Computers, Materials & Continua CMC, vol.60, no.3, pp.1003-1013, 2019

A Recommendation System Based on Fusing Boosting Model and DNN Model

Aziguli Wulam[1, 2], Yingshuai Wang[1, 2], Dezheng Zhang[1, 2, *], Jingyue Sang[3] and Alan Yang[4]

Abstract: In recent years, the models combining traditional machine learning with the deep learning are applied in many commodity recommendation practices. It has been proved better performance by the means of the neural network. Feature engineering has been the key to the success of many click rate estimation model. As we know, neural networks are able to extract high-order features automatically, and traditional linear models are able to extract low-order features. However, they are not necessarily efficient in learning all types of features. In traditional machine learning, gradient boosting decision tree is a typical representative of the tree model, which can construct new features related before and after tree. Convolutional neural networks have a better perception of local features. In this paper, we take advantage of convolutional networks to capture the local features. The features are constructed by the node leaf of gradient boosting decision tree. This paper employs the tree leaf node to mine the user behavior path features, and uses the deep model to extract the user abstract features. Based on a Kaggle competition, our model performs better in the test data than any other model.

Keywords: Deep tree joint network, gradient boosting decision tree, convolution neural network, recommendation systems, attention network.

1 Introduction

In the field of e-commerce, the prediction of Click Though Rate (CTR) is an important part of the recommendation engine. One CTR application is descripted as follows: The input query is a set of user's contextual information, and the output is a ranked list of items. Another application is as follows: When a user enters the channel, the recommendation is to find the relevant items in a database and then rank the items based on certain objects. It is difficult to capture user's diverse interests effectively from a variety of historical behaviors. In order to tackle this challenge, deep interest network [Zhou, Zhu, Song et al.

[1] School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing, 100083, China.

[2] Beijing Key Laboratory of Knowledge Engineering for Materials Science, Beijing, 100083, China.

[3] Industrial Engineering, Shanghai University, Shanghai, 200444, China.

[4] Amphenol Assemble Tech, Houston, TX 77070, US.

[*] Corresponding Author: Dezheng Zhang. Email: zdzchina@126.com.

(2017)] with a local activation unit is designed to learn the user interests, and applied to the online display advertising system in Alibaba. Considering the changing of the environment and users' interests, a novel model named deep interest evolution network [Zhou, Mou, Fan et al. (2018)] is proposed, and has been deployed in the display advertisement system of Taobao. In the recent time, some researchers propose an FGCNN model (Feature Generation by Convolutional Neural Network) [Liu, Tang, Chen et al. (2019)], which includes feature generation and deep classifier. Another model named xDeepFM [Lian, Zhou, Zhang et al. (2018)] aims to generate feature interactions in an explicit fashion. xDeepFM contains a Compressed Interaction Network (CIN), which shares some functionalities with convolutional neural network and recurrent neural networks. Logistic regression [Peng, Lee and Ingersoll (2002)] is widely used because it is simple, scalable and interpretable. Wide & Deep [Cheng, Koc, Harmsen et al. (2016)] presents learning jointly trained by wide linear models and deep neural networks. Wide & Deep combines the benefit of memory with the advance of generalization for recommender systems. Deep & Cross network [Wang, Fu, Fu et al. (2017)] keeps the benefits of a DNN model, and introduces a novel cross network that is more efficient in learning certain bound-degree feature interactions. Factorization machine [Rendle (2010, 2012)] maps sparse features into low dimensional dense vectors, and learns feature interactions from vector inner products. Field-aware factorization machines [Zhang, Liu and Zhang (2018); Juan, Zhuang, Chin et al. (2016)] set a field for each feature. Product-based neural networks [Hsiao and Huang (2002)] with an embedding layer to capture interactive patterns between categories, and further feed forward network to explore high-order feature interactions. All models above, including linear logistic regression, non-linear gradient boosting decision trees and factorization machines, have been proved better on low-order features, and deep neural network [LeCun, Bengio and Hinton (2015)] performs better on high-order features. However, these models mostly depend on feature engineering so as to capture feature accurately. In this paper, it is possible to construct an end-to-end learning model, with no need of feature engineering besides raw features.

