FuxiCTR: An Open Benchmark for Click-Through Rate Prediction

Jieming Zhu
Huawei Noah's Ark Lab
Shenzhen, China
jmzhu@ieee.org

Jinyang Liu
The Chinese University of Hong Kong
Hong Kong, China
jy.liu@link.cuhk.edu.hk

Shuai Yang
Peking University
Beijing, China
ethanyang@pku.edu.cn

Qi Zhang
Huawei Noah's Ark Lab
Shenzhen, China
zhangqi193@huawei.com

Xiuqiang He
Huawei Noah's Ark Lab
Shenzhen, China
hexiuqiang1@huawei.com

ABSTRACT

In many applications, such as recommender systems, online advertising, and product search, click-through rate (CTR) prediction is a critical task, because its accuracy has a direct impact on both platform revenue and user experience. In recent years, with the prevalence of deep learning, CTR prediction has been widely studied in both academia and industry, resulting in an abundance of deep CTR models. Unfortunately, there is still a lack of a standardized benchmark and uniform evaluation protocols for CTR prediction. This leads to the non-reproducible and even inconsistent experimental results among these studies. In this paper, we present an open benchmark (namely FuxiCTR) for reproducible research and provide a rigorous comparison of different models for CTR prediction. Specifically, we ran over 4,600 experiments for a total of more than 12,000 GPU hours in a uniform framework to re-evaluate 24 existing models on two widely-used datasets, Criteo and Avazu. Surprisingly, our experiments show that many models have smaller differences than expected and sometimes are even inconsistent with what reported in the literature. We believe that our benchmark could not only allow researchers to gauge the effectiveness of new models conveniently, but also share some good practices to fairly compare with the state of the arts. We will release all the code and benchmark settings[1].

1 INTRODUCTION

In many applications such as recommender systems, online advertising, and product search, click-through rate (CTR) is a key factor in business valuation, which measures the probability of a user clicking or interacting with a candidate item. CTR prediction is extremely important because, for applications with a large user base, even a small improvement in prediction accuracy can potentially lead to a large increase in the overall revenue. However, making accurate CTR prediction remains a great challenge. In contrast to other data types such as images and texts, data in CTR prediction problems are usually of large scale and high sparsity, involving many categorical features of different fields (e.g., billions of samples with millions of features in app recommendation of Google Play [4]).

The importance and unique challenges of CTR prediction have attracted a lot of research efforts from both academia and industry, leading to numerous models ranging from simple logistic regression (LR) [25, 33], factorization machines (FM) [15, 32], to deep neural networks (DNN) [6]. In particular, with the recent success of deep learning, many deep models have been proposed, such as Wide&Deep [4], DeepFM [10], DCN [38], and xDeepFM [19], in order to make more and more accurate CTR prediction.

Unfortunately, there is still a lack of a standardized benchmark and uniform evaluation protocols for CTR prediction. This leads to the non-reproducible and even inconsistent experimental results among these studies. Motivated by this, in this paper, we build an open benchmark (namely FuxiCTR) for reproducible research and provide a rigorous comparison of different models for CTR prediction. Specifically, we ran over 4,600 experiments for more than 12,000 GPU hours in a uniform framework to re-evaluate 24 existing models on two widely-used datasets, Criteo and Avazu. Our experiments show somewhat surprising results: many models have smaller differences than expected and sometimes are even inconsistent with what reported in the literature. A recent study [7] also pointed out similar problems, in terms of reproducibility of the results and optimization of the baselines used for comparison, in current research on recommender systems. In contrast to this work, we take one step further to build an open benchmark as well as release the most comprehensive benchmarking results to facilitate future research. In particular, we identify the key points to performance tuning, which are valuable for new beginners. We believe that our benchmark, along with the results, could not only allow researchers to gauge the effectiveness of new models conveniently, but also share some good practices to fairly compare with the state of the art. We will release all the code and benchmark settings on Github.

In summary, the main contributions of this work are as follows:

- To the best of our knowledge, our work presents the first open benchmark for CTR prediction.
- We release the most comprehensive benchmarking results and will open source all the benchmark code and configuration files for reproducible research.

[1]The link will be open soon.

Manuscript, July 2017, Washington, DC, USA
2020. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

- Our bechmarking results show the non-reproducibility and inconsistency issues in existing studies, and shed light on future research on CTR prediction.

The remainder of this paper is organized as follows. Section 2 introduces the background of CTR prediction. Section 3 describes the details of our FuxiCTR benchmark as well as the benchmarking results. We then make some discussions in Section 4 and review the related work in Section 5. We finally conclude the paper in Section 6.

2 CTR PREDICTION

In this section, we provide an overview of CTR prediction and then briefly review some of the representative models.

2.1 Overview

The objective of CTR prediction is to predict the probability a user will click a given item. However, how to make accurate CTR prediction remains a challenging research problem. In contrast to other data types, such as images and texts, data in CTR prediction problems are large-scale and highly sparse. In general, CTR prediction is formulated as a binary-classification problem and consists of the following key parts.

2.1.1 Feature Embedding. Features in each field may be categorical, numeric, or multi-valued (e.g., multiple tags of an item). Since most features are sparse and high-dimensional after one-hot or multi-hot encoding, it is common to apply feature embedding to mapping these features into low-dimensional dense vectors. We summarize the embedding process of the three types of features in the following.

- *Categorical*: For a categorical feature field i, given a one-hot feature vector x_i, we have its embedding as $e_i = V_i x_i$, where the embedding matrix $V_i \in \mathbb{R}^{dxn}$ has vocabulary size n and embedding dimension d.
- *Numeric*: For a numeric feature field j, there are two typical choices for feature embedding: 1) one can bucketize numeric values into discrete features (e.g., age 1319 as teenager) and then embedding them as categorical features; 2) Given a normalized scalar value x_j, we set its embedding as $e_j = v_j x_j$, where $v_j \in \mathbb{R}^d$ is the embedding vector of field j.
- *Multi-valued*: For a multi-valued field h, each feature can be represented as a sequence. We obtain its embedding as $e_h = V_h[x_{h1}, x_{h2}, \ldots, x_{hk}] \in \mathbb{R}^{dxk}$, given x_{hk} as a one-hot encoded vector of the sequence element and k denoting the maximal length of the sequence. Then the embedding e_h can be further aggregated to a d-dimensional vector, e.g., through mean/sum pooling.

