FeatNet: Feature Interaction Pruning Networks for CTR Prediction

ABSTRACT

Learning effective high-order feature interactions is crucial for click-through rate (CTR) prediction in recommender systems. Existing methods that learn meaningful high-order feature combinations by reassembling low-order feature combinations, i.e., 2-order feature cross, suffer from both high computational cost to calculate the interaction weight of all pairwise feature interactions and exhaustive search space to identify effective crosses, especially when dealing with the sparse and large number of features in real industry recommender systems. Some recent methods attempt to reduce the combinatorial search space of low-order feature combinations by using DNNs, which is considered as universal function approximators to potentially learn all kinds of feature interactions, however, it had been proved to be inefficient to even approximately the low-order interactions, i.e., 2-order or 3rd-order feature crosses. To learn the arbitrary-order feature interactions in a effective way, we propose a novel method: **Feat**ure Interaction Pruning **Net**works (**FeatNet**) for CTR prediction. Specifically, we design Feature Subset Generation Networks (FSGN) with a soft-threshold pruning operation to select the meaningful feature combinations in each subset, which maintains the diversity of different feature combinations as well as reduces the computational costs in the later high-order feature interaction. Then Set Attentive Pruning Networks (SAPN) is used to generate fine-grained arbitrary-order feature interactions for for CTR prediction. FeatNet is flexible to maintain high coverage ratio of feature combinations and denoise the irrelevant feature interactions by two-stage soft-threshold pruning strategy, making it have the ability to learn from a large number of feature combinations. Extensive experiments on three real-world datasets, including a large-scale industrial dataset from WeChat platform with billions of feature dimensions, showing the superiority of FeatNet to deal with high dimensional and sparse data for CTR prediction.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Computing methodologies** → **Neural networks**;

KEYWORDS

High-Order Interaction, Feature Pruning, Recommender Systems

ACM Reference Format:
. 2022. FeatNet: Feature Interaction Pruning Networks for CTR Prediction. In *Proceedings of International World Wide Web Conference (WWW'22).* ACM, France, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

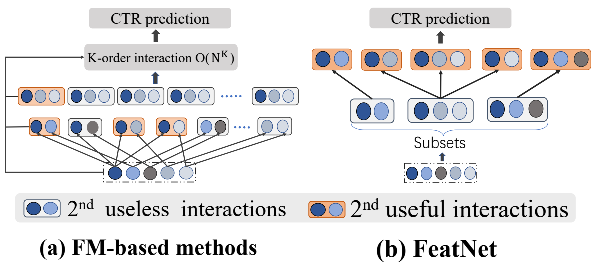


Figure 1: Complexity of computing high-order interactions in (a) escalated model architecture v.s. (b) de-escalated model architecture.

1 INTRODUCTION

Predicting the probabilities of users clicking on items (a.k.a., CTR: click-through rate prediction) is an important problem in recommender systems [1, 3, 4, 26, 40]. The key to achieve better performance in CTR prediction is learning effective feature interactions [6, 7, 15, 22, 36], and has attracted great attention in recent years. One of the biggest challenges lies in the high computational cost to compute the high-order interactions of raw features because the number of feature combinations increases exponentially when the number of raw features increase. In real-world applications, raw features are usually high sparse with millions of dimensions. For example, identifier features like the ID of the user/item is very sparse after being encoded as a one-hot vector; so are multi-filed vectors built from upstream tasks such as visual information. It is very time-consuming to calculate the high-order feature interactions on such sparse features with millions of dimensions and there is a high rick of model overfitting problem for CTR prediction.

Great efforts have been made to solve this problem, which can be generally classified into three categories: factorization machines based methods, such as FMs [34], DeepFM [14], XDeepFM [23] and FibiNet [18]; self-attention based method such as AutoInt [37], AFM [8] and HoAFM [42]; and neural feature crossing methods like DCN [44] and its improved version DCNV2 [45]. As showed in Fig. 1 (a), all of them can be summarized in an interaction *escalated* model architecture, in which the high-order information is derived by reassembling low-order feature interactions. In other words, we need to enumerate all 2-order feature interactions, leading to a high computational cost and a high risk of introducing noise interactions. Some methods [7, 14] attempt to reduce the combinatorial search space of low-order feature combinations by using DNNs, which is considered as universal function approximators to potentially learn all kinds of feature interactions [30], however, it had been proved to be inefficient to even approximately the low-order interactions, i.e., 2-order or 3rd-order feature crosses [1, 44].

To learn the arbitrary-order feature interactions in a effective way, we propose a novel interaction *de-escalated* model FeatNet with two-stage pruning operations to decrease the model complexity, and meanwhile, improve the ability of the model to learn from the useful feature interactions for CTR prediction. As shown in Fig. 1 (b), in the first pruning stage, we design a Feature Subset Generation Networks (FSGN) to select the meaningful subsets of different features, which maintains the diversity of feature combinations, and successfully reduces the feature size as well as removes noisy features before making interactions. In the second pruning stage, the interactions of de-escalated features in each sub-optimized feature subset are further fed into a Set Attentive Pruning Networks (SAPN) to generate fine-grained arbitrary-order feature interactions for CTR prediction. With two stage soft-threshold pruning operations in FSGN and SAPN respectively, we de-escalate both the number of features and the invalid high order interactions, so as to reduce the model complexity and promote the model performance.

A very important benefit brought by the two-stage pruning operations is the flexibility of our model to deal with any number of input features, which may induce a high time complexity in other *escalated* models [23, 34]. Thus, FeatNet can be easily adapted to real-world recommender scenarios with millions of dimensions of input features. Our contributions are summarized as follows:

- We design a novel *de-escalated* model FeatNet with two-stage pruning operations to decrease the model complexity, making our model applicable in large-scale industrial recommender scenarios with high dimensional input features.
- Our model can maintain high coverage ratio of feature combinations and can automatically learn the explicit arbitrary-order feature interactions for CTR prediction, which offers good model explainability.
- We conduct extensive experiments on two public datasets (Criteo and Avazu) and a large-scale industrial dataset from WeChat platform (termed as Production) with billions of input feature dimensions, showing the superiority of FeatNet to deal with high dimensional and sparse data for CTR prediction.