2 Method

Feature interactions plays an important role in recommendation system. Some feature interactions can be easily understood and engineered by experts, such as the product recommendation scene: girls like to browse cosmetics while boys like to browse basketball. However, other feature interactions is hidden in data, such as the classic association rule: "diaper and beer" is mined from a large amount of data and difficult to discover. Even for easily understanding feature interactions, it is unlikely to model them well, especially when the raw feature is huge. In order to interpret feature visually, general linear model FTRL [Mcmahan, Holt, Sculley et al. (2013)] is proposed, and has shown good performance in some practice. However, a general linear model lacks the ability to learn feature interactions better, feature interactions needs manual design. With the development of artificial intelligence, the deep neural network has shown the potential ability to learn sophisticated feature interactions automatically. Some ideas proposed CNN [Liu, Yu, Wu et al. (2015)] and RNN [Zhang, Dai, Xu et al. (2014)] for CTR prediction [Akopyan and Khashba (2017)], but the CNN model is biased to the interaction between neighbor locally while RNN based on the sequential dependency. In recommendation business data, there

is no relationship between each column features. In this situation, our paper employs CNN as wide part, which is based on the GBDT leaf nodes to construct local features. The main neural network structure is as follows:

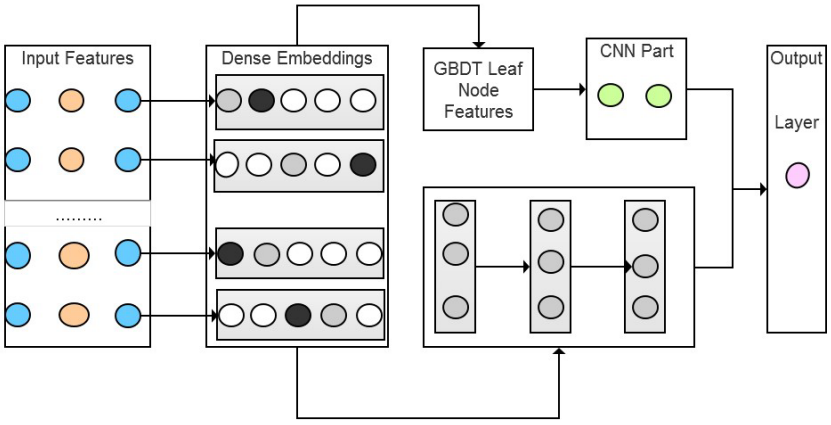

Figure1: GBDT_CNN_DNN rough structure

To reduce the feature dimension, we employ an embedding procedure to transform the category feature into dense vector, then add the numeric feature together. Based on the embedding, GBDT_CNN part extracts the low order feature and DNN part extracts the high order feature, as shown in Fig. 1.

In deep learning, one of the most important things is to prevent over fitting. An overfit model has poor performance since it overeats to the giving training data. We employ L2-norm to regularize the objective function. On the other hand, deep neural network is also easy to overfit. In this paper, we use a simple strategy to avoid neural network from over fitting, which is known as dropout. We also early stopping to prevent the overfitting in deep part. Therefore, in our GBDT_CNN_DNN framework as shown in Fig. 1, we adopt L2-norm to regularize the generalize linear component and adopt dropout and early stopping for the deep component.

3 Deep wide GBDT_CNN network

3.1 Model structure

Based on the above GBDT_CNN_DNN architecture, linear part, gradient boosting decision tree part and deep network part are jointly trained in our model, the main network structure as Fig. 2.