2.1.2 Feature Interaction. It is straightforward to apply a classification model for CTR prediction after feature embedding. Nevertheless, for CTR prediction tasks, interactions between features are central to boost classification performance. In factorization machines (FM) [32], inner products are shown as an effective way to capture pairwise feature interactions. Since the success of FM, a large body of work has been done to capture interactions among features in different manners. Typical examples include inner product and outer product layers in PNN [30], Bi-interaction in NFM [11], crossing network in DCN [38], compressed interaction in xDeepFM [19], convolution in FGCNN [21], circular convolution in HFM [36], bilinear interaction in FiBiNET [14], self-attention in AutoInt [35], graph neural network in FiGNN [18], hierarchical attention in InterHAt [17], etc.

2.1.3 Loss Function. The binary cross-entropy loss function is widely used in CTR prediction models, which is defined as follows:

$$\mathcal{L} = -\frac{1}{N} \sum_{(\hat{y}, y) \in \mathbb{D}} \left(y log\hat{y} + (1 - y) log(1 - \hat{y}) \right), \tag{1}$$

where $\mathbb{D}$ is the training set with N samples. y and $\hat{y}$ denote the ground truth and the estimated click probability, respectively. We define $\hat{y} = \sigma(\phi(x))$, where $\phi(x)$ represents the model function that outputs the logit value given input features x and $\sigma(\cdot)$ is the sigmoid function to map $\hat{y}$ to $[0, 1]$.

2.2 Representative Models

In this section, we summarize the representative models that we have reproduced and evaluated in this work.

2.2.1 Shallow Models. It is common that industrial CTR prediction tasks have large-scale data. Therefore, shallow models have been in widespread use due to their simplicity and efficiency. Even today, LR [33] and FM [32] are still two strong baseline models deployed in industry. We describe the shallow models as follows:

- **LR**. Logistic regression (LR) is a simple baseline model for CTR prediction [33]. With the online learning algorithm, FTRL [25], proposed by Google, LR has been widely adopted in industry.
- **FM**. While LR fails to capture non-linear feature interactions, Rendle et al. propose factorization machine (FM) [32] that embeds features into dense vectors and models pairwise feature interactions as inner products of the corresponding embedding vectors.
- **FFM**. Field-aware factorization machine (FFM) [15] is an extension of FM that considers field information for feature interactions. It was a winner model in several contests on CTR prediction.
- **HOFM**. Since FM only captures second-order feature interactions, HOFM [2] aims to extend FM to high-order factorization machines. However, it results in exponential feature combinations that consume huge memory and take long running time.
- **FwFM**. Recently, Pan et al. [28] extends FM by considering the weights of features interactions. Compared with FFM, it achieves comparable performance but uses much less model parameters.
- **LorentzFM**. LorentzFM [40] has recently been proposed to embed features into a hyperbolic space and model feature interactions via triangle inequality of Lorentz distance.

2.2.2 Deep Models. With the success of deep learning, deep neural networks have been widely studied for CTR prediction. Compared with shallow models, deep models are powerful in capturing sophisticated high-order feature interactions with nonlinear activation functions and thus yield better performance.

- **DNN**. DNN is a straightforward deep model reported in [6], which applies a fully-connected network (termed DNN) after the concatenation of feature embeddings for CTR prediction.

Table 1: Reproducibility requirements met by existing studies (✓ | – | ✗ means totally | partially | not met, respectively)

Reproducibility requirements	xDeepFM	FGCNN	AutoInt+	FiGNN	ONN	FiBiNET	LorentzFM	AFN+	InterHAt	FuxiCTR
Data partition	–	–	✓	–	–	–	–	–	✓	✓
Data preprocessing	✗	✗	✓	✗	–	✗	–	✗	✓	✓
Model code	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓
Training code	✓	✗	✓	✓	✗	✗	✗	✓	✓	✓
Model hyper–parameters	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Baseline hyper–parameters	✓	✓	–	–	✗	–	–	–	✗	✓
Baseline code	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓

- **CCPM**. CCPM [22] reports the first attempt to use convolution for CTR prediction, where feature embeddings are aggregated hierarchically through convolution networks.
- **Wide&Deep**. Wide&Deep [4] is a general learning framework proposed by Google that combines a wide (or shallow) network and deep network to achieve the advantages of both.
- **PNN**. PNN [30] is a product-based network that feed the inner (or outer) products of features embeddings as the input of DNN.
- **DeepCross**. Inspired by residual networks, [34] propose deep crossing that adds residual connections between layers of DNNs.
- **NFM**. Similar to PNN, NFM [11] proposes a Bi-interaction layer that pools the pairwise feature interactions to a vector and then feed it to a DNN for CTR prediction.
- **AFM**. Instead of treating all feature interactions equally as in FM, AFM [39] learns the weights of feature interactions via attentional networks. Different from FwFM, AFM adjusts the weights dynamically according to the input data sample.
- **DeepFM**. DeepFM [10] is an extension of Wide&Deep that substitutes LR with FM to explicitly model second-order feature interactions.
- **DCN**. In DCN [38], a cross network is proposed to perform high-order feature interactions in an explicit way. In addition, it also integrates a DNN network following the Wide&Deep framework.
- **xDeepFM**. While high-order feature interactions modelled by DCN are bit-wise, xDeepFM [19] proposes to capture high-order feature interactions in a vector-wise way via a compressed interaction network (CIN).
- **HFM+**. HFM [36] proposes holographic representation and computes compressed outer products via circular convolution to model pairwise feature interactions. HFM+ further integrates a DNN network with HFM.
- **FGCNN**. FGCNN [21] applies convolution networks and recombination layers to generate additional combinatorial features to enrich existing feature representations.
- **AutoInt+**. AutoInt [35] leverages self-attention networks to learn high-order features interactions. AutoInt+ integrates AutoInt with a DNN network.
- **FiGNN**. FiGNN [18] leverages the message passing mechanism of graph neural networks to learn high-order features interactions.
- **ONN**. ONN (a.k.a., NFFM) [41] is a model built on FFM. It feeds the interaction outputs from FFM to a DNN network for CTR prediction.
- **FiBiNET**. FiBiNET [14] leverages squeeze-excitation network to capture important features, and proposes bilinear interactions to enhance feature interactions.
- **AFN+**. AFN [5] applies logarithmic transformation layers to learn adaptive-order feature interactions. AFN+ further integrates AFN with a DNN network.
- **InterHAt**. InterHAt [17] employs hierarchical attention networks to model high-order feature interactions in an efficient manner.

