and deep neural networks to combine the benefits of memorization and generalization for recommender systems. DeepFM[6] replaces the wide part of Wide & Deep model with FM and shares the feature embedding between the FM and deep component. Some works explicitly introduce high-order feature interactions by subnetwork. For example, Deep & Cross Network (DCN)[17] and DCN v2[18] efficiently capture feature interactions of bounded degrees in an explicit fashion. The eXtreme Deep Factorization Machine (xDeepFM) [12] also models the low-order and high-order feature interactions in an explicit way by proposing a novel Compressed Interaction Network (CIN) part. Similarly, FiBiNet[9] dynamically learns the importance of features via the Squeeze-Excitation network (SENET) and feature interactions via bi-linear function.

Though many models have been proposed, seldom works fairly compare these models' performance. FuxiCTR[22] performs open benchmarking for CTR prediction and experimental results[22] show that FiBiNet is one of the best performance models, which outperforms all other 23 models on Avazu dataset.

However, we argue that FiBiNet has too many model parameters, which hinders its wider usage in real-life applications. In real-world systems, both the smaller size and the cost of training and inference times are important factors to be considered. Therefore, our works aims to redesign the model structure to greatly reduce model size while improving its performance.

In this paper, we propose a novel FiBiNet++ model to address these challenges as shown in Figure 1. First, we reconstruct the model structure by removing the bi-linear module on SENet and the linear part in FiBiNet, which directly reduces model parameters. More importantly, we upgrade the bi-linear function into the bi-linear+ module by changing the hadamard product to inner product and bringing a "Low Rank Layer" on feature interaction into it. In Section 3.2, we will demonstrate that the proposed "Low Rank Layer" is primarily responsible for high model compression ratio.

Our inspiration for this approach stems from the LoRA[8], which reveals that LLM models possess a low rank "intrinsic dimension," enabling them to learn effectively even after undergoing random projection into a smaller subspace. We put forth the hypothesis that feature interactions in CTR models similarly exhibit a low intrinsic rank during training and propose incorporating a "Low Rank Feature Interaction Layer" into bi-linear+ modules, which greatly reduces model parameters while keeps model's performance. Finally, we introduce feature normalization and the upgraded SENet+ module to further enhance model performance.

We summarize our major contributions as below: (1) The proposed FiBiNet++ greatly reduces model size of FiBiNet by 12x to 16x on three datasets. (2) Compared with FiBiNet, our proposed FiBiNet++ model increases mode's training and reference efficiencies by +37.50% to 81.03% on three datasets. (3) FiBiNet++ yields remarkable improvements compared to state-of-the-art models.

2 PRELIMINARIES

DNN model is always used as a sub-component in most current DNN ranking systems[3, 6, 9, 12, 14, 17, 19] and it contains two components: feature embedding and MLP.

(1) Feature Embeddding. We map one-hot sparse features to dense, low-dimensional embedding vectors and obtain feature embedding $\mathbf{v}_i$ for one-hot vector $\mathbf{x}_i$ via: $\mathbf{v}_i = \mathbf{W}_e \mathbf{x}_i \in \mathbb{R}^{1\times d}$, where $\mathbf{W}_e \in \mathbb{R}^{n\times d}$ is the embedding matrix of n features and d is the dimension of field embedding.

(2) MLP. To learn high-order feature interactions, multiple feed-forward layers are stacked on the concatenation of dense features represented as $\mathbf{H}_0 = concat[\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_f]$, where f denotes field number. Then, the feed forward process of MLP is $\mathbf{H}_l = ReLU(\mathbf{W}_l \mathbf{H}_{l-1} + \beta_l)$, where l is the depth and ReLU is the activation function. $\mathbf{W}_l, \beta_l, \mathbf{H}_l$ are weighting matrix, bias and output of the l-th layer.

For binary classifications, the loss function of CTR prediction is the log loss:

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

where N is the total number of training instances, y_i is the ground truth of i-th instance and $\hat{y}_i$ is the predicted CTR.

