Efficient Feature Interactions Learning with Gated Attention Transformer

Chao Long, Yanmin Zhu(✉), Haobing Liu, and Jiadi Yu

Shanghai Jiao Tong University, Shanghai 200240, China
{longchao,yzhu,liuhaobing,jdyu}@sjtu.edu.cn

Abstract. Click-through rate (CTR) prediction plays a key role in many domains, such as online advising and recommender system. In practice, it is necessary to learn feature interactions (i.e., cross features) for building an accurate prediction model. Recently, several self-attention based transformer methods are proposed to learn feature interactions automatically. However, those approaches are hindered by two drawbacks. First, Learning high-order feature interactions by using self-attention will generate many repetitive cross features because k-order cross features are generated by crossing *(k–1)*-order cross features and *(k–1)*-order cross features. Second, introducing useless cross features (e.g., repetitive cross features) will degrade model performance. To tackle these issues but retain the strong ability of the Transformer, we combine the vanilla attention mechanism with the gated mechanism and propose a novel model named Gated Attention Transformer. In our method, k-order cross features are generated by crossing *(k–1)*-order cross features and *1*-order features, which uses the vanilla attention mechanism instead of the self-attention mechanism and is more explainable and efficient. In addition, as a supplement of the attention mechanism that distinguishes the importance of feature interactions at the vector-wise level, we further use the gated mechanism to distill the significant feature interactions at the bit-wise level. Experiments on two real-world datasets demonstrate the superiority and efficacy of our proposed method.

Keywords: Feature interactions · CTR prediction · Transformer

1 Introduction

Click-through rate (CTR) prediction is a critical task in many domains, such as online advertising and recommender systems, since a small improvement to the task will bring a lot of revenue [7,21]. Taking online advertising as an example, when publishers have displayed the advertisements provided by the advertisers, their revenue depends on whether the users will click on these advertisements. That is to say, advertisers only pay for ads that users have clicked on. Considering the large amounts of existing users, a minor improvement to the prediction model means millions of additional users will click on advertisements, which will bring a large amount of revenue. Thus, this field attracts more and more interest from both academia and industry.

Cross features, especially high-order cross features (i.e., feature interactions), are the critical factors of a successful CTR prediction model. For example, suppose we have a record including three features {Age = 20, Gender = Male, ProductCategory=Auto advertisement}, the two-order cross features ⟨Age = 20, Gender = Male⟩ is obviously more predictive than the raw features ⟨Gender = Male⟩ for the auto advertisement. Thus, using only raw features often leads to sub-optimal results [18].

To model high-order cross features, recently several self-attention based methods are proposed, such as AutoInt [24] and InterHat [16]. Despite the superior performance of these methods, two drawbacks will hinder model performance. First, these methods generate k-order feature interactions by crossing *(k–1)*-order feature interactions and *(k-1)*-order feature interactions, which is counter-intuitive and will introduce large amount repetitive cross features. Second, introducing useless feature interactions will degrade model performance.

To cope with the problems above, we propose a novel method named Gated Attention Transformer. The key idea is to generate k-order feature interactions by crossing *(k–1)*-order feature interactions and *1*-order features, which is more intuitive and efficient. In addition, since each representation vector represents multiple cross features when modeling feature interactions automatically, the attention mechanism that distinguishes the importance of feature interactions at the vector-wise level loses its efficiency. Thus, the gated mechanism that distills the significant feature interactions at the bit-wise level is used to complement the attention mechanism. Specifically, the core of the proposed method is a transformer-like encoder, which consists of a gated multi-head attention layer and a gated distillation network. The gated multi-head attention layer first crosses the *(k–1)*-order cross features and the *1*-order features by using the vanilla attention mechanism. Afterwards, it learns an update gate for each head, which merges different head adaptively. Though cross features can be learned by the gated multi-head attention layer, each vector represents multiple cross features. Thus, the vanilla attention mechanism loses its efficiency for distinguishing the importance of feature interactions at the vector-wise level. To tackle these problems, a gated distillation network that alleviates the effects of useless cross features at the bit-wise level is used. By stacking multiple such encoders, the method can learn feature interactions of different orders.

To summarize, we make the following contributions:

- We propose a novel method to learn feature interactions efficiently by combining the gated mechanism and the Transformer architecture.
- We design a Transformer-like encoder, which not only retains the Transformer's architecture but tackles the drawbacks of the self-attention mechanism. Besides, we combine the attention mechanism with the gated mechanism for overcoming the limitation of the attention mechanism in the process of learning feature interactions automatically.

