AutoInt: Automatic Feature Interaction Learning via Self-Attentive Neural Networks

Weiping Song[1], Chence Shi[1], Zhiping Xiao[2], Zhijian Duan[1], Yewen Xu[1], Ming Zhang[1], Jian Tang[3,4]
[1]Peking University, China
[2]University of California, Los Angeles, US
[3]Montreal Institute for Learning Algorithms (Mila), Canada
[4]HEC Montreal, Canada
{songweiping,chenceshi,zjduan,xuyewen,mzhang_cs}@pku.edu.cn,patriciaxiao@g.ucla.edu,jian.tang@hec.ca

ABSTRACT

Click-through rate (CTR) prediction, which aims to predict the probability of a user clicking an ad or an item, is critical to many online applications such as online advertising and recommender systems. The problem is very challenging since (1) the input features (e.g., the user id, user age, item id, item category) are usually sparse and high-dimensional, and (2) an effective prediction relies on high-order combinatorial features (*a.k.a.* cross features), which are very time-consuming to hand-craft by domain experts and are impossible to be enumerated. Therefore, there have been efforts in finding low-dimensional representations of the sparse and high-dimensional raw features and their meaningful combinations.

In this paper, we propose an effective and efficient algorithm to automatically learn the high-order feature combinations of input features. Our proposed algorithm is very general, which can be applied to both numerical and categorical input features. Specifically, we map both the numerical and categorical features into the same low-dimensional space. Afterward, a multi-head self-attentive neural network with residual connections is proposed to explicitly model the feature interactions in the low-dimensional space. With different layers of the multi-head self-attentive neural networks, different orders of feature combinations of input features can be modeled. The whole model can be efficiently fit on large-scale raw data in an end-to-end fashion. Experimental results on four real-world datasets show that our proposed approach not only outperforms existing state-of-the-art approaches for prediction but also offers good explainability.

KEYWORDS

High-order feature interactions, Self attention, CTR prediction, Explainable recommendation

1 INTRODUCTION

Predicting the probabilities of users clicking ads or items (a.k.a., click-through rate prediction) is a critical problem for many applications such as online advertising and recommender systems. The performance of the prediction has a direct impact on the final revenue of the business providers. Due to its importance, it has attracted growing interest in both academia and industry communities.

Machine learning has been playing a key role in click-through rate prediction, which is usually formulated as supervised learning with user profiles and item attributes as input features. The problem is very challenging for several reasons. First, the input features are extremely sparse and high-dimensional [7, 10, 12, 18, 29]. In real-world applications, a considerable percentage of user's demographics and item's attributes are usually discrete and/or categorical. To make supervised learning methods applicable, these features are first converted to a one-hot encoding vector, which can easily result in features with millions of dimensions. Taking the well-known CTR prediction data Criteo[1] as an example, the feature dimension is approximately 30 million with sparsity over 99.99%. With such sparse and high-dimensional input features, the machine learning models are easily overfitted. Second, as shown in extensive literature [7, 10, 16, 29], high-order feature combinations are crucial for a good performance. For example, it is reasonable to recommend *Mario Bros.*, a famous video game, to *David*, who is a ten-year-old boy. In this case, the third-order combinatorial feature *<Gender=Male, Age=10, ProductCategory=VideoGame>* is very informative for prediction. However, finding such meaningful high-order combinatorial features heavily relies on domain experts. Moreover, it is almost impossible to hand-craft all the meaningful combinations [7, 23]. One may ask that we can enumerate all the possible high-order features and let machine learning models select the meaningful ones. However, enumerating all the possible high-order features will exponentially increase the dimension and sparsity of the input features, leading to a more serious problem of model overfitting. Therefore, there has been extensive efforts in the communities in finding low-dimensional representations of the sparse and high-dimensional input features and meanwhile modeling different orders of feature combinations.

For example, Factorization Machines (FM) [23], which combine polynomial regression models with factorization techniques, are developed to model feature interactions and have been proved effective for various tasks [24, 25]. However, limited by its polynomial fitting time, it is only effective for modeling low-order feature interactions and impractical to capture high-order feature interactions. Recently, many works [7, 10, 12, 34] based on deep neural networks have been proposed to model the high-order feature interactions. Specifically, multiple layers of non-linear neural networks are usually used to capture the high-order feature interactions.

WOODSTOCK'97, July 1997, El Paso, Texas USA

ACM ISBN 123-4567-24-567/08/06.
https://doi.org/10.475/123_4

[1]http://labs.criteo.com/2014/09/kaggle-contest-dataset-now-available-academic-use/

However, such kinds of methods suffer from two limitations. First, fully-connected neural networks have been shown inefficient in learning multiplicative feature interactions [3]. Second, since these models learn the feature interactions in an implicit way, they lack good explanation on which feature combinations are meaningful. Therefore, we are looking for an approach that is able to explicitly model different orders of feature combinations, represent the entire features into low-dimensional spaces, and meanwhile offer good model explainability.

In this paper, we propose such an approach based on the multi-head self-attention mechanism [32]. Our proposed approach learns effective low-dimensional representations of the sparse and high-dimensional input features and is applicable to both the categorical and/or numerical input features. Specifically, both the categorical and numerical features are first embedded into low-dimensional spaces, which reduces the dimension of the input features and meanwhile allows different types of features to interact with each other via vector arithmetic (e.g., summation and inner product). Afterward, we propose a novel interacting layer to promote the interactions between different features. Within each interacting layer, each feature is allowed to interact with all the other features and is able to automatically identify relevant features to form meaningful higher-order features via the multi-head attention mechanism [32]. Moreover, the multi-head mechanism projects a feature into multiple subspaces, and hence it can capture different feature interactions in different subspaces. Such an interacting layer models the one-step interaction between the features. By stacking multiple interacting layers, we are able to model different orders of feature interactions. In practice, the residual connection [11] is added to the interacting layer, which allows combining different orders of feature combinations. We use the attention mechanism for measuring the correlations between features, which offers good model explainability.