3 OUR PROPOSED MODEL

The architecture of the proposed FiBiNet++ is shown in Figure 1. The original feature embedding is first normalized before being sent to the following components. Then, Bi-linear+ module models feature interactions and SENet+ module computes bit-wise feature importance. The outputs of two branches are concatenated as input of the following MLP layers.

3.1 FiBiNet++

Feature Normalization. We introduce feature normalization into FiBiNet++ to enhance model's training stability as follows:

$$\mathbf{N}(\mathbf{V}) = concat[\mathbf{N}(\mathbf{v}_1), \mathbf{N}(\mathbf{v}_2), ..., \mathbf{N}(\mathbf{v}_f)] \in \mathbb{R}^{1\times fd} \tag{2}$$

where $\mathbf{N}(\cdot)$ is layer normalization[1] for numerical feature and batch normalization[10] operation for categorical feature.

Bi-Linear+ Module. FiBiNet models interaction between feature $\mathbf{x}_i$ and feature $\mathbf{x}_j$ by bi-linear function which introduces an extra learned matrix $\mathbf{W}$ as follows:

$$\mathbf{p}_{i,j} = \mathbf{v}_i \circ \mathbf{W} \otimes \mathbf{v}_j \in \mathbb{R}^{1\times d} \tag{3}$$

where $\circ$ and $\otimes$ denote inner product and element-wise hadamard product, respectively. In order to effectively reduce model size, we upgrade bi-linear function into bi-linear+ module by following two methods. First, the hadamard product is replaced by another inner product as: $\mathbf{p}_{i,j} = \mathbf{v}_i \circ \mathbf{W} \circ \mathbf{v}_j \in \mathbb{R}^{1\times 1}$. We think the feature interaction is rather sparse and one bit information as representation is enough instead of a vector. It's easy to see the parameters of $\mathbf{p}_{i,j}$ decrease greatly from $\mathbf{d}$ dimensional vector to 1 bit for each feature interaction. Suppose the input instance has f fields and we have the following vector after bi-linear feature interaction:

$$\mathbf{P} = concat[\mathbf{p}_{1,2}, \mathbf{p}_{1,3}, ..., \mathbf{p}_{f-1,f}] \in \mathbb{R}^{1\times \frac{f\times(f-1)}{2}} \tag{4}$$

Inspired by LoRA[8], which has demonstrated that LLM models possess a low "intrinsic dimension" and exhibit efficient learning despite undergoing random projections to smaller subspaces, we posit that updates to the feature interaction layer during training also exhibit a low "intrinsic rank." Therefore, we propose integrating a thin "Low Rank Layer" into Bi-Linear+, thereby reducing model parameters significantly while maintaining optimal model performance. Specifically, we introduce "Low Rank Layer" stacking on feature interaction vector $\mathbf{P}$ as follows:

$$\mathbf{H}^{LRL} = \sigma_1(\mathbf{W}_1 \mathbf{P}) \in \mathbb{R}^{1\times m} \tag{5}$$

where $\mathbf{W}_1 \in \mathbb{R}^{m\times \frac{f\times(f-1)}{2}}$ is a learning matrix of thin MLP layer with small size m and $\sigma_1(\cdot)$ is an identity function. "Low Rank Layer" projects feature interactions from sparse space into low rank space to greatly reduce the storage.

SENet+ Module. SENet+ module consists of four phases: squeeze, excitation, re-weight and fuse. **(1) Squeeze.** SENet collects one bit information by mean pooling from each feature embedding as "summary statistics". However, we improve the original squeeze step by providing more useful information. Specifically, we first segment each normalized feature embedding $\mathbf{v}_i \in \mathbb{R}^{1\times d}$ into $\mathbf{g}$ groups, which is a hyper-parameter, as: $\mathbf{v}_i = concat[\mathbf{v}_{i,1}, \mathbf{v}_{i,2}, ..., \mathbf{v}_{i,g}]$, where $\mathbf{v}_{i,j} \in \mathbb{R}^{1\times \frac{d}{g}}$ means information in the j-th group of the i-th feature. Let $\mathbf{k} = \frac{d}{g}$ denotes the size of each group. Then, we select the max value $\mathbf{z}_{i,j}^{max}$ and average pooling value $\mathbf{z}_{i,j}^{avg}$ in $\mathbf{v}_{i,j}$ as representation of the group as: $\mathbf{z}_{i,j}^{max} = \max_t \left\{\mathbf{v}_{i,j}^t\right\}_{t=1}^{k}$ and $\mathbf{z}_{i,j}^{avg} = \frac{1}{k}\sum_{t=1}^{k} \mathbf{v}_{i,j}^t$. The concatenated representative information of each group forms the "summary statistic" $\mathbf{Z}_i$ of feature embedding $\mathbf{v}_i$:

$$\mathbf{Z}_i = concat[\mathbf{z}_{i,1}^{max}, \mathbf{z}_{i,1}^{avg}, \mathbf{z}_{i,2}^{max}, \mathbf{z}_{i,2}^{avg}, ..., \mathbf{z}_{i,g}^{max}, \mathbf{z}_{i,g}^{avg}] \in \mathbb{R}^{1\times 2g} \tag{6}$$

Finally, we can concatenate each feature's summary statistic $\mathbf{Z} = concat[\mathbf{Z}_1, \mathbf{Z}_2, ..., \mathbf{Z}_f] \in \mathbb{R}^{1\times 2gf}$ as the input of SENet+ module.

(2) Excitation. The excitation step in SENet computes each feature's weight according to the statistic vector $\mathbf{Z}$, which is a field-wise attention. However, we improve this step by changing the field-wise attention into a more fine-grained bit-wise attention. Similarly, we also use two fully connected (FC) layers to learn the weights as follows: $\mathbf{A} = \sigma_3(\mathbf{W}_3 \sigma_2(\mathbf{W}_2 \mathbf{Z})) \in \mathbb{R}^{1\times fd}$, where

$\mathbf{W}_2 \in \mathbb{R}^{\frac{2gf}{r} \times 2gf}$ denotes learning parameters of the first FC layer, which is a thin layer and r is reduction ratio. $\mathbf{W}_3 \in \mathbb{R}^{fd \times \frac{2gf}{r}}$ means learning parameters of the second FC layer, which is a wider layer with a size of fd. Here $\sigma_2(\cdot)$ is $ReLu(\cdot)$ and $\sigma_3(\cdot)$ is an identity function without non-linear transformation.In this way, each bit in input embedding can dynamically learn the corresponding attention score provided by $\mathbf{A}$.

(3) Re-Weight. Re-weight step does element-wise multiplication between the original field embedding and the learned attention scores as follows: $\mathbf{V}^w = \mathbf{A} \otimes \mathbf{N}(\mathbf{V}) \in \mathbb{R}^{1 \times fd}$, where $\otimes$ is an element-wise multiplication between two vectors and $\mathbf{N}(\mathbf{V})$ denotes original embedding after normalization.

(4) Fuse. An extra "fuse" step is introduced in order to better fuse the information contained both in original feature embedding and weighted embedding. Specifically, we first use skip-connection to merge two embedding as follows: $\mathbf{v}_i^s = \mathbf{v}_i^o \oplus \mathbf{v}_i^w$, where $\mathbf{v}_i^o$ donates the i-th normalized feature embedding, $\mathbf{v}_i^w$ denotes embedding after re-weight step, $\oplus$ is an element-wise addition operation. Then, another feature normalization is applied on feature embedding $\mathbf{v}_i^s$ for a better representation: $\mathbf{v}_i^u = \mathbf{LN}(\mathbf{v}_i^s)$. Note we adopt layer normalization here no matter what type of feature it belongs to, numerical or categorical feature. Finally, we concatenate all the fused embeddings as the output of the SENet+ module:

$$\mathbf{V}^{SENet+} = concat[\mathbf{v}_1^u, \mathbf{v}_2^u, ..., \mathbf{v}_f^u] \in \mathbb{R}^{1 \times fd} \quad (7)$$

Concatenation Layer. We concatenate the output of two branches to form the input of the following MLP layers:

$$\mathbf{H}_0 = concat[\mathbf{H}^{LRL}, \mathbf{V}^{SENet+}] \quad (8)$$

3.2 Discussion

In this section, we discuss the model size difference between FiBiNet and FiBiNet++. Note only non-embedding parameter is considered, which really demonstrates model complexity.

