Input Enhanced Logarithmic Factorization Network for CTR Prediction

Xianzhuang Li[1], Zhen Wang[2], Xuesong Wu[3], Bo Yuan[1], and Xueqian Wang[1(✉)]

[1] Shenzhen International Graduate School, Tsinghua University, Shenzhen, China
lixz19@mails.tsinghua.edu.cn, yuanbo@ieee.org, wang.xq@sz.tsinghua.edu.cn
[2] School of Computer Science, The University of Sydney, Sydney, Australia
zwan4121@uni.sydney.edu.au
[3] Internet Business Department, Xiaomi, Beijing, China
wuxuesong@xiaomi.com

Abstract. Factorization-based methods, which can automatically model second-order or higher-order cross features, have been the benchmark models for click-through rate (CTR) prediction. In general, they enumerate all cross features with a predetermined order and filter out useless interactions through model training. However, two significant challenges remain. First, a maximum order for feature interactions needs to be defined in advance, imposing a trade-off between computational cost and the expression ability of models; Second, enumerating all feature interactions may introduce unwanted noise. In this work, we propose a novel model called *Input Enhanced Logarithmic Factorization Network* (ILFN), which can effectively learn arbitrary-order feature interactions and identify useful cross features. More importantly, ILFN can take full advantage of the original feature distribution in a discriminative way.

The core of ILFN is the *Input Enhanced Component* (IEC), which can represent the impact of each feature in a feature interaction as a trainable coefficient to learn arbitrary-order cross features. Moreover, IEC can reduce the demand for logarithmic neurons by exploiting the essential raw information and does not need to incorporate the deep neural network (DNN) to model high-order interactions. Therefore, ILFN is more effective and efficient and can converge to satisfactory results faster. Extensive experiments on four real-world datasets demonstrate that our ILFN model can outperform start-of-the-art methods. The effectiveness of each proposed component is also verified by hyper-parameter and ablation studies.

Keywords: Factorization machines · Logarithmic neural networks · Recommender systems · Feature interactions

J. Gama et al. (Eds.): PAKDD 2022, LNAI 13282, pp. 42–54, 2022.
https://doi.org/10.1007/978-3-031-05981-0_4

1 Introduction

Recommender systems are essential to many Internet companies. Among many real-world recommender systems such as news ranking, financial analysis and online advertising, Click-through rate (CTR) prediction plays a central role. The key of CTR prediction is to estimate the probability of a user clicking the recommended item, which has a direct impact on the final revenue [3].

Features play a vital role in the success of many CTR prediction tasks, and how to effectively represent features is one of the keys to obtaining optimal results. Although handcrafted feature interactions [5] are effective, data scientists need to spend great efforts on the transformation of original features to generate the best prediction systems. What is even worse, for large-scale datasets, such as the popular benchmark dataset *Criteo* with more than two million features, it is impractical to work on these features manually. The *Factorization Machine* (FM) [20] model was proposed to search for feasible transformations of the raw features. Due to its robust processing capabilities for sparse features and good scalability, the FM model has been widely adopted and has been proved effective for various tasks [20,21]. However, limited by its polynomial regression part, it only leverages first- and second-order feature interactions. In contrast, the higher-order cross features may play a more critical role in boosting the accuracy of CTR prediction [12,18].

With the success of deep neural networks (DNNs) in many fields such as natural language processing and computer vision, using deep learning for feature interactions has been a research trend in CTR prediction. Various DNN-based FM methods [3,6,13,18] have been proposed in recent years to model both low- and high-order cross features. For those models, in general, a maximum order is predefined, and then all combinatorial features within this order are enumerated while irrelevant interactions are filtered out by training. However, two significant challenges remain in the above methods. First, predefining an order, which is generally small (such as 3), may restrict the exploitation of higher-order cross features. Second, since not all useless feature combinations can be successfully removed, enumerating all cross features may introduce harmful noise and complicate the training. To this end, the logarithmic neural network (LNN) [15] was introduced to learn arbitrary-order feature interactions [4].

