MaskNet: Introducing Feature-Wise Multiplication to CTR Ranking Models by Instance-Guided Mask

Zhiqiang Wang, Qingyun She, Junlin Zhang
Sina Weibo Corp
Beijing, China
roky2813@sina.com,qingyun_she@163.com,junlin6@staff.weibo.com

ABSTRACT

Click-Through Rate(CTR) estimation has become one of the most fundamental tasks in many real-world applications and it's important for ranking models to effectively capture complex high-order features. Shallow feed-forward network is widely used in many state-of-the-art DNN models such as FNN, DeepFM and xDeepFM to implicitly capture high-order feature interactions. However, some research has proved that addictive feature interaction, particular feed-forward neural networks, is inefficient in capturing common feature interaction. To resolve this problem, we introduce specific multiplicative operation into DNN ranking system by proposing instance-guided mask which performs element-wise product both on the feature embedding and feed-forward layers guided by input instance. We also turn the feed-forward layer in DNN model into a mixture of addictive and multiplicative feature interactions by proposing MaskBlock in this paper. MaskBlock combines the layer normalization, instance-guided mask, and feed-forward layer and it is a basic building block to be used to design new ranking model under various configurations. The model consisting of MaskBlock is called MaskNet in this paper and two new MaskNet models are proposed to show the effectiveness of MaskBlock as basic building block for composing high performance ranking systems. The experiment results on three real-world datasets demonstrate that our proposed MaskNet models outperform state-of-the-art models such as DeepFM and xDeepFM significantly, which implies MaskBlock is an effective basic building unit for composing new high performance ranking systems.

ACM Reference Format:
Zhiqiang Wang, Qingyun She, Junlin Zhang. 2021. MaskNet: Introducing Feature-Wise Multiplication to CTR Ranking Models by Instance-Guided Mask. In *Proceedings of DLP-KDD 2021.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Click-through rate (CTR) prediction is to predict the probability of a user clicking on the recommended items. It plays important role in personalized advertising and recommender systems. Many models have been proposed to resolve this problem such as Logistic Regression (LR) [16], Polynomial-2 (Poly2) [17], tree-based models [7], tensor-based models [12], Bayesian models [5], and Field-aware Factorization Machines (FFMs) [11]. In recent years, employing DNNs for CTR estimation has also been a research trend in this field and some deep learning based models have been introduced such as Factorization-Machine Supported Neural Networks(FNN)[24], Attentional Factorization Machine (AFM)[3], wide & deep(W&D)[22], DeepFM[6], xDeepFM[13] etc.

Feature interaction is critical for CTR tasks and it's important for ranking model to effectively capture these complex features. Most DNN ranking models such as FNN , W&D, DeepFM and xDeepFM use the shallow MLP layers to model high-order interactions in an implicit way and it's an important component in current state-of-the-art ranking systems.

However, Alex Beutel et.al [2] have proved that addictive feature interaction, particular feed-forward neural networks, is inefficient in capturing common feature crosses. They proposed a simple but effective approach named "latent cross" which is a kind of multiplicative interactions between the context embedding and the neural network hidden states in RNN model. Recently, Rendle et.al's work [18] also shows that a carefully configured dot product baseline largely outperforms the MLP layer in collaborative filtering. While a MLP can in theory approximate any function, they show that it is non-trivial to learn a dot product with an MLP and learning a dot product with high accuracy for a decently large embedding dimension requires a large model capacity as well as many training data. Their work also proves the inefficiency of MLP layer's ability to model complex feature interactions.

Inspired by "latent cross"[2] and Rendle's work [18], we care about the following question: Can we improve the DNN ranking systems by introducing specific multiplicative operation into it to make it efficiently capture complex feature interactions?

In order to overcome the problem of inefficiency of feed-forward layer to capture complex feature cross, we introduce a special kind of multiplicative operation into DNN ranking system in this paper. First, we propose an instance-guided mask performing element-wise production on the feature embedding and feed-forward layer. The instance-guided mask utilizes the global information collected from input instance to dynamically highlight the informative elements in feature embedding and hidden layer in a unified manner. There are two main advantages for adopting the instance-guided mask: firstly, the element-wise product between the mask and hidden layer or feature embedding layer brings multiplicative operation into DNN ranking system in unified way to more efficiently capture complex feature interaction. Secondly, it's a kind of fine-gained bit-wise attention guided by input instance which can both

DLP-KDD 2021, August 15, 2021, Singapore

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

weaken the influence of noise in feature embedding and MLP layers while highlight the informative signals in DNN ranking systems.

By combining instance-guided mask, a following feed-forward layer and layer normalization, MaskBlock is proposed by us to turn the commonly used feed-forward layer into a mixture of addictive and multiplicative feature interactions. The instance-guided mask introduces multiplicative interactions and the following feed-forward hidden layer aggregates the masked information in order to better capture the important feature interactions. While the layer normalization can ease optimization of the network.

MaskBlock can be regarded as a basic building block to design new ranking models under some kinds of configuration. The model consisting of MaskBlock is called MaskNet in this paper and two new MaskNet models are proposed to show the effectiveness of MaskBlock as basic building block for composing high performance ranking systems.