Although we enumerate only a part of the existing models here, they cover a wide spectrum of studies on CTR prediction. For completeness, we introduce some others in the related work in Section 5.

3 FUXICTR BENCHMARK

In this section, we first describe the motivation of our work, and then present our evaluation protocol of FuxiCTR. Finally, we report the benchmarking results of 24 models on two widely-used datasets.

3.1 Motivation of FuxiCTR

CTR prediction has been widely studied in recent years. Unfortunately, there is a lack of rigour on reproducibility in current studies. As shown in Table 1, many of them fail to meet all of the reproducibility requirements.

- **Data partition and preprocessing**: Most work split the data randomly, but no details (e.g. random seed) are open. Even more, data preprocessing steps may be missing, making it difficult for others to reproduce the data.
- **Model code and training code**: Some studies open source their models. Some others have been implemented by third parties on Github. But in many cases, we can only found the model code, but not the training code.
- **Model hyper-parameters**: Most studies describe the hyperparameters of their own models clearly. But the hyper-parameters are usually tuned on an unknown data split.
- **Baseline hyper-parameters and baseline code**: Many studies report the details of their own model, but pay little attention to the baseline models. We noticed that existing studies seldom open source the baseline models they used, which makes it non-trivial to fairly compare their performance.

To facilitate reproducible research and fairness of comparison, in this work, we aim to provide a standard benchmark for CTR prediction as well as the most comprehensive benchmarking results.

Table 2: Dataset statistics

Dataset	#Instances	#Fields	#Features	Positive Ratio
Criteo	46M	39	5.55M	26%
Avazu	40M	24	8.37M	17%

3.2 Evaluation Protocol

3.2.1 Datasets. We use two real-world datasets for our evaluation: Criteo[2] and Avazu[3]. Both of them are open datasets released by two leading ad companies, and have been widely used in the previous work (e.g., [10, 19, 35, 38]). We choose them because they are collected or sampled from real click logs in production, making the benchmarking results meaningful to industrial applications. Besides, they both have tens of millions of samples to train a deep CTR prediction model. Table 2 summarizes the data statistical information.

3.2.2 Data partition. Since most of the existing studies , we follow them to randomly split both Criteo and Avazu into 8:1:1 as the training set, validation set, and test set, respectively. To make it exactly reproducible and easy to compare with existing work, we use the code provided by AutoInt [35] and control the random seed (i.e., seed=2018) for splitting.

3.2.3 Data preprocessing. We mostly follow the work in [35] to preprocess the data. However, we found some defects in [35] that finally make our process slightly different.

- **Criteo**. The Criteo dataset consists of ad click data over a week. It comprises 26 categorical feature fields and 13 numerical feature fields. For categorical fields, we replace infrequent features (we set less than 2 after tuning) with a default "<UNK>" feature. For numerical fields, we follow the winner solution of the Criteo contest to discretize each integer value x to $\lfloor \log^2(x) \rfloor$, if $x > 2$; $x = 1$ otherwise. In contrast, numeric values are only normalized in [35].
- **Avazu**. Avazu contains 10 days of click logs. It has a total of 23 fields with categorical features such as app id, app category, device id, etc. Especially, we remove the sample_id field that is useless for CTR prediction. But it is ignored and retained by [35]. In addition, we transform and expand the timestamp field into three fields: hour, weekday, and is_weekend. For all categorical fields, we replace infrequent features (we set less than 1 after tuning) with a default "<UNK>" feature.

3.2.4 Evaluation metrics. We employ two commonly-used metrics, AUC and logloss, to evaluate the performance of each CTR prediction model.

- **AUC** is the Area Under the ROC Curve, a common metric to measure the probability that a randomly chosen positive sample is ranked higher than a randomly chosen negative sample. Higher AUC indicates better CTR prediction performance.
- **Logloss**, also known as the binary cross-entropy loss, is defined as Equation 1. Lower logloss indicates better CTR prediction performance.

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

It is worth noting that, when considering a large user base, an improvement of AUC at 1‰ level is generally considered as practically significant for an industrial CTR prediction task. This fact has been recognized by several existing studies from Google [4, 10, 38], Microsoft [20], and Huawei [10].

3.2.5 Implementation of FuxiCTR. The benchmark code of FuxiCTR consists of the following parts: 1) The data processing part reads the raw data from csv, transforms all features into numpy arrays and stores the transformed data into the hdf5 format. 2) All the models under study are implemented in Pytorch. 3) The training part is implemented to read batches of data, compute forward and backward passes, and perform early stopping and learning rate scheduling if necessary. 4) The hyper-parameter tuning part provides a configurable interface to allow grid search of hyper-parameters given by users. We integrate all parts as a unified framework, allowing researchers to easily reuse our code, build new models, or adding new datasets.