2 THE PROPOSED MODEL

In this section, we first provide an overview of our proposed model FeatNet, then we describe in more detail how FeatNet learns arbitrary-order feature interactions for CTR prediction.

2.1 Overview

As shown in Fig. 2, FeatNet is composed of three components: input feature embedding component, feature de-escalated component and prediction component. Different from interaction **escalated** models, in which the high-order interaction information is derived by first computing all the low-order feature interactions, then selecting the useful part (shown in Fig. 1(a)), we redefine the problem into a **de-escalated** model architecture, that is, a Feature Subset Generation Network (FSGN) is designed to firstly select the useful subsets of features, which reduces the feature size as well as removes noisy features before making interactions. Different features are pruned by a learnable-threshold in each subset to ensure the diversity of feature combinations. Then a Set Attentive Pruning Networks (SAPN) is adopted in each subset to generate fine-grained arbitrary high-order feature interactions for further CTR prediction. The details of each component is described in the following part.

2.2 Feature Embedding Component

Generally, the input features can be divided into three parts, numerical data such as age, gender of users; multi-field category data such as region or ID information, which usually is very sparse since many of them are identifiers; and vector data built by upstream tasks such as sequence data of a particular user, cross modality vector like visual representation of items. We first represent different fields of data as dense vectors, which are concatenated to obtain the final input feature $\mathbf{x}$, represented as:

$$\mathbf{x} = [\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, ..., \mathbf{x_M}], \tag{1}$$

where M is the number of dense features, $\mathbf{x_i}$ is the representation of the i^{th} dense feature.

2.3 Feature De-escalated Component

Instead of making interactions on all the high-dimensional input raw features, we use different feature subsets to obtain arbitrary number of features for further diversified interactions. We first de-escalate the number of features by feature subset generation networks (FSGN). Each subset network acts as a domain expert [28], and explore feature combinations from different perspective. To obtain the explicit high-order feature interactions, Set Attentive Pruning Networks (SAPN) is used to generate fine-grained arbitrary high-order feature interactions. With two stage soft-threshold pruning operations in FSGN and SAPN respectively, we de-escalate both the number of features and the invalid high order interactions, so as to reduce the model complexity and promote the model performance.

2.3.1 Feature Subset Generation Networks (FSGN). Feature Subset Generation Networks (FSGN) is designed to generate feature subsets, which contains different number of feature combinations. The combinations of different features in all subsets are expected to be diverse so as to provide more information for the further interaction. Besides, the number of features in each subset is also expected to be different, that is to say, providing both high-order and low-order feature combination information. FSGN generates the weights for each feature in different subsets.

Given the raw input feature matrix $\mathbf{x} \in \mathbb{R}^{M \times D}$, M is the number of features, D is the embedding size. Inspired by [17], we use nonlinear transformation with expand ratio to improve the representational capacity of input feature $\mathbf{x}$ as:

$$\mathbf{x}_r = \sigma(\mathbf{W}_1 \cdot \mathbf{x}^\top), \tag{2}$$

where σ is the Sigmoid activation function, $\mathbf{W}_1 \in \mathbb{R}^{r \times D}$ is the matrix weights and r is the expand ratio.

The value in matrix $\mathbf{p} \in \mathbb{R}^{M \times S}$ defines the probability of a certain dense feature $\mathbf{x}_i$ belongs to each subset, computed as:

$$\mathbf{p} = \mathrm{ReLU}(\mathbf{W}_2 \cdot \mathbf{x}_r), \tag{3}$$

where $\mathbf{W}_2 \in \mathbb{R}^{S \times r}$ is the matrix weights and S is the number of predefined subsets. We use ReLU as the activation function to eliminate those features with negative importance weights.

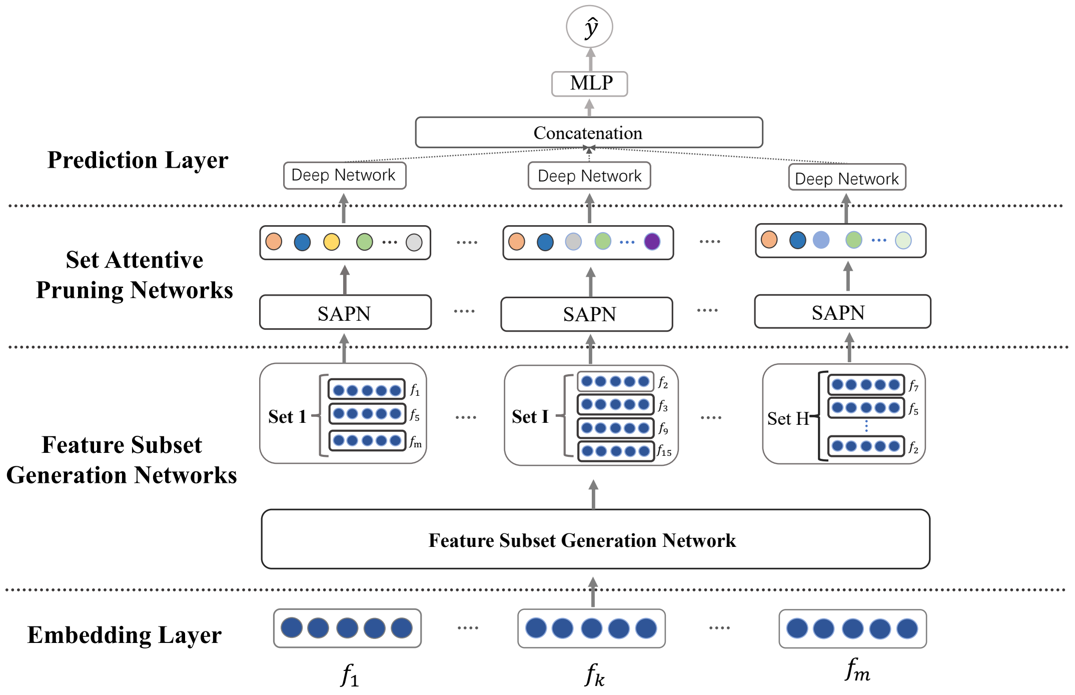


Figure 2: The overview architecture of FeatNet. Feature Subset Generation Network (FSGN) and Set Attentive Pruning Networks (SAPN) are designed to learn arbitrary-order useful feature interactions.