Unfortunately, LNN cannot handle negative or zero vectors. As a result, the AFN model [4], learning cross features via LNN, must convert negative embedding vectors to their absolute values and add noise to zero embedding vectors, leading to the lack and distortion of the initial embedding information. In this paper, we propose a novel model named *Input Enhanced Logarithmic Factorization Network* (ILFN), which can effectively and adaptively learn arbitrary-order cross features and their weights from data. The core of ILFN is the *Input Enhanced Component* (IEC), which can convert the impact of each feature in a feature interaction into a learnable coefficient with the following benefits: First, IEC can build arbitrary-order feature interactions without distortion of the initial distribution. Second, we use the attention mechanism to integrate the arbitrary-order cross features and natural embedding features, which shows

the importance of these features. Besides, it reduces the need for logarithmic neurons and does not need to ensemble the deep neural network (DNN) to mine high-order feature interactions. Compared to existing methods, ILFN is more expressive yet remains efficient and simple. Third, we perform batch normalization (BN) on the output of logarithmic transformation and exponential transformation to eliminate the influence of embedding vectors close to 0. To summarize, we make the following contributions:

- We propose a novel model called *Input Enhanced Logarithmic Factorization Network* (ILFN), which can effectively learn arbitrary-order feature interactions and identify useful cross features.
- The core of ILFN is the *Input Enhanced Component* (IEC), which can make full use of the original data distribution to model feature combinations efficiently and discriminatively without destroying the initial data distribution like LNN.
- IEC reduces the demand for logarithmic neurons and does not need to ensemble DNNs, resulting in fast convergence speed.
- Extensive experiments on four real-world datasets demonstrate that ILFN outperforms several state-of-the-art methods consistently. An ablation study verifies the effectiveness of each proposed component.
- We provide a case study and share analysis of ILFN to present the impact of vital components (such as IEC and LNN) on performance in the field of CTR prediction.

2 Related Work

Logistic Regression (**LR**[1]) [9] is a linear model that only models the linear combination of features for CTR prediction. By contrast, *Factorization Machine* (**FM**) [20], which works well on large sparse data, projects features into low-dimensional vectors and learns second-order interactions by inner products. It has received significant research interest from the community due to its excellent performance on sparse data sets, low time complexity, and low memory storage. Afterwards, different variants of factorization networks have been proposed. As an extension of the standard FM, *Field-aware Factorization Machine* **FFM** [16] introduces field-aware latent vectors to model the interactions between features from different fields. **HOFM** [21] can model high-order cross features, but one major downside is that it builds all cross features, including both useful and useless interactions.

With the great success of deep neural networks (DNNs), many researchers have proposed various DNN-based methods for CTR prediction. FNN [23] uses FM to pre-train the embedding data. **PNN** [11] explores the high-order cross features via a product layer between the embedding layer and the DNN layers. **NFM** [13] introduces neural networks into the FM model to enhance its ability

[1] The models in bold are baselines in the experimental part.

to leverage high-order feature interactions. **CrossNet** [18] models cross features by the outer product. **Deep&Cross** [18] integrates CrossNet with DNNs. **Wide&Deep** [3] combines LR with DNNs, where LR models the linear feature combinations while DNNs learn high-order non-linear cross features. **DeepFM** [6] combines FM with DNNs, which uses FM to replace LR in Wide&Deep. **xDeepFM** [12], an improved version of DeepFM, can explicitly model higher-order feature interactions by introducing the **CIN** component.

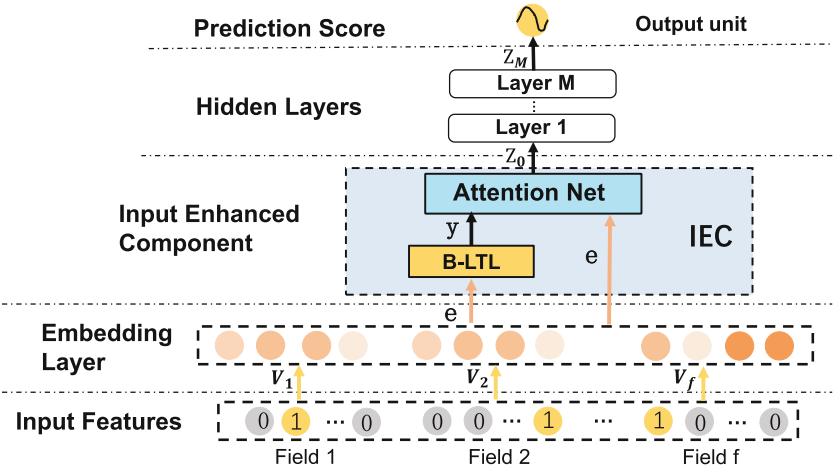


Fig. 1. The network structure of the ILFN model

However, these DNN-based models generally need to enumerate all possible cross features, introducing noise and complicating the training. To alleviate this challenge, **AFM** [14] adopts attention mechanism [1] to filter out the noise by assigning different coefficients to diverse feature interactions. Nevertheless, it can only handle second-order cross features. To this end, **AFN** [4] was proposed to learn arbitrary-order cross features with the logarithmic transformation layer, which results in the lack and distortion of the initial embedding information. In this work, we propose an effective model named *Input Enhanced Logarithmic Factorization Network* (ILFN), which can effectively and adaptively learn arbitrary-order cross features and their weights for CTR prediction. More importantly, it takes full advantage of the original information with lower time complexity.

