InteractionNN: A Neural Network for Learning Hidden Features in Sparse Prediction

Xiaowang Zhang* , **Qiang Gao** and **Zhiyong Feng**

College of Intelligence and Computing, Tianjin University, Tianjin 300350, China
Tianjin Key Laboratory of Cognitive Computing and Application, Tianjin 300350, China
{xiaowangzhang, qianggao, zyfeng}@tju.edu.cn

Abstract

In this paper, we present a neural network (InteractionNN) for sparse predictive analysis where hidden features of sparse data can be learned by multilevel feature interaction. To characterize multilevel interaction of features, InteractionNN consists of three modules, namely, nonlinear interaction pooling, layer-lossing, and embedding. Nonlinear interaction pooling (NI pooling) is a hierarchical structure and, by shortcut connection, constructs low-level feature interactions from basic dense features to elementary features. Layer-lossing is a feed-forward neural network where high-level feature interactions can be learned from low-level feature interactions via correlation of all layers with target. Moreover, embedding is to extract basic dense features from sparse features of data which can help in reducing our proposed model computational complex. Finally, our experiment evaluates on the two benchmark datasets and the experimental results show that InteractionNN performs better than most of state-of-the-art models in sparse regression.

1 Introduction

Sparse prediction (sparse predictive analysis), as an important regression problem in machine learning field [Pearl, 2018], applies machine learning to estimate the relationships between features and targets when features of data is in a sparse representation [Pan *et al.*, 2018]. Sparse data has features in a sparse representation. Formally, in [Rendle, 2011], we describe sparse data as follows: Suppose there is a dataset $D = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$, where n is the number of samples. Let $m(x_i)$ be the number of non-zero elements in the i-th feature vector x and $\overline{m}(x)$ be the average number of non-zero elements $m(x)$ of all vector $x \in D$. High sparsity ($\overline{m}(x_i) \ll n$) appears in many real world data like feature vectors of event transactions or text analysis. Recently, sparse prediction becomes more and more interesting since sparse data widely exists in our real world due to the multi-field category of practical data. For instance, consider some data with three multi-field categorical features:

[*Gender=Male*, *Country=China*, *Weekday=Thursday*],

by using one-hot encoding [Bayer *et al.*, 2017], we can convert it as a sparse representation as follows:

$$\underbrace{[0,1]}_{\text{Gender=Male}} \quad \underbrace{[0,0,0,1,...,0,0]}_{\text{Country=China}} \quad \underbrace{[0,0,0,1,0,0,0]}_{\text{Weekday=Thursday}}$$

Compared to regression in dense data, it is not easy to extract hidden features of sparse data due to highly dispersed distribution of each sample [Juan *et al.*, 2016]. To learn hidden features of sparse data, there are recently many existing works roughly classified as the following three classes: (1) manually feature engineering; (2) (non)Linear models for low-level features; and (3) neural networks for multilevel features. Manually feature engineering is firstly and directly employed in crafting hidden features such as bid-ask model and taxonomy-aware FE. Compared to manually feature engineering, (non)linear models and newly neural networks are proposed to extract hidden features automatically. Linear models and non-linear models mainly extract features in a low-level such as factorization machines (FM) [Rendle, 2012], field-aware factorization machines [Juan *et al.*, 2016], even with the help of feature engineering, such as logistic regression [Cheng *et al.*, 2010], follow the regularized leader (FTRL) [Mcmahan *et al.*, 2013]. Neural networks-based models are recently applied to extract hidden features of sparse data in a multiple level [Zhang *et al.*, 2016; Ying *et al.*, 2016; Xiao *et al.*, 2017]. Those neural networks-based (NN-based) models can learn high & low-level feature interactions by operating feature vectors [He *et al.*, 2017]. Wide&deep learning [Cheng *et al.*, 2016] presents a sum of feature vectors, product-based factorization machines (PNN) [Qu *et al.*, 2016] product of feature vectors. However, those existing NN-based models are limited to utilize only current feature in extracting higher level features [Zhang *et al.*, 2016]. Thus, it is not easy to characterize the transitivity and reusability of features due to history features missing.