FSGN acts as a multi-class classifier, which means each dense feature is classified into multiple subsets according to the probabilities matrix. Considering only the features with negative probabilities are pruned by ReLU. To further remove the noisy features, we use soft threshold reparameterization [20] to further prune the features with small positive probability.

2.3.2 Soft-Threshold Pruning Operation. To select the useful information and reduce the model complexity, we adopt the soft-threshold pruning method by eliminating the small probability scores with a learnable threshold variable. For a value $p \in \mathbf{p}$, the soft-threshold probability $S_g(p, \tau)$ is computed as:

$$S_g(p, \tau) = sign(p) \cdot \text{ReLU}(|p| - g(\tau)), \tag{4}$$

where ReLU is the activation function, the probability p with absolute value smaller than τ would be pruned. $\tau \in \mathbb{R}^{M \times S}$ is the learnable threshold variable, and all the probability score owns individual pruning weight. $g(\tau) : \mathbb{R} \to \mathbb{R}$ is the learning function, which controls the pruning strategy that we want. Here we choose sigmoid function cause it helps us to control the pruning strength: for example, if we set the initial value of τ to a large negative number, the pruning function won't prune any weight until all the embedding is already well trained. if we set the initial values of τ close to zero, the pruning function will be activated at the start of the training. Setting τ to a large number is not suggested cause it might prune all the weight to zero and cause the failure of model.

Finally, a dense feature is classified into the subsets with non-zero values in the pruned probability matrix $\mathbf{p}^s$ computed in Eq. 4. The features in all subsets with re-weighted pruned probability, can be formulated as:

$$\mathbf{X} = \text{Reweight}(\mathbf{x}) = [\mathbf{p}_1^s \cdot \mathbf{x}, \mathbf{p}_2^s \cdot \mathbf{x}, ..., \mathbf{p}_M^s \cdot \mathbf{x}], \tag{5}$$

where $\mathbf{p}_i^s \in \mathbb{R}^{1 \times M}$ is a weighted vector in subset i, and $\mathbf{X} \in \mathbb{R}^{S \times M \times D}$ is the final re-weight feature matrix of all subsets.

2.3.3 Set Attentive Pruning Networks (SAPN). We had generated useful subsets of features in FSGN, which reduces the feature size as well as removes noisy features before making high-order interactions. Considering that the output of the FSGN is several sets with dynamic number of elements, our model need to deal with set-input problem of permutation invariant, as well as be capable of processing input sets of any size, which means Deep Neural Networks is not suitable for this stage. Hence we propose we propose Set Attentive Pruning Networks (SAPN) which improves the self-attention module to learn more possible combinations of different feature interaction.

The traditional self-attention can be described as followed: assuming the feature is $\mathbf{X}_i \in \mathbf{X}$ in a specific subset, we generate the key and query by:

$$\mathbf{K} = \mathbf{X}_i \cdot \mathbf{W}_K^\top, \tag{6}$$

$$\mathbf{Q} = \mathbf{X}_i \cdot \mathbf{W}_Q^\top, \tag{7}$$

where $\mathbf{W}_K$ and $\mathbf{W}_Q$ are the transformation matrix.

Then the attentive similarity score can be computed as:

$$\text{Score} = \text{Softmax}(\mathbf{Q} \cdot \mathbf{K}^\top), \tag{8}$$

where Score $\in \mathbb{R}^{M \times M}$.

A dense feature $\mathbf{X}_{ik} \in \mathbb{R}^{1\times D}$ is a row of feature matrix $\mathbf{X}_i \in \mathbb{R}^{M\times D}$, the high-order feature interactions of $\mathbf{X}_{ik}$ with all dense features can be computed as:

$$\mathbf{X}^{'}_{ik} = \sum_{j=1}^{M} \text{Score}_{kj}\mathbf{X}_{ij}, \tag{9}$$

where $\text{Score}_k \in \mathbb{R}^{1XM}$ is a row of the Score and $\mathbf{X}^{'}_{ik} \in \mathbb{R}^{1\times D}$ is new interacted feature.

Since each row of the Score can be considered as one combination of feature interactions, in which the weight is different for each dense feature. The number of combinations in traditional self-attention is M. Comparing with the number of all the possible feature combinations 2^M, we find the feature combinations in traditional self-attention method only cover a small proportion, *i.e.,* $\frac{M}{2^M}$ of all the possible feature combinations, and it will collapse exponentially with the increase of number of dense features M.

Considering that the different feature combinations will bring different information, and more information will be provided in sufficient feature combinations, which may be crucial to improve the model performance. We propose a new perspective to evaluate the quality of feature interactions that takes the number of interaction combinations into consideration except for the orders of feature interactions. We define the **coverage ratio** of feature combinations as:

$$\text{Coverage Ratio} = \frac{\#\text{Interaction Combinations}}{\#\text{All Feature Combinations}}, \tag{10}$$

where the number of all possible feature combinations is 2^M and M is the number of dense features, which can be up to hundreds in large scale industry recommendation systems.

To improve coverage ratio of feature combinations, so as to learn more information from different feature combinations, we improve the self-attention mechanism by using learnable query parameters, allowing us to control the coverage ratio of feature combinations. The improved induced-attentive similarity score [21] can be reformulated as:

$$\text{Score}' = \text{ReLU}(\mathbf{I} \cdot \mathbf{K}^{\top}), \tag{11}$$

where $\mathbf{I} \in \mathbb{R}^{E\times D}$ is the learnable query parameters used to control the number of combinations. The number of interaction combinations is determined by E.