The major parameter of FiBiNet comes from two components: one is the connection between the first MLP layer and the output of two bi-linear modules, and the other is the linear part. Suppose we denote $h = 400$ as the size of the first MLP layer, $f = 50$ as field number, $d = 10$ as feature embedding size, and $t = 1$ *million* as feature number. Therefore, the parameter number in these two parts is nearly 10.8 million:

$$\mathbf{T}^{FiBiNet} = \underbrace{\mathbf{f \times (f-1) \times d \times h} +}_{MLP\ and\ bi-linear} \underbrace{t}_{linear} = 10.8\ millions \quad (9)$$

For FiBiNet++, the majority of model parameter comes from following three components: the connection between the first MLP layer and embedding produced by SENet+ module(1-th part), the connection between the first MLP layer and "Low Rank Layer"(2-th part), and parameters between "Low Rank Layer" and bi-linear feature interaction results(3-th part). Let $m = 50$ denote the size of "Low Rank Layer". We have the parameter number of these components as follows:

$$\mathbf{T}^{FiBiNet++} = \underbrace{\mathbf{f \times d \times h} +}_{1-th\ part} \underbrace{\mathbf{m \times h}}_{2-th\ part} + \underbrace{\frac{\mathbf{f \times (f-1)}}{\mathbf{2}} \times \mathbf{m}}_{3-th\ part} = 0.28\ millions \quad (10)$$

Table 1: Overall performance (AUC) of different models

	Avazu		Criteo		KDD12	
Model	AUC(%)	Paras.	AUC(%)	Paras.	AUC(%)	Paras.
FM	78.17	1.54M	78.97	1.0M	77.65	5.46M
DNN	78.67	<u>0.74M</u>	80.73	<u>0.48M</u>	79.54	<u>0.37M</u>
DeepFM	78.64	2.29M	80.58	1.48M	79.40	5.84M
xDeepFM	78.88	4.06M	80.64	4.90M	79.51	6.91M
DCN	78.68	0.75M	80.73	0.48M	79.58	0.37M
AutoInt+	78.62	0.77M	80.78	0.48M	<u>79.69</u>	0.38M
DCN v2	78.98	4.05M	<u>80.88</u>	0.65M	79.66	0.39M
FiBiNet	<u>79.12</u>	10.27M	80.73	7.25M	79.52	6.41M
FiBiNet++	**79.15**	0.81M	**81.10**	0.56M	**79.98**	0.40M
Improv.	+0.03	12.7x	+0.37	12.9x	+0.46	16x

We can see that the above-mentioned methods to reduce model size greatly decrease model size from 10.8 million to 0.28 million, which is nearly 39x model compression. In addition, the larger the field number f is, the larger the model compression ratio we can achieve. It's easy to see that "Low Rank Layer" is the key to the high compression ratio while it can also be applied into other CTR models with long feature interaction layers such as ONN[20] and FAT-DeepFFM[21] for feature interaction compression.

4 EXPERIMENTAL RESULTS

4.1 Experiment Setup

Datasets Three datasets are used in our experiments and we randomly split instances by 8:1:1 for training, validation and testing: **(1) Criteo[1]:** As a display ad dataset, there are 26 anonymous categorical fields and 13 continuous feature fields. **(2) Avazu[2]:** The Avazu dataset contains 23 fields that indicate elements of a single ad impression. **(3) KDD12[3]:** KDD12 dataset has 13 fields spanning from user id to ad position for a clicked data.