In this paper, we propose a novel neural network InteractionNN for sparse predictive analysis for learning multilevel feature interactions by utilizing history features to capture transitivity and reusability of features. The main contributions of this paper are summarized as follows:

- We propose a shortcut connection-based hierarchical structure (NI pooling) to learn low-level feature inter-

*Corresponding Author

actions by utilizing history features. Experimental results show that NI pooling outperforms FM with a 3.0% improvement, and performs better than BI-Interaction pooling in NFM with a 4.4% improvement.
- We design a fully connected feed-forward neural network (Layer-lossing) to learn the higher level feature interactions from all higher level features in hidden layer. Experimental results show that layer-lossing performs better than general NN with a 0.9% improvement.
- We present a hybrid embedding approach to learn basic dense features from sparse data where two embedding methods are employed to linear and nonlinear features respectively. Finally, our experiments on Frappe and MovieLens show that InteractionNN performs better than FM with a 6.3%, DeepFM with a 5.8%, and NFM with a 0.8% improvement respectively.

2 Related Works

In this section, we compare our approach to existing works.

Neural Network-based Mmodels [Chen *et al.*, 2016] presents a DeepCTR model based on neural network by concatenating image feature vectors at embedding layer. Wide & deep learning (WDL) [Cheng *et al.*, 2016] presents a merging strategy for training jointly linear model and deep learning. Compared to those models with the help of feature engineering, InteractionNN can constructs features automatically. Factorisation-machine supported neural networks (FNN) [Zhang *et al.*, 2016] present feature division based on field to reduce dimension of input, meanwhile combining embedding vectors to extract basic dense features. Product-based neural network (PNN) [Qu *et al.*, 2016] presents a product layer based on product of feature vectors to learn basic dense features. Compared to those models extracting only high-level feature interactions, InteractionNN can also extract low-level feature interactions which is useful to characterize the semantics of sparse data.

Hierarchical representation model (HRM) [He *et al.*, 2016] and neural network-based collaborative filtering (NCF) [Liu *et al.*, 2018] capture the feature interactions via a simply average of embedding vectors. DeepFM [Guo *et al.*, 2017] combines FM and deep learning, which can obtain feature interactions via inner product and concatenate of feature vectors. Neural factorization machines (NFM) [He *et al.*, 2017] proposes a BI-Interaction, which computes element-wise product and sum of feature interactions. Attentional factorization machines (AFM) [Xiao *et al.*, 2017] learns feature interaction contribution via attention mechanism by calculating element-wise product and weighted sum of feature interactions.

CCPM [Liu *et al.*, 2015], based on CNN, can learn some feature interactions between local features by convolution kernel. Deep crossing [Ying *et al.*, 2016] proposes multiple residual units based on ResNet [He *et al.*, 2016] to learn high-level feature interactions. Deep cross network (DCN) [Wang *et al.*, 2017] presents a novel cross network to deepen the low-level feature interactions learned via product of feature interactions. Compared to those models constructing higher level features depending on only current features, InteractionNN can capture the transitivity and reusability of features.

Other Models The family of FM [He *et al.*, 2016; Liu *et al.*, 2018] can learn feature interactions via inner product of embedding vectors. FTRL [Mcmahan *et al.*, 2013] presents a per-coordinate learning rates which has excellent sparsity and convergence properties. Feature engineering [Shi *et al.*, 2014], such as bid-ask model and taxonomy-aware FE, manually design some shallow hidden features. Compared to those classical machine learning approaches to extracting shallow features, InteractionNN can also extract high-level features.

3 Overview of InteractionNN

In this section, we introduce the framework of InteractionNN model that captures multilevel feature interactions by extracting information layer by layer during sparse data modeling. InteractionNN mainly contains three modules, namely, nonlinear interaction pooling, Layer-lossing, and embedding, as shown in Figure 1.

Next, we introduce the three modules of InteractionNN as follows:
- Embedding is to extract basic dense features from sparse features of data which can help in reducing our proposed model computational complex.
- NI pooling is a hierarchical structure and, by shortcut connection [Huang *et al.*, 2017], which constructs low-level feature interaction from basic dense features in Embedding.
- Layer-lossing is a feed-forward neural network where high-level feature interactions can be learned from low-level feature interactions extracted in NI pooling via correlation of all layers with target.

Besides, the linear model of InteractionNN is to learn linear features fused together with nonlinear features extracted in Layer-lossing in the combination layer for regression.

4 InteractionNN for Sparse Prediction

In this section, we introduce InteractionNN in detail.

4.1 Embedding

Embedding contains two layers: the 1st converts the sparse feature vector into dense vector representation and the 2nd converts the shared input into concentrated matrix expression.