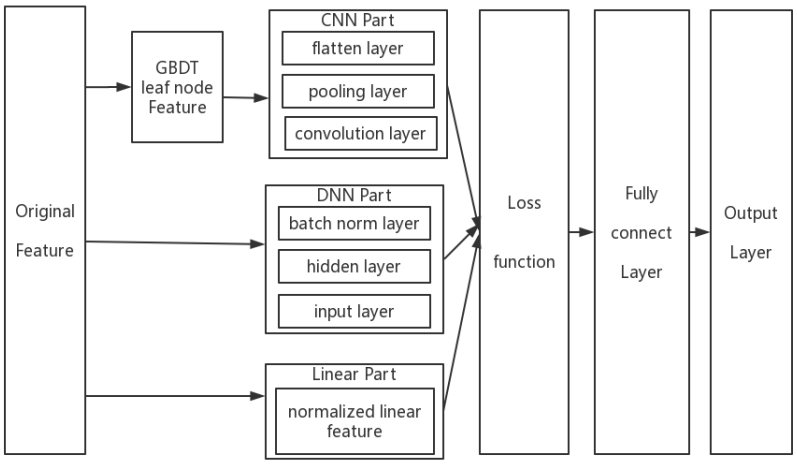

Figure 2: Detail model architecture

Fig. 2 illustrates the detail neural network architecture of our proposed GBDT_CNN_DNN model, which combines the low order feature with the high order feature in the area of recommendation. In the pooling process, this paper employs the method of segmental flexible pooling for the first time. Specially, the pooling window is acquired by neural network learning. In the model, the convolution kernels are 1 by 1, 3 by 3, and 5 by 5, and the number of convolution kernels is learned by neural network. It performs better than a manual setting.

3.2 Convolution on boosting features

Boosting algorithm [Chen and Guestrin (2016)] is consist of iteratively learning weak classifiers with respect to a distribution. The significance of the model is that the latter tree learns the residual of the previous tree. The leaf node of the latter tree is locally related to the previous tree, and the convolutional neural network [Sutskever, Hinton and Krizhevsky (2012)] is suitable for local relationship characteristics.

$$f_M(x) = \sum_{i-1}^{M} T_i\,(x;\theta_m) \tag{1}$$

where $T(x;\theta_m)$ is m decision tree, θ_m represents the parameters of the decision tree, M is the number of trees, and strong classifier $f_M(x)$ can be composed of multiple weak classifiers $T(x;\theta_m)$ linear added.

The m gradient boosting tree has a relationship with the m-1 tree, as follows:

$$f_m(x) = f_{m-1}(x) + T(x;\theta_m) \tag{2}$$

Let y_i is the real value, $y_i \approx \sum_{i=1}^{M} T(x;\theta_m)$, furtherly we can calculate the following formula:

$$T_{M-2}(x;\theta_{m-2}) + T_{M-1}(x;\theta_{m-1}) + T_M(x;\theta_m) = y_i - \sum_{i=1}^{M-3} T_i\,(x;\theta_i) \tag{3}$$

We can deduce the following formula:

$$T_{M-1}(x;\theta_{m-1}) = y_i - \sum_{i=1}^{M-3} T_i\,(x;\theta_i) - T_{M-2}(x;\theta_{m-2}) - T_M(x;\theta_m) \tag{4}$$

Considering that convolution neural network is good at capturing such local perceptual features related before and after, we employ the convolution neural network on boosting features.

3.3 Deep neural network on original features

The deep neural network can extract abstract high-order features, and the network with embeddings [Frome, Corrado, Shlens et al. (2013)] can generate unseen features better. The deep component is a feed-forward neural network, as shown in Fig. 1, for the categorical feature and numerical feature. The original inputs are converted into a low dimensional dense vector, which also called the embedding vector. In this paper, the embedding vectors are initialized with norm distribution, and then the values are trained to minimize the total loss function. These dense embedding vectors are fed into the hidden layers of a deep neural network in the forward progress. Each hidden layer is computed by the following equations:

$$y^{(l+1)} = f\left(x * w^{(l)} + b^{(l)}\right) \tag{5}$$

where l is the number of the layer and f is the activation function [Schmidt-Hieber (2017)]. In this paper, we employ the rectified linear units.