- We conduct extensive experiments on two real-world datasets to evaluate our method, and the results demonstrate that our model outperforms several representative state-of-art methods.

The rest of this paper is organized as follows: Sect. 2 gives a brief review of the related work in the field of CTR prediction. Section 3 provides some important concepts and a problem formulation. Section 4 introduces our proposed method. Section 5 presents the results of extensive experiments on two public datasets. Section 6 concludes and points out some future research directions.

2 Related Work

In this section, we briefly review the existing approaches in the field of CTR prediction.

2.1 Traditional Models for CTR Prediction

Linear Regression (LR) is widely used in CTR prediction for its simplicity and efficiency, which learns a unique weight for each feature. However, human efforts are needed to generate high-order cross features, which is unacceptable for its high cost. To model feature interactions automatically, Factorization machine (FM) [23] and many variants are proposed [11–13]. FM models two-order feature interactions based on LR, which learns a latent vector for each raw feature and uses inner product operation to model two-order feature interactions. Field-aware FM (FFM) [13] considers the field information and learns a field-aware latent vector for each raw feature. Attentional FM (AFM) [28] considers that different two-order cross features have different significance. Therefore, it uses an attention network to acquire a significance score for each cross feature [1]. Though HOFM [3] can extend FM to model high-order feature interactions by efficient training algorithms, it is hard to apply it to real predictive systems [3].

2.2 Deep Neural Network Based Models for CTR Prediction

With the success of deep neural network (DNN) in various fields [8,19,30], a lot of methods using DNN to model high-order feature interactions are proposed. For example, Product-based Neural Network (PNN) [22] concatenates the raw features and the two-order cross features and feeds them into a DNN to model high-order cross features. Neural FM (NFM) [9] uses a Bi-interaction pooling layer to model two-order feature interactions and feeds them into a DNN to model high-order feature interactions. Considering both low-order and high-order cross features are important for prediction, Wide&Deep [4] uses a linear part to model low-order feature interactions and a deep part, which is a DNN, to model high-order feature interactions. DeepFM [7] combines FM with DNN to model both low- and high-order cross features jointly.

Neural Architecture Search (NAS) is proposed to search the most proper architecture of a neural network. Inspired by NAS, some methods utilizing

AutoML to seek useful feature interactions are proposed. For example, AutoGroup [18] first find multiple subsets of features, where the feature interactions among each feature subset are effective. Then, it models high-order feature interactions based on these effective feature subsets.

2.3 Self-attention Based Models for CTR Prediction

The Attention mechanism learns a function that measures the importance of different things. It is originally proposed for neural machine translation (NMT) [6] and is widely used in different domains, such as recommender system [16,20], due to its capability to learn the importance of things.

Self-attention is a form of the attention mechanism. Researchers from Google have designed the Transformer [25], which is entirely based on multi-head self-attention and achieves state-of-the-art performance on multiple NLP tasks. Inspired by the success of Transformer in different domains, several self-attention based methods are proposed to model high-order feature interactions. AutoInt [24] uses the multi-head self-attention mechanism to learn high-order feature interactions. By stacking the different number of self-attention layers, it can learn different orders of feature interactions. InterHat [16] first uses a self-attention based transformer encoder to model the polysemy of features and then learns different orders of feature interactions by hierarchical attention. Though self-attention based methods achieve state-of-the-art performance, some drawbacks remain to be tackled.

3 Preliminaries

In this section, we briefly introduce some notations and the problem statement.

Definition 1: Field and Feature. In this paper, the term *field* refers to the name of attributes, such as *gender,city*. The term *feature* refers to values of the corresponding field, such as the feature *male* for the field *gender* and the feature *Washington* for the field *city*.

Definition 2: P-order Cross Features. Given an input vector $x \in \mathbb{R}^m$ that x_i is the feature of field i, a p-order cross feature is defined as $g(x_{i1}, \dots, x_{ip})$, where x_{ik} is an arbitrary feature of x, p is the number of distinct features involved, and $g(\cdot)$ is a non-additive interaction function, such as inner product [16,21,23] and outer product [17]. A p-order cross feature models the p-order feature interaction among the corresponding features.

Problem Statement. We assume that the training dataset composes of m categorical fields (i.e., user id, item id, etc.) and N records. Therefore, each record of the dataset is consisted of m categorical features and an associated label $y \in \{0, 1\}$. We suppose that all fields involved here are categorical because most fields in this problem are either categorical or can be made categorical through discretization. Let $x \in \mathbb{R}^m$ denotes the concatenation of categorical features of a record, where x_i is the feature of field i. Our goal is to predict a $\hat{y} \in (0, 1)$ for the record based on the input vector x.