To summarize, in this paper we make the following contributions:

- We propose to study the problem of explicitly learning high-order feature interactions and meanwhile finding models with good explainability for the problem.
- We propose a novel approach based on self-attentive neural network, which can automatically learn high-order feature interactions and efficiently handle large-scale high-dimensional sparse data.
- We conducted extensive experiments on several real-world data sets. Experimental results on the task of CTR prediction show that our proposed approach not only outperforms existing state-of-the-art approaches for prediction but also offers good model explainability.

Our work is organized as follows. In Section 2, we summarize the related work. Section 3 formally defines our problem. Section 4 presents the proposed approach to learn feature interactions. In Section 5, we present the experimental results and detailed analysis. We conclude this paper and point out the future work in Section 6.

2 RELATED WORK

Our work is relevant to three lines of work: 1) Click-through rate prediction in recommender systems and online advertising, 2) techniques for learning feature interactions, and 3) self-attention mechanism and residual networks in the literature of deep learning.

2.1 Click-through Rate Prediction

Predicting click-through rates is important to many Internet companies, and various systems have been developed by different companies [7–9, 13, 18, 26, 39]. For example, Google developed the Wide&Deep[7] learning system for recommender systems, which combines the advantages of both the linear shallow models and deep models. The system achieves remarkable performance in APP recommendation. The problem also receives a lot of attention in the academic communities. For example, Shan et al. [28] proposed a context-aware CTR prediction method which factorized three-way <user, ad, context> tensor. Oentaryo et al. [21] developed hierarchical importance-aware factorization machine to model dynamic impacts of ads.

2.2 Learning Feature Interactions

Learning feature interactions is a fundamental problem and therefore extensively studied in the literature. A well-known example is Factorization Machines (FM) [23], which were proposed to mainly capture the first- and second-order feature interactions and have been proved effective for many tasks in recommender systems [24, 25]. Afterward, different variants of factorization machines have been proposed. For example, Field-aware Factorization Machines (FFM) [14] modeled fine-grained interactions between features from different fields. GBFM [6] and AFM [36] considered the importance of different second-order feature interactions. However, all these approaches focus on modeling low-order feature interactions.

There are some recent works that model high-order feature interactions. For example, NFM [12] stacked deep neural networks on top of the output of the second-order feature interactions to model higher-order features. Similarly, PNN [22], FNN [37], DeepCrossing [29], Wide&Deep [7] and DeepFM [10] utilized feed-forward neural networks to model high-order feature interactions. However, all these approaches learn the high-order feature interactions in an implicit way and therefore lack good model explainability. On the contrary, there are three lines of works that learn feature interactions in an explicit fashion. First, Deep&Cross [34] and xDeepFM [16] took outer product of features at the bit- and vector-wise level respectively. Although they perform explicit feature interactions, it is not trivial to explain which combinations are useful. Second, some tree-based methods [35, 38, 40] combined the power of embedding-based models and tree-based models but had to break training procedure into multiple stages. Third, HOFM [4] proposed efficient training algorithms for high-order factorization machines. However, HOFM requires too many parameters and only its low-order (usually less than 5) form can be practically used. We compare with all these methods in the experiments.

2.3 Attention and Residual Networks

Our proposed model makes use of the latest techniques in the literature of deep learning: attention [2] and residual networks [11]. Attention is first proposed in the context of neural machine translation [2] and has been proved effective in a variety of tasks such as question answering [31], text summarization [27], and recommender systems [39]. Vaswani et al. [32] further proposed multi-head self-attention to model complicated dependencies between words in machine translation.

Residual networks [11] achieved state-of-the-art performance in the ImageNet contest. Since the residual connection, which can be simply formalized as $y = F(x) + x$, encourages gradient flow through interval layers, it becomes a popular network structure for training very deep neural networks.

3 PROBLEM DEFINITION

We first formally define the problem of click-through rate (CTR) prediction as follows:

DEFINITION 1. (**CTR Prediction**) Let $\mathbf{x} \in \mathbb{R}^n$ denotes the concatenation of user u's features and item v's features, where categorical features are represented with one-hot encoding, and n is the dimension of concatenated features. The problem of *click-through rate prediction* aims to predict the probability of user u clicking item v according to the feature vector $\mathbf{x}$.

A straightforward solution for CTR prediction is to treat $\mathbf{x}$ as the input features and deploy the off-the-shelf classifiers such as logistic regression. However, since the original feature vector $\mathbf{x}$ is very sparse and high-dimensional, the model will be easily overfitted. Therefore, it is desirable to represent the raw input features in low-dimensional continuous spaces. Moreover, as shown in existing literature, it is crucial to utilize the higher-order combinatorial features to yield good prediction performance [5, 7, 10, 20, 23, 29]. Specifically, we define the high-order combinatorial features as follows:

DEFINITION 2. (**p-order Combinatorial Feature**) Given input feature vector $\mathbf{x} \in \mathbb{R}^n$, a *p-order combinatorial feature* is defined as $\langle x_{i_1}, x_{i_2}, ..., x_{i_p} \rangle = g(x_{i_1}, ..., x_{i_p})$, where each feature comes from a distinct field, p is the number of involved feature fields, and $g(\cdot)$ can be any combination function, such as multiplication [23] and outer product [16, 34].

Traditionally, meaningful high-order combinatorial features are hand-crafted by domain experts. However, this is very time-consuming and hard to generalize to other domains. Besides, it is almost impossible to hand-craft all meaningful high-order features. Therefore, we aim to develop an approach that is able to automatically discover the meaningful high-order combinatorial features and meanwhile map all these features into low-dimensional continuous spaces. Formally, we define our problem as follows:

DEFINITION 3. (**Problem Definition**) Given an input feature vector $\mathbf{x} \in \mathbb{R}^n$ for click-through rate prediction, our goal is to learn a low-dimensional representation of $\mathbf{x}$, which models the high-order combinatorial features.

4 MODEL

In this section, we first give an overview of the proposed approach *AutoInt*, which can automatically learn feature interactions for CTR prediction. Next, we present a comprehensive description of how to learn a low-dimensional representation that models high-order combinatorial features without manual feature engineering.