Formally, let F denote the feature size, K the embedding size, and FS the number of feature values, the two layers $x_{em_1} \in R^F$ and $x_{em_2} \in R^{F \times K}$ are defined as follows:

$$x_{em_1} = x_{in} \otimes vec_{em_1}, \qquad x_{em_2} = x_{in} \oslash mat_{em_2}. \quad (1)$$

Here $vec_{em_1} \in R^{FS}$ is vector weight parameter, and $mat_{em_2} \in R^{FS \times K}$ is matrix weight parameter. Note that $\otimes$ and $\oslash$ are used to perform parallel lookups on the list of tensor in x_{in} [Liu *et al.*, 2018]. Intuitively, x_{em_1} and x_{em_2} represent a dense feature vector and a dense feature matrix, where an original sparse feature vector x_{in} is input. Note that x_{em_2} is utilized to learn high-level nonlinear feature interaction while x_{em_1} is used to learn low-level linear feature.

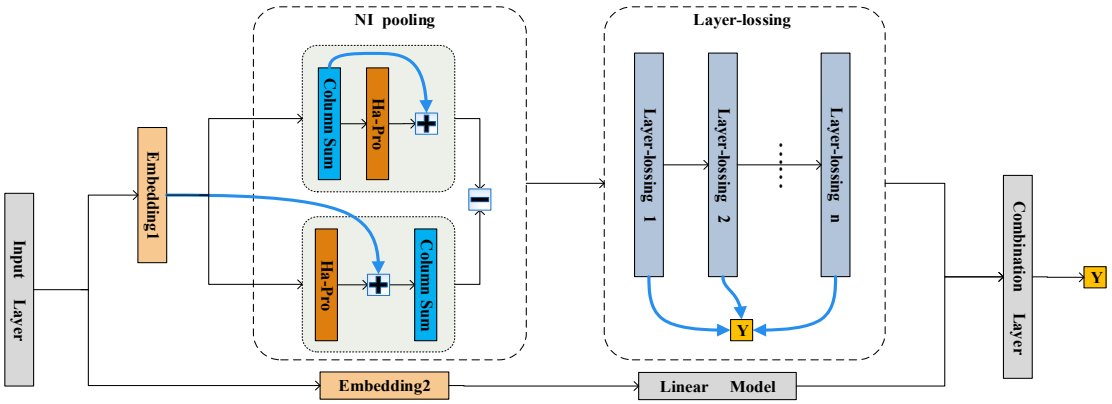

Figure 1: The framework of InteractionNN

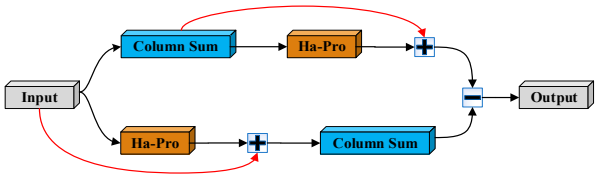

Figure 2: The structure of NI pooling

4.2 Nonlinear Interaction Pooling (NI pooling)

NI pooling is a hierarchical extractor (shown in Figure 2) where feature vectors (low-level features) are extracted from feature matrix (sparse features) generated in embedding. NI pooling consists of two elementary unary operator: *column sum* (denoted by $\oplus$) and *hadamard product* (denoted by $\odot$).

Formally, the two operators *column sum* ($\oplus$) and *Ha-Pro* ($\odot$) are defined as follows: Let $[M_{ij}]_{n\times m}$ be a $n \times m$ matrix.

- $\oplus\ [M_{ij}]_{n\times m} = [M_i]_m = [\ \sum_{i=1}^{n} M_{ij}\ ]_m$.
- $\odot\ [M_{ij}]_{n\times m} = [M_{ij}^2]_{n\times m}$.

Thus NI pooling is formally expressed as the following: where x_p is the output of NI pooling,

$$x_p = [\odot[\oplus(x_{em_2})] + \oplus(x_{em_2})] - \oplus[\odot(x_{em_2}) + x_{em_2}], \quad (2)$$

Equation (2) can be equivalently simplified to the following:

$$x_p = 2 \sum_{i,k=1,i\neq k}^{n} (M_{ij} \cdot M_{kj}) \propto \sum_{i,k=1,i\neq k}^{n} (M_{ij} \cdot M_{kj}).$$

[He *et al.*, 2017] converts a feature matrix into a feature vector by introducing a latent semantic space. In this paper, we consider shortcut connection to capture the transitivity and reusability of features in pooling.