3 Our Approach

In this section, we introduce our proposed *Input Enhanced Logarithmic Factorization Network* (ILFN). To effectively learn arbitrary-order feature interactions and take full advantage of the original information, we propose the *Input*

Enhanced Component (IEC), the core of ILFN. The *Input Enhanced Component* contains two parts: the BN-Logarithmic Transformation Layer (B-LTL) and Attention Layer. The former can represent the impact of each feature in a feature interaction as a trainable coefficient to learn arbitrary-order cross features. The latter can integrate outputs of the former and embedding layer, which handles data distortion and shows the importance of various features. In addition, the Attention Layer can fit classic DNN to learn high-order interactions by changing weights of different vectors, so ILFN no need to extra integrate DNN components like AFN. Therefore, ILFN is more effective and can converge to satisfactory results faster.

In summary, our ILFN can discriminatively learn arbitrary-order cross features without lack and distortion of the original data distribution. Next, we introduce the ILFN model in detail from the following aspects: (1) Input and Embedding Layer; (2) Input Enhanced Component (IEC); (3) Hidden Layers; (4) Prediction Score and Learning; (5) Relationship with FM and HOFM. The overall structure of ILFN is shown in Fig. 1.

3.1 Input and Embedding Layer

Input Features. Firstly, user profiles and item attributes are represented as sparse vectors. Specifically: $\boldsymbol{x} = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_f]$, where f is the number of all feature fields and x_i denotes the i-th field. If the i-th field is categorical, x_i is a one-hot vector. Otherwise, it is a scalar value.

Embedding Layer. Since raw features are normally very sparse, a common practice is to project them into low-dimensional spaces by a classic embedding layer. The output of the embedding layer is a wide concatenated field vector: $\boldsymbol{e} = [\boldsymbol{e}_1, \boldsymbol{e}_2, ..., \boldsymbol{e}_f]$, where f denotes the number of all feature fields and $\boldsymbol{e}_i \in \mathbb{R}^k$ is the embedding of the i-th field, and k denotes the dimension of field embedding. By introducing the embedding layer, instances of various feature lengths can be transformed into unified embedding vectors with the same size $f \times k$.

3.2 Input Enhanced Component

The Input Enhanced Component (IEC) includes two parts: the BN-Logarithmic Transformation Layer (B-LTL) and Attention Layer.

BN-Logarithmic Transformation Layer. Figure 2 shows the structure of B-LTL. It contains multiple logarithmic neurons, which can adaptively learn arbitrary-order cross features. Due to the characteristics of logarithmic neurons, two positive-value operations are introduced in the preprocessing part: First, all embedding values should be converted to their absolute values. Second, a small positive value (such as 1e−7) is added to each zero embedding vector.

In B-LTL, the output of the i-th logarithmic neuron is:

$$y_i = exp(\sum_{j}^{f} w_{ij} ln \boldsymbol{e}_j) = \prod_{j}^{f} \boldsymbol{e}_j{}^{w_{ij}} = \boldsymbol{e}_1{}^{w_{i1}} \odot \boldsymbol{e}_2{}^{w_{i2}} \odot ... \odot \boldsymbol{e}_f{}^{w_{if}} \tag{1}$$