4 Method

In this section, we first give an overview of our proposed model depicted in Fig. 1. Afterward, we present a comprehensive description of each component.

4.1 Overview

We design a novel model to learn feature interactions efficiently with the following considerations: (1) Though Transformer is successfully applied to various domains, the self-attention mechanism fails to learn feature interactions efficiently in CTR prediction. (2) Useless feature interactions will degrade model performance. The attention mechanism loses its efficiency when learning feature interactions automatically.

Figure 1 shows an overview of our method. The sparse input features are first mapped into a low-dimensional vector space by an embedding layer. Thus, each feature is represented with a dense vector. Afterward, we feed these embedding vectors into several encoders to model feature interactions of different orders. The details of this encoder are introduced in Sect. 4.4. Finally, we employ an aggregation layer to aggregate all cross features of different orders and predict the result based on it. Next, we introduce the details of each component.

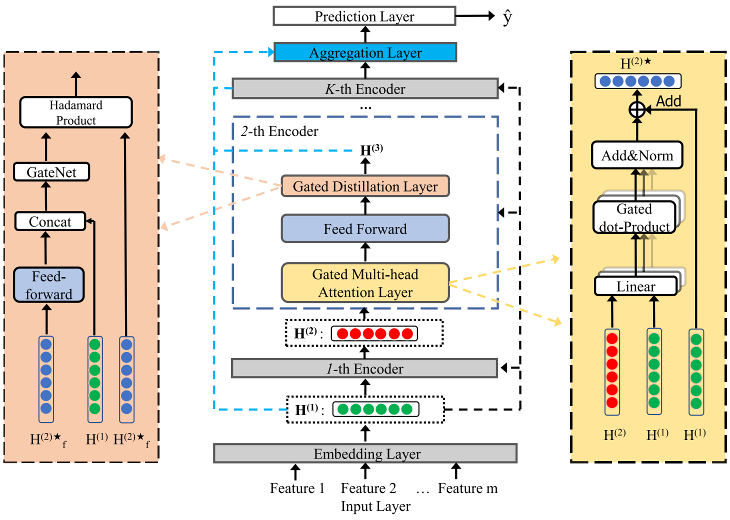


Fig. 1. The overview of our proposed method. By stacking multiple encoders, multi-order feature interactions can be modeled.

4.2 Input Layer

Suppose each record contains m fields, then the input is the concatenation of all fields. Specifically,

$$x = [x_1, x_2, \ldots, x_m]^T, \tag{1}$$

where x_i is the feature of the i-th field. x_i is a one-hot vector if the i-th field is categorical. Otherwise, x_i is a scalar value if the i-th field is numerical.

4.3 Embedding Layer

Since the one-hot representation of features is sparse and high-dimensional, it is necessary to represent each feature as a dense and low-dimensional vector. Taking the record {UserId=95, Gender=male, ItemId=27} as an example, the record becomes sparse and high-dimensional after one-hot encoding:

$$[\underbrace{0, 0, \cdots, 1, \cdots, 0, 0}_{UserId}, \underbrace{1, 0}_{Gender}, \underbrace{0, 0, \cdots, 1, \cdots, 0, 0}_{ItemId}], \tag{2}$$

which is hard to train. Thus, we apply an embedding layer to represent each feature as a dense and low-dimensional vector. Specifically,

$$e_i = V_i \times x_i, \tag{3}$$

where V_i is the embedding lookup table of the i-th field, and x_i the one-hot representation of the feature of the i-th field. If the field is multivalent, we take the average of the corresponding feature representations as the result. Take the field *Genre* as an example, a movie having multiple genres (e.g., Comedy and Romance) corresponds to a multi-hot representation. We consider the Comedy and Romance as a new feature of the filed *Genre*, and represent it as

$$e_{CR} = \frac{1}{2} V_{genre} \times x_{CR}, \tag{4}$$

where V_{genre} is the embedding lookup table of the field *Genre*, and x_{CR} is the multi-hot representation of the feature Comedy and Romance. In this way, the result of the embedding layer is an matrix:

$$H^{(1)} = [e_1, e_2, \ldots, e_m]^T, \tag{5}$$

where $H^{(1)} \in \mathbb{R}^{m \times d}$, $e_i \in \mathbb{R}^d$ denotes the embedding vector of the feature of the i-th field, and d denotes the dimension of feature embedding.