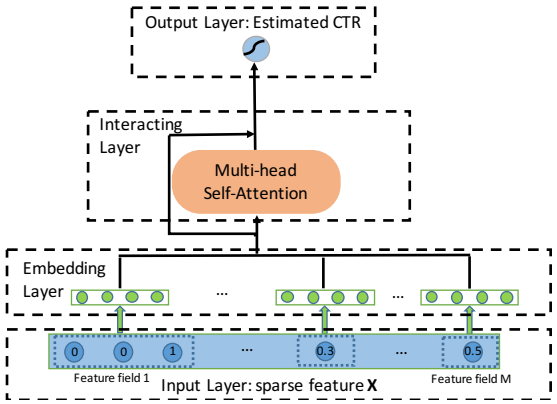


Figure 1: Overview of our proposed model AutoInt. The details of embedding layer and interacting layer are illustrated in Figure 2 and Figure 3 respectively.

4.1 Overview

The goal of our approach is to map the original sparse and high-dimensional feature vector into low-dimensional spaces and meanwhile model the high-order feature interactions. As shown in Figure 1, our proposed method takes the sparse feature vector $\mathbf{x}$ as input, followed by an embedding layer that projects all features (i.e., both categorical and numerical features) into the same low-dimensional space. Next, we feed embeddings of all fields into a novel interacting layer, which is implemented as a multi-head self-attentive neural network. For each interacting layer, high-order features are combined through the attention mechanism, and different kinds of combinations can be evaluated with the multi-head mechanisms, which map the features into different subspaces. By stacking multiple interacting layers, different orders of combinatorial features can be modeled.

The output of the final interacting layer is the low-dimensional representation of the input feature, which models the high-order combinatorial features and is further used for estimating the click-through rate through a sigmoid function. Next, we introduce the details of our proposed method.

4.2 Input Layer

We first represent user's profiles and item's attributes as a sparse vector, which is the concatenation of all fields. Specifically,

$$\mathbf{x} = [\mathbf{x_1}; \mathbf{x_2}; ...; \mathbf{x_M}], \tag{1}$$

where M is the number of total feature fields, and $\mathbf{x_i}$ is the feature representation of the i-th field. $\mathbf{x_i}$ is a one-hot vector if the i-th field is categorical (e.g., $\mathbf{x_1}$ in Figure 2). $\mathbf{x_i}$ is a scalar value if the i-th field is numerical (e.g., $\mathbf{x_M}$ in Figure 2).

4.3 Embedding Layer

Since the feature representations of the categorical features are very sparse and high-dimensional, a common way is to represent them

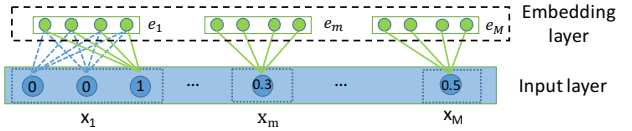


Figure 2: Illustration of input and embedding layer, where both categorical and numerical fields are represented by low-dimensional dense vectors.

into low-dimensional spaces (e.g., word embeddings). Specifically, we represent each categorical feature with a low-dimensional vector, i.e.,

$$\mathbf{e_i} = \mathbf{V_i}\mathbf{x_i}, \tag{2}$$

where $\mathbf{V_i}$ is an embedding matrix for field i, and $\mathbf{x_i}$ is an one-hot vector.

To allow the interaction between categorical and numerical features, we also represent the numerical features in the same low-dimensional feature space. Specifically, we represent the numerical feature as

$$\mathbf{e_m} = \mathbf{v_m}x_m, \tag{3}$$

where $\mathbf{v_m}$ is an embedding vector for field m, and x_m is a scalar value.

By doing this, the output of the embedding layer would be a concatenation of multiple embedding vectors, as presented in Figure 2.

4.4 Interacting Layer

Once the numerical and categorical features live in the same low-dimensional space, we move to model high-order combinatorial features in the space. The key problem is to determine which features should be combined to form meaningful high-order features. Traditionally, this is accomplished by domain experts who create meaningful combinations based on their knowledge. In this paper, we tackle this problem with a novel method, the multi-head self-attention mechanism [32].

Multi-head self-attentive network [32] has recently achieved remarkable performance in modeling complicated relations. For example, it shows superiority for modeling arbitrary word dependency in machine translation [32] and sentence embedding [17], and has been successfully applied to capturing node similarities in graph embedding [33]. Here we extend this latest technique to model the correlations between different feature fields.

Specifically, we adopt the key-value attention mechanism [19] to determine which feature combinations are meaningful. Taking the feature m as an example, next we explain how to identify multiple meaningful high-order features involving feature m. We first define the correlation between feature m and feature k under a specific attention head h as follows:

$$\begin{aligned} \alpha^{(\mathbf{h})}_{\mathbf{m},\mathbf{k}} &= \frac{exp(\psi^{(h)}(\mathbf{e_m}, \mathbf{e_k}))}{\sum_{l=1}^{M} exp(\psi^{(h)}(\mathbf{e_m}, \mathbf{e_l}))}, \\ \psi^{(h)}(\mathbf{e_m}, \mathbf{e_k}) &= \langle \mathbf{W}^{(\mathbf{h})}_{\mathbf{Query}}\mathbf{e_m}, \mathbf{W}^{(\mathbf{h})}_{\mathbf{Key}}\mathbf{e_k} \rangle, \end{aligned} \tag{4}$$

where $\psi^{(h)}(\cdot,\cdot)$ is an attention function which defines the similarity between the feature m and k. It can be defined as a neural network or as simple as inner product, i.e., $\langle \cdot, \cdot \rangle$. In this work, we use inner product due to its simplicity and effectiveness. $\mathbf{W}^{(\mathbf{h})}_{\mathbf{Query}}$, $\mathbf{W}^{(\mathbf{h})}_{\mathbf{Key}} \in \mathbb{R}^{d' \times d}$ in Equation 4 are transformation matrices which map the original embedding space $\mathbb{R}^d$ into a new space $\mathbb{R}^{d'}$. Next, we update the representation of feature m in subspace h via combining all relevant features guided by coefficients $\alpha^{(\mathbf{h})}_{\mathbf{m},\mathbf{k}}$:

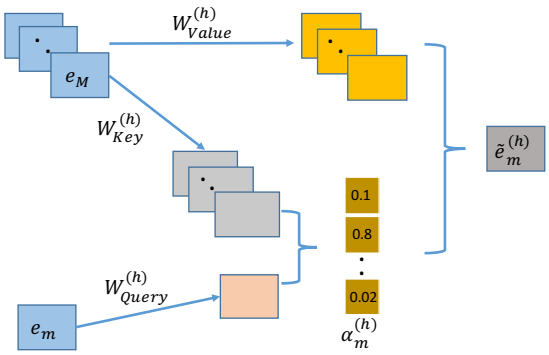


Figure 3: The architecture of interacting layer. Combinatorial features are conditioned on attention weights, i.e., $\alpha^{(\mathbf{h})}_{\mathbf{m}}$.

$$\widetilde{\mathbf{e}}^{(\mathbf{h})}_{\mathbf{m}} = \sum_{k=1}^{M} \alpha^{(\mathbf{h})}_{\mathbf{m},\mathbf{k}} (\mathbf{W}^{(\mathbf{h})}_{\mathbf{Value}}\mathbf{e_k}), \tag{5}$$

where $\mathbf{W}^{(\mathbf{h})}_{\mathbf{Value}} \in \mathbb{R}^{d' \times d}$.

Since $\widetilde{\mathbf{e}}^{(\mathbf{h})}_{\mathbf{m}} \in \mathbb{R}^{d'}$ is a combination of feature m and its relevant features (under head h), it represents a new combinatorial feature learned by our method. Furthermore, a feature is also likely to be involved in different combinatorial features, and we achieve this by using multiple heads, which create different subspaces and learn distinct feature interactions separately. We collect combinatorial features learned in all subspaces as follows:

$$\widetilde{\mathbf{e}}_{\mathbf{m}} = \widetilde{\mathbf{e}}^{(\mathbf{1})}_{\mathbf{m}} \oplus \widetilde{\mathbf{e}}^{(\mathbf{2})}_{\mathbf{m}} \oplus \cdots \oplus \widetilde{\mathbf{e}}^{(\mathbf{H})}_{\mathbf{m}}, \tag{6}$$

where $\oplus$ is the concatenation operator, and H is the number of total heads.

To preserve previously learned combinatorial features, including raw individual features, we add standard residual connections in our network. Formally,

$$\mathbf{e}^{\mathbf{Res}}_{\mathbf{m}} = ReLU(\widetilde{\mathbf{e}}_{\mathbf{m}} + \mathbf{W}_{\mathbf{Res}}\mathbf{e_m}), \tag{7}$$

where $\mathbf{W}_{\mathbf{Res}} \in \mathbb{R}^{d'H \times d}$ is the projection matrix in case of dimension mismatching [11], and $ReLU(z) = max(0, z)$ is a non-linear activation function.

With such an interacting layer, the representation of each feature $\mathbf{e_m}$ will be updated into a new feature representation $\mathbf{e}^{\mathbf{Res}}_{\mathbf{m}}$, which is a representation of high-order features. We can stack multiple such layers with the output of the previous interacting layer as the input of the next interacting layer. By doing this, we can model arbitrary-order combinatorial features.

Time Complexity Analysis. The main cost of one-step feature interaction is two-fold. First, calculating attention weights for one head takes $O(Mdd' + M^2d')$ time. Afterward, forming combinatorial features under one head also takes $O(Mdd' + M^2d')$ time. Because we have H heads, it takes $O(MHd'(M + d))$ time altogether. It is therefore efficient because H, d and d' are usually small.

4.5 Output Layer

The output of the interacting layer is a set of feature vectors $\{\mathbf{e}_\mathbf{m}^\mathbf{Res}\}_{m=1}^{M}$, which includes raw individual features reserved by residual block and combinatorial features learned via the multi-head self-attention mechanism. For final CTR prediction, we simply concatenate all of them and then apply a non-linear projection as follows:

$$\hat{y} = \sigma(\mathbf{w}^\mathbf{T}(\mathbf{e}_\mathbf{1}^\mathbf{Res} \oplus \mathbf{e}_\mathbf{2}^\mathbf{Res} \oplus \cdots \oplus \mathbf{e}_\mathbf{M}^\mathbf{Res}) + b), \tag{8}$$

where $\mathbf{w} \in \mathbb{R}^{d'HM}$ is a column projection vector which linearly combines concatenated features, b is the bias, and $\sigma(x) = (1 + exp(-x))^{-1}$ predicts the click-through rate.

4.6 Training

Our loss function is *Log loss*, which is defined as follows:

$$Logloss = -\frac{1}{N}\sum_{i=1}^{N}(y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i)), \tag{9}$$

where y_i and $\hat{y}_i$ are ground truth of user clicks and estimated CTR respectively, i indexes the training samples, and N is the total number of training samples. The parameters to learn in our model are $\{\mathbf{V_i}, \mathbf{v_m}, \mathbf{W}_\mathbf{Query}^\mathbf{(h)}, \mathbf{W}_\mathbf{Key}^\mathbf{(h)}, \mathbf{W}_\mathbf{Value}^\mathbf{(h)}, \mathbf{W_{Res}}, \mathbf{w}, b\}$, which are updated via minimizing the total *Logloss* using gradient descent.

5 EXPERIMENT

In this section, we move forward to evaluate the effectiveness of our proposed approach. We aim to answer the following questions:

- **RQ1** How does our proposed AutoInt perform on the problem of CTR prediction? Is it efficient for large-scale sparse and high-dimensional data?
- **RQ2** What are the influences of different model configurations?
- **RQ3** What are the dependency structures between different features? Is our proposed model explainable?
- **RQ4** Will integrating implicit feature interactions further improve the performance?

We first describe the experimental settings before answering these questions.