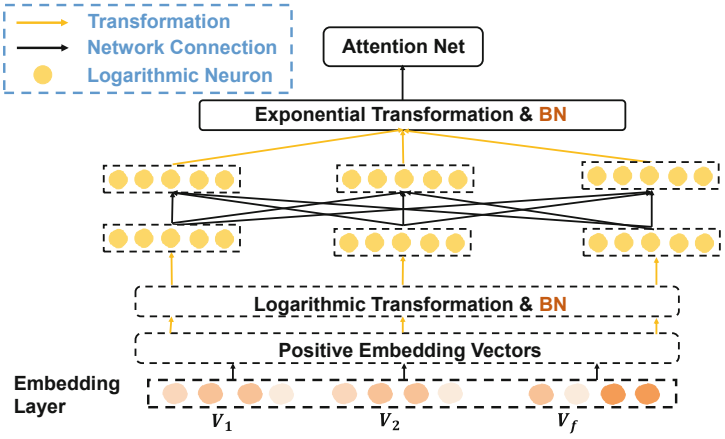


Fig. 2. The structure of the BN-Logarithmic Transformation Layer (B-LTL)

where w_{ij} denotes the power term of the i-th neuron on the j-th field, and $\odot$ denotes the bit-wise production operation. Note that the output $\boldsymbol{y}_i$ of the i-th logarithmic neuron can represent arbitrary-order cross features, and when w_i is adjusted to 0, the i-th feature will be filtered out, so it can learn useful combined features. For example, with w_{i1} and w_{i2} set to 1 and the rest set to 0 in Eq. 1, we have: $\boldsymbol{y}_i = \boldsymbol{e}_1 \odot \boldsymbol{e}_2$. This indicates that the output $\boldsymbol{y}_i$ of the i-th logarithmic neuron can represent a second-order cross-feature for the first two raw feature fields. Therefore, we can use multiple logarithmic neurons to obtain various feature combinations in any order as the output of this layer.

In addition, we perform BN on the output of logarithmic transformation and exponential transformation. We do this based on the following consideration. In real-world datasets, the feature embedding vectors are usually initialized close to zero. After logarithmic transformation, embedding vectors often involve minimal negative values, detrimental to optimizing parameters in successive layers. Since normalization can scale and convert the output to a normalized value, it is imperative to the training process of ILFN.

Attention Layer. To unify the representation, we rename vectors of the embedding layer as follow:

$$[\boldsymbol{y}_{n+1}, \boldsymbol{y}_{n+2}, ..., \boldsymbol{y}_{n+f}] = [\boldsymbol{e}_1, \boldsymbol{e}_2, ..., \boldsymbol{e}_f] \tag{2}$$

Then we use the attention mechanism to integrate the outputs of B-LTL and the embedding layer to show different importants of various features. The weights of different vectors can be defined as:

$$a'_i = \langle h, Relu(\boldsymbol{W}_a \boldsymbol{y}_i + b_a) \rangle \tag{3a}$$

$$a_i = \frac{exp(a')}{\sum_{i=1}^{n+f}(exp(a'_i))} \tag{3b}$$

where $\boldsymbol{W}_a \in \mathbb{R}^{(n+f)\times e}$, $h \in \mathbb{R}^e$, $b_a \in \mathbb{R}^e$ are model parameters, and e is the number of hidden units in attention network. Softmax function is used to normalize the attention score, and Relu function is used as an activation function, which empirically performs good performance.

After this layer, the outputs of B-LTL and the embedding vectors are concatenated together:

$$\boldsymbol{z}_0 = [a_1\boldsymbol{y}_1, a_2\boldsymbol{y}_2, ..., a_n\boldsymbol{y}_n, a_{n+1}\boldsymbol{y}_{n+1}, ..., a_{n+f}\boldsymbol{y}_{n+f}] \tag{4}$$

where n is the number of logarithmic neurons, and f denotes the number of total fields.

In this layer, $\boldsymbol{z}_0$ contains not only arbitrary-order cross features but also the original embedding information. Therefore, the distortion of original information caused by the logarithmic neural network has been compensated. More importantly, an attention mechanism is introduced to integrate embedding features and processed arbitrary-order cross features, which learns the importance of different parts and reduces the number of logarithmic neurons required for the model to achieve optimal results.

3.3 Hidden Layers

We stack several fully connected hidden layers upon the *Input Enhanced Component* to combine the formed feature interactions. So, $\boldsymbol{z}_0$ is fed into M hidden layers:

$$\begin{aligned} \boldsymbol{z}_1 &= \sigma(\boldsymbol{W}_1\boldsymbol{z}_0 + \boldsymbol{b}_1) \\ &..... \\ \boldsymbol{z}_M &= \sigma(\boldsymbol{W}_M\boldsymbol{z}_{M-1} + \boldsymbol{b}_M) \end{aligned} \tag{5}$$

where σ is the activation function, W_M denotes the weight matrix while b_M is the bias vector of the M-th layer. In addition, we also perform BN on the output of hidden layers. Since we use a multilayer neural network after IEC, performing BN on the hidden layer's output helps alleviate the covariance shift problem, which empirically leads to faster convergence and better model performance.