4.4 Encoder

The k-th encoder models $(k+1)$-order feature interactions. It takes $H^{(1)} \in \mathbb{R}^{m \times d}$ and $H^{(k)} \in \mathbb{R}^{m \times d}$ as inputs, and output $H^{(k+1)} \in \mathbb{R}^{m \times d}$, where $H^{(1)}$ denotes the embedding vectors of raw features, $H^{(k)}$ denotes the representations of k-order cross features, and $H^{(k+1)}$ denotes the representations of $(k+1)$-order cross features. As shown in Fig. 1, each encoder is composed of two different layers. Next, we introduce each layer in detail.

Gated Multi-head Attention Layer. The multi-head self-attention mechanism is first proposed by Google [25] and proved effective in different domains, such as machine translation [6] and recommender systems [20,27]. Considering the drawbacks of the self-attention mechanism, the vanilla attention mechanism is more efficient here. In addition, inspired by GaAN [29] that learns a importance score for each head since heads are not equally important, we combine the attention mechanism with the gated mechanism and propose a novel gated multi-head attention layer. Different from the GaAN, the gated multi-head attention layer learns a gated vector for each head and weighted each head at the bit-wise level, since each head models multiple feature interactions in each subspace.

Specifically, for the gated multi-head attention layer of the k-th encoder, the inputs are the representations of k-order cross features and the embedding vectors of raw features, which are denoted as $H^{(k)} \in \mathbb{R}^{m \times d}$ and $H^{(1)} \in \mathbb{R}^{m \times d}$ respectively. In contrast to the vanilla multi-head attention network, we integrate a neural gating structure:

$$g_i = sigmoid(GateNet_1([H^{(k)}||H^{(1)}])), \tag{6}$$

where $||$ denotes the concatenation operation, $GateNet_1$ is a feed-forward neural network, and g_i is the gate learned for head i. Then, we distinguish the importance of feature interactions by using both the attention mechanism and the gating mechanism:

$$\alpha_i = softmax(\frac{Q_i K_i^T}{\sqrt{d_K}}), \tag{7}$$

$$Q_i = W_i^{(Q)} H^{(k)}, K_i = W_i^{(K)} H^{(1)}, \tag{8}$$

where $W_i^{(Q)} \in \mathbb{R}^{d_K \times d}, W_i^{(K)} \in \mathbb{R}^{d_K \times d}$ are parameters to learn for head i, d_K denotes the dimension of each head, and α_i is the final importance score of cross features. Finally, we obtain the representations by concatenating the outputs of all heads:

$$H_i^{(k)*} = g_i \odot \alpha_i V_i, \tag{9}$$

$$H^{(k)*} = Norm([H_1^{(k)*}||H_2^{(k)*}||\dots||H_h^{(k)*}] + H^{(1)}), \tag{10}$$

$$V_i = W_i^{(V)} H^{(k)}, \tag{11}$$

where $W_i^{(V)} \in \mathbb{R}^{d_K \times d}$ is the parameter of head i, h is the number of heads, and $Norm$ represents the normalization.

Gated Distillation Layer. Insignificant features may lead to suboptimal results [18]. Thus, how to minimize the influence of those useless features becomes valuable. The attention mechanism can distinguish the importance of features and give an importance score at the vector-wise level. However, in the process of the high-order cross features modeling, each representation vector represents multiple cross features. Take the representation matrix $H^{(k)} \in \mathbb{R}^{m \times d}$ as an example, $H_i^{(k)}$ represents all the k-order cross features related to the feature of the

i-th field. Thus, the importance score at the vector level fails to distinguish those mixed cross features.

The gating mechanism is widely used in LSTM [10] and GRU [5] to control the information transmission. The core idea of it is to learn an update gate for the input at the bit-wise level, which controls how much the information can be transmitted. In this paper, we adopt the gating mechanism to alleviate the effect of useless cross features. Specifically, we learn a gate for each cross feature by a feed-forward neural network and employ the gate to control the transmission of information.

For the gated distillation layer of the k-th encoder, we take the output of the corresponding feed-forward neural network and the embedding vectors of raw features as inputs. Then, the output of the gated distillation layer is obtained by:

$$gate = sigmoid(GateNet_2(W_{gh}H_f^{(k)\star} + W_{go}H^{(1)}), \tag{12}$$

$$H^{(k+1)} = gate \odot H_f^{(k)\star}, \tag{13}$$

where $W_{gh} \in \mathbb{R}^{d \times d}, Wgo \in \mathbb{R}^{d \times d}$ are trainable parameters of the gated distillation layer, $GateNet_2$ is a feed-forward neural network, and $\odot$ is the Hadamard Product. The matrix $H^{(k+1)} \in \mathbb{R}^{m \times d}$ is also the output of the k-th encoder.