As a summary of NI pooling, we conclude the followings:

- *Column sum* ($\oplus$) is to obtain statistical information of the embedding matrix.
- *Hadamard product* ($\odot$) is to strengthen features based on matrix factorization. It splits embedding matrix into two types of matrices. The first represents each row vector separately, and the second is a diagonal matrix which can strengthen features. Finally, it merges the feature vectors together to a matrix. For instance,

$$\odot \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} = \begin{bmatrix} a_1 & a_2 \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} a_1 & 0 \\ 0 & a_2 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ a_3 & a_4 \end{bmatrix} \cdot \begin{bmatrix} a_3 & 0 \\ 0 & a_4 \end{bmatrix} = \begin{bmatrix} a_1^2 & a_2^2 \\ a_3^2 & a_4^2 \end{bmatrix}$$

  Intuitively speaking, each matrix is decomposed into several sparse matrices. It makes the implicit relationships between features easier to learn.
- "+" is a sum, which adds the previous feature matrices (or vectors) to current feature matrices (or vectors). In Figure 2, red arrow above and "+" indicates that the feature of "column sum" is passed to the current layer, and red arrow below and "+" denotes that the feature of "input" is passed to the current layer in NI pooling. As a shortcut connection, it can use history features to capture the transitivity and reusability of features.
- "-" is a subtraction, which subtracts the results of two vectors after two conversion methods. It can reinforce important features and weaken unimportant features.

In a short, each operation of NI pooling is interpretable in mathematic and then it is also reliable to learn effective low-level feature interactions. Besides, the computational complexity of NI pooling is $O(n)$. It is easy to update model parameters with general optimizer and new samples.

Batch Normalization Batch Normalization (BN) [Ioffe *et al.*, 2015] is an acceleration strategy for neural network, which can tune the distribution of hidden layer to be the same during training process. In order to reduce "internal covariate shift", we introduce BN in NI pooling to detect the performance of NI pooling and whether BN can accelerate the convergence speed of the pooling. We will clarify the impact of BN on NI pooling by through experiments in Section 5.3. We utilize BN in NI pooling to make the same distribution of different hidden layers. Essentially, it is the standardization of information at each layer level. Furthermore, BN can better extract implicit relationships between features.

Dropout Dropout [Srivastava *et al.*, 2014] is a regularization strategy to prevent over-fitting for neural network, which can randomly select some features into training process in each iteration. When neural network propagates forward, a neuron stops working with a certain probability p. It can enhance the generalization ability of model, because it does not rely too much on local features. In InteractionNN, we introduce dropout on NI pooling to avoid over-fitting. More importantly, we can find much different crucial information in each iteration because of randomness of selecting features. In Section 5.3, our experimental results illustrate the impact of dropout regularization. Of course, we also employ BN and dropout strategy on Layer-lossing to learn significant feature interactions and prevent over-fitting.

Layer-lossing Layer-lossing is a modified neural network structure, which is responsible for learning high-level and nonlinear interactions between features (as shown in Figure 3). Obviously, the linear model plays a key role to model the relevance between each hidden layer and final target. By obtaining information of each hidden layer, it can get more important features. Meanwhile, the model allows the loss of each hidden layer to be added to the objective loss function. In Section 4.3, we will introduce the objective loss function in detail.

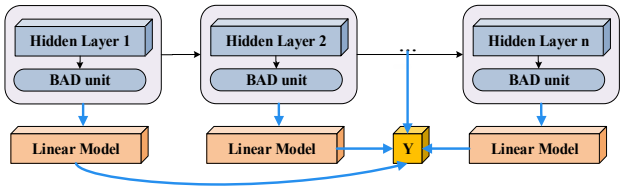

Figure 3: The architecture of the Layer-lossing.

Layer-lossing is to extract higher-level feature interactions from each hidden layer, considering the transitivity and reusability of features. Formally, it is defined as follows:

$$\begin{aligned} h_1 &= bad(w_1 \times x_{ni} + b_1) \\ h_2 &= bad(w_2 \times h_1 + b_2) \\ &\cdots\cdots \\ h_L &= bad(w_n \times h_{L-1} + b_L) \end{aligned} \tag{3}$$

where L is the number of hidden layers, h_i is the output of the $i\text{-}th$ hidden layer. w_i and b_i denote the connection matrix and bias, respectively. The bad unit means batch normalization, activation function and dropout, respectively. Among them, the role of BN and dropout is described in Section 4.2. The activation function represents the functional relationship between previous layer and current layer. We extract complex nonlinear feature interactions by setting a nonlinear activation at each hidden layer. It can make Layer-lossing more effective. In this paper, we employ three nonlinear activation functions (sigmoid, tanh, and relu) to improve the network's ability to predict. In Section 5.4, experimental results show the effects of different activation functions on performance.