3.4 Prediction Score and Learning

Finally, we get the prediction results from the hidden layers: $\hat{y} = \boldsymbol{z}_M$, where $\boldsymbol{z}_M$ is the output of the hidden layer.

For binary classification tasks, a popular objective function is the Logarithmic Loss, defined as:

$$Logloss = -\frac{1}{N}\sum_{j=1}^{N}(y_j log(\sigma(\hat{y_j})) + (1-y_j)log(1-\sigma(\hat{y_j}))) \tag{6}$$

where $y_j \in \{0, 1\}$ is the true value of the j-th instance while $\sigma(\hat{y_j}) \in (0, 1)$ is the predicted value (here σ is the *sigmoid* function) and N is the number of all training instances.

3.5 Relationship with FM and HOFM

Suppose we remove IEC and only use the logarithmic neurons to produce second-order cross features and approximate a sum function for the hidden layers. In that case, our ILFN model will be downgraded to the FM model. Similarly, if the logarithmic neurons are set to deliver all feature interactions within a maximum order, and the hidden layer only simulates a summation function, our ILFN model will be functionally identical to HOFM.

4 Experiments

4.1 Experimental Settings

Datasets. We conducted experiments on four public benchmark datasets: *Criteo*[2], *Avazu*[3], *Movielens*[4] and *Frappe*[5]. We split them randomly by 8:1:1 for training, validation and test, respectively. **Criteo Dataset** is a famous industry benchmarking dataset containing the click records of 45 million users, which includes 13 numerical feature fields and 26 categorical feature fields. **Avazu Dataset** has 40 million mobile user behaviours, 24 feature fields containing the user features and item attributes. **Movielens Dataset** consists of two million users' records of movies. We focused on the personalized tag recommendation by converting each tag record (user ID, movie ID, tag) into a feature vector. **Frappe Dataset** contains the application usage logs of more than 90,000 users in different contexts.

Evaluation Metrics. We adopted two evaluation metrics in our experiments: **AUC** (Area Under ROC) and **Logloss** (cross entropy) [10]. Note that **a slight improvement in AUC or a decrease in Logloss at 0.001-level is conventionally regarded as being significant** for CTR prediction because it will bring a large increase in a company's revenue if the company has a large user base. **The proposal of some crucial models in CTR prediction tasks only increases the AUC by a few thousandths** [3,4,11,14,17,19,22].

Baselines. We compared our ILFN with four classes of the existing methods: (i) first-order models that just linearly sum up raw input features; (ii) second-order approaches that model first- and second-order feature interactions; (iii) high-order methods that can capture high-order cross features; (iv) ensemble models that involve DNNs as the counterpart. These models are shown in Table 1.

Implementation Details. We implemented our methods using Tensorflow[6]. We used Adam with a learning rate of 0.001 and a mini-batch size of 4096 in all models. We also fixed the network structure (i.e., three layers, 400-400-400) in

[2] http://labs.criteo.com/2014/02/.
[3] https://www.kaggle.com/c/avazu-ctr-prediction.
[4] https://grouplens.org/datasets/movielens/.
[5] http://baltrunas.info/research-menu/frappe.
[6] https://www.tensorflow.org/.

all models containing DNNs. The embedding size was 10 for all approaches. The default numbers of logarithmic neurons were 1000, 800, 600, and 500 for *Criteo*, *Avazu*, *Movielens*, and *Frappe* datasets. To avoid overfitting, we also performed early-stopping according to the AUC on the validation set. The maximum order in HOFM was set to 3. All the other hyper-parameters were tuned according to the validation set to achieve the best performance. We ran the CTR prediction experiments three times for each model and reported the average value as the final result.

4.2 Performance Comparison

We compared our ILFN model with several classical models featuring linear and high-order feature interactions. The results are shown in Table 1, from which we have three critical observations:

First, feature interactions play an essential role in the improvement of the CTR model's performance. For example, LR performed worse than other models with feature interactions. Based on the result that high-order models had larger AUC and smaller Logloss than first- and second-order models, higher-order cross features play a more critical role in CTR prediction tasks. Besides, DNNs can be employed to enhance the performance of cross feature-based models; those ensembled methods usually performed better than others.