The original activation function to compute the attention similarity score in Eq. 8 is softmax function. Considering it will take all the features into consideration even those with small attention weights. To select the most useful feature combinations as well as reduce the computational costs, we adopt similar strategy that used in FSGN: using ReLU and the soft-threshold method S_g in Eq. 4 to fine-pruned features for make further high-order feature interactions. The fine-pruned attention weights are:

$$\text{Score}'' = S_g(\text{Score}'). \tag{12}$$

Learnable interaction combinations of feature $\mathbf{X}_{ik}$ with all dense features in Eq. 9 can be reformulated as:

$$\mathbf{X}^{'}_{ik} = \sum_{j=1}^{M} \text{Score}''_{kj}\mathbf{X}_{ij}. \tag{13}$$

Table 1: Statistics of the datasets.

Datasets	# Instances	# Fields	# Dimensions
Criteo	45,840,617	39	2,086,936
Avazu	40,428,967	22	1,544,250
WeChat Production	10,000,000	400+	Billions+

In summary, in our feature de-escalated component, we adopted a coarse-to-fine two-stage pruning strategy in FSGN and SAPN. FeatNet is capable of capturing both high-order combination (coarse), as well as low-order feature combinations (fine), which achieves the goal of generating arbitrary-order of interactions. Besides, by adjusting the hyper-parameter properly in Eq. 11, our model could obtain the high coverage ratio of feature combinations for more interaction information, and meanwhile select the most meaningful information by two-stage pruning strategy.

2.4 Prediction Component

The interaction features in all subsets are fed into the final prediction component, represented as $[\mathbf{X}^{'}_1, \mathbf{X}^{'}_2, ..., \mathbf{X}^{'}_M]$. For each subset, we apply a separate deep neural networks to capture the high-order feature interaction information. The low-order interaction feature $\mathbf{X}^{'}_i \in R^{M\times D}$ is firstly squeezed into a vector $\mathbf{X}^{''}_i \in R^{1\times(M\times D)}$, then fed into an MLP layer. Let $\mathbf{a}^{(0)}_i = X^{''}_i$, it can be fomulated as:

$$\mathbf{a}^{(l)} = \sigma(\mathbf{W}^{(l)}_i \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}). \tag{14}$$

We concatenate the output vector of each subset to get the final output vector $\mathbf{h}$ and compute the final probability $\hat{p}$ as follows:

$$\hat{p} = \sigma(\mathbf{W}_o^{\top} \cdot \mathbf{h} + b), \tag{15}$$

where $\mathbf{W}_o$ and b is the transformation matrix and bias.

2.5 Loss Optimization

We use cross entropy loss function to optimize our model, represented as:

$$Loss = -\frac{1}{N}\sum_{i=1}^{N}(y_i \cdot \log(\hat{p}_i) + (1 - y_i) \cdot \log(1 - \hat{p}_i)), \tag{16}$$

where y_i is the ground truth of i-th instance, $\hat{p}_i$ is the predicted CTR, and N is the total size of samples. We employ the Adam optimizer with dynamic learning rate to accelerate model convergence [19].

3 EXPERIMENTS

We evaluate FeatNet on three real-world datasets with comparation of several representative models to show the superiority of our model to learn the high-order effective interactions for CTR prediction.

3.1 Experimental Settings

3.1.1 Dataset. We conduct experiments on three datasets, including two public datasets, *i.e.,* Criteo and Avazu, and a large scale industry dataset from WeChat platform, termed as Production. The statistics details of three datasets are summarized in Table 2.

Table 2: Comparison of different methods. Higher coverage ratio of interaction will provide more information to promote the model performance. M is the number of fields, H, S and E are the real number hyper-parameters in AFN and FeatNet.

Model	High-Order	De-Noise	# Approximated Coverage Ratio
DeepFM	✓	×	High (C_M^2)
xDeepFM	✓	×	High (C_M^2+ O(M))
FiBiNet	×	✓	High (C_M^2)
AutoInt	✓	×	Low (O(M))
AFN	✓	×	Adaptive (O(H))
DCNV2	✓	×	NA
FeatNet	✓	✓	Adaptive (S*E)

- **Criteo**[1]: this is a popular industry benchmark dataset for CTR prediction, which contains 13 numerical feature fields and 26 categorical feature fields.
- **Avazu**[2]: it contains users' click records on mobile advertisements. It has 22 feature fields including user features and advertisement attributes.
- **Production**: this data is collected on WeChat[3] platform, which is a production scenario with hundreds of feature fields and more than one hundred million daily active users.

3.1.2 Evaluation metrics. We adopt the classical metric for performance evaluation: AUC (Area Under the ROC curve) [13]. Note that a slight increase in AUC at 0.001-level is considered to be a significant improvement for the CTR prediction task [18, 23]

3.1.3 Baseline Methods Compared. We compare our model with different types of state-of-the-art methods for CTR prediction, including:

- DeepFM [14]: it captures the low-order feature interactions in FM and high-order interactions in a deep neural networks for CTR prediction.
- xDeepFM [23]: it is a generalization of DeepFM by learning the linear regression weights for the FM layer.
- AFN [8]: it models adaptive-order feature interactions in the logarithmic space.
- AutoInt [37]: it leverages self-attention networks to learn high-order features interactions.
- FiBiNet [18]: it uses squeeze-excitation network to capture important features, and proposes bilinear interactions to enhance feature interactions.
- DCNV2 [45]: it is the state-of-the-art method in CTR prediction, in which the the feature interactions are automatically learned by a cross neural networks.