The contributions of our work are summarized as follows:

(1) In this work, we propose an instance-guided mask performing element-wise product both on the feature embedding and feed-forward layers in DNN models. The global context information contained in the instance-guided mask is dynamically incorporated into the feature embedding and feed-forward layer to highlight the important elements.
(2) We propose a basic building block named MaskBlock which consists of three key components: instance-guided mask, a following feed-forward hidden layer and layer normalization module. In this way, we turn the widely used feed-forward layer of a standard DNN model into a mixture of addictive and multiplicative feature interactions.
(3) We also propose a new ranking framework named MaskNet to compose new ranking system by utilizing MaskBlock as basic building unit. To be more specific, the serial MaskNet model and parallel MaskNet model are designed based on the MaskBlock in this paper. The serial rank model stacks MaskBlock block by block while the parallel rank model puts many MaskBlocks in parallel on a sharing feature embedding layer.
(4) Extensive experiments are conduct on three real-world datasets and the experiment results demonstrate that our proposed two MaskNet models outperform state-of-the-art models significantly. The results imply MaskBlock indeed enhance DNN model's ability of capturing complex feature interactions through introducing multiplicative operation into DNN models by instance-guided mask.

The rest of this paper is organized as follows. Section 2 introduces some related works which are relevant with our proposed model. We introduce our proposed models in detail in Section 3. The experimental results on three real world datasets are presented and discussed in Section 4. Section 5 concludes our work in this paper.

2 RELATED WORK

2.1 State-Of-The-Art CTR Models

Many deep learning based CTR models have been proposed in recent years and it is the key factor for most of these neural network based models to effectively model the feature interactions.

Factorization-Machine Supported Neural Networks (FNN)[24] is a feed-forward neural network using FM to pre-train the embedding layer. Wide & Deep Learning[22] jointly trains wide linear models and deep neural networks to combine the benefits of memorization and generalization for recommender systems. However, expertise feature engineering is still needed on the input to the wide part of Wide & Deep model. To alleviate manual efforts in feature engineering, DeepFM[6] replaces the wide part of Wide & Deep model with FM and shares the feature embedding between the FM and deep component.

While most DNN ranking models process high-order feature interactions by MLP layers in implicit way, some works explicitly introduce high-order feature interactions by sub-network. Deep & Cross Network (DCN)[21] efficiently captures feature interactions of bounded degrees in an explicit fashion. Similarly, eXtreme Deep Factorization Machine (xDeepFM) [13] also models the low-order and high-order feature interactions in an explicit way by proposing a novel Compressed Interaction Network (CIN) part. AutoInt[19] uses a multi-head self-attentive neural network to explicitly model the feature interactions in the low-dimensional space. FiBiNET[9] can dynamically learn feature importance via the Squeeze-Excitation network (SENET) mechanism and feature interactions via bilinear function.

2.2 Feature-Wise Mask Or Gating

Feature-wise mask or gating has been explored widely in vision [8, 20], natural language processing [4] and recommendation system[14, 15]. For examples, Highway Networks [20] utilize feature gating to ease gradient-based training of very deep networks. Squeeze-and-Excitation Networks[8] recalibrate feature responses by explicitly multiplying each channel with learned sigmoidal mask values. Dauphin et al.[4] proposed gated linear unit (GLU) to utilize it to control what information should be propagated for predicting the next word in the language modeling task. Gating or mask mechanisms are also adopted in recommendation systems. Ma et al. [15] propose a novel multi-task learning approach, Multi-gate Mixture-of-Experts (MMoE), which explicitly learns to model task relationships from data. Ma et al.[14] propose a hierarchical gating network (HGN) to capture both the long-term and short-term user interests. The feature gating and instance gating modules in HGN select what item features can be passed to the downstream layers from the feature and instance levels, respectively.

2.3 Normalization

Normalization techniques have been recognized as very effective components in deep learning. Many normalization approaches have been proposed with the two most popular ones being BatchNorm [10] and LayerNorm [1] . Batch Normalization (Batch Norm or BN)[10] normalizes the features by the mean and variance computed within a mini-batch. Another example is layer normalization (Layer Norm or LN)[1] which was proposed to ease optimization of recurrent neural networks. Statistics of layer normalization are not computed across the N samples in a mini-batch but are estimated in a layer-wise manner for each sample independently. Normalization methods have shown success in accelerating the training of deep networks.

3 OUR PROPOSED MODEL

In this section, we first describe the feature embedding layer. Then the details of the instance-guided mask, MaskBlock and MaskNet structure we proposed will be introduced. Finally the log loss as a loss function is introduced.

3.1 Embedding Layer

The input data of CTR tasks usually consists of sparse and dense features and the sparse features are mostly categorical type. Such features are encoded as one-hot vectors which often lead to excessively high-dimensional feature spaces for large vocabularies. The common solution to this problem is to introduce the embedding layer. Generally, the sparse input can be formulated as:

$$x = [x_1, x_2, ..., x_f] \tag{1}$$

where f denotes the number of fields, and $x_i \in \mathbb{R}^n$ denotes a one-hot vector for a categorical field with n features and $x_i \in \mathbb{R}^n$ is vector with only one value for a numerical field. We can obtain feature embedding e_i for one-hot vector x_i via:

$$e_i = W_e x_i \tag{2}$$

where $W_e \in \mathbb{R}^{k \times n}$ is the embedding matrix of n features and k is the dimension of field embedding. The numerical feature x_j can also be converted into the same low-dimensional space by:

$$e_j = V_j x_j \tag{3}$$

where $V_j \in \mathbb{R}^k$ is the corresponding field embedding with size k.