Second, learning arbitrary-order cross features is more effective than learning fixed-order feature interactions. It can be verified by the fact that ILFN performed better than methods with the preset maximum orders on *Criteo*, *Avazu*,

Table 1. The performance comparison of different CTR models

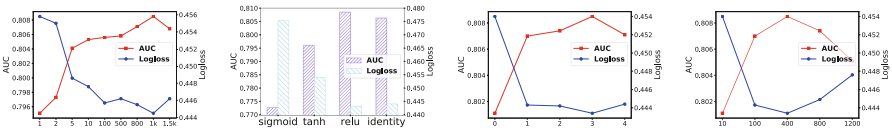
Model class	Model	Criteo		Avazu		Movielens		Frappe	
		AUC	Logloss	AUC	Logloss	AUC	Logloss	AUC	Logloss
First-order	LR [9]	0.7859	0.4635	0.7313	0.4066	0.9215	0.3080	0.9330	0.2858
Second-order	FM [20]	0.7933	0.4573	0.7496	0.3740	0.9388	0.2798	0.9641	0.2144
	FFM [16]	0.8045	0.4478	0.7501	0.3733	0.9427	0.2961	0.9687	0.2132
	AFM [14]	0.7954	0.4554	0.7454	0.3765	0.9299	0.2833	0.9640	0.2292
Ensembled	Wide&Deep [3]	0.8062	0.4453	0.7629	0.3744	0.9381	0.3309	0.9728	0.2038
	Deep&Cross [18]	0.8059	0.4463	0.7550	0.3721	0.9421	0.2789	0.9402	0.2809
	DCN V2 [2]	0.8072	0.4443	0.7557	0.3719	0.9481	0.2736	0.9772	0.1805
	DeepFM [6]	0.8026	0.4509	0.7535	0.3741	0.9424	0.3130	0.9720	0.2108
	AFN$^+$ [4]	0.8071	0.4450	0.7555	0.3717	0.9500	0.2585	0.9783	0.1762
	xDeepFM [12]	0.8069	0.4443	0.7535	0.3738	0.9448	0.2717	0.9738	0.2098
	FiBiNET [19]	0.8067	0.4445	0.7540	0.3737	**0.9513***	**0.2564***	0.9782	0.1761
High-order	HOFM [21]	0.7960	0.4550	0.7517	0.3756	0.9409	0.3090	0.9711	0.2138
	CrossNet [18]	0.7915	0.4585	0.7498	0.3756	0.9323	0.2931	0.9395	0.2838
	PNN [11]	0.8026	0.4509	0.7526	0.3737	0.9471	0.2789	0.9735	0.2011
	NFM [13]	0.7968	0.4536	0.7531	0.3762	0.9439	0.3003	0.9725	0.2080
	AFN [4]	0.8058	0.4457	0.7512	0.3731	0.9477	0.2753	0.9759	0.1784
	AutoInt [17]	0.8039	0.4476	0.7530	0.3761	0.9492	0.2611	0.9698	0.2354
	CIN [12]	0.8040	0.4473	0.7532	0.3757	0.9494	0.2601	0.9707	0.2339
	ILFN	**0.8085***	**0.4434***	**0.7573***	**0.3701***	0.9509	0.2571	**0.9792***	**0.1749***

and *Frappe* datasets. ILFN achieved the second-best performance on *Movielens* dataset. *Movielens* only contains three feature fields, and the benefit of ILFN to mine higher-order feature interactions may be marginal. As shown in Table 1, more straightforward methods had a better prediction effect, we conjecture that the predictions on *Movielens* dataset rely more on lower-order cross features, and the advantages of ILFN are thus restricted.

At last, the introduction of the *Input Enhanced Component* (IEC) can effectively and efficiently learn arbitrary-order cross features and identify useful cross features. Beyond that, it can fully use the original data distribution to build arbitrary-order combinations without destroying the initial data distribution. More importantly, ILFN can use IEC to model high-order feature combinations in a discriminative way with a faster convergence speed. Therefore, as a non-ensembled model, ILFN had better performance than those ensemble methods on *Criteo*, *Avazu*, and *Frappe* datasets.

4.3 Hyper-parameter Study

We then studied the impact of hyper-parameters on ILFN, including (1) the number of logarithmic neurons; (2) activation functions; (3) the depth of hidden layers; (4) the number of hidden neurons. Given that the results on the four datasets were similar, we only show the results on the *Criteo* dataset here.