4.5 Aggregation Layer and Prediction Layer

Aggregation Layer. Each encoder generates a representation matrix $H^{(t)} \in \mathbb{R}^{m \times d}$, where t ranges from 2 to M, M is the number of encoders, and the matrix $H^{(t)}$ denotes the representations of t-order cross features. We regard the raw features as the one-order cross features. Thus, we aggregate these matrices by an aggregation layer:

$$H = \sum_{t=1}^{M} W_t H^{(t)}, \tag{14}$$

where $W_t \in \mathbb{R}$ are trainable parameters, $H^{(1)} \in \mathbb{R}^{m \times d}$ is the embedding matrix of raw features, and $H \in \mathbb{R}^{m \times d}$.

Prediction Layer. The output of the aggregation layer is the matrix $H \in \mathbb{R}^{m \times d}$, in which $H_i \in \mathbb{R}^d$ represents all cross features related to the raw feature of field i. For each H_i, we obtain the attention score by an attention network:

$$\alpha_i = \frac{exp(AttentionNet(H_i))}{\sum_s^m exp(AttentionNet(H_s))}, \tag{15}$$

where $AttentionNet$ is a single hidden layer feed-forward neural network. Then, we obtain the final prediction by:

$$\hat{y} = sigmoid(\sum_i^m \alpha_i PredictionNet(H_i)), \tag{16}$$

where $PredictionNet$ is also a single hidden layer feed-forward neural network.

Table 1. Statistics of evaluation datasets

Datasets	#Samples	#Fields	#Features
Frappe	288,609	10	5,382
Avazu	40,428,967	22	1,544,488

4.6 Training

We adopt Logloss to optimize the model parameters. Formally, the objective function of our method is defined as follows:

$$Logloss = -\frac{1}{N}\sum_{j=1}^{N}(y_j log(\hat{y_j}) + (1-y_j)log(1-\hat{y_j})) + \frac{\lambda}{2}(||\Theta||_F^2), \qquad (17)$$

where y_j and $\hat{y_j}$ are ground truth of the j-th sample and the estimated CTR respectively, j is the index of samples, N is the number of samples, and Θ denotes all the parameters of our method.

5 Experiments

In this section, we aim to answer the following questions:

- **RQ1:** Can our proposed method perform better than the baselines?
- **RQ2:** Are the critical components (e.g., gated multi-head attention layer) really effective for improving the model performance?
- **RQ3:** Is our proposed method more efficient than the self-attention based method?
- **RQ4:** How do the critical hyper-parameters (e.g., the number of encoders) affect the model performance?

5.1 Experiment Setup

Datasets. To answer the three questions, we conduct extensive experiments on two public real-world datasets: Frappe[1] and Avazu[2]. The Frappe [2] dataset contains records about users' app usage behaviors that whether a app is used by a user. The Avazu dataset contains records about users' ads click behaviors that whether a displayed mobile ad is clicked by a user. All the fields involved in both datasets are categorical. For the two datasets, we randomly select 80% of all samples for training, 10% for validating and 10% for testing. The statistics of Frappe and Avazu are shown in Table 1.

[1] http://baltrunas.info/research-menu/frappe.
[2] https://www.kaggle.com/c/avazu-ctr-prediction.

Table 2. Hyper-parameters of each model.

Model	Frappe			Avazu		
	emb	lr	L2	emb	lr	L2
FM	32	1e–3	0	32	1e–3	1e–4
PNN	32	1e–3	0	32	1e–3	1e–4
DeepFM	32	1e–3	0	32	1e–3	1e–4
xDeepFM	32	1e–3	1e–4	32	1e–3	1e–4
AutoInt	32	1e–3	1e–4	16	1e–3	1e–5
InterHat	32	5e–3	1e–4	32	1e–3	5e–5
Our model	32	5e–3	1e–4	16	1e–3	1e–4

Note: emb = dimension of the embedding vectors, lr=learning rate, $L_2 = l_2$ regularization.

Evaluation Metrics. We use AUC (Area Under the ROC Curve) and Logloss (cross entropy) to evaluate the performance of all methods. AUC measures the probability that a randomly selected positive record will be ranked higher than a randomly selected negative record. Higher AUC indicates better performance. Logloss measures how much the difference between the predicted score and the ground truth. Lower Logloss indicates better performance. It is noticeable that a improvement of AUC at **0.001-level** is regarded as significant for CTR prediction [4,7,15,24,26].