3.2.6 Training details and hyper-parameters tuning. During training, we apply the Reduce-LR-on-Plateau scheduler to reduce learning rate by a factor of 10 when the given metric stops improving. To avoid overfitting, early stopping is employed when the metric on the validation set stops improving in two consecutive epochs. The default learning rate is $1.e-3$. The batch size is initially set to 10000 and then decreases gradually using [5000, 2000, 1000] if an OOM error occurs in CUDA. We found that for CTR prediction models trained on millions of samples, using a large batch size usually makes the model run much faster and converge to a better result. Given the large number of features, feature embeddings usually take up most of the model parameters. For fairness of comparison, we keep the default embedding size to 40. We also empirically study the impact of embedding size in Section 3.4. We also found that regularization weight makes a large effect on the prediction performance. Thus, we carefully tuned it among [0, 1.e-8, 1.e-7, 1.e-6, 1.e-5, 1.e-4, 1.e-3]. For the other hyper-parameters of all the models under study, we also carefully tuned within a range. To avoid exponential combination space, we usually tune important hyper-parameters first and then the other ones. On average, we run 97.8 experiments for each model to obtain the best result. To test their robustness, we report the mean and standard deviation of five different runs by setting different random seeds. All the experiments were run on a GPU cluster with 8~16 GPUs (each with 16GB GPU memory) available depending on the occupancy rate.

3.2.7 Reproducibility. To ensure reproduciblility, we keep the md5sum values of our data partitions. We explicitly set the random seed during each running and record the data preprocessing thresholds and model hyper-parameters into configure files. In particular, compared with tensorflow, Pytorch offer better ability to reduce non-determinism for our models when running on a GPU device. We will release the source code of FuxiCTR together with the configuration files to the community for reproducible research (currently undergoing a open-sourcing process of our company).

Table 3: FuxiCTR benchmarking results

(Bold numbers indicate the best results in each column; the underscored numbers show the results that have not been successfully reproduced)

Criteo

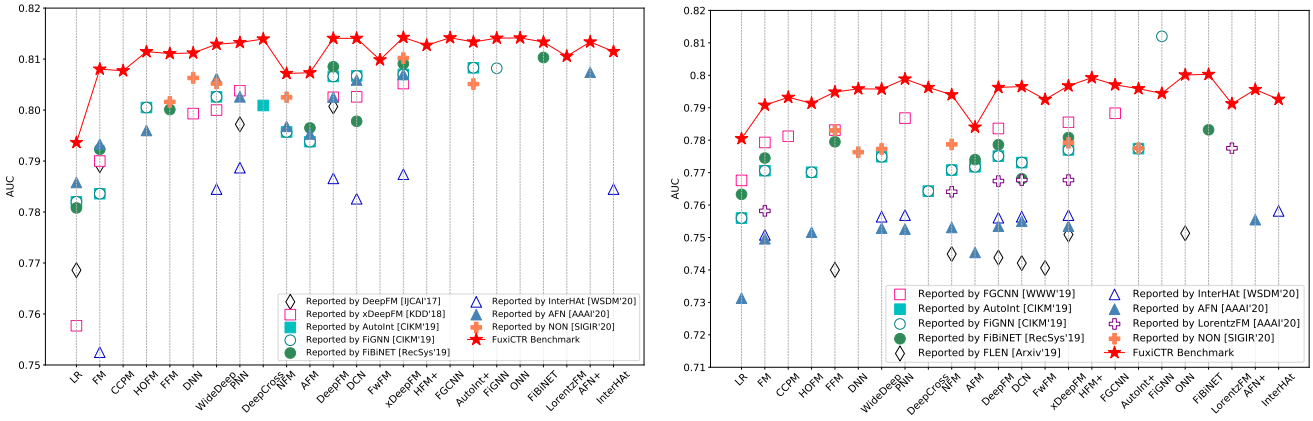
		Best Reported		Reproduced		Retuned				Training
Year	Model	Logloss	AUC	Logloss	AUC	Logloss	AUC	#Params	#Runs	Time x Epochs
–	LR	0.4474	0.7858	–	–	0.45651 ± 0.00003	0.79363 ± 0.00003	5.5M	12	7m x 12
2010	FM	0.4464	0.7933	–	–	0.44432 ± 0.00010	0.80804 ± 0.00015	227.5M	20	18m x 5
2015	CCPM	–	–	–	–	0.44396 ± 0.00012	0.80771 ± 0.00011	222.0M	24	2h33m x 1
2016	HOFM	0.4508	0.8005	–	–	0.44042 ± 0.00006	0.81146 ± 0.00007	255.3M	17	1h42m x 26
2016	FFM	0.4525	0.8001	–	–	0.44091 ± 0.00031	0.81109 ± 0.00031	638.2M	20	3h59m x 5
2016	DNN	0.4491	0.7993	–	–	0.44071 ± 0.00006	0.81120 ± 0.00009	226.5M	64	13m x 2
2016	WideDeep	0.4453	0.8062	0.44290	0.80878	0.43889 ± 0.00013	0.81291 ± 0.00010	231.1M	80	11m x 4
2016	PNN	0.4532	0.8038	0.44510	0.80903	0.43884 ± 0.00008	0.81324 ± 0.00006	258.5M	38	32m x 2
2016	DeepCross	0.4425	0.8009	0.44408	0.80794	0.43804 ± 0.00059	0.81395 ± 0.00060	284.4M	138	26m x 3
2017	NFM	0.4537	0.7968	0.44953	0.80178	0.44431 ± 0.00003	0.80718 ± 0.00005	229.6M	64	18m x 2
2017	AFM	0.4541	0.7965	–	–	0.44430 ± 0.00074	0.80732 ± 0.00080	227.5M	10	31m x 22
2017	DeepFM	0.4445	0.8085	0.44948	0.80236	0.43779 ± 0.00005	0.81407 ± 0.00007	229.1M	128	18m x 9
2017	DCN	0.4419	0.8067	0.44214	0.80946	0.43783 ± 0.00017	0.81407 ± 0.00012	245.1M	544	14m x 9
2018	FwFM	–	–	0.44428	0.80740	0.44189 ± 0.00009	0.80985 ± 0.00010	222.0M	14	19m x 1
2018	xDeepFM	0.4418	0.8091	**0.43912**	**0.81302**	**0.43776 ± 0.00019**	**0.81426 ± 0.00020**	232.3M	156	1h9m x 8
2019	HFM+	–	–	–	–	0.43910 ± 0.00005	0.81273 ± 0.00010	260.2M	74	1h13m x 2
2019	FGCNN	–	–	0.44139	0.81033	0.43814 ± 0.00021	0.81418 ± 0.00031	317.4M	87	3h9m x 6
2019	AutoInt+	0.4434	0.8083	0.44250	0.80916	0.43854 ± 0.00018	0.81336 ± 0.00017	285.4M	120	16m x 1
2019	FiGNN	0.4411	0.8082	0.44276	0.80890	0.43792 ± 0.00010	0.81407 ± 0.00007	222.7M	40	1h40m x 14
2019	ONN	**0.43577**	**0.8123**	0.43954	0.81273	0.43805 ± 0.00016	0.81415 ± 0.00027	436.7M	108	2h13m x 8
2019	FiBiNET	0.4423	0.8103	0.44306	0.80881	0.43857 ± 0.00021	0.81335 ± 0.00008	482.5M	78	2h21m x2
2020	LorentzFM	–	–	0.44558	0.80594	0.44132 ± 0.00004	0.81055 ± 0.00005	222.0M	8	2h12m x 20
2020	AFN+	0.4451	0.8074	0.44317	0.80836	0.43865 ± 0.00015	0.81336 ± 0.00007	238.1M	114	38m x 6
2020	InterHAt	0.4577	0.7845	0.44411	0.80732	0.44013 ± 0.00011	0.81167 ± 0.00009	222.1M	90	18m x 20