Through the aforementioned method, an embedding layer is applied upon the raw feature input to compress it to a low dimensional, dense real-value vector. The result of embedding layer is a wide concatenated vector:

$$\mathrm{V}_{emb} = concat(\mathrm{e}_1, \mathrm{e}_2, ..., \mathrm{e}_i, ..., \mathrm{e}_f) \tag{4}$$

where f denotes the number of fields, and $\mathrm{e}_i \in \mathbb{R}^k$ denotes the embedding of one field. Although the feature lengths of input instances can be various, their embedding are of the same length $f \times k$, where k is the dimension of field embedding.

We use instance-guided mask to introduce the multiplicative operation into DNN ranking system and here the so-called "instance" means the feature embedding layer of current input instance in the following part of this paper.

3.2 Instance-Guided Mask

We utilize the global information collected from input instance by instance-guided mask to dynamically highlight the informative elements in feature embedding and feed-forward layer. For feature embedding, the mask lays stress on the key elements with more information to effectively represent this feature. For the neurons in hidden layer, the mask helps those important feature interactions to stand out by considering the contextual information in the input instance. In addition to this advantage, the instance-guided mask also introduces the multiplicative operation into DNN ranking system to capture complex feature cross more efficiently.

As depicted in Figure 1, two fully connected (FC) layers with identity function are used in instance-guided mask. Notice that the input of instance-guided mask is always from the input instance, that is to say, the feature embedding layer.

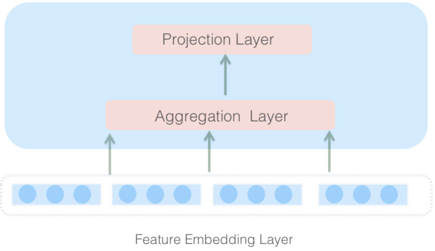

Figure 1: Neural Structure of Instance-Guided Mask

The first FC layer is called "aggregation layer" and it is a relatively wider layer compared with the second FC layer in order to better collect the global contextual information in input instance. The aggregation layer has parameters W_{d1} and here d denotes the d-th mask. For feature embedding and different MLP layers, we adopt different instance-guided mask owning its parameters to learn to capture various information for each layer from input instance.

The second FC layer named "projection layer" reduces dimensionality to the same size as feature embedding layer V_{emb} or hidden layer V_{hidden} with parameters W_{d2}, Formally,

$$V_{mask} = W_{d2}(Relu(W_{d1}V_{emb} + \beta_{d1})) + \beta_{d2} \tag{5}$$

where $V_{emb} \in \mathbb{R}^{m=f \times k}$ refers to the embedding layer of input instance, $W_{d1} \in \mathbb{R}^{t \times m}$ and $W_{d2} \in \mathbb{R}^{z \times t}$ are parameters for instance-guided mask, t and z respectively denotes the neural number of aggregation layer and projection layer, f denotes the number of fields and k is the dimension of field embedding. $\beta_{d1} \in \mathbb{R}^{t \times m}$ and $\beta_{d2} \in \mathbb{R}^{z \times t}$ are learned bias of the two FC layers. Notice here that the aggregation layer is usually wider than the projection layer because the size of the projection layer is required to be equal to the size of feature embedding layer or MLP layer. So we define the size $r = t/z$ as reduction ratio which is a hyper-parameter to control the ratio of neuron numbers of two layers.

Element-wise product is used in this work to incorporate the global contextual information aggregated by instance-guided mask into feature embedding or hidden layer as following:

$$\begin{aligned} \mathrm{V}_{maskedEMB} &= \mathrm{V}_{mask} \odot \mathrm{V}_{emb} \\ \mathrm{V}_{maskedHID} &= \mathrm{V}_{mask} \odot \mathrm{V}_{hidden} \end{aligned} \tag{6}$$

where V_{emb} denotes embedding layer and V_{hidden} means the feed-forward layer in DNN model, $\odot$ means the element-wise production between two vectors as follows:

$$V_i \odot V_j = [V_{i1} \cdot V_{j1}, V_{i2} \cdot V_{j2}, ..., V_{iu} \cdot V_{ju}] \tag{7}$$

here u is the size of vector V_i and V_j

The instance-guided mask can be regarded as a special kind of bitwise attention or gating mechanism which uses the global context information contained in input instance to guide the parameter optimization during training. The bigger value in V_{mask} implies that the model dynamically identifies an important element in feature embedding or hidden layer. It is used to boost the element in vector V_{emb} or V_{hidden}. On the contrary, small value in V_{mask} will suppress the uninformative elements or even noise by decreasing the values in the corresponding vector V_{emb} or V_{hidden}.

The two main advantages in adopting the instance-guided mask are: firstly, the element-wise product between the mask and hidden layer or feature embedding layer brings multiplicative operation into DNN ranking system in unified way to more efficiently capture complex feature interaction. Secondly, this kind of fine-gained bit-wise attention guided by input instance can both weaken the influence of noise in feature embedding and MLP layers while highlight the informative signals in DNN ranking systems.