5.1 Experiment Setup

5.1.1 Data Sets. We use four public real-world data sets. The statistics of the data sets are summarized in Table 1.

Criteo[2] This is a benchmark dataset for CTR prediction, which has 45 million users' click records on displayed ads. It contains 26 categorical feature fields and 13 numerical feature fields.

Avazu[3] This dataset contains users' mobile behaviors including whether a displayed mobile ad is clicked by a user or not. It has 23 feature fields spanning from user/device features to ad attributes.

KDD12[4] This data set was released by KDDCup 2012, which originally aimed to predict the number of clicks. Since our work focuses on CTR prediction rather than the exact number of clicks, we treat this problem as a binary classification problem (1 for clicks>0, 0 for without click), which is similar to FFM [14].

Table 1: Statistics of evaluation data sets.

Data	#Samples	#Fields	#Features (Sparse)
Criteo	45,840,617	39	998,960
Avazu	40,428,967	23	1,544,488
KDD12	149,639,105	13	6,019,086
MovieLens-1M	739,012	7	3,529

MovieLens-1M[5] This dataset contains users' ratings on movies. During binarization, we treat samples with a rating less than 3 as negative samples because a low score indicates that the user does not like the movie. We treat samples with a rating greater than 3 as positive samples and remove neutral samples, i.e., a rating equal to 3.

Data Preparation First, we remove the infrequent features (appearing in less than *threshold* instances) and treat them as a single feature "<unknown>", where *threshold* is set to {10, 5, 10} for Criteo, Avazu and KDD12 data sets respectively. Second, since numerical features may have large variance and hurt machine learning algorithms, we normalize numerical values by transforming a value z to $log^2(z)$ if $z > 2$, which is proposed by the winner of Criteo Competition[6]. Third, we randomly select 80% of all samples for training and randomly split the rest into validation and test sets of equal size.

5.1.2 Evaluation Metrics. We use two popular metrics to evaluate the performance of all methods.

AUC Area Under the ROC Curve (AUC) measures the probability that a CTR predictor will assign a higher score to a randomly chosen positive item than a randomly chosen negative item. A higher AUC indicates a better performance.

Logloss Since all models attempt to minimize the *Logloss* defined by Equation 9, we use it as a straightforward metric.

It is noticeable that a slightly higher AUC or lower *Logloss* at ***0.001-level*** is regarded significant for CTR prediction task, which has also been pointed out in existing works [7, 10, 34].

5.1.3 Competing Models. We compare the proposed approach with three classes of previous models. (A) the linear approach that only uses individual features. (B) factorization machines-based methods that take into account second-order combinatorial features. (C) techniques that can capture high-order feature interactions. We associate the model classes with model names accordingly.

LR (A). LR only models the linear combination of raw individual features.

FM [23] (B). FM uses factorization techniques to model second-order feature interactions.

AFM [36] (B). AFM[7] is one of the state-of-the-art models that capture second-order feature interactions. It extends FM by using attention mechanism to distinguish the different importance of second-order combinatorial features.

DeepCrossing [29] (C). DeepCrossing utilizes deep fully-connected neural networks with residual connections to learn non-linear feature interactions in an implicit fashion.

[2] https://www.kaggle.com/c/criteo-display-ad-challenge
[3] https://www.kaggle.com/c/avazu-ctr-prediction
[4] https://www.kaggle.com/c/kddcup2012-track2
[5] https://grouplens.org/datasets/movielens/
[6] https://www.csie.ntu.edu.tw/~r01922136/kaggle-2014-criteo.pdf
[7] https://github.com/sunchenglong/attentional_factorization_machine

Table 2: Effectiveness Comparison of Different Algorithms. We highlight that our proposed model almost outperforms all baselines across four data sets and both metrics. Further analysis is provided in Section 5.2.

Model Class	Model	Criteo		Avazu		KDD12		MovieLens-1M	
		AUC	Logloss	AUC	Logloss	AUC	Logloss	AUC	Logloss
First-order	LR	0.7820	0.4695	0.7560	0.3964	0.7361	0.1684	0.7716	0.4424
Second-order	FM [23]	0.7836	0.4700	0.7706	0.3856	0.7759	0.1573	0.8252	0.3998
	AFM[36]	0.7938	0.4584	0.7718	0.3854	0.7659	0.1591	0.8227	0.4048
High-order	DeepCrossing [29]	0.8012	0.4513	0.7643	0.3889	0.7715	0.1591	0.8453	0.3814
	NFM [12]	0.7957	0.4562	0.7708	0.3864	0.7515	0.1631	0.8357	0.3883
	CrossNet [34]	0.7907	0.4591	0.7667	0.3868	0.7773	0.1572	0.7968	0.4266
	CIN [16]	0.8009	0.4517	**0.7758**	0.3829	0.7800	0.1566	0.8286	0.4108
	HOFM [4]	0.8005	0.4508	0.7701	0.3854	0.7707	0.1586	0.8304	0.4013
	AutoInt (ours)	**0.8061**	**0.4454**	0.7752	**0.3823**	**0.7881**	**0.1545**	**0.8460**	**0.3784**

NFM [12] (C). NFM[8] stacks deep neural networks on top of second-order feature interaction layer. High-order feature interactions are captured implicitly by the non-linear activations of neural networks.

CrossNet [34] (C). Cross Network, which is the core of Deep&Cross model, takes outer product of concatenated feature vector at the bit-wise level to model feature interactions explicitly.

CIN [16] (C). Compressed Interaction Network, which is the core of xDeepFM model, takes outer product of stacked feature matrix at vector-wise level.

We will compare with the full models of CrossNet and CIN, i.e., Deep&Cross and xDeepFM, in a joint training setting later.

HOFM [4] (C). HOFM proposes efficient kernel-based algorithms for training high-order factorization machines. Follow settings in Blondel et al. [4] and He and Chua [12], we build a third-order factorization machine using public implementation[9].