We compare with the above methods on three important aspects: the ability to capture explicit high-order interactions, denoising, and the approximated coverage ratio (defined in Eq. 10) of feature combinations. The comparison of different methods is summarized in Table 2, we can see that: the interaction coverage ratio is high (*i.e.,* O(M^2)) in DeepFM, xDeepFM and FiBiNet, because they explicitly model all 2^{nd} order interactions. For DeepFM and xDeepFM, high-order information is captured by deep neural networks, but they cannot denoise the irrelevant feature interaction. FiBiNet uses squeeze-excitation network to calculate feature importance, and denoises the irrelevant information from the large amount of feature interactions, but it cannot capture the high-order feature interactions. As for AutoInt, although it constructs high-order interactions, only O(M) combinations of features can be learned, which leads to low coverage ratio of feature interactions. AFN can reach high coverage ratio as well as high-order interactions by setting the hyper-parameter H, but it fails to eliminate the noise. For DCNV2, since it uses cross networks to learn high-order feature interactions, it is hard to explicitly compute its coverage ratio. FeatNet considers all three important aspects: it uses a learned query parameter in Eq. 11 to ensure that the high-order information can be learned from abundant feature combinations by enlarging the coverage ratio in each subset, and the soft-threshold pruning operation in FSGN and SAPN enable FeatNet to denoise the irrelevant information, so as to learn the arbitrary-order feature interactions.

3.1.4 Experimental setting. Following the experimental settings in [39], we randomly split the instances by 8:1:1 for training, validation and testing respectively. For each baseline method, grid search is applied to find the optimal settings. These include latent dimensions D from $\{8, 16, 32\}$, and the learning rate from $\{0.1, 0.01, 0.001, 0.0001\}$. We report the result of each method with its optimal hyperparameter settings on the validation data. In FeatNet model, we set the dimensions D of latent vector to 16, and learning rate is 0.0001, the number of subsets S is 10, expand ratio r is set to 4, the depth for MLP of each subset is 3, and the number of neurons is 128 , the learnable query variable E in Eq. 11 to control the coverage rate is set to 16 for three datasets. All the models are implemented in Pytorch 1.7.0 , CUDA v11.2, and run on hardware with AMD EPYC 7K62 48-Core Processor and GPU A100-SXM4-40GB. Due to space limit, more details of the parameter settings of baseline methods are provided in supplementary material.

3.1.5 Model Complexity Analysis. Here, we use k and M to represent the order of the feature interactions and the number of dense features, D as the embedding size of the feature respectively. In FeatNet, for Feature Subset Generation Networks (FSGN), the computation cost depends on the expand ratios r in Eq. 2. If we assume the number of predefined subset is S, then the computational cost equals to O(M*S/r+M*S). For the second pruning module: Set Attentive Pruning Networks (SAPN), since we only focus on the cost of computing interaction combination rather than the cost of interaction method, the cost of SAPN is O(S*E*M), where E is the dimension of predefined induced query. Therefore the total time complexity that FeatNet needs to compute feature combinations at arbitrary order is O(M*S*(1+1/r+E)), if M is large, FeatNet only take linear time complexity of O(M) to generate arbitrary-order feature combinations. As for high-order FMs based methods, like HOFMs [2], supposing n is the maximum order of feature combinations as predefined, it takes $O(kM^n)$ time to deliver a prediction which can be reduced to $O(kMn^2)$ with dynamic programming. Note that the time complexity of HOFMs is highly correlated with

[1]http://labs.criteo.com/downloads/2014-kaggle-display- advertising-challenge-dataset
[2]http://www.kaggle.com/c/avazu-ctr-prediction
[3]https://mp.weixin.qq.com

Table 3: Performance comparison of different methods.

Datasets	Criteo		Avazu		Production	
Models	AUC	LogLoss	AUC	LogLoss	AUC	LogLoss
DeepFM	0.7922	0.40929	0.7623	0.39522	0.6737	2.2388
xDeepFM	0.7910	0.40899	0.7611	0.39567	0.6745	2.2624
AutoInt	0.7888	0.41318	0.7622	0.39575	0.6796	2.2401
AFN	0.7941	0.40822	0.7622	0.39555	0.6777	2.3177
FiBiNet	0.7932	0.40824	<u>0.7651</u>	0.39417	0.6815	2.2376
DCNV2	<u>0.7960</u>	**0.40687**	<u>0.7651</u>	**0.39391**	<u>0.6837</u>	2.2215
FeatNet	**0.7995**	0.40697	**0.7657**	0.39417	**0.6857**	**2.2127**

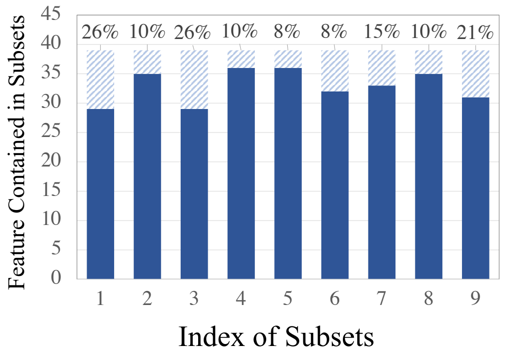
Note a slight increase in AUC at 0.001-level is known to be a significant improvement for the CTR prediction.

the maximum order n of cross features, making it difficult to balance the high-orders interaction of features and model complexity. While in FeatNet, both S and E are only determined by the model structure due to its adaptive cross features generation manner.

3.2 Main Results

We present the results of AUC and LogLoss in Table 3. We have the following observations:

- The overall performance of DeepFM and xDeepFM is poor on three datasets, which may be due to their incapability to denoise the irrelevant information from all 2^{nd} feature combinations. The performance is not consistent in three datasets, possibly due to the fact that more combinations are constructed in Avazu and Production datasets with more feature fields.
- AutoInt performs better than xDeepFM on Production dataset, showing the effectiveness of self-attention mechanism in capturing the high-order information. However, it loses advantages on Criteo and Avazu datasets, which may be due to the insufficient information provided in low coverage ratio of feature combinations.
- AFN achieves better performance than AutoInt on Criteo dataset, and performs competitively on other datasets, implying the effectiveness to enlarging the coverage ratio of feature combinations.
- FiBiNet outperforms AutoInt on three datasets, showing the importance of the large amount of feature combinations, which may provide sufficient information.
- DCNV2 is the state-of-the art baseline, with high capacity to learn feature interactions using the cross networks. The original first order features are directly fed into the following layers, which is similar to residual networks. This is effective for denoising, and improves the performance of DCNV2.
- FeatNet achieves the best AUC results on all datasets, showing that the de-escalated architecture designed in FeatNet can effectively learn the useful arbitrary-order feature interactions. For datasets with large number of feature fields like Production, FeatNet outperforms the state-of-the-art method DCNV2 by more than 0.002, showing the capability of FeatNet to deal with high dimensional and sparse input features in large scale industrial recommender systems.