3.3 MaskBlock

To overcome the problem of the inefficiency of feed-forward layer to capture complex feature interaction in DNN models, we propose a basic building block named MaskBlock for DNN ranking systems in this work, as shown in Figure 2 and Figure 3. The proposed MaskBlock consists of three key components: layer normalization module ,instance-guided mask, and a feed-forward hidden layer. The layer normalization can ease optimization of the network. The instance-guided mask introduces multiplicative interactions for feed-forward layer of a standard DNN model and feed-forward hidden layer aggregate the masked information in order to better capture the important feature interactions. In this way, we turn the widely used feed-forward layer of a standard DNN model into a mixture of addictive and multiplicative feature interactions.

First, we briefly review the formulation of LayerNorm.

Layer Normalization:

In general, normalization aims to ensure that signals have zero mean and unit variance as they propagate through a network to reduce "covariate shift" [10]. As an example, layer normalization (Layer Norm or LN)[1] was proposed to ease optimization of recurrent neural networks. Specifically, let $x = (x_1, x_2, ..., x_H)$ denotes the vector representation of an input of size H to normalization layers. LayerNorm re-centers and re-scales input x as

$$\begin{aligned} &\mathrm{h} = \mathrm{g} \odot N(\mathrm{x}) + \mathrm{b}, \quad N(\mathrm{x}) = \frac{\mathrm{x} - \mu}{\delta}, \\ &\mu = \frac{1}{H}\sum_{i=1}^{H} x_i, \quad \delta = \sqrt{\frac{1}{H}\sum_{i=1}^{H}(x_i - \mu)^2} \end{aligned} \tag{8}$$

where h is the output of a LayerNorm layer. $\odot$ is an element-wise production operation. μ and δ are the mean and standard deviation of input. Bias b and gain g are parameters with the same dimension H.

As one of the key component in MaskBlock, layer normalization can be used on both feature embedding and feed- forward layer. For the feature embedding layer, we regard each feature's embedding as a layer to compute the mean, standard deviation, bias and gain of LN as follows:

$$LN\_EMB(\mathrm{V}_{emb}) = concate\left(LN(\mathrm{e}_1), LN(\mathrm{e}_2), ..., LN(\mathrm{e}_i), ..., LN(\mathrm{e}_f)\right) \tag{9}$$

As for the feed-forward layer in DNN model, the statistics of LN are estimated among neurons contained in the corresponding hidden layer as follows:

$$LN\_HID(\mathrm{V}_{hidden}) = ReLU(LN(\mathrm{W}_i\mathrm{X})) \tag{10}$$

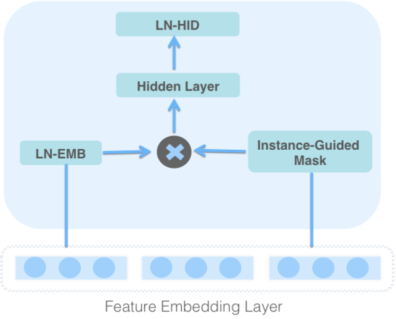

Figure 2: MaskBlock on Feature Embedding

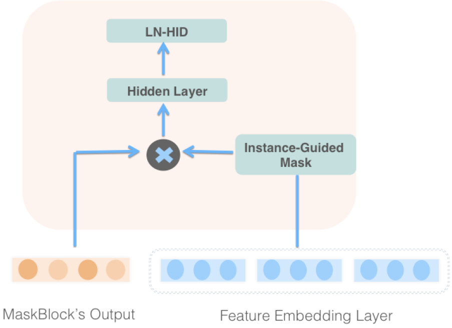

Figure 3: MaskBlock on MaskBlock

where $\mathrm{X} \in \mathbb{R}^t$ refers to the input of feed-forward layer, $\mathrm{W}_i \in \mathbb{R}^{m\times t}$ are parameters for the layer, t and m respectively denotes the size of input layer and neural number of feed-forward layer. Notice that we have two places to put normalization operation on the MLP: one place is before non-linear operation and another place is after non-linear operation. We find the performance of the normalization before non-linear consistently outperforms that of the normalization after non-linear operation. So all the normalization used in MLP part is put before non-linear operation in our paper as formula (4) shows.

MaskBlock on Feature Embedding:

We propose MaskBlock by combining the three key elements: layer normalization, instance-guided mask and a following feed-forward layer. MaskBlock can be stacked to form deeper network. According to the different input of each MaskBlock, we have two kinds of MaskBlocks: MaskBlock on feature embedding and MaskBlock on Maskblock. We will firstly introduce the MaskBlock on feature embedding as depicted in Figure 2 in this subsection.

The feature embedding V_{emb} is the only input for MaskBlock on feature embedding. After the layer normalization operation on embedding V_{emb}. MaskBlock utilizes instance-guided mask to highlight the informative elements in V_{emb} by element-wise product, Formally,

$$\mathrm{V}_{maskedEMB} = \mathrm{V}_{mask} \odot LN\_EMB(\mathrm{V}_{emb}) \tag{11}$$

where $\odot$ means an element-wise production between the instance-guided mask and the normalized vector $LN_EMB(\mathrm{V}_{emb})$, $\mathrm{V}_{maskedEMB}$ denote the masked feature embedding. Notice that the input of instance-guided mask V_{mask} is also the feature embedding V_{emb}.