5.1.4 Implementation Details. [10] All methods are implemented in TensorFlow[1]. We use an embedding dimension of 16 and batch size of 1024 for all methods. Hidden units d' is set to 32. We use Adam [15] to optimize all deep neural network-based models. DeepCrossing has four feed-forward layers, each with 100 hidden units. We use one hidden layer of size 200 on top of Bi-Interaction layer for NFM as recommended by their paper. For CN and CIN, we use three interaction layers consistently. To prevent overfitting, we add dropout[30] with ratio 0.5 for a small dataset, i.e., MovieLens-1M, and we found it not necessary for other three large data sets. Except for special mention, we stack three interacting layers in the following experiments and use two attention heads in each layer.

5.2 Quantitative Results (RQ1)

Evaluation of Effectiveness

The performance of different algorithms is summarized in Table 2. We have the following observations:

(1) FM and AFM, which explore second-order feature interactions, consistently outperform LR by a large margin on all datasets, which indicates that individual features are insufficient in CTR prediction.

(2) An interesting observation is the inferiority of some models which capture high-order feature interactions. For example, although DeepCrossing and NFM use the deep neural network as a core component to learning high-order feature interactions, they do not guarantee improvement over FM and AFM. This may attribute to the fact that they learn feature interactions in an implicit fashion. On the contrary, CIN does it explicitly and outperforms low-order models consistently.

(3) HOFM outperforms FM in most cases except for KDD12 dataset, which indicates that modeling third-order feature interactions is beneficial to prediction performance.

(4) AutoInt achieves the best performance overall baseline methods on three of four real-world data sets. On Avazu data set, CIN performs a little better than AutoInt in AUC evaluation, but we get lower *Logloss*. Note that our proposed AutoInt shares the same structures as DeepCrossing except the feature interacting layer, which indicates using the attention mechanism to learn explicit combinatorial features is crucial.

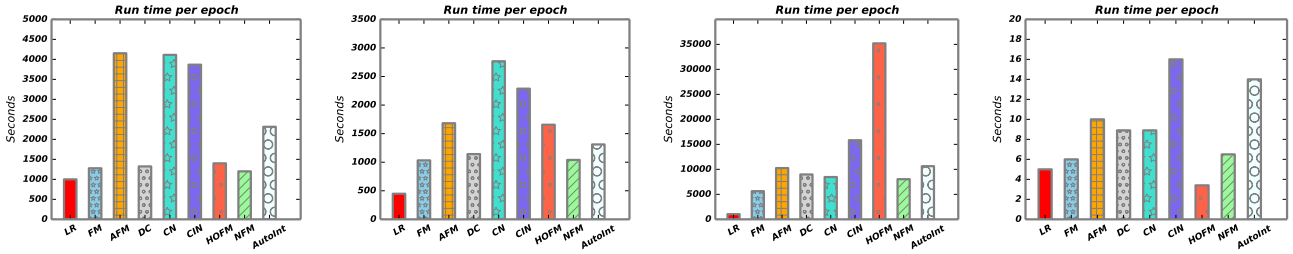
Evaluation of Model Efficiency

We present the runtime results of different algorithms on four data sets in Figure 4. Unsurprisingly, LR is the most efficient algorithm due to its simplicity. FM and NFM perform similarly in terms of runtime because NFM only stacks a single feed-forward hidden layer on top of the second-order interaction layer. Among all listed methods, CIN, which achieves the best performance for prediction among all the baselines, is much more time-consuming due to its complicated crossing layer. This may make it impractical in the industrial scenarios. Note that AutoInt is sufficiently efficient, which is comparable to the efficient algorithms DeepCrossing and NFM.

We also compare the sizes of different models (i.e., the number of parameters) as another criterion for efficiency evaluation. As shown in Table 3, comparing to the best model CIN in the baseline models, the number of parameters in AutoInt is much smaller.

To summarize, our proposed AutoInt achieves the best performance among all the compared models. Compared to the most competitive baseline model CIN, AutoInt requires much fewer parameters and is much more efficient during online inference.

[8] https://github.com/SyncWorld/neural_factorization_machine
[9] https://github.com/geffy/tffm
[10] We will release our code upon publishing.



(a) Criteo (b) Avazu (c) KDD12 (d) MovieLens-1M

Figure 4: Efficiency Comparison of Different Algorithms in terms of *Run Time*. "DC" and "CN" are DeepCrossing and CrossNet for short, respectively. Since HOFM cannot be fit on one GPU card for the KDD12 dataset, extra communication cost makes it most time-consuming. Further analysis is presented in Section 5.2.

Table 3: Efficiency Comparison of Different Algorithms in terms of *Model Size* on Criteo data set. "DC" and "CN" are DeepCrossing and CrossNet for short, respectively. We only take the parameters above the embedding layer into account.

Model	DC	CN	CIN	NFM	AutoInt
#Params	1.6×10^5	3×10^3	1.9×10^6	4×10^3	3.9×10^4

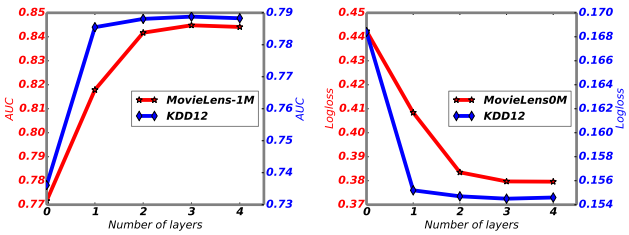
Table 4: Ablation study comparing the performance of AutoInt with and without residual connections. $\text{AutoInt}_{\text{w/}}$ is the complete model while the $\text{AutoInt}_{\text{w/o}}$ is the model without residual connection.

Data Sets	Models	AUC	Logloss
Criteo	$\text{AutoInt}_{\text{w/}}$	0.8061	0.4454
	$\text{AutoInt}_{\text{w/o}}$	0.8033	0.4478
Avazu	$\text{AutoInt}_{\text{w/}}$	0.7752	0.3823
	$\text{AutoInt}_{\text{w/o}}$	0.7729	0.3836
KDD12	$\text{AutoInt}_{\text{w/}}$	0.7888	0.1545
	$\text{AutoInt}_{\text{w/o}}$	0.7831	0.1557
MovieLens-1M	$\text{AutoInt}_{\text{w/}}$	0.8460	0.3784
	$\text{AutoInt}_{\text{w/o}}$	0.8299	0.3959

5.3 Analysis (RQ2)

To further validate and gain deep insights into the proposed model, we conduct ablation study and compare several variants of AutoInt.