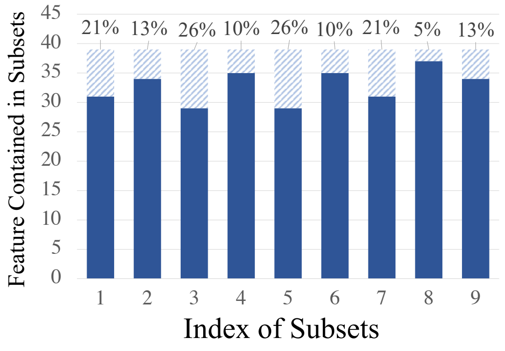
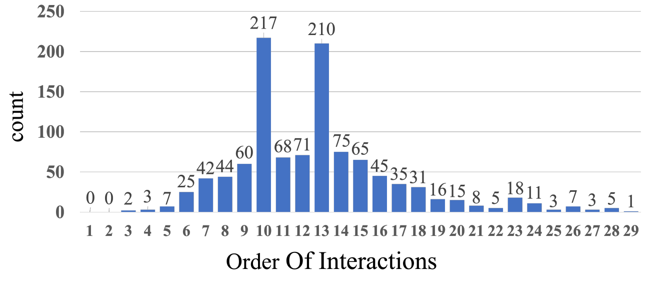
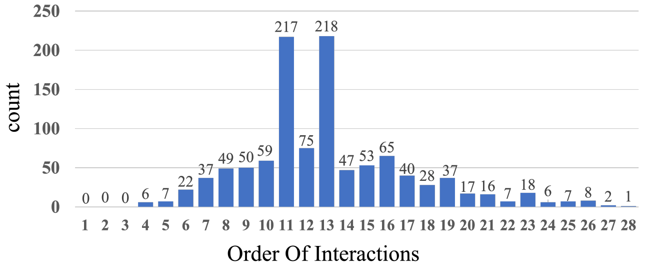


Figure 3: The number of features in different subsets.

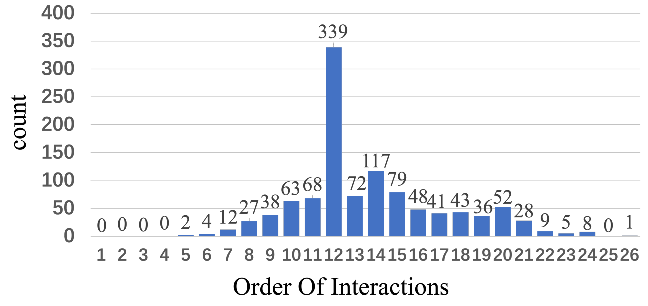
- Considering the model properties summarized in Table 2, we find that higher coverage ratio of feature combinations can provide more information, which is useful for CTR prediction; but without the denoising operation, it will damage the model results (xDeepFM v.s. AutoInt). With denoising, it will enhance the model capability (FiBiNet v.s. AutoInt). Besides, compared with implicit high-order information, the explicit high-order information is more crucial for model performance: the model that learns explicit high-order information achieves better performance (AFN/AutoInt v.s. DeepFM/xDeepFM).



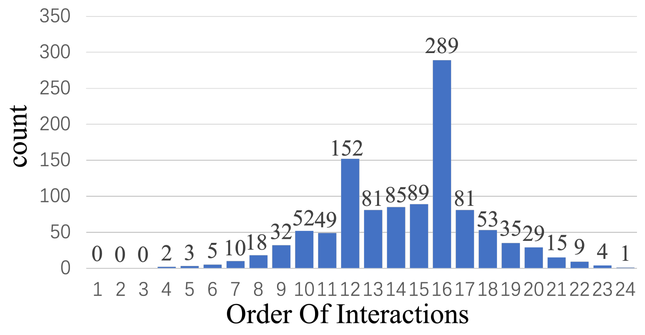
(a)



(b)



(c)



(d)

Figure 4: Different number of interaction orders after pruning operation.

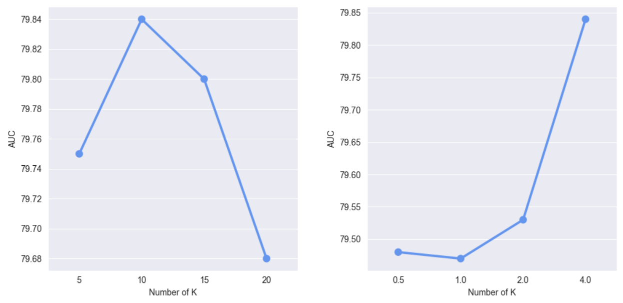
3.3 Experimental Analysis

To examine the effectiveness of FeatNet in filtering out irrelevant noise and learning the high-order feature interactions, we further investigate the number of features in each subsets and display the explicit order information learned in FeatNet.

3.3.1 Learning De-escalated Features in Subsets. FeatNet uses different subsets to select the useful feature combinations, which reduces the feature size before making high-order feature interactions. The features pruned in each subset varies from 5% to 30%, with an average number of 10% in all three datasets. As we mentioned in the model part, we hope the feature subset generation networks to maintain proper quantity of features to ensure the diversity of features in each subset, but also needs to eliminate usefulness features so as to de-noise. We present two instances randomly selected from Criteo dataset for this analysis. As shown in Fig. 3, we can see that the feature numbers are different in each subset, and the features are pruned at least 8% and at most 26%. It illustrates that FSGN will get rid of different features for each subset and did ensure the criteria that we mentioned above so as to provide adequate feature information for the next stage of fine-grained pruning.