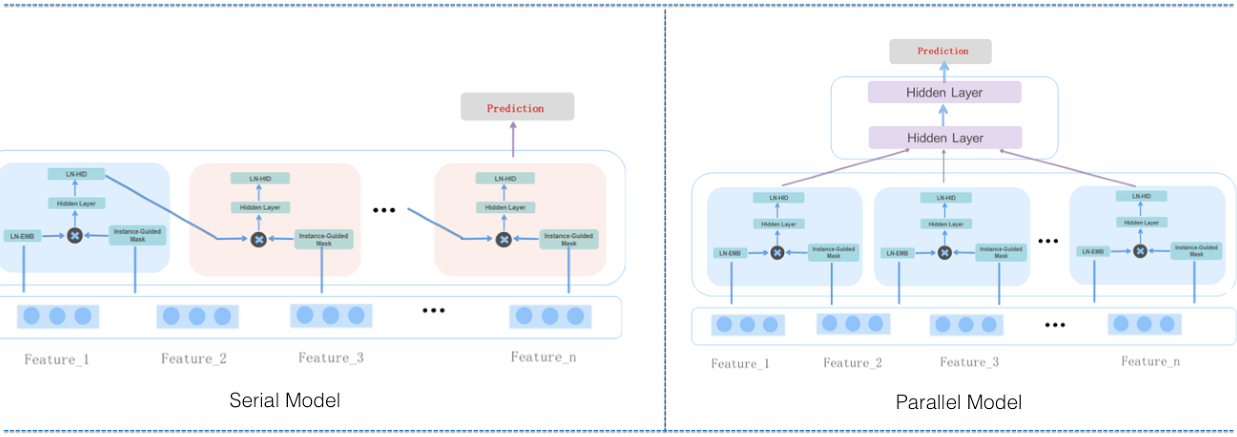

Figure 4: Structure of Serial Model and Parallel Model

We introduce a feed-forward hidden layer and a following layer normalization operation in MaskBlock to better aggregate the masked information by a normalized non-linear transformation. The output of MaskBlock can be calculated as follows:

$$\begin{aligned} \mathrm{V}_{output} &= LN\_HID(W_i V_{maskdEMB}) \\ &= ReLU(LN(\mathrm{W}_i(\mathrm{V}_{mask} \odot LN\_EMB(\mathrm{V}_{emb})))) \end{aligned} \tag{12}$$

where $\mathrm{W}_i \in \mathbb{R}^{q \times n}$ are parameters of the feed-forward layer in the i-th MaskBlock, n denotes the size of $\mathrm{V}_{maskedEMB}$ and q means the size of neural number of the feed-forward layer.

The instance-guided mask introduces the element-wise product into feature embedding as a fine-grained attention while normalization both on feature embedding and hidden layer eases the network optimization. These key components in MaskBlock help the feed-forward layer capture complex feature cross more efficiently.

MaskBlock on MaskBlock:

In this subsection, we will introduce MaskBlock on MaskBlock as depicted in Figure 3. There are two different inputs for this MaskBlock: feature embedding V_{emb} and the output V^p_{output} of the previous MaskBlock. The input of instance-guided mask for this kind of MaskBlock is always the feature embedding V_{emb}. MaskBlock utilizes instance-guided mask to highlight the important feature interactions in previous MaskBlock's output V^p_{output} by element-wise product, Formally,

$$V_{maskedHID} = V_{mask} \odot V^p_{output} \tag{13}$$

where $\odot$ means an element-wise production between the instance-guided mask V_{mask} and the previous MaskBlock's output V^p_{output}, $V_{maskedHID}$ denote the masked hidden layer.

In order to better capture the important feature interactions, another feed-forward hidden layer and a following layer normalization are introduced in MaskBlock . In this way, we turn the widely used feed-forward layer of a standard DNN model into a mixture of addictive and multiplicative feature interactions to avoid the ineffectiveness of those addictive feature cross models. The output of MaskBlock can be calculated as follows:

$$\begin{aligned} \mathrm{V}_{output} &= LN\_HID(W_i V_{maskdHID}) \\ &= ReLU(LN(\mathrm{W}_i(\mathrm{V}_{mask} \odot \mathrm{V}^p_{output}))) \end{aligned} \tag{14}$$

where $W_i \in \mathbb{R}^{q \times n}$ are parameters of the feed-forward layer in the i-th MaskBlock, n denotes the size of $\mathrm{V}_{maskedHID}$ and q means the size of neural number of the feed-forward layer.

3.4 MaskNet

Based on the MaskBlock, various new ranking models can be designed according to different configurations. The rank model consisting of MaskBlock is called MaskNet in this work. We also propose two MaskNet models by utilizing the MaskBlock as the basic building block.

Serial MaskNet:

We can stack one MaskBlock after another to build the ranking system , as shown by the left model in Figure 4. The first block is a MaskBlock on feature embedding and all other blocks are MaskBlock on Maskblock to form a deeper network. The prediction layer is put on the final MaskBlock's output vector. We call MaskNet under this serial configuration as SerMaskNet in our paper. All inputs of instance-guided mask in every MaskBlock come from the feature embedding layer V_{emb} and this makes the serial MaskNet model look like a RNN model with sharing input at each time step.