Avazu

		Best Reported		Reproduced		Retuned				Training
Year	Model	Logloss	AUC	Logloss	AUC	Logloss	AUC	#Params	#Runs	Time x Epochs
–	LR	0.3868	0.7676	–	–	0.37991 ± 0.00001	0.78048 ± 0.00006	8.4M	90	5m x 26
2010	FM	0.3740	0.7793	–	–	0.37363 ± 0.00011	0.79079 ± 0.00018	343.3M	187	18m x 1
2015	CCPM	0.3800	0.7812	–	–	0.37210 ± 0.00045	0.79321 ± 0.00072	335.0M	24	57m x 1
2016	HOFM	0.3756	0.7701	–	–	0.37329 ± 0.00023	0.79139 ± 0.00031	385.2M	31	1h47m x 1
2016	FFM	0.3781	0.7831	–	–	0.37114 ± 0.00022	0.79484 ± 0.00044	778.7M	32	1h54m x 1
2016	DNN	–	–	–	–	0.37048 ± 0.00009	0.79583 ± 0.00008	338.9M	252	12m x 1
2016	WideDeep	0.3744	0.7749	0.37173	0.79354	0.37034 ± 0.00004	0.79575 ± 0.00006	345.9M	52	14m x 1
2016	PNN	0.3737	0.7868	0.37300	0.79178	0.36839 ± 0.00005	0.79885 ± 0.00007	336.1M	36	11m x 1
2016	DeepCross	0.3889	0.7643	0.37350	0.79096	0.37002 ± 0.00008	0.79620 ± 0.00010	342.6M	45	12m x 1
2017	NFM	0.3761	0.7708	0.37483	0.78871	0.37146 ± 0.00005	0.79399 ± 0.00006	346.4M	54	16m x 1
2017	AFM	0.3766	0.7740	–	–	0.37809 ± 0.00067	0.78403 ± 0.00129	343.3M	67	20m x 2
2017	DeepFM	0.3742	0.7836	0.37186	0.79339	0.37013 ± 0.00010	0.79627 ± 0.00008	373.2M	234	16m x 1
2017	DCN	0.3721	0.7681	0.37297	0.79161	0.36993 ± 0.00013	0.79652 ± 0.00011	336.9M	496	7m x 1
2018	FwFM	0.3988	0.7406	0.38472	0.77436	0.37240 ± 0.00024	0.79253 ± 0.00035	334.9M	15	11m x 1
2018	xDeepFM	0.3737	0.7855	0.37037	0.79576	0.36970 ± 0.00013	0.79674 ± 0.00024	344.1M	288	31m x 1
2019	HFM+	–	–	–	–	0.36831 ± 0.00004	0.79925 ± 0.00003	355.3M	92	43m x 1
2019	FGCNN	0.3746	0.7883	0.37124	0.79479	0.36961 ± 0.00018	0.79705 ± 0.00027	374.5M	84	2h10m x 1
2019	AutoInt+	0.3811	0.7774	0.37322	0.79127	0.37031 ± 0.00017	0.79587 ± 0.00007	337.9M	75	15m x 1
2019	FiGNN	0.3817	**0.8120**	0.37438	0.78928	0.37113 ± 0.00009	0.79442 ± 0.00014	335.0M	64	2h25m x 1
2019	ONN	0.3945	0.7513	**0.36843**	**0.79898**	0.36767 ± 0.00041	0.80012 ± 0.00009	406.3M	120	1h52 x 1
2019	FiBiNET	0.3786	0.7832	0.37240	0.79372	**0.36753 ± 0.00024**	**0.80028 ± 0.00030**	395.9M	54	36m x 1
2020	LorentzFM	0.3828	0.7775	0.38024	0.77991	0.37417 ± 0.00092	0.79122 ± 0.00199	334.9M	9	46m x 17
2020	AFN+	**0.3718**	0.7555	0.37151	0.79391	0.37061 ± 0.00017	0.79561 ± 0.00029	363.5M	190	35m x 1
2020	InterHAt	0.3910	0.7582	0.38581	0.77034	0.37225 ± 0.00023	0.79273 ± 0.00034	335.1M	56	17m x 1

3.3 Results Analysis

In this section, we report our benchmarking results of 24 models on two datasets, as shown in Table 3. To be specific, the "Best Reported" column shows the best results copied from existing studies. The "Reproduced" column reports the results we reproduced on our data partitions according to the hyper-parameters used in the original papers. The "Retuned" column reports the mean and standard deviation results re-tuned in our experiments. In addition,