(a) Number of logarithmic neurons (b) Activation functions (c) Depth of hidden layers (d) Number of hidden neurons

Fig. 3. Impact of network hyper-parameters on the performance of ILFN on the Criteo dataset

Number Logarithmic Neurons. Figure 3(a) demonstrates the impact of the number of logarithmic neurons on *Criteo* dataset. The appropriate number of logarithmic neurons on *Criteo* is 1000, which is smaller than 1500 in AFN. Thus, preserving original data information makes the model use fewer logarithmic neurons to achieve optimal performance. This indicates that the introduction of logarithmic neural networks has led to the loss of some crucial features, and the model needs more logarithmic neurons to make up for it.

Activation Functions. We compared the performance of different activation functions in ILFN, as shown in Eq. 5. According to Fig. 3(b), *ReLU* is indeed the most appropriate for neurons in ILFN.

Depth of Hidden Layers. Figure 3(c) demonstrates the effect of the depth of hidden layers. The performance of ILFN increased when the depth of the network was varied from 0 to 3. However, when the number of hidden layers was greater than 3, the model's performance degraded.

Number of Hidden Neurons. Figure 3(d) provides the results over different numbers of hidden neurons in hidden layers. The performance of our ILFN increased when the number of neurons was increased from 10 to 400. When the parameter size of the hidden layer exceeds 400, the performance begins to decline, which is caused by overfitting.

4.4 Ablation Study

ILFN strategically integrates the BN-LTL and the Attention Layer into a unified end-to-end model for learning arbitrary-order cross features. Unlike AFN, ILFN can model high-order feature interactions by IEC; therefore, we do not involve DNN in ILFN. To gain deep insights into ILFN, we conducted a series of ablation experiments. Table 2 presents the performance of ILFN *w.r.t.* specific components. From which we can reach the following conclusions: First, introducing the logarithmic neurons for adaptively learning arbitrary-order feature interactions is effective. Second, it makes sense to introduce the attention mechanism to joint embedding vectors with the output of the B-LTL. It retains the original distribution of the data. Lastly, DNN can bring additional improvement when the model use LNN to learn cross features. Our proposed IEC can effectively learn arbitrary-order feature interactions and identify valuable combinations. Moreover, by changing a_i in Eq. 4, IEC can build high-order feature interactions like DNN. This can be verified by adding DNN to ILFN, which has almost no extra effect. In summary, IEC, learning cross features in an expressive and discriminative manner, significantly boosts the performance for CTR prediction.

Table 2. The performance of different components in ILFN

Datasets	Criteo	
Model	AUC	Logloss
FM	0.7933	0.4573
HOFM	0.7960	0.4550
ILFN-no-att	0.8063	0.4452
ILFN-no-BN	0.8075	0.4446
LNN	0.8058	0.4457
LNN-DNN	0.8071	0.4450
ILFN-DNN	0.8084	0.4436
ILFN	**0.8085***	**0.4434***

Table 3. Relative improvements (RI) of extra components on Criteo and Frappe

Method	Component	Criteo RI	Frappe RI
FM	–	–	–
FFM	Field-aware	20.8	1.9
AFM	Attention	4.2	−6.7
DeepFM	DNN	14.0	−0.3
NFM	DNN	8.1	−5.9
DCN	CrossNet	24.0	5.1
AFN	LNN	25.4	2.4
ILFN	**IEC**	**30.4***	**10.4***

4.5 Component Comparison

In ILFN and other baseline models, each extra model component has its unique functionality: DNN models high-order feature interactions; Logarithmic neural network (LNN) focuses on the arbitrary-order cross features. To more intuitively observe the different performance improvement effects brought by various components. We first train all models without using the extra parts (i.e., simulating the standard FM model). Then we freeze feature embeddings and introduce the additional components only. Upon convergence, the relative improvements (RI) of each model on *Criteo* and *Frappe* are listed in Table 3. Note that values in Table 3 are a few thousandths because the improvement at 0.001-level is conventionally regarded as significant in the CTR estimation scenario. It is clear that with the help of the IEC, ILFN achieves 30.4‰ and 10.4‰ improvement on the two datasets, respectively. Our proposed IEC outperforms other widely recognised components by the public by learning arbitrary-order cross features in a discriminative way without destroying original data distribution.