Parallel MaskNet:

We propose another MaskNet by placing several MaskBlocks on feature embedding in parallel on a sharing feature embedding layer, as depicted by the right model in Figure 4. The input of each block is only the shared feature embedding V_{emb} under this configuration. We can regard this ranking model as a mixture of multiple experts just as MMoE[15] does. Each MaskBlock pays attention to specific kind of important features or feature interactions. We collect the information of each expert by concatenating the output of each MaskBlock as follows:

$$\mathrm{V}_{merge} = concate(\mathrm{V}^1_{output}, \mathrm{V}^2_{output}, ..., \mathrm{V}^i_{output}, ..., \mathrm{V}^u_{output}) \tag{15}$$

where $\mathrm{V}^i_{output} \in \mathbb{R}^q$ is the output of the i-th MaskBlock and q means size of neural number of feed-forward layer in MaskBlock, u is the MaskBlock number.

To further merge the feature interactions captured by each expert, multiple feed-forward layers are stacked on the concatenation information V_{merge}. Let $\mathrm{H}_0 = \mathrm{V}_{merge}$ denotes the output of the concatenation layer, then H_0 is fed into the deep neural network and the feed forward process is:

$$\mathrm{H}_l = ReLU(\mathrm{W}_l \mathrm{H}_{l-1} + \beta_l) \tag{16}$$

where l is the depth and ReLU is the activation function. $\mathrm{W}_t, \beta_t, \mathrm{H}_l$ are the model weight, bias and output of the l-th layer. The prediction layer is put on the last layer of multiple feed-forward networks. We call this version MaskNet as "ParaMaskNet" in the following part of this paper.

3.5 Prediction Layer

To summarize, we give the overall formulation of our proposed model' s output as:

$$\hat{y} = \delta(w_0 + \sum_{i=1}^{n} w_i x_i) \tag{17}$$

where $\hat{y} \in (0, 1)$ is the predicted value of CTR, δ is the sigmoid function, n is the size of the last MaskBlock's output(SerMaskNet) or feed-forward layer(ParaMaskNet), x_i is the bit value of feed-forward layer and w_i is the learned weight for each bit value.

For binary classifications, the loss function is the log loss:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i) \tag{18}$$

where N is the total number of training instances, y_i is the ground truth of i-th instance and $\hat{y}_i$ is the predicted CTR. The optimization process is to minimize the following objective function:

$$\mathfrak{L} = \mathcal{L} + \lambda \|\Theta\| \tag{19}$$

where λ denotes the regularization term and Θ denotes the set of parameters, including those in feature embedding matrix, instance-guided mask matrix, feed-forward layer in MaskBlock, and prediction part.

4 EXPERIMENTAL RESULTS

In this section, we evaluate the proposed approaches on three real-world datasets and conduct detailed ablation studies to answer the following research questions:

- **RQ1** Does the proposed MaskNet model based on the MaskBlock perform better than existing state-of-the-art deep learning based CTR models?
- **RQ2** What are the influences of various components in the MaskBlock architecture? Is each component necessary to build an effective ranking system?
- **RQ3** How does the hyper-parameter of networks influence the performance of our proposed two MaskNet models?
- **RQ4** Does instance-guided mask highlight the important elements in feature embedding and feed-forward layers according to the input instance?

In the following, we will first describe the experimental settings, followed by answering the above research questions.

4.1 Experiment Setup

4.1.1 Datasets. The following three data sets are used in our experiments:

(1) **Criteo**[1] **Dataset:** As a very famous public real world display ad dataset with each ad display information and corresponding user click feedback, Criteo data set is widely used in many CTR model evaluation. There are 26 anonymous categorical fields and 13 continuous feature fields in Criteo data set.
(2) **Malware**[2] **Dataset:** Malware is a dataset from Kaggle competitions published in the Microsoft Malware prediction. The goal of this competition is to predict a Windows machine's probability of getting infected. The malware prediction task can be formulated as a binary classification problem like a typical CTR estimation task does.
(3) **Avazu**[3] **Dataset:** The Avazu dataset consists of several days of ad click- through data which is ordered chronologically. For each click data, there are 23 fields which indicate elements of a single ad impression.

We randomly split instances by 8 : 1 : 1 for training , validation and test while Table 1 lists the statistics of the evaluation datasets.

Table 1: Statistics of the evaluation datasets

Datasets	#Instances	#fields	#features
Criteo	45M	39	30M
Avazu	40.43M	23	9.5M
Malware	8.92M	82	0.97M

4.1.2 Evaluation Metrics. AUC (Area Under ROC) is used in our experiments as the evaluation metric. AUC's upper bound is 1 and larger value indicates a better performance.

RelaImp is also as work [23] does to measure the relative AUC improvements over the corresponding baseline model as another evaluation metric. Since AUC is 0.5 from a random strategy, we can remove the constant part of the AUC score and formalize the RelaImp as:

$$RelaImp = \frac{AUC(Measured\ Model) - 0.5}{AUC(Base\ Model) - 0.5} - 1 \tag{20}$$

4.1.3 Models for Comparisons. We compare the performance of the following CTR estimation models with our proposed approaches: FM, DNN, DeepFM, Deep&Cross Network(DCN), xDeepFM and AutoInt Model, all of which are discussed in Section 2. FM is considered as the base model in evaluation.

4.1.4 Implementation Details. We implement all the models with Tensorflow in our experiments. For optimization method, we use the Adam with a mini-batch size of 1024 and a learning rate is set to 0.0001. Focusing on neural networks structures in our paper, we make the dimension of field embedding for all models to be a fixed value of 10. For models with DNN part, the depth of hidden layers is set to 3, the number of neurons per layer is 400, all activation function is ReLU. For default settings in MaskBlock, the reduction ratio of instance-guided mask is set to 2. We conduct our experiments with 2 Tesla $K40$ GPUs.