5.3.1 Influence of Residual Structure. The standard AutoInt utilizes residual connections, which carry through all learned combinatorial features and therefore allow modeling very high-order combinations. To justify the contribution of residual units, we tease apart them from our standard model and keep other structures as they are. As presented in Table 4, we observe that the performance decrease on all datasets if residual connections are removed. Specifically, the full model outperforms the variant by a large margin on the KDD12 and MovieLens-1M data, which indicates residual connections are crucial to model high-order feature interactions in our proposed method.



(a) AUC (b) Logloss

Figure 5: Performance w.r.t. the number of interacting layers. Results on Criteo and Avazu data sets are similar and hence omitted.

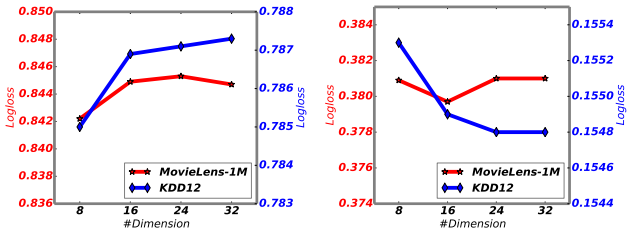
5.3.2 Influence of Network Depths. Our model learns high-order feature combinations by stacking multiple interacting layers (introduced in Section 4). Therefore, we are interested in how the performance change w.r.t. the number of interacting layers, i.e., the order of combinatorial features. Note that when there is no interacting layer (i.e., *Number of layers* equals zero), our model takes the weighted sum of raw individual features as input, i.e., no combinatorial features are considered.

The results are summarized in Figure 5. We can see that if one interacting layer is used, i.e., feature interactions are taken into account, the performance increase dramatically on both data sets, showing that combinatorial features are very informative for prediction. As the number of interacting layers further increases, i.e., higher-order combinatorial features are taken into account, the performance of the model further increases. When the number of layers reaches three, the performance becomes stable, showing that adding extremely high-order features are not informative for prediction.

5.3.3 Influence of Different Dimensions. Next, we investigate the performance w.r.t. the parameter d, which is the output dimension of the embedding layer. On the KDD12 dataset, we can see that the performance continuously increase as we increase the dimension size since larger models are used for prediction. The results are different on the MovieLens-1M dataset. When the dimension size reaches 24, the performance begins to decrease. The reason is that this data set is small, and the model is overfitted when too many parameters are used.

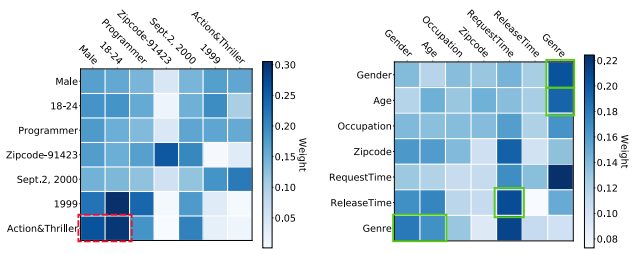
Table 5: Results of Integrating Implicit Feature Interactions. We indicate the base model behind each method. The last two columns are average changes of AUC and *Logloss* compared to corresponding base models ("+": increase, "-": decrease).

Model	Criteo		Avazu		KDD12		MovieLens-1M		Avg. Changes	
	AUC	Logloss	AUC	Logloss	AUC	Logloss	AUC	Logloss	AUC	Logloss
Wide&Deep (LR)	0.8026	0.4494	0.7749	0.3824	0.7549	0.1619	0.8300	0.3976	+0.0292	-0.0213
DeepFM (FM)	0.8066	0.4449	0.7751	0.3829	0.7867	0.1549	0.8437	0.3846	+0.0142	-0.0113
Deep&Cross (CN)	0.8067	0.4447	0.7731	0.3836	0.7869	0.1550	0.8446	0.3809	+0.0199	-0.0164
xDeepFM (CIN)	0.8070	0.4447	0.7768	0.3832	0.7820	0.1560	0.8467	0.3800	+0.0068	-0.0068
AutoInt+ (ours)	**0.8080**	**0.4437**	**0.7771**	**0.3811**	**0.7892**	**0.1544**	**0.8486**	**0.3757**	+0.0019	-0.0014



(a) AUC (b) Logloss

Figure 6: Performance w.r.t. number of embedding dimensions. Results on Criteo and Avazu data sets are similar and hence omitted.



(a) Label=1, Predicted CTR=0.89 (b) Overall feature interactions

Figure 7: Heat maps of attention weights for both case- and global-level feature interactions on MovieLens-1M. The axises represent feature fields <*Gender, Age, Occupation, Zipcode, RequestTime, RealeaseTime, Genre*>. We highlight some learned combinatorial features in rectangles.

5.4 Explainable Recommendations (RQ3)

A good recommender system can not only provide good recommendations but also offer good explainability. Therefore, in this part, we present how our AutoInt is able to explain the recommendation results. We take the MovieLens-1M dataset as an example.

Let's look at a recommendation result suggested by our algorithm, i.e., a user likes an item. Figure 7 (a) presents the correlations between different fields of input features, which are obtained by the attention score. We can see that AutoInt is able to identify the meaningful combinatorial feature <*Gender=Male, Age=[18-24), MovieGenre=Action&Triller*> (i.e., red dotted rectangle). This is very reasonable since young men are very likely to prefer action&triller movies.

We are also interested in what the correlations between different feature fields in the data are. Therefore, we measure the correlations between the feature fields according to their average attention score in the entire data. The correlations between different fields are summarized into Figure 7 (b). We can see that <*Gender, Genre*>, <*Age, Genre*>, <*RequestTime, ReleaseTime*> and <*Gender, Age, Genre*> (i.e., solid green region) are strongly correlated, which are the explainable rules for recommendation in this domain.