Baselines. As described in Sect. 2, we categorize the existing approaches into three types: (A) Tradional models; (B) Deep learning-based models; (C) Self-attention based models. We select the following representative models of the three types to compare with ours.
FM [23] learns a latent vector of each raw feature and models two-order cross features by inner product operation.
PNN [22] uses DNN to model high-order cross features.
DeepFM [7] combines FM with DNN to model low-order cross features and high-order cross features jointly.
xDeepFM [17] utilizes multiple Compressed Interaction Networks to model high-order cross features in an explicit fashion.
AutoInt [24] utilizes multi-head attention mechanism to explicitly model high-order cross features.
InterHat [16] utilizes the hierarchical attention mechanism to explain cross features, which aggregates cross features first and then takes inner product operation to model higher-order cross features at each step.

Implementation Details. All methods are implemented by Pytorch. We adopt early-stopping to avoid overfitting and implement FM, PNN, DeepFM, and xDeepFM following [24]. We use Adam [14] to optimize all methods, and the batch size is 1024 for both Frappe and Avazu. For our method, the $GateNet_1$

Table 3. Results of model comparision, the best results are highlighted.

Model type	Model	Frappe		Avazu	
		AUC	Logloss	AUC	Logloss
Traditional	FM	0.9719	0.1889	0.7356	0.4041
DNN-based	PNN	0.9827	0.1447	0.7440	0.3969
	DeepFM	0.9772	0.1787	0.7414	0.3983
	xDeepFM	0.9808	0.1631	0.7443	0.3980
Self-attention based	AutoInt	0.9810	0.1862	0.7441	0.3974
	InterHat	0.9805	0.1638	0.7417	0.3990
	Our model	**0.9848**	**0.1506**	**0.7468**	**0.3957**

and $GateNet_2$ are two single hidden layer feed-forward neural networks. Besides, the number of heads is 8 for the gated multi-head attention layer. The other hyper-parameters are summarized in Table 2, which are tuned on the validation dataset to obtain the best result of each model.

5.2 Performance Comparison(RQ1)

In this section, we compare our method with six representative baselines. The results on the Frappe and the Avazu datasets are summarized in Table 3, from which we have the following observations:

- All methods of modeling high-order feature interaction are superior to FM that only models second-order cross features, which indicates that high-order cross features are essential for a successful CTR prediction model.
- An interesting observation is that some self-attention based models are inferior to some DNN-based models on the Frappe and Avazu datasets. It indicates that there is no significant performance difference between the DNN-based models and the self-attention based models.
- Our method achieves the best performance on both Frappe and Avazu datasets, which indicates that combining the attention mechanism with the gated mechanism can boost the model performance.

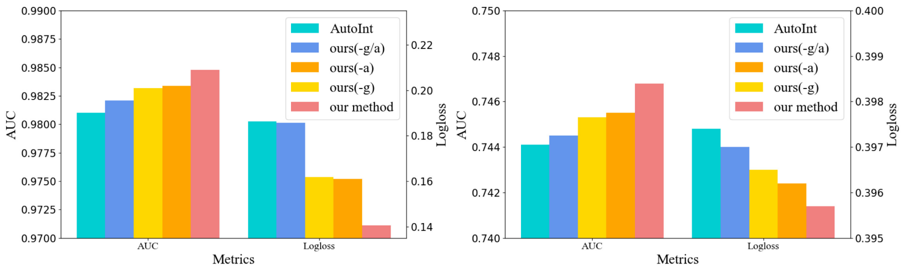
5.3 Ablation Study(RQ2)

In this section, we aim to explore whether the critical components of our method are effective for improving the model performance. Thus, we conduct an ablation study.

- ours(-g/a): on the one hand, the encoder uses a vanilla multi-head key-value attention layer that removes the gate neural network from the gated multi-head attention layer. On the other hand, the encoder removes the gated distillation layer.

- ours(-g): the encoder removes the gated distillation layer but retains the gated multi-head attention layer.
- ours(-a): the encoder uses a vanilla multi-head attention layer and retains the gated distillation layer.

The results of each variant of our method and the baseline are shown in Fig. 2, from which we can obtain the following observations: (1) ours(-g/a) performs better than AutoInt, which indicates that the vanilla attention mechanism is proper than the self-attention mechanism in CTR prediction. (2) ours(-g) outperform ours(-g/a), which indicates that the gated multi-head attention mechanism is more effective than vanilla multi-head attention mechanism. (3) ours(-a) outperform ours(-g/a), which indicates that the gated distillation layer is effective for boosting model performance.



(a) Frappe (b) Avazu

Fig. 2. Results of the ablation study.

5.4 Efficiency Comparison(RQ3)

In real-world scenarios, efficiency is an important factor that affects whether a model could be used. Thus, we record the training cost (running time per epoch) as the criterion of measuring the efficiency of methods.