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

Table 2: Overall performance (AUC) of different models on three datasets(feature embedding size=10,our proposed two models both have 3 MaskBlocks with same default settings.)

	Criteo		**Malware**		**Avazu**	
	AUC	RelaImp	AUC	RelaImp	AUC	RelaImp
FM	0.7895	0.00%	0.7166	0.00%	0.7785	0.00%
DNN	0.8054	+5.35%	0.7246	+3.70%	0.7820	+1.26%
DeepFM	0.8057	+5.46%	0.7293	+5.86%	0.7833	+1.72%
DCN	0.8058	+5.49%	0.7300	+6.19%	0.7830	+1.62%
xDeepFM	0.8064	+5.70%	0.7310	+6.65%	0.7841	+2.01%
AutoInt	0.8051	+5.39%	0.7282	+5.36%	0.7824	+1.40%
SerMaskNet	0.8119	+7.74%	**0.7413**	**+11.40%**	**0.7877**	**+3.30%**
ParaMaskNet	**0.8124**	**+7.91%**	0.7410	+11.27%	0.7872	+3.12%

4.2 Performance Comparison (RQ1)

The overall performances of different models on three evaluation datasets are show in the Table 2. From the experimental results, we can see that:

(1) Both the serial model and parallel model achieve better performance on all three datasets and obtains significant improvements over the state-of-the-art methods. It can boost the accuracy over the baseline FM by 3.12% to 11.40%, baseline DeepFM by 1.55% to 5.23%, as well as xDeepFM baseline by 1.27% to 4.46%. We also conduct a significance test to verify that our proposed models outperforms baselines with the significance level $\alpha = 0.01$.
Though maskNet model lacks similar module such as CIN in xDeepFM to explicitly capture high-order feature interaction, it still achieves better performance because of the existence of MaskBlock. The experiment results imply that MaskBlock indeed enhance DNN Model's ability of capturing complex feature interactions through introducing multiplicative operation into DNN models by instance-guided mask on the normalized feature embedding and feed-forward layer.

(2) As for the comparison of the serial model and parallel model, the experimental results show comparable performance on three evaluation datasets. It explicitly proves that MaskBlock is an effective basic building unit for composing various high performance ranking systems.

4.3 Ablation Study of MaskBlock (RQ2)

In order to better understand the impact of each component in MaskBlock, we perform ablation experiments over key components of MaskBlock by only removing one of them to observe the performance change, including mask module, layer normalization(LN) and feed-forward network(FFN). Table 3 shows the results of our two full version MaskNet models and its variants removing only one component.

From the results in Table 3, we can see that removing either instance-guided mask or layer normalization will decrease model's performance and this implies that both the instance-guided mask and layer normalization are necessary components in MaskBlock for its effectiveness. As for the feed-forward layer in MaskBlock, its effect on serial model or parallel model shows difference. The Serial model's performance dramatically degrades while it seems do no harm to parallel model if we remove the feed-forward layer in MaskBlock. We deem this implies that the feed-forward layer in MaskBlock is important for merging the feature interaction information after instance-guided mask. For parallel model, the multiple feed-forward layers above parallel MaskBlocks have similar function as feed-forward layer in MaskBlock does and this may produce performance difference between two models when we remove this component.

Table 3: Overall performance (AUC) of MaskNet models removing different component in MaskBlock on Criteo dataset(feature embedding size=10, MaskNet model has 3 MaskBlocks.)

Model Name	SerMaskNet	ParaMaskNet
Full	0.8119	0.8124
-w/o Mask	0.8090	0.8093
-w/o LN	0.8106	0.8103
-w/o FFN	0.8085	0.8122

4.4 Hyper-Parameter Study(RQ3)

In the following part of the paper, we study the impacts of hyper-parameters on two MaskNet models, including 1) the number of feature embedding size; 2) the number of MaskBlock; and 3) the reduction ratio in instance-guided mask module. The experiments are conducted on Criteo dataset via changing one hyper-parameter while holding the other settings. The hyper-parameter experiments show similar trend in other two datasets.

Number of Feature Embedding Size. The results in Table 4 show the impact of the number of feature embedding size on model performance. It can be observed that the performances of both models increase when embedding size increases at the beginning. However, model performance degrades when the embedding size is set greater than 50 for SerMaskNet model and 30 for ParaMaskNet model. The experimental results tell us the models benefit from larger feature embedding size.

Table 4: Overall performance (AUC) of different feature embedding size of MaskNet Models on Criteo dataset(the number of MaskBlock is 3)

Embedding Size	10	20	30	50	80
SerMaskNet	0.8119	0.8123	0.8121	0.8125	0.8121
ParaMaskNet	0.8124	0.8128	0.8131	0.8129	0.8129

Table 5: Overall performance (AUC) of different numbers of MaskBlocks in MaskNet model on Criteo dataset(embedding size= 10)

Block Number	1	3	5	7	9
SerMaskNet	0.8110	0.8119	0.8126	0.8117	0.8115
ParaMaskNet	0.8113	0.8124	0.8127	0.8128	0.8132

Table 6: Overall performance (AUC) of different size of Hidden Layer in Mask Module of MastBlock on Criteo dataset.(embedding size= 10, number of MaskBlock is 3)

Reduction ratio	1	2	3	4	5
SerMaskNet	0.8118	0.8119	0.8120	0.8117	0.8119
ParaMaskNet	0.8124	0.8124	0.8122	0.8122	0.8124

Number of MaskBlock. For understanding the influence of the number of MaskBlock on model's performance, we conduct experiments to stack MaskBlock from 1 to 9 blocks for both MaskNet models. The experimental results are listed in the Table 5. For SerMaskNet model, the performance increases with more blocks at the beginning until the number is set greater than 5. While the performance slowly increases when we continually add more MaskBlock into ParaMaskNet model. This may indicates that more experts boost the ParaMaskNet model's performance though it's more time consuming.