(a) Criteo dataset (b) Avazu dataset

Figure 1: Comparison between FuxiCTR and the state-of-the-arts

"#Params" denotes the number of parameters used in the re-tuned model. "#Runs" records the number of experiments we run during performance tuning. We also report training time in terms of time per epoch and number of epochs. From the results, we have a number of surprising observations:

- The best results reported by existing studies show certain inconsistency. For example, InterHAt has worse performance than LR on both datasets; PNN performs worse than LR on Criteo; DeepCross is worse than LR as well. They are largely due to the different data partitions or data preprocessing steps, even though they use the same datasets. This reveals that standard data partition and data preprocessing is necessary to fair and consistent comparisons among models.
- The "Reproduced" column shows that, in most cases, we can reproduce the model performance reported from existing studies. However, even using the same data partition and data preprocessing in these experiments, some inconsistency occurs. For example, using the hyper-parameters from the papers on Criteo, DeepFM becomes much worse than the results reported before while xDeepFM achieves improved results. This indicates that it is necessary to re-tune the hyper-parameters when test a model on a new data partition (even for the same dataset). However, it is not uncommon that studies (e.g., [14]) state that they follow the baseline hyper-parameters from their papers for fairness of comparison, yet they experiment with a different data partition. A common benchmark is thus desired to alleviate this issue.
- We re-tuned all the models on our data partitions through extensive experiments. The results show a quite large improvement (up to 1%) over the existing results (either best reported or reproduced), considering that 1‰ improvement is deemed to be significantly practical in many studies [4, 20, 38]. After re-tuning, we found that the differences among the state-of-the-art models become small. For example, DeepFM, DCN, xDeepFM, FGCNN, FiGNN, and ONN all achieve the same level of accuracy (~0.814 AUC) on Criteo, while FiBiNET and ONN attain comparable performance (~0.800 AUC) on Avazu. We run many experiments with different hyper-parameters, but do not obtain sufficiently distinct results. Especially, some recent models, such as InterHAt, AFN+, and LorentzFM obtain even worse results than some previous state-of-the-arts. Our results raise questions about the published claims and also suggest that baseline models should be more carefully tuned to make more convincing comparisons.
- Figure 1 illustrates a clear comparison between FuxiCTR and the state-of-the-art results. For each dataset, we plot the AUC results from 8~9 papers. We can see that the results vary a lot among different papers. The X-axis is arranged sequentially according to the publication year, but there is no clear pattern on the improvement. Remarkably, our FuxiCTR achieves the best performance among all models. Nevertheless, it shows that recent models only obtain diminished improvements and sometimes even lead to performance drops.
- Memory consumption and model efficiency are aspects that are equally important to industrial CTR prediction tasks. We report the number of parameters and training time used in each model in Table 3. Some models run very slow (hours per epoch) due to the use of convolution networks (e.g., CCPM, FGCNN, HFM+), fieldwise interactions (e.g., FFM, ONN), graph neural networks (e.g., FiGNN), and so on. Some others take up more parameters, such as FFM, ONN, FGCNN, etc. These drawbacks minght hinder their wide adoption in industry. Note that these numbers provide a reference only, because they depends heavily on the model hyper-parameters. For example, it is possible to reduce the number of parameters and training time at the expense of decreased accuracy, by reducing the number of layers or setting a smaller hidden size.

3.4 Key Points to Performance Tuning

During our evaluation, we identify the following key points that are critical for performance tuning.

- **Data preprocessing**. Data often determine the upper bound of a model. However, existing work seldom tunes the thresholds during data preprocessing. After extensive experiments, we set an appropriate threshold for rare feature filtering (i.e., 2 for Criteo and 1 for Avazu), which yields much better performance.

- **Batch size**. In our experiments, we observe that a large batch size usually results in better performance. For example, we set it to 10000 if no OOM (Out-Of-Memory) error occurs.
- **Embedding size**. While existing work usually set 10 or 16 in the experiments, we found that it performs better by setting to a larger one (i.e., 40) within the GPU memory constraints.
- **Regularization weight and dropout rate**. Regularization and dropout are two key techniques to reduce model overfitting. They have a large impact on prediction performance on CTR models. We exhaustively search the optimal value within a range.
- **Batch normalization**. In some cases, adding batch normalization between hidden layers of a DNN model can boost prediction performance.

More detailed settings about the hyper-parameters will be available in the configuration files of FuxiCTR benchmarks.

4 DISCUSSION

In this section, we discuss about the limitations of FuxiCTR and provide some directions for further exploration.

More datasets: In this work, we evaluate and benchmark existing CTR prediction models on two widely-used datasets, Criteo and Avazu. However, both datasets are anonymized and there is a lack of the corresponding user field and item field information. Therefore, they are able to be used to benchmark some models that require explicit user-item interaction (e.g., FLEN [3]) and user behavior sequence (e.g., DIN [44]). Meanwhile, both datasets correspond to the scenario of display advertising, limiting the diversity of our benchmark. We plan to extend more datasets from different applications (e.g., product search) to make FuxiCTR a more comprehensive benchmark for CTR prediction. We emphasize that although some studies employ small datasets (e.g., Frappe) for CTR prediction evaluation, we advocate for more research on industrial-scale datasets.

Data splits: To keep it consistent with most existing research, we split the datasets randomly to benchmark CTR prediction models. We do so with the following consideration. With randomly splitting, the data distributions between train, validation, and test sets are more consistent. This helps to better reveal the effectiveness of a CTR prediction model on capturing feature interactions, because in production it is necessary to perform CTR calibration after prediction if the train-test distributions vary largely. As part of future work, we will evaluate the models by splitting data sequentially over time and also perform CTR calibration as necessary.

User interests modeling: Recently, some work [9, 43, 44] proposes to model user history interaction sequence to enhance CTR prediction. However, we cannot evaluate these models on Criteo and Avazu datasets, due to the lack of user id and item id information in the two mostly-widely used datasets. We will extend more datasets in FuxiCTR to benchmark these models.