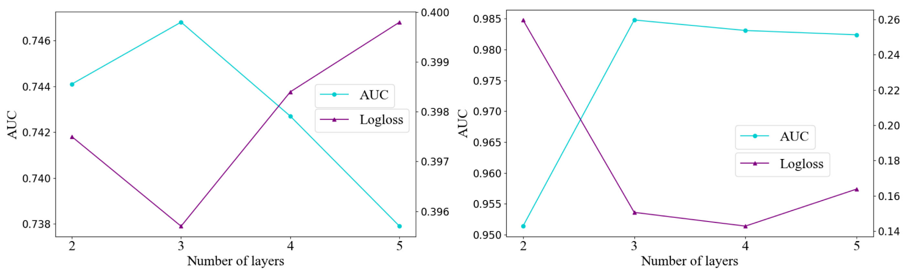
For a fair comparison, all the models are trained in the same machine with a TITAN Xp GPU. Table 4 shows the training cost of attention-based methods. As can be seen, our method is significantly more efficient than self-attention based methods, which proves that self-attention mechanism may be the bottleneck of the method.

5.5 Hyper-Parameter Study(RQ4)

In this section, we study the impact of hyper-parameters on our proposed method, including the number of encoders and the dimension of embedding vectors.

Table 4. Efficiency comparison on two datasets.

Method	Dataset	Training cost(s)
AutoInt	Frappe	70.69
	Avazu	5985.26
InterHat	Frappe	60.89
	Avazu	5564.63
Ours	Frappe	44.51
	Avazu	4918.48

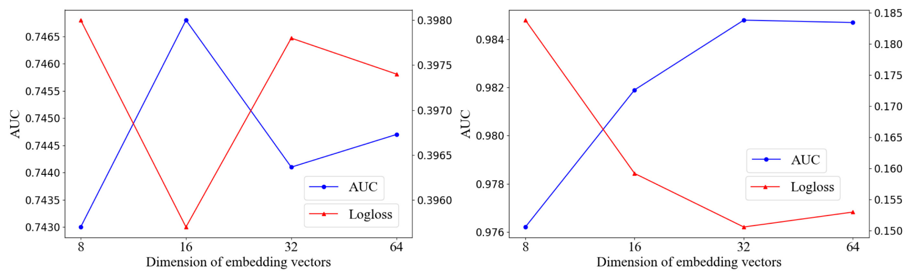


(a) Avazu (b) Frappe

Fig. 3. The impact of number of encoders on our method with respect to AUC and Logloss.

Number of Modules. Figure 3 shows the impact of the number of encoders. We observe that the model performance shows an increasing trend, followed by a decreasing trend when the number of encoders is larger than 3. Because the number of encoders determines the maximal order of feature interactions that our method can learn, it indicates that 4-order feature interactions are good enough to predict the final result. Besides, an interesting observation is that for both the Frappe dataset and the Avazu dataset, our method thinks that 4-order feature interactions are enough to obtain a good prediction.

Dimension of Embedding Vectors. As shown in Fig. 4, the performance of our method first grows with the dimension of embedding vectors, which is attributed to the better learning ability of our method. The performance starts to degrade when the dimension is larger than 32 for the Frappe dataset and 16 for the Avazu dataset, which is caused by the overfitting.



(a) Avazu (b) Frappe

Fig. 4. Performance of our method affected by the dimension of embedding vectors.

6 Conclusions

In this paper, we point out that the self-attention mechanism is less efficient to learn feature interactions and propose a novel Transformer-like method. The method combines the key-value attention mechanism with the gated mechanism, which makes it more efficient and effective to learn feature interactions.

For future works, we are interested in proposing new technologies to alleviate the effect of irrelevant cross features. For example, a hot direction is automatic feature selection.

Acknowledgements. This research is supported in part by the 2030 National Key AI Program of China (2018AAA0100503), Shanghai Municipal Science and Technology Commission (No. 19510760500, and No. 19511101500), National Science Foundation of China (No. 62072304, No. 61772341), and Zhejiang Aoxin Co. Ltd.