The linear model is simple linear regression, which can predict the output of each hidden layer as accurately as possible. Formally, the definition of linear regression is as follow:

$$\overline{y} = w_i \times h_i + b_i \tag{4}$$

where w_i and b_i denote the weight and bias vector, and $\overline{y}$ is predicted continuous value. Compared with the general neural network, it can learn more significant feature interactions from hidden layers.

4.3 Others

The linear model in Figure 1 is wide component in InteractionNN, which can learn essential and linear features. The combination layer sums the output of wide component and layer-lossing. For a regression problem, the predicted output of the model is:

$$\overline{y} = h_L + x_{wide} + b, \tag{5}$$

where $\overline{y}$ is predicted continuous value, b denotes bias, h_L and x_{wide} denote the output of layer-lossing and wide component.

In this paper, we use the square loss as the model's loss function, and calculate the loss of each hidden layer in layer-lossing:

$$\mathcal{L} = \sum_{x \in D} (\overline{y}(x) - y(x))^2 + \sum_{i=1}^{L} \sum_{x \in D} (h_i(x) - y)^2, \tag{6}$$

where L denotes the numbers of hidden layer, and $h_i(x)$ is predicted target value of the $i\text{-}th$ hidden layer.

5 Experiments and Evaluation

We implement InteractionNN by using TensorFlow and construct three sets of experiments to evaluate our proposed model on learning feature interactions. The first experiment aims to evaluate the effectiveness of InteractionNN by comparing to the state-of-the-art model. The second aims to appraise the ability of NI pooling to learn low-level feature interactions. The third aims to evaluate the availability of Layer-lossing for learning high-level feature interactions compared with general neural network.

5.1 Experiment Setup

We perform experiments with two benchmark datasets:

- Frappe: It includes 96,203 applications usage records for various contexts. It results in 5,382 features via one-hot encoding. In each sample, only 10 feature values are 1.
- MovieLens: It contains 668,953 applications. It produces 90,445 features via one-hot encoding. In each sample, it has only 3 non-zero elements.

We randomly split the datasets into three parts: 70% (training), 20% (validation), and 10% (test). To evaluate the performance, we employ two evaluation metrics: RMSE (Root Mean Square Error) and AUC (Area Under ROC). We compare InteractionNN with traditional machine learning methods and state-of-the-art deep learning models: LR, FM, DeepFM, PNN, and NFM.

The loss functions of all models are square loss for a fair comparison. The optimizer is mini-batch Adagrad, which

can adaptively tune learning rate. The initial learning rate is searched in [0.001, 0.005, 0.01, 0.05]. The batch size of both datasets is 128. Except for the size of attention factor is 64, other embedding size is set to 256. In order to avoid over-fitting, we use dropout is searched in [0.2, 0.4, 0.6, 0.8] in both NI pooling and Layer-lossing. Besides, we utilize an early stopping mechanism based on the validation error, where InteractionNN stops iterating if the validation error continuously rising in five epochs.

5.2 Performance Comparison

In this section, we compare the performance of different methods on the test set (as shown in Table 1). For InteractionNN, we only set a hidden layer with bad unit. We have three major observation as follows:

Method	Frappe		MovieLens	
	RMSE	AUC	RMSE	AUC
LR	0.5832	0.9358	0.5979	0.9255
FM	0.3766	0.9799	0.4649	0.9578
PNN	0.3550	0.9741	0.4587	0.9521
DeepFM	0.3782	0.9631	0.4543	0.9582
NFM	0.3095	0.9810	0.4443	0.9599
InteractionNN	**0.3057**	**0.9838**	**0.4383**	**0.9617**

Table 1: Test error of different models.