Efficiency evaluation: Current models for CTR prediction has become more and more complex in structure, after using components such as attention [17, 35], convolution [21], and graph neural network [18]. In this version, we mainly evaluate the efficiency of these models through their training time. It is also desirable to test their inference time in future. Due to the strict latency constraints of CTR prediction in real-time applications, efficiency benchmarking would not only help practitioners choose an appropriate model, but also facilitate researchers to design effective yet efficient models.

Auto-tuning of hyper-parameters: As the experimental results shown in Section 3, hyper-parameter tuning is critical to the performance of CTR prediction models. How to quickly find the optimal hyper-parameter remains an open research problem. When data evolve with time, re-tuning of model hyper-parameters is also required to adapt to the new data. In our benchmark, we mainly apply grid search to find the best hyper-parameters of each model. It is highly expected to explore some advanced automl techniques (e.g., bayesian optimization in the NNI project [1]) to further boost hyper-parameter tuning process in future.

5 RELATED WORK

In this section, we review the related work on CTR prediction and open benchmarks.

5.1 CTR Prediction

During the last decade, CTR prediction models have been widely studied and evolved from shallow models to deep models. We have introduced some representative models in Section 2. Here, we present a review of more related studies on CTR prediction, which are summarized into the following categories.

5.1.1 Feature Interaction Learning. While simple linear models such as LR [33] and FTRL [25] have been widely used due to their simplicity and efficiency, they have difficulty to capture non-linear feature mappings and conjunctions. He et al. [12] propose the GBDT + LR approach that applies Gradient Boosting Decision Tree (GBDT) to extract meaningful feature conjunctions.

FM [32] is an effective model that captures pairwise feature interactions via inner products of feature vectors. Due to its success, many follow-up models have been proposed from different aspects, such as field awareness (e.g., FFM [15], FwFM [28]), importance of feature interactions (e.g., AFM [39], IFM [13]), outer-products based interaction (HFM [36]), robustness (RFM [29]), and interpretability (SEFM [16]). However, it is non-trivial for these models to capture high-order feature interactions in practice [2].

Recently, deep learning has become a hot research topic of recommender systems [42], yielding an abundance of deep models for CTR prediction, including YoutubeDNN [6], Wide&Deep [4], PNN [30], DeepFM [10], NFM [11], etc. Some of them aim to capture different orders of feature interactions explicitly (e.g., DCN [38], xDeepFM [19]). Some other models explore the use of convolutional networks (e.g., CCPM [22], FGCNN [21]), recurrent networks (e.g., [9, 43]), or attention networks (e.g., AutoInt [35], FiBiNET [14]) to learn implicit feature interactions.

5.1.2 User Interests Modeling. The previous user behaviours have a large effect on predicting the click probability on the next item. To better capture such history behaviours (i.e., item sequences), some recent studies propose user interests modeling for CTR prediction via attention, LSTM, GRU, and memory networks. Typical examples include DIN [44], DIEN [43], DSIN [9], HPMN [31]) and DSTN [27].

5.1.3 Multi-Task Learning. In many recommender systems, users may have diverse behaviors beyond click, such as browsing, favorite, add-to-cart, and purchase. To improve the performance of CTR

prediction, it is desirable to leverage other types of user feedbacks to enrich the supervision signals for CTR prediction. Towards this goal, some work proposes multi-task learning models to learn task relationships among different user behaviors, such as ESMM [24], MMoE [23] and DUPN [26].

5.2 Open Benchmarks and Reproducibility

With the prevalence of deep learning, new models are emerging at a increasingly rapid pace. There is a high demand for a common benchmark to fairly compare against baseline models. Open benchmarks are valuable to promote research developments. For example, ImageNet [8] and GLUE [37] are two well-known benchmarks that contribute much to the progress in computer vision and nature language processing, respectively. In recommender systems, some datasets (e.g., Criteo and Avazu) are widely used. However, there is still a lack of standard preprocessing and evaluation protocols, which results in the inconsistency and reproducibility issues of existing studies [7]. In this work, we take an important step towards reproducible research by releasing the FuxiCTR benchmark as well as the benchmarking results over 20 models.

6 CONCLUSION

In this paper, we present the first open benchmark, namely FuxiCTR, for CTR prediction. We aim to alleviate the issues of non-reproducible and inconsistent results raised in current studies. We formulate the evaluation protocol of FuxiCTR in details and evaluate 24 existing models by running over 4,600 experiments for more than 12,000 GPU hours on two widely-used real-world datasets Criteo and Avazu. We provide the most comprehensive benchmarking results that compare existing models in a rigorous manner. The results show that the difference between many models are smaller than expected and inconsistent results exist in existing papers. We believe that our benchmark would not only facilitate reproducible research but also help new beginners to learn the state-of-the-art CTR prediction models.

REFERENCES

[1] 2020. Neural Network Intelligence âĂŤ An open source AutoML toolkit. https://github.com/Microsoft/nni

[2] Mathieu Blondel, Akinori Fujino, Naonori Ueda, and Masakazu Ishihata. 2016. Higher-Order Factorization Machines. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*. 3351–3359.

[3] Wenqiang Chen, Lizhang Zhan, Yuanlong Ci, and Chen Lin. 2019. FLEN: Leveraging Field for Scalable CTR Prediction. *CoRR* abs/1911.04690 (2019).

[4] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, et al. 2016. Wide & Deep Learning for Recommender Systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (DLRS@RecSys)*. 7–10.

[5] Weiyu Cheng, Yanyan Shen, and Linpeng Huang. 2020. Adaptive Factorization Network: Learning Adaptive-Order Feature Interactions. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*. 3609–3616.

[6] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys)*. 191–198.

[7] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. 2019. Are we really making much progress? A worrying analysis of recent neural recommendation approaches. In *Proceedings of the 13th ACM Conference on Recommender Systems (RecSys)*. 101–109.

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. 2009. ImageNet: A large-scale hierarchical image database. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. 248–255.

[9] Yufei Feng, Fuyu Lv, Weichen Shen, Menghan Wang, Fei Sun, Yu Zhu, and Keping Yang. 2019. Deep Session Interest Network for Click-Through Rate Prediction. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI)*. 2301–2307.