Reduction Ratio in Instance-Guided Mask. In order to explore the influence of the reduction ratio in instance-guided mask, We conduct some experiments to adjust the reduction ratio from 1 to 5 by changing the size of aggregation layer. Experimental results are shown in Table 6 and we can observe that various reduction ratio has little influence on model's performance. This indicates that we can adopt small reduction ratio in aggregation layer in real life applications for saving the computation resources.

4.5 Instance-Guided Mask Study(RQ4)

As discussed in Section in 3.2, instance-guided mask can be regarded as a special kind of bit-wise attention mechanism to highlight important information based on the current input instance. We can utilize instance-guided mask to boost the informative elements and suppress the uninformative elements or even noise in feature embedding and feed-forward layer.

To verify this, we design the following experiment: After training the SerMaskNet with 3 blocks, we input different instances into the model and observe the outputs of corresponding instance-guided masks.

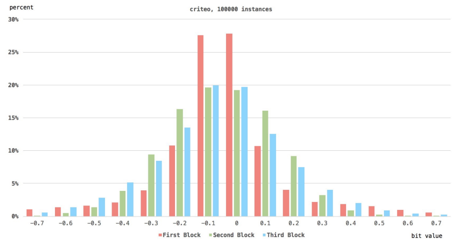

Figure 5: Distribution of Mask Values

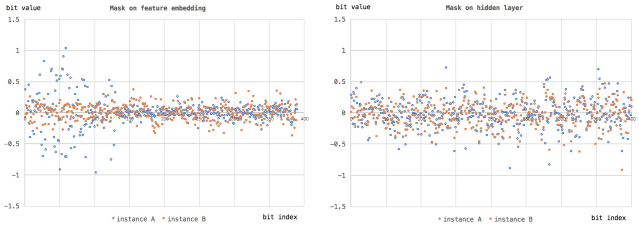

Figure 6: Mask Values of Two Expamples

Firstly, we randomly sample 100000 different instances from Criteo dataset and observe the distributions of the produced values by instance-guided mask from different blocks. Figure 5 shows the result. We can see that the distribution of mask values follow normal distribution. Over 50% of the mask values are small number near zero and only little fraction of the mask value is a relatively larger number. This implies that large fraction of signals in feature embedding and feed-forward layer is uninformative or even noise which is suppressed by the small mask values. However, there is some informative information boosted by larger mask values through instance-guided mask.

Secondly, we randomly sample two instances and compare the difference of the produced values by instance-guided mask. The results are shown in Figure 6. We can see that: As for the mask values for feature embedding, different input instances lead the mask to pay attention to various areas. The mask outputs of instance A pay more attention to the first few features and the mask values of instance B focus on some bits of other features. We can observe the similar trend in the mask values in feed-forward layer. This indicates the input instance indeed guide the mask to pay attention to the different part of the feature embedding and feed-forward layer.

5 CONCLUSION

In this paper, we introduce multiplicative operation into DNN ranking system by proposing instance-guided mask which performs element-wise product both on the feature embedding and feed-forward layers. We also turn the feed-forward layer in DNN model into a mixture of addictive and multiplicative feature interactions by proposing MaskBlock by bombing the layer normalization, instance-guided mask, and feed-forward layer. MaskBlock is a basic building block to be used to design new ranking model. We also propose two specific MaskNet models based on the MaskBlock. The experiment results on three real-world datasets demonstrate that our proposed models outperform state-of-the-art models such as DeepFM and xDeepFM significantly.

REFERENCES

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

[2] 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*. 46–54.

[3] 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.

[4] Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 933–941.

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

[6] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247* (2017).

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

[8] Jie Hu, Li Shen, and Gang Sun. 2018. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7132–7141.

[9] Tongwen Huang, Zhiqi Zhang, and Junlin Zhang. 2019. FiBiNET: combining feature importance and bilinear feature interaction for click-through rate prediction. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 169–177.

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

[11] 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.

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

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

[14] Chen Ma, Peng Kang, and Xue Liu. 2019. Hierarchical gating networks for sequential recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 825–833.

[15] 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*. 1930–1939.

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

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

[18] Steffen Rendle, Walid Krichene, Li Zhang, and John Anderson. 2020. Neural Collaborative Filtering vs. Matrix Factorization Revisited. *arXiv preprint arXiv:2005.09683* (2020).

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

[20] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *arXiv preprint arXiv:1505.00387* (2015).

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

[22] 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).

[23] Ruobing Xie, Cheng Ling, Yalong Wang, Rui Wang, Feng Xia, and Leyu Lin. 2020. Deep Feedback Network for Recommendation. 2491–2497. https://doi.org/10.24963/ijcai.2020/345

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