5.5 Integrating Implicit Interactions (RQ4)

Feed-forward neural networks are capable of modeling implicit feature interactions and have been widely integrated into existing CTR prediction methods [7, 10, 16]. To investigate whether integrating implicit feature interactions further improves the performance, we combine AutoInt with a two-layer feed-forward neural network by joint training. We name the joint model *AutoInt+* and compare it with the following algorithms:

- Wide&Deep [7]. Wide&Deep integrates the outputs of logistic regression and feed-forward neural networks.
- DeepFM [10]. DeepFM combines FM and feed-forward neural network, with a shared embedding layer.
- Deep&Cross [34]. Deep&Cross is the extension of CrossNet by integrating feed-forward neural networks.
- xDeepFM [16]. xDeepFM is the extension of CIN by integrating feed-forward neural networks.

Table 5 presents the evaluation results of joint-training models. We have the following observations: 1) the performance of our method improves by joint training with feed-forward neural networks on all datasets. This indicates that integrating implicit feature interactions indeed boosts the predictive ability of our proposed model. However, as can be seen from last two columns, the magnitude of performance improvement is fairly small compared to other models, showing that our individual model AutoInt is quite powerful. 2) after integrating implicit feature interactions, AutoInt+ outperforms all competitive methods, and achieves new state-of-the-art performances on used CTR prediction data sets.

6 CONCLUSION

In this work, we propose a novel CTR prediction model based on self-attention mechanism, which can automatically learn high-order feature interactions in an explicit fashion. The key to our method is the newly-introduced interacting layer, which allows each feature to interact with the others and to determine the relevance through learning. Experimental results on four real-world

data sets demonstrate the effectiveness and efficiency of our proposed model. Besides, we provide good model explainability via visualizing the learned combinatorial features. When integrating with implicit feature interactions captured by feed-forward neural networks, we achieve better offline AUC and *Logloss* scores compared to the previous state-of-the-art methods. In the future, we are interested in incorporating contextual information into our method and improving its performance for online recommender systems.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. TensorFlow: A System for Large-Scale Machine Learning.. In *OSDI*, Vol. 16. 265–283.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[3] Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Jia Li, Vince Gatto, and Ed H Chi. 2018. Latent Cross: Making Use of Context in Recurrent Recommender Systems. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 46–54.

[4] Mathieu Blondel, Akinori Fujino, Naonori Ueda, and Masakazu Ishihata. 2016. Higher-order factorization machines. In *Advances in Neural Information Processing Systems*. 3351–3359.

[5] Mathieu Blondel, Masakazu Ishihata, Akinori Fujino, and Naonori Ueda. 2016. Polynomial Networks and Factorization Machines: New Insights and Efficient Training Algorithms. In *International Conference on Machine Learning*. 850–858.

[6] Chen Cheng, Fen Xia, Tong Zhang, Irwin King, and Michael R Lyu. 2014. Gradient boosting factorization machines. In *Proceedings of the 8th ACM Conference on Recommender systems*. ACM, 265–272.

[7] 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.

[8] 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.

[9] 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.

[10] 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).

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

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

[13] 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.

[14] Yuchin 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*. ACM, 43–50.

[15] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[16] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems. *arXiv preprint arXiv:1803.05170* (2018).

[17] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130* (2017).

[18] H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, 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*. ACM, 1222–1230.

[19] Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126* (2016).

[20] Alexander Novikov, Mikhail Trofimov, and Ivan Oseledets. 2016. Exponential machines. *arXiv preprint arXiv:1605.03795* (2016).

[21] Richard J Oentaryo, Ee-Peng Lim, Jia-Wei Low, David Lo, and Michael Finegold. 2014. Predicting response in mobile advertising with hierarchical importance-aware factorization machine. In *Proceedings of the 7th ACM international conference on Web search and data mining*. ACM, 123–132.

[22] 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 *Data Mining (ICDM), 2016 IEEE 16th International Conference on*. IEEE, 1149–1154.

[23] Steffen Rendle. 2010. Factorization machines. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*. IEEE, 995–1000.

[24] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*. ACM, 811–820.

[25] Steffen Rendle, Zeno Gantner, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2011. Fast context-aware recommendations with factorization machines. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. ACM, 635–644.

[26] 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*. ACM, 521–530.

[27] Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685* (2015).

[28] Lili Shan, Lei Lin, Chengjie Sun, and Xiaolong Wang. 2016. Predicting ad click-through rates via feature-based fully coupled interaction tensor factorization. *Electronic Commerce Research and Applications* 16 (2016), 30–42.

[29] Ying Shan, T Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and JC Mao. 2016. Deep crossing: Web-scale modeling without manually crafted combinatorial features. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 255–262.

[30] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[31] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*. 2440–2448.

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. 6000–6010.

[33] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph Attention Networks. *stat* 1050 (2017), 20.

[34] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & Cross Network for Ad Click Predictions. *arXiv preprint arXiv:1708.05123* (2017).

[35] Xiang Wang, Xiangnan He, Fuli Feng, Liqiang Nie, and Tat-Seng Chua. 2018. TEM: Tree-enhanced Embedding Model for Explainable Recommendation. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 1543–1552.

[36] 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).

[37] Weinan Zhang, Tianming Du, and Jun Wang. 2016. Deep learning over multi-field categorical data. In *European conference on information retrieval*. Springer, 45–57.

[38] Qian Zhao, Yue Shi, and Liangjie Hong. 2017. GB-CENT: Gradient Boosted Categorical Embedding and Numerical Trees. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 1311–1319.

[39] Guorui Zhou, Chengru Song, Xiaoqiang Zhu, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2017. Deep interest network for click-through rate prediction. *arXiv preprint arXiv:1706.06978* (2017).

[40] Jie Zhu, Ying Shan, JC Mao, Dong Yu, Holakou Rahmanian, and Yi Zhang. 2017. Deep embedding forest: Forest-based serving with deep embedding features. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1703–1711.