[10] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. In *International Joint Conference on Artificial Intelligence (IJCAI)*. 1725–1731.

[11] Xiangnan He and Tat-Seng Chua. 2017. Neural Factorization Machines for Sparse Predictive Analytics. In *Proceedings of the 40th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 355–364.

[12] Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, 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, (ADKDD)*. 5:1–5:9.

[13] Fuxing Hong, Dongbo Huang, and Ge Chen. 2019. Interaction-Aware Factorization Machines for Recommender Systems. In *The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI)*. 3804–3811.

[14] Tongwen Huang, Zhiqi Zhang, and Junlin Zhang. 2019. FiBiNET: combining feature importance and bilinear feature interaction for click-through rate prediction. In *Proceedings of ACM Conference on Recommender Systems (RecSys)*. 169–177.

[15] Yu-Chin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. 2016. Field-aware Factorization Machines for CTR Prediction. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys)*. 43–50.

[16] Liang Lan and Yu Geng. 2019. Accurate and Interpretable Factorization Machines. In *The AAAI Conference on Artificial Intelligence (AAAI)*. 4139–4146.

[17] Zeyu Li, Wei Cheng, Yang Chen, Haifeng Chen, and Wei Wang. 2020. Interpretable Click-Through Rate Prediction through Hierarchical Attention. In *The International Conference on Web Search and Data Mining (WSDM)*. 313–321.

[18] 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 (CIKM)*. 539–548.

[19] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, et al. 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, (KDD)*. 1754–1763.

[20] Xiaoliang Ling, Weiwei Deng, Chen Gu, Hucheng Zhou, Cui Li, and Feng Sun. 2017. Model Ensemble for Click Prediction in Bing Search Ads. In *Proceedings of the 26th International Conference on World Wide Web Companion (WWW)*. 689–698.

[21] Bin Liu, Ruiming Tang, Yingzhi Chen, Jinkai Yu, Huifeng Guo, and Yuzhou Zhang. 2019. Feature Generation by Convolutional Neural Network for Click-Through Rate Prediction. In *The World Wide Web Conference, (WWW)*. 1119–1129.

[22] Qiang Liu, Feng Yu, Shu Wu, and Liang Wang. 2015. A Convolutional Click Prediction Model. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management (CIKM)*. 1743–1746.

[23] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H. Chi. 2018. Modeling Task Relationships in Multi-task Learning with Multi-gate Mixture-of-Experts. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*. 1930–1939.

[24] Xiao Ma, Liqin Zhao, Guan Huang, Zhi Wang, Zelin Hu, Xiaoqiang Zhu, and Kun Gai. 2018. Entire Space Multi-Task Model: An Effective Approach for Estimating Post-Click Conversion Rate. In *Proceedings of the 41st International Conference on Research & Development in Information Retrieval (SIGIR)*. 1137–1140.

[25] H. Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, Sharat Chikkerur, Dan Liu, Martin Wattenberg, Arnar Mar Hrafnkelsson, Tom Boulos, and Jeremy Kubica. 2013. Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 1222–1230.

[26] Yabo Ni, Dan Ou, Shichen Liu, Xiang Li, Wenwu Ou, Anxiang Zeng, and Luo Si. 2018. Perceive Your Users in Depth: Learning Universal User Representations from Multiple E-commerce Tasks. In *Proceedings of the SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*. 596–605.

[27] Wentao Ouyang, Xiuwu Zhang, Li Li, Heng Zou, Xin Xing, Zhaojie Liu, and Yanlong Du. 2019. Deep Spatio-Temporal Neural Networks for Click-Through Rate Prediction. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*. 2078–2086.

[28] 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 on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*. 1349–1357.

[29] Surabhi Punjabi and Priyanka Bhatt. 2018. Robust Factorization Machines for User Response Prediction. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web (WWW)*. 669–678.

[30] 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 *Proceedings of the IEEE 16th International Conference on Data Mining (ICDM)*. 1149–1154.

[31] Kan Ren, Jiarui Qin, Yuchen Fang, Weinan Zhang, Lei Zheng, Weijie Bian, Guorui Zhou, Jian Xu, Yong Yu, Xiaoqiang Zhu, and Kun Gai. 2019. Lifelong Sequential Modeling with Personalized Memorization for User Response Prediction. In

Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR). 565–574.

[32] Steffen Rendle. 2010. Factorization Machines. In *Proceedings of the 10th IEEE International Conference on Data Mining (ICDM)*. 995–1000.

[33] Matthew Richardson, Ewa Dominowska, and Robert Ragno. 2007. Predicting clicks: estimating the click-through rate for new ads. In *Proceedings of the 16th International Conference on World Wide Web (WWW)*. 521–530.

[34] Ying Shan, T. Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and J. C. 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 (KDD)*. 255–262.

[35] 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 (CIKM)*. 1161–1170.

[36] Yi Tay, Shuai Zhang, Anh Tuan Luu, Siu Cheung Hui, Lina Yao, and Tran Dang Quang Vinh. 2019. Holographic Factorization Machines for Recommendation. In *The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI)*. 5143–5150.

[37] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *7th International Conference on Learning Representations (ICLR)*.

[38] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & Cross Network for Ad Click Predictions. In *Proceedings of the 11th International Workshop on Data Mining for Online Advertising (ADKDD)*. 12:1–12:7.

[39] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. 2017. Attentional Factorization Machines: Learning the Weight of Feature Interactions via Attention Networks. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, (IJCAI)*. 3119–3125.

[40] Canran Xu and Ming Wu. 2020. Learning Feature Interactions with Lorentzian Factorization Machine. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*. 6470–6477.

[41] 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.

[42] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep Learning Based Recommender System: A Survey and New Perspectives. *ACM Comput. Surv.* 52, 1 (2019), 5:1–5:38.

[43] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2018. Deep Interest Evolution Network for Click-Through Rate Prediction. *CoRR* abs/1809.03672 (2018).

[44] Guorui Zhou, Xiaoqiang Zhu, Chengru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep Interest Network for Click-Through Rate Prediction. In *Proceedings of the SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*. 1059–1068.