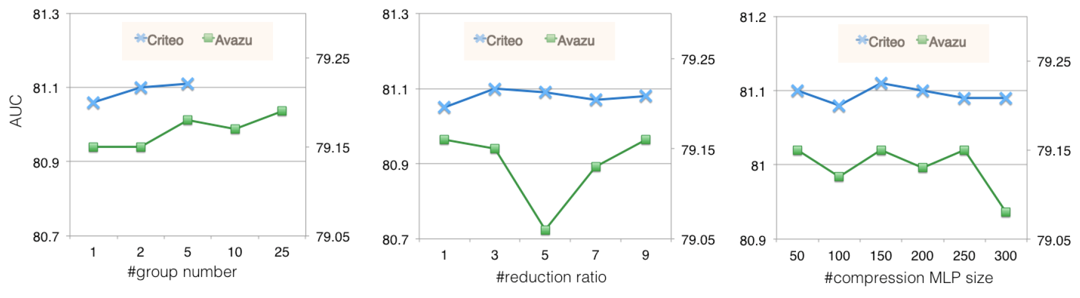
Models for Comparisons We compare the performance of the FM[15], DNN, DeepFM[6], DCN [17], AutoInt [16], DCN V2 [18], xDeepFM [12] and FiBiNet [9] models as baselines and AUC is used as the evaluation metric.

Implementation Details For the optimization method, we use the Adam with a mini-batch size of 1024 and 0.0001 as learning rates. We make the dimension of field embedding for all models to be a fixed value of 10 for Criteo dataset, 50 for Avazu dataset and 10 for KDD12 dataset. For models with DNN part, the depth of hidden layers is set to 3, the number of neurons per layer is 400, and all activation functions are ReLU. In SENet+, the reduction ratio is set to 3 and the group number is 2 as the default settings. In Bi-linear+ module, we set size of the low rank layer as 50.

4.2 Results and Analysis

Performance Comparison. Table 1 shows the performances of different SOTA baselines and FiBiNet++. The best results are in bold, and the best baseline results are underlined. We can see that:(1) FiBiNET++ model outperforms all the compared SOTA methods and achieves the best performance on all three benchmarks. (2) Among

[1]Criteo http://labs.criteo.com/downloads/download-terabyte-click-logs/
[2]Avazu http://www.kaggle.com/c/avazu-ctr-prediction
[3]KDD12 https://www.kaggle.com/c/kddcup2012-track2



(a) Effect of Group Number (b) Effect of Reduction Ratio (c) Effect of Compression MLP Size

Figure 2: Effect of Hyper-Parameters

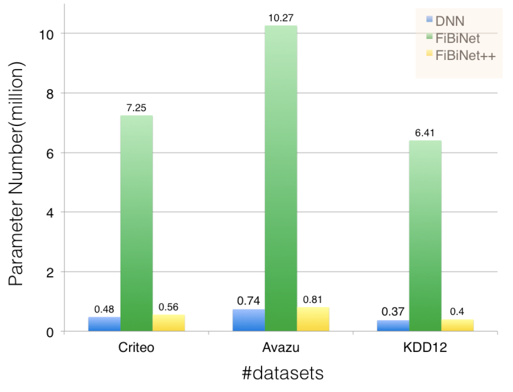


Figure 3: Comparison of Model Size

Table 2: Training and reference efficiency comparison

	Avazu		Criteo		KDD12	
Model	Train(ms)	Refer(ms)	Train(ms)	Refer(ms)	Train(ms)	Refer(ms)
FiBiNet	97	12	191	58	33	8
FiBiNet++	40	7	72	11	20	5
Improv.	+58.76%	+41.66%	+62.30%	+81.03%	+39.39%	+37.50%

all the strong baselines with a similar amount of parameters such as Autoint+, DCN v2 and DeepFM, FiBiNet++ is the best performance model on all three datasets. This demonstrates the reason why FiBiNet++ outperforms other models is because of its designed components instead of more parameter. (3) Compared with FiBiNet model, FiBiNet++ can achieve better performance on all datasets though it has much fewer parameters, which indicates that our proposed methods to enhance model performance are effective.

Model Size Comparison. We compare the non-embedding model size of different methods in Table 1 and Figure 3. FiBiNet++ provides orders of magnitude improvement in model size while improving the quality of the model compared with FiBiNet. Specifically, FiBiNet++ reduce the model size of FiBiNet by 12.7x, 12.9x and 16x in terms of the number of parameters on three datasets, respectively, which demonstrates that our proposed methods to reduce model parameter in this paper are effective. Now FiBiNet++ has a comparable model size with DNN model while outperforms all other models on three datasets at the same time.