- Firstly, InteractionNN performs best compared to the other popular models. To be specific, InteractionNN outperforms FM with a 6.3% improvement. InteractionNN betters the current best model, NFM, with a 0.8% improvement. This confirms the importance of features' transmissibility and reusability. By extracting features layer by layer and shortcut connection, InteractionNN can learn more effective feature interactions.
- Secondly, DeepFM combines FM model and general neural network, which learn low-level feature interactions and high-level feature interactions, respectively. As can be seen from Table 1, DeepFM and FM performs similarly. It demonstrates that reasonable low-level feature interactions has a crucial role. Meanwhile, it is our purpose of designing Layer-lossing to learn high-level feature interactions based on important low-level feature interactions.
- Finally, shallow models (NFM, InteractionNN) can performs better than deep learning models (PNN, DeepFM) with a 5.4% relative improvement. This demonstrates that the better low-level feature interactions is the premise of extracting high-level feature interactions. Meanwhile, it can simplify the learning process of deep learning to extract high-level feature interactions.

5.3 Evaluating NI Pooling

In this section, we will pay attention to analyze the effect of NI pooling for learning low-level feature interactions. In order to compare the performance of NI pooling to extract low-level feature interactions separately, we remove wide component and Layer-lossing from InteractionNN.

Firstly, we compare different methods for learning low-level feature interactions on the test set (as shown in Table 2). The main observations are summarized as follows:

Method	Frappe		MovieLens	
	RMSE	AUC	RMSE	AUC
None(LR)	0.5832	0.9358	0.5979	0.9255
Inner-product	0.3766	0.9799	0.4649	0.9578
Outer-product	0.4732	0.9585	0.4953	0.9482
Ab pooling	0.446	0.9728	0.5370	0.9474
BI pooling	0.3745	0.9795	0.4945	0.9533
NI pooling	**0.3390**	**0.9816**	**0.4586**	**0.9598**

Table 2: Test error of different methods for learning low-level feature interactions. (Ab pooling is attention-based pooling in AFM)

- NI pooling achieves the best performance compared to the other methods. Specially, NI pooling improves performance by 4.8% and 4.0% over the most advanced method (BI pooling) on Frappe and MovieLens, respectively. It points that NI pooling can effectively learn feature interactions, and the shortcut connection can automatically learn extra information.
- NI pooling and BI pooling performs much better than inner product and outer product with a great improvement on Frappe and MovieLens, respectively. It indicates that the implicit relationship between features is quite complicated, and simple product of feature vectors can only extract shallow feature interactions.
- All methods for learning feature interactions are much better than LR. Moreover, LR relies heavily on feature engineering. Designing effective method to learn more important feature interactions is a prospective research direction to ease humans burden.

Besides, we also explore the effect of dropout, batch normalization and embedding size (as shown in Figure 4). We have the following observations:

- To avoid over-fitting, we use dropout regularization method in NI pooling (as shown in Figure 4(a)). We can see that dropout ratio affects the performance of InteractionNN. Specifically, the better optimal dropout ratio is 0.4 and 0.6 on Frappe and MovieLens, respectively. This verifies that dropout is obligatory to NI pooling, which increases level of feature interactions. Due the randomness of feature selection, NI pooling can combine different features to learn various feature interactions.
- Figure 4(b) and 4(c) indicate the validation error of each epoch of NI pooling with and without BN on Frappe and MovieLens, respectively. The dropout radio is set to 0.4 and 0.6, and the learning rate is set to 0.001 and 0.005, respectively. Obviously, BN can speed up the iteration, which accelerate the convergence of NI pooling. Specially, on Frappe, when BN is applied, the validation error of epoch 20 is even lower than that of the same epoch

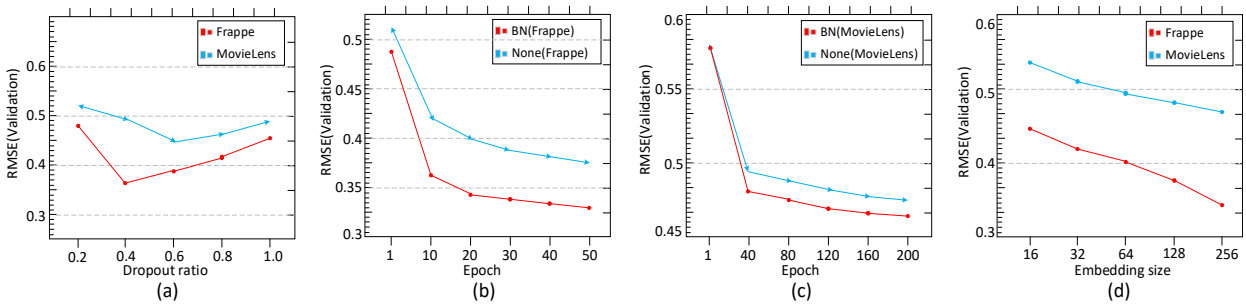

Figure 4: The performance of NI pooling using different parameters.

without BN with a 0.05 reduction.