5 Conclusion

In this paper, we presented a novel model called *Input Enhanced Logarithmic Factorization Network* (ILFN) for CTR prediction. It can effectively learn arbitrary-order cross features and identify beneficial feature interactions. Furthermore, it takes full advantage of the original embedding information in a discriminative way. The key to our method is the *Input Enhanced Component* (IEC), which joints the original embedding vectors and the output of the BN-Logarithmic Transformation Layer (B-LTL) via the Attention mechanism. In addition, we also revealed that ILFN could generalize to FM and HOFM by simplifying calculations. Extensive experiments on four real-world datasets confirmed that ILFN outperforms the representative and state-of-the-art deep learning approaches such as DCN, DeepFM, xDeepFM, and FiBiNET consistently for CTR prediction. We also analyzed the impact of hyper-parameters and conducted an ablation study, which demonstrates the effectiveness of our proposed components.

References

1. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: ICLR (2015)
2. Wang, R., Shivanna, R., et al.: DCN V2: improved deep & cross network and practical lessons for web-scale learning to rank systems. In: WWW, pp. 1785–1797 (2021)
3. Cheng, H.-T., Koc, L., et al.: Wide & deep learning for recommender systems. In: Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, pp. 7–10 (2016)
4. Cheng, W., Shen, Y., Huang, L.: Adaptive factorization network: learning adaptive-order feature interactions. In: AAAI, pp. 3609–3616 (2020)

5. McMahan, H.B., Holt, G., et al.: Ad click prediction: a view from the trenches. In: The 19th SIGKDD, 11–14 August 2013 (2013)
6. Guo, H., Tang, R., Ye, Y., Li, Z., He, X.: DeepFM: a factorization-machine based neural network for CTR prediction. In: IJCAI, pp. 1725–1731 (2017)
7. Graepel, T., Candela, J.Q., Borchert, T., Herbrich, R.: Web-scale Bayesian click-through rate prediction for sponsored search advertising in Microsoft's Bing search engine. Omnipress (2010)
8. Wang, Z., Zhang, R., Qi, J., Yuan, B.: DBSVEC: density-based clustering using support vector expansion. In: ICDE, pp. 280–291. IEEE (2019)
9. Lee, K., Orten, B., Dasdan, A., Li, W.: Estimating conversion rate in display advertising from past erformance data. In: KDD, pp. 768–776 (2012)
10. Wang, Z., Liu, L., Tao, D.: Deep streaming label learning. In: International Conference on Machine Learning (ICML) (2020)
11. Qu, Y., et al.: Product-based neural networks for user response prediction. In: ICDM, pp. 1149–1154. IEEE (2016)
12. Lian, J., Zhou, X., Zhang, F., Chen, Z., Xie, X., Sun, G.: xDeepFM: combining explicit and implicit feature interactions for recommender systems. In: KDD, pp. 1754–1763 (2018)
13. He, X., Chua, T.-S.: Neural factorization machines for sparse predictive analytics. In: SIGIR, pp. 355–364 (2017)
14. Xiao, J., Ye, H., He, X., Zhang, H., et al.: Attentional factorization machines: learning the weight of feature interactions via attention networks. In: IJCAI (2017)
15. Hines, J.W.: A logarithmic neural network architecture for unbounded non-linear function approximation. In: IEEE ICNN 1996 (1996)
16. Juan, Y., Zhuang, Y., Chin, W.-S., Lin, C.-J.: Field-aware factorization machines for CTR prediction. In: RecSys (2016)
17. Song, W., Shi, C., et al.: AutoInt: automatic feature interaction learning via self-attentive neural networks. In: CIKM (2019)
18. Wang, R., Fu, B., Fu, G., Wang, M.: Deep & cross network for ad click predictions. In: KDD, pp. 1–7 (2017)
19. Huang, T., Zhang, Z., Zhang, J.: FiBiNET: combining feature importance and bilinear feature interaction for click-through rate prediction. In: ACM RecSys (2019)
20. Rendle, S.: Factorization machines. In: 2010 IEEE International Conference on Data Mining, pp. 995–1000. IEEE (2010)
21. Blondel, M., Fujino, A., Ueda, N., Ishihata, M.: Higher-order factorization machines. In: NIPS (2016)
22. Yu, Y., Wang, Z., Yuan, B.: An input-aware factorization machine for sparse prediction. In: IJCAI (2019)
23. Zhang, W., Du, T., Wang, J.: Deep learning over multi-field categorical data. In: Ferro, N., et al. (eds.) ECIR 2016. LNCS, vol. 9626, pp. 45–57. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-30671-1_4