Training/Reference Efficiency. Efficiency is an essential concern in industrial applications and we conduct experiments to compare the training and inference time between our proposed FiBiNet++ and FiBiNet. We leverage time(millisecond) of processing one batch of examples during the training and reference as an efficiency metric. The average training and inference times of the two models are illustrated in Table 2. Compared with FiBiNet model, the training efficiency of FiBiNet++ increases by 58.76%, 62.30% and 39.39% while reference efficiency increases by 41.66%, 81.03% and 37.50% on three datasets respectively. Our FiBiNet++ model shows a significant advantage in training and inference efficiency, which makes it more practical to be applied in real life.

Hyper-Parameters of FiBiNet++. Next, we study hyperparameter sensitivity of FiBiNet++. **(1) Group Number.** Figure 2a shows a slight performance increase with the increase of group number, which indicates that more group number benefits model performance because we can input more useful information in feature embedding into SENet+ module.

(2) Reduction Ratio. We conduct some experiments to adjust the reduction ratio in SENet+ module from 1 to 9 and Figure 2b shows the result. It can be seen that the performance is better if we set the reduction ratio to 3 or 9. **(3) Size of Low Rank Layer.** The results in Figure 2c show the impact when we adjust the size of the Low Rank Layer in bi-linear+ module. We can observe that the performance begins to decrease when the size is set greater than 150, which demonstrates the feature interaction represented in low rank space indeed works.

5 CONCLUSION

In this paper, we propose FiBiNet++ model in order to greatly reduce the model size while improving the model performance. Experimental results show that FiBiNet++ provides orders of magnitude improvement in model size while improving the quality of the model.

REFERENCES

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

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

[3] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*. ACM, 191–198.

[4] Wei Deng, Junwei Pan, Tian Zhou, Deguang Kong, Aaron Flores, and Guang Lin. 2021. Deeplight: Deep lightweight feature interactions for accelerating ctr predictions in ad serving. In *Proceedings of the 14th ACM international conference on Web search and data mining*. 922—-930.

[5] Thore Graepel, Joaquin Quinonero Candela, Thomas Borchert, and Ralf Herbrich. 2010. Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft’s bing search engine. Omnipress.

[6] 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).

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

[8] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685* (2021).

[9] 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, RecSys 2019, Copenhagen, Denmark, September 16-20, 2019*. ACM, 169–177. https://doi.org/10.1145/3298689.3347043

[10] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).

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

[12] 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*. ACM, 1754–1763.

[13] H. Brendan McMahan, Gary Holt, D. Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, and et al. 2013. Ad Click Prediction: A View from the Trenches. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Chicago, Illinois, USA) *(KDD ’13)*. Association for Computing Machinery, New York, NY, USA, 1222–1230. https://doi.org/10.1145/2487575.2488200

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

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

[16] 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.

[17] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD’17*. ACM, 12.

[18] Ruoxi Wang, Rakesh Shivanna, Derek Zhiyuan Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed H. Chi. 2021. DCN V2: Improved Deep & Cross Network and Practical Lessons for Web-scale Learning to Rank Systems. In *WWW ’21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*. 1785–1797.

[19] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. 2017. Attentional factorization machines: Learning the weight of feature interactions via attention networks. *arXiv preprint arXiv:1708.04617* (2017).

[20] Yi Yang, Baile Xu, Shaofeng Shen, Furao Shen, and Jian Zhao. 2020. Operation-aware Neural Networks for user response prediction. *Neural Networks* 121 (2020), 161–168. https://doi.org/10.1016/j.neunet.2019.09.020

[21] Junlin Zhang, Tongwen Huang, and Zhiqi Zhang. 2019. FAT-DeepFFM: Field Attentive Deep Field-aware Factorization Machine. In *Industrial Conference on Data Mining*. https://api.semanticscholar.org/CorpusID:155099971

[22] Jieming Zhu, Jinyang Liu, Shuai Yang, Qi Zhang, and Xiuqiang He. 2020. FuxiCTR: An Open Benchmark for Click-Through Rate Prediction. *ArXiv* abs/2009.05794 (2020).