- Figure 4(d) describes the validation error of different embedding sizes. The dropout radio is set to 0.4 and 0.6, and the learning rate is set to 0.001 and 0.005, respectively. Meanwhile, BN is also applied. We can see that the larger the embedding size, the better NI pooling performs. This confirms that the more the number of features, the better the implicit relationship between features is extracted.

5.4 Impact of Layer-lossing

Layer-lossing of InteractionNN is crucial for learning higher-level feature interactions. In order to show the effect of Layer-lossing on the result, we use 1, 3, and 5 hidden layers respectively. We call the model with i hidden layers LL-i, and the general neural network is named NN-i.

NN	Frappe		Frappe	
	RMSE	AUC	RMSE	AUC
NI pooling	0.3390	0.9816	0.4586	0.9598
NN-1	0.3110	0.9810	0.4413	0.9606
LL-1	**0.3057**	**0.9838**	**0.4383**	**0.9617**
NN-3	0.3292	0.9757	0.4848	0.9507
LL-3	0.3187	0.9769	0.4732	0.9550
NN-5	0.3338	0.9740	0.5361	0.9500
LL-5	0.3256	0.9752	0.4972	0.9502

Table 3: Test error of Layer-lossing and general neural network.

In order to understand how the depth of neural network affects performance and whether InteractionNN performs better than general neural network, we employ LL-1, LL-3, and LL-5 respectively (as shown in Table 3). To avoid the influence of NI pooling, we fix hyper-parameters in NI pooling. For instance, the learning rate is set to 0.001, the dropout ratio in NI pooling is set to 0.4, and the embedding size is set to 256. Meanwhile, the dropout ratio in each hidden layer is the same. Obviously, when the number of hidden layers is the same, Layer-lossing outperforms better than general neural network. It shows that the features retained in the hidden layer are important.

There is an interesting appearance that the more hidden layers, the worse the performance. We believe that the gradient in back propagation is more likely to disappear under sparse settings. Although we also try other successful models, the performance of InteractionNN isn't improved. This indicates low-level feature interactions play a key role for Layer-lossing to capture high-level feature interactions.

In addition, Table 4 shows the test error of LL-1 using different activation functions in hidden layer. First and foremost, we observe that by using different nonlinear activations, LL-1 performs better than NI pooling, respectively. This indicates LL-1 can learn the higher-level interactions between features via nonlinear activation function and modeling the relationship between each hidden layer and final target. Among the three nonlinear activation functions, Relu and Sigmoid perform better on Frappe and MovieLens. Note that the nonlinear feature interactions are different for different datasets.

Activation Function	Frappe		Frappe	
	RMSE	AUC	RMSE	AUC
LL-0(NI pooling)	0.3390	0.9838	0.4586	0.9585
LL-1-sigmoid	0.3121	0.9821	**0.4383**	**0.9617**
LL-1-tanh	0.3135	0.9826	0.4458	0.9579
LL-1-relu	**0.3057**	**0.9836**	0.4403	0.9599

Table 4: Test error of InteractionNN using different activation functions in hidden layers

6 Conclusion

In this paper, we present a novel neural network model InteractionNN for sparse prediction, where hidden features of sparse data can be learned by multilevel feature interactions. InteractionNN model considers the transitivity and reusability of features. It provides a promising research direct to improve NN-based models even general machine learning model.

Acknowledgments

This paper is supported by National Key Research and Development Program of China (2017YFC0908401) and National Natural Science Foundation of China (NSFC) (61672377).

References

[Bayer *et al.*, 2017] Immanuel Bayer, Xiangnan He, Bhargav Kanagal, and Steffen Rendle (2017). A generic coordinate descent framework for learning from implicit feedback. In *Proc. of WWW'17*, pp.1341–1350.

[Chen *et al.*, 2016] Junxuan Chen, Baigui Sun, Hao Li, Hongtao Lu, and Xian-Sheng Hua (2016). Deep ctr prediction in display advertising. In *Proc. of MM'16*, pp.811–820.