3.3.2 Learning Arbitrary Feature Orders. Different from Self-attention based methods, FeatNet is capable of explicitly construct arbitrary-order of feature interactions. In order to explore the statistic distribution of the order that SAPN generated, we use the attention matrix in SAPN to calculate the non-zero number in each interaction pairs as the order of corresponding interactions. We randomly select four instances from Criteo dataset to show the learned feature order information. As is shown in Fig. 4, quite reasonable arbitrary interaction orders are learned by our two stage pruning strategy. The interaction orders learned in our model vary from 3 to 29 (maximum order is 39), and the middle order range varies from 8 to 18. To be more specific, FeatNet keeps less low-order interactions since some of them can be represented by middle-order interactions, and it also drops most of the high-order interactions that more than 20 orders, which may due to the high-order feature combinations may incorporate the irrelevant information. Specifically, for different samples, the order distribution is quite different: for the instance (a), (b) and (d) , they follow bi-modal distribution while (c) follows uni-modal distribution from the perspective of distribution mode, showing the arability of our model to capture arbitrary order feature combinations.

3.3.3 Hyperparameters Analysis. We analyze the effect of the number of subsets S and the dimension E of the learnable query parameter in Eq. 11, which controls the coverage ratio of feature combinations on the performance of FeatNet. As shown in Fig. 5, we observe that when the number of subsets increases, the performance of FeatNet increases first and then decreases when the number of subsets becomes too large. This indicates that the number of subsets should be set appropriately to make a trade-off between increasing feature coverage ratio and introducing more noise in order to achieve the best performance. For the learnable query parameter in Eq. 11, it controls the order of interactions to be learned in FeatNet. We can see that the higher order learned in model, the



(a) The number of subsets. (b) Learnable query parameter.

Figure 5: The influence of hyper-parameters.

better the performance. We set learnable query parameter E to 40 in our experiments to balance the computational costs and good model performance.

4 RELATED WORK

The researches for CTR prediction [5, 9, 29, 35, 41] can be generally divided into two classes. One type of mainstream algorithms focuses on how to capture higher-order feature interactions, while the other tries to find better ways to learn the importance of feature interactions for better feature combinations.

4.1 Learning High-Order Interactions.

The classical feature interaction method is the Factorization Machines (FM) [34], which tends to introduce second-order interaction through enumerating all second-order interactions and perform dot product on each interaction pair. To capture the high-order information, HOFM [2] provides an efficient method to construct high-order feature interactions, however, the drawback of HOFM is obvious: when building K-order interaction, it needs to compute all K-order feature pairs, which endues very high computational costs. To address this problem, DeepFM [14] combines FM with Deep Neural Networks. It uses FM to construct second-order interaction and DNN to capture high-order interaction implicitly. XDeepFM [23] improves DeepFM by using a CIN Module to explicitly enumerate construct second-order interaction. In order to reduce the number of interactions pairs, CIN sums up the results of the interaction then make further higher-order interactions. Although CIN might reduce the computation cost of high-order interaction, it still does not solve the problem raised by FM, i.e. the need to enumerate all the combinations of feature pairs. DCN [44], as well as its variant DCNV2 [45], squeezes the whole feature matrix into a vector and uses cross-layer information to extract feature interactions, which achieves better performance in capturing implicit interactions. Recently, with the Transformer model [43], AutoInt [37] is proposed to utilize Multi-head self-attention block to construct feature interactions. Query and Key are used to evaluate feature similarity, then features are added up to obtain the high-order interaction information. However, it suffers from the high number of combinations of all the weighted features, in which the features with low weights may introduce noise. To pursue more fine-grained feature interaction, AFM [8] maps embedding space into logarithmic space and eliminate the feature with zero weight. However, It is almost impossible for the model to learn zero weight without any further constraints.

4.2 Learning Weights of Interactions.

It is very useful to evaluate the importance weight of feature interactions, which can be used to remove irreverent information for CTR prediction. Many studies focused on mining the importance weight of feature interactions. FwFM [33] improves FM by assigning learnable weight to each interaction pair. IAFM [16] considers the feature aspect and field aspect to learn flexible interactions weight on two levels. FiBiNet [18] uses Squeeze-Excitation [17] networks to learn weights and conduct bilinear layer to model feature interactions. AutoFIS [24] automatically identify important feature interactions for factorization models with low computational costs.

4.3 Pruning methods.

We adopt soft-threshold pruning strategy [20, 46, 47] to select the benefit feature interactions in our model. There are many algorithms in literature focusing on learning effective pruning.[10–12, 31, 32] A practical method proposed in [27] uses L_0 norm regularization in neural networks, and prunes the network during training by encouraging weights to become exactly zero. L0-SIGN [38] computes feature interaction importance using DNN then apply the above pruning method to construct useful interaction. A simpler soft-threshold method (STR) is proposed in [20] to use soft threshold weight to reparameterize the method to learn from sparse data. It will prune the neuron to zero when the value of it is smaller than the threshold. STR has been utilized in recommendation system [25] to prune the embedding parameters. They use different embedding sizes for different fields, which maintain high accuracy while decreasing more than 95% embedding parameters.

5 CONCLUSION

In this paper, a novel de-escalated model architecture is proposed in FeatNet. It first selects the useful feature combinations and then learns the meaningful high-order feature interactions. FeatNet is flexible to maintain high coverage ratio of feature combinations and denoise the irrelevant feature interactions by a soft-threshold pruning strategy. This improves the capacity of our model to learn from a large number of feature combinations. FeatNet avoids the weaknesses of high computational costs caused by calculating all the 2-order interactions. It can be easily adopted to real-world industrial recommender scenarios with high-dimensional input features. While the idea of using de-escalated model architecture is validated, our implementation can be further improved in the future, namely with respect to automatically determining the number of subsets.