References

1. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: ICLR (2015)
2. Baltrunas, L., Church, K., Karatzoglou, A., Oliver, N.: Frappe: understanding the usage and perception of mobile app recommendations in-the-wild (2015). CoRR abs/1505.03014
3. Blondel, M., Fujino, A., Ueda, N., Ishihata, M.: Higher-order factorization machines. In: NIPS, pp. 3351–3359 (2016)
4. Cheng, H., et al.: Wide & deep learning for recommender systems. In: RecSys, pp. 7–10 (2016)
5. Chung, J., Gülçehre, Ç., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling (2014). CoRR abs/1412.3555
6. Cui, H., Iida, S., Hung, P., Utsuro, T., Nagata, M.: Mixed multi-head self-attention for neural machine translation. In: EMNLP-IJCNLP, pp. 206–214 (2019)
7. Guo, H., Tang, R., Ye, Y., Li, Z., He, X.: Deepfm: a factorization-machine based neural network for CTR prediction. In: IJCAI, pp. 1725–1731 (2017)
8. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR, pp. 770–778 (2016)

9. He, X., Chua, T.: Neural factorization machines for sparse predictive analytics. In: SIGIR, pp. 355–364 (2017)
10. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
11. Hong, F., Huang, D., Chen, G.: Interaction-aware factorization machines for recommender systems. In: AAAI, pp. 3804–3811 (2019)
12. Jiao, L., Yu, Y., Zhou, N., Zhang, L., Yin, H.: Neural pairwise ranking factorization machine for item recommendation. In: Nah, Y., Cui, B., Lee, S.-W., Yu, J.X., Moon, Y.-S., Whang, S.E. (eds.) DASFAA 2020. LNCS, vol. 12112, pp. 680–688. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-59410-7_46
13. Juan, Y., Zhuang, Y., Chin, W., Lin, C.: Field-aware factorization machines for CTR prediction. In: RecSys, pp. 43–50 (2016)
14. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: ICLR (2015)
15. Li, Z., Cui, Z., Wu, S., Zhang, X., Wang, L.: Fi-gnn: modeling feature interactions via graph neural networks for CTR prediction. In: CIKM, pp. 539–548 (2019)
16. Li, Z., Cheng, W., Chen, Y., Chen, H., Wang, W.: Interpretable click-through rate prediction through hierarchical attention. In: WSDM, pp. 313–321 (2020)
17. Lian, J., Zhou, X., Zhang, F., Chen, Z., Xie, X., Sun, G.: xdeepfm: combining explicit and implicit feature interactions for recommender systems. In: KDD, pp. 1754–1763 (2018)
18. Liu, B., et al.: Autogroup: automatic feature grouping for modelling explicit high-order feature interactions in CTR prediction. In: SIGIR, pp. 199–208 (2020)
19. Liu, H., Zhu, Y., Xu, Y.: Learning from heterogeneous student behaviors for multiple prediction tasks. In: Nah, Y., Cui, B., Lee, S.-W., Yu, J.X., Moon, Y.-S., Whang, S.E. (eds.) DASFAA 2020. LNCS, vol. 12113, pp. 297–313. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-59416-9_18
20. Luo, X., Sha, C., Tan, Z., Niu, J.: Multi-head attentive social recommendation. In: Cheng, R., Mamoulis, N., Sun, Y., Huang, X. (eds.) WISE 2020. LNCS, vol. 11881, pp. 243–258. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-34223-4_16
21. Pan, J., et al.: Field-weighted factorization machines for click-through rate prediction in display advertising. In: WWW, pp. 1349–1357 (2018)
22. Qu, Y., et al.: Product-based neural networks for user response prediction. In: ICDM, pp. 1149–1154 (2016)
23. Rendle, S.: Factorization machines. In: ICDM, pp. 995–1000 (2010)
24. Song, W., et al.: Autoint: automatic feature interaction learning via self-attentive neural networks. In: CIKM, pp. 1161–1170 (2019)
25. Vaswani, A., et al.: Attention is all you need. In: NIPS, pp. 5998–6008 (2017)
26. Wang, R., Fu, B., Fu, G., Wang, M.: Deep & cross network for ad click predictions. In: KDD, pp. 12:1–12:7 (2017)
27. Wu, C., Wu, F., Ge, S., Qi, T., Huang, Y., Xie, X.: Neural news recommendation with multi-head self-attention. In: EMNLP-IJCNLP, pp. 6388–6393 (2019)
28. Xiao, J., Ye, H., He, X., Zhang, H., Wu, F., Chua, T.: Attentional factorization machines: learning the weight of feature interactions via attention networks. In: IJCAI, pp. 3119–3125 (2017)
29. Zhang, J., Shi, X., Xie, J., Ma, H., King, I., Yeung, D.: Gaan: gated attention networks for learning on large and spatiotemporal graphs. In: UAI 2018, Monterey, California, USA, 6–10 August 2018, pp. 339–349. AUAI Press (2018)
30. Zhang, J., Zheng, Y., Qi, D.: Deep spatio-temporal residual networks for citywide crowd flows prediction. In: Thirty-first AAAI conference on artificial intelligence (2017)