[Cheng *et al.*, 2010] Haibin Cheng and Erick Cantú-Paz (2010). Personalized click prediction in sponsored search. In *Proc. of WSDM'10*, pp.351–360.

[Cheng *et al.*, 2016] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah (2016). Wide & deep learning for recommender systems. In *Proc. of RecSys'16*, pp.7–10.

[Guo *et al.*, 2017] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He (2017). DeepFM: A Factorization-machine based neural network for CTR prediction. In *Proc. of IJCAI'17*, pp:1725–1731.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016). Deep residual learning for image recognition. In *Proc. of CVPR'16*, pp.770–778.

[He *et al.*, 2016] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua (2016). Fast matrix factorization for online recommendation with implicit feedback. In *Proc. of SIGIR'16*, pp.549–558.

[He *et al.*, 2017] Xiangnan He and Tat-Seng Chua (2017). Neural factorization machines for sparse predictive analytics. In *Proc. of SIGIR'17*, pp.355–364.

[Huang *et al.*, 2017] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger (2017). Densely Connected Convolutional Networks. In *Proc. of CVPR'17*, pp.2261–2269.

[Ioffe *et al.*, 2015] Sergey Ioffe and Christian Szegedy (2015). Batch normalization: accelerating deep network training by reducing internal covariate shift. In *Proc. of ICML'15*, pp.448–456.

[Juan *et al.*, 2016] Yu-Chin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin (2016). Field-aware factorization machines for ctr prediction. In *Proc. of RecSys'16*, pp.43–50.

[Liu *et al.*, 2015] Qiang Liu, Feng Yu, Shu Wu, and Liang Wang (2015). A convolutional click prediction model. In *Proc. of CIKM'15*, pp.1743–1746.

[Liu *et al.*, 2018] Han Liu, Xiangnan He, Fuli Feng, Liqiang Nie, Rui Liu, and Hanwang Zhang (2018). Discrete factorization machines for fast feature-based recommendation. In *Proc. of IJCAI'18*, pp.3449–3455.

[Liu *et al.*, 2018] Yang Liu, Linfeng Li, and Jun Liu (2018). Bilateral neural embedding for collaborative filtering-based multimedia recommendation. *Multimedia Tools Appl.*, 77(10):12533–12544.

[Mcmahan *et al.*, 2013] H. Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, Sharat Chikkerur, Dan Liu, Martin Wattenberg, Arnar Mar Hrafnkelsson, Tom Boulos, and Jeremy Kubica (2013). Ad click prediction: A view from the trenches. In *Proc. of KDD'13*, pp.1222–1230.

[Pan *et al.*, 2018] 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 *Proc. of WWW'18*, pp.1349–1357.

[Pearl, 2018] Judea Pearl (2018) Theoretical impediments to machine learning with seven sparks from the causal revolution. In *Proc. of WSDM'18*, Article 3.

[Qu *et al.*, 2016] Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang (2016). Product-based neural networks for user response prediction. In *Proc. of ICDM'16*, pp.1149–1154.

[Rendle, 2011] Steffen Rendle (2011). Factorization machines. In *Proc. of SIGIR'11*, pp.635–644.

[Rendle, 2012] Steffen Rendle (2012). Factorization machines with libFM. *ACM TIST*, 3(3):57:1–57:22.

[Srivastava *et al.*, 2014] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever and Ruslan Salakhutdinov (2014). Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res*, 15(1):1929–1958.

[Shi *et al.*, 2014] Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, Joaquin Quiñonero Candela (2014). Practical lessons from predicting clicks on ads at facebook. In *Proc. of ADKDD'14*, pp.5:1–5:9.

[Wang *et al.*, 2017] Ruoxi Wang, Bin Fu, Gang Fu and Mingliang Wang (2017). Deep & cross network for AD click predictions. In *Proc. of ADKDD'17*, pp.12:1–12:7.

[Xiao *et al.*, 2017] 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. In *Proc. of IJCAI'17*, pp.3119–3125.

[Ying *et al.*, 2016] Ying Shan, T. Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and J. C. Mao (2016). Deep crossing: web-scale modeling without manually crafted combinatorial features. In *Proc. of SIGKDD'16*, pp.255–262.

[Zhang *et al.*, 2016] Weinan Zhang, Tianming Du, and Jun Wang (2016). Deep learning over multi-field categorical data: A case study on user response prediction. In *Proc. of ECIR'16*, pp.45–57.