REFERENCES

[1] 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.
[2] Mathieu Blondel, Akinori Fujino, Naonori Ueda, and Masakazu Ishihata. 2016. Higher-order factorization machines. In *Advances in Neural Information Processing Systems*. 3351–3359.

[3] Andrei Z Broder. 2008. Computational advertising and recommender systems. In *Proceedings of the 2008 ACM conference on Recommender systems*. 1–2.

[4] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*. 129–136.

[5] Olivier Chapelle, Eren Manavoglu, and Romer Rosales. 2014. Simple and scalable response prediction for display advertising. *ACM Transactions on Intelligent Systems and Technology (TIST)* 5, 4 (2014), 1–34.

[6] Ting Chen, Ji Lin, Tian Lin, Song Han, Chong Wang, and Denny Zhou. 2018. Adaptive mixture of low-rank factorizations for compact neural modeling. (2018).

[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*. 7–10.

[8] Weiyu Cheng, Yanyan Shen, and Linpeng Huang. 2020. Adaptive factorization network: Learning adaptive-order feature interactions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 3609–3616.

[9] Xiaotie Deng, Paul Goldberg, Yang Sun, Bo Tang, and Jinshan Zhang. 2017. Pricing ad slots with consecutive multi-unit demand. *Autonomous Agents and Multi-Agent Systems* 31, 3 (2017), 584–605.

[10] Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. 2020. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*. PMLR, 2943–2952.

[11] Jonathan Frankle and Michael Carbin. 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635* (2018).

[12] Trevor Gale, Erich Elsen, and Sara Hooker. 2019. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574* (2019).

[13] 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.

[14] 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).

[15] 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*. 355–364.

[16] Fuxing Hong, Dongbo Huang, and Ge Chen. 2019. Interaction-aware factorization machines for recommender systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 3804–3811.

[17] 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.

[18] 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.

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

[20] Aditya Kusupati, Vivek Ramanujan, Raghav Somani, Mitchell Wortsman, Prateek Jain, Sham Kakade, and Ali Farhadi. 2020. Soft threshold weight reparameterization for learnable sparsity. In *International Conference on Machine Learning*. PMLR, 5544–5555.

[21] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. 2019. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*. PMLR, 3744–3753.

[22] Zeyu Li, Wei Cheng, Yang Chen, Haifeng Chen, and Wei Wang. 2020. Interpretable click-through rate prediction through hierarchical attention. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 313–321.

[23] 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*. 1754–1763.

[24] Bin Liu, Chenxu Zhu, Guilin Li, Weinan Zhang, Jincai Lai, Ruiming Tang, Xiuqiang He, Zhenguo Li, and Yong Yu. 2020. Autofis: Automatic feature interaction selection in factorization models for click-through rate prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2636–2645.

[25] Siyi Liu, Chen Gao, Yihong Chen, Depeng Jin, and Yong Li. 2021. Learnable Embedding Sizes for Recommender Systems. *arXiv preprint arXiv:2101.07577* (2021).

[26] Tie-Yan Liu. 2011. Learning to rank for information retrieval. (2011).

[27] Christos Louizos, Max Welling, and Diederik P Kingma. 2017. Learning sparse neural networks through L_0 regularization. *arXiv preprint arXiv:1712.01312* (2017).

[28] 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.

[29] 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*. 1222–1230.

[30] Hrushikesh N Mhaskar. 1996. Neural networks for optimal approximation of smooth and analytic functions. *Neural computation* 8, 1 (1996), 164–177.

[31] Hesham Mostafa and Xin Wang. 2019. Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. In *International Conference on Machine Learning*. PMLR, 4646–4655.

[32] Sharan Narang, Erich Elsen, Gregory Diamos, and Shubho Sengupta. 2017. Exploring sparsity in recurrent neural networks. *arXiv preprint arXiv:1704.05119* (2017).

[33] Junwei Pan, Jian Xu, Alfonso Lobos Ruiz, Wenliang Zhao, Shengjun Pan, Yu Sun, and Quan Lu. 2018. Field-weighted factorization machines for click-through rate prediction in display advertising. In *Proceedings of the 2018 World Wide Web Conference*. 1349–1357.

[34] Steffen Rendle. 2010. Factorization machines. In *2010 IEEE International conference on data mining*. IEEE, 995–1000.

[35] 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*. 521–530.

[36] 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*. 255–262.

[37] 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.

[38] Yixin Su, Rui Zhang, Sarah Erfani, and Zhenghua Xu. 2021. Detecting Beneficial Feature Interactions for Recommender Systems. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI)*.

[39] Yang Sun, Junwei Pan, Alex Zhang, and Aaron Flores. 2021. FM2: Field-matrixed Factorization Machines for Recommender Systems. In *Proceedings of the Web Conference 2021*. 2828–2837.

[40] Yang Sun, Yunhong Zhou, and Xiaotie Deng. 2014. Optimal reserve prices in weighted GSP auctions. *Electronic Commerce Research and Applications* 13, 3 (2014), 178–187.

[41] Yang Sun, Yunhong Zhou, Ming Yin, and Xiaotie Deng. 2012. On the convergence and robustness of reserve pricing in keyword auctions. In *Proceedings of the 14th Annual International Conference on Electronic Commerce*. 113–120.

[42] Zhulin Tao, Xiang Wang, Xiangnan He, Xianglin Huang, and Tat-Seng Chua. 2020. HoAFM: a high-order attentive factorization machine for CTR prediction. *Information Processing & Management* 57, 6 (2020), 102076.

[43] 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*. 5998–6008.

[44] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*. 1–7.

[45] Ruoxi Wang, Rakesh Shivanna, Derek Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed Chi. 2021. DCN V2: Improved Deep & Cross Network and Practical Lessons for Web-scale Learning to Rank Systems. In *Proceedings of the Web Conference 2021*. 1785–1797.

[46] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. 2018. Generative image inpainting with contextual attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5505–5514.

[47] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. 2019. Free-form image inpainting with gated convolution. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 4471–4480.