In the actual data set, some features fluctuate greatly, we try the following three feature transformations:

(a) For each feature f_i, we design the following piecewise function. According to the range of the feature variables, the feature should add one to handle features whose value is zero. We also normalize the features to have mean value zero and standard deviation value one. The mean and standard deviation value was measured to the train set, and then applied to both train set and valid set. Some features have significant outliers, so any feature value that is more five standard deviations than the mean is truncated to five. We find these feature processing skills improve better in the test data.

$$\begin{cases} \log\left(f_i^2 + 1\right), 0 \le f_i < 1 \\ \log(f_i + 1), f_i > 1 \\ -\log\left(f_i^2 + 1\right), -1 \le f_i < 0 \\ -\log(|f_i| + 1), f_i < -1 \end{cases} \tag{6}$$

(b) In order to reduce the fluctuation of continuous features, we also try the binning and bucket technology. The specific steps are as follows: Firstly, calculate the range of values for each column feature. Secondly, design the number of buckets. Thirdly, according to the equal frequency method, the data is placed in different buckets.

(c) For each feature, it needs to scan all the data instance to estimate the information gain of all possible split points, which consume more time. Our method is Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB). By the method of GOSS, we exclude a significant proportion of data instances with small gradients, and use the rest to estimate the information gain. GOSS can obtain quite an accurate estimation of the information gain with a much smaller data size. By the means of EFB, we bind mutually exclusive features to reduce the number of features. In our training data, the above three feature transformation methods can reduce the fluctuation well as a configuration file. In different data distribution, users can experiment each method and then choose the best one.

We try two ways as the DNN input: The first way is entering hidden vector into DNN directly, and the second way is applying second order feature result as DNN input. In large-scale data samples, the second way can extract more abstract features and achieve better results.

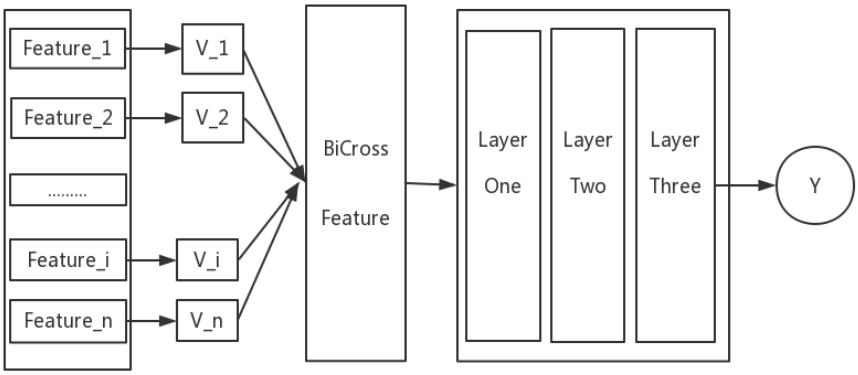

Figure3: DNN part on Bi_Cross Feature

The Fig. 3. has five layers: input feature vector layer, embedding layer, bi-cross feature layer, hidden layer and prediction score layer. The embedding layer projects each feature to a dense vector representation. The $v_i \in R^k$ is the i-th embedding feature, the math form is as follows:

$$V_x = \{x_1 v_1, x_2 v_2, \ldots\ldots, x_i v_i, x_n v_n\} \tag{7}$$

We then feed the embedding set V_x into Bi-cross feature layer, which coverts embedding vector into Bi-cross layer value:

$$f(V_x) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} x_i v_i \circ x_j v_j \tag{8}$$

The Bi-cross feature layer is a stack of fully connected layers, which are capable of learning higher order interactions between features. We put the result into the hidden layer.

3.4 Linear model on original features

Linear model can extract relatively low-order features, which can remember obvious business rules with a few parameters. Its mathematical formula is as follows:

$$y = w^T x + \sum_{i=1}^{n} \sum_{j=i+1}^{n} x_i v_i \circ x_j v_j + b \tag{9}$$

where y is the predicted value, x denotes the combination of the feature vectors, and w and b are the model parameters. Some features are the original inputs and the rest are statistical features [Han and Bhanu (2004)].

In practice, it is known that not all features are relevant to prediction, so we propose the attention mechanism on feature interactions by performing a weighted sum on the interacted vectors:

$$f_{att} = \sum_{(i,j) \in R_x} a_{ij} (v_i \circ v_j) x_i x_j \tag{10}$$

where a_{ij} is the attention score for cross feature w_{ij}, and it shows the importance of w_{ij} in predicting the target. We construct an attention network to estimate the a_{ij}, and it is shown in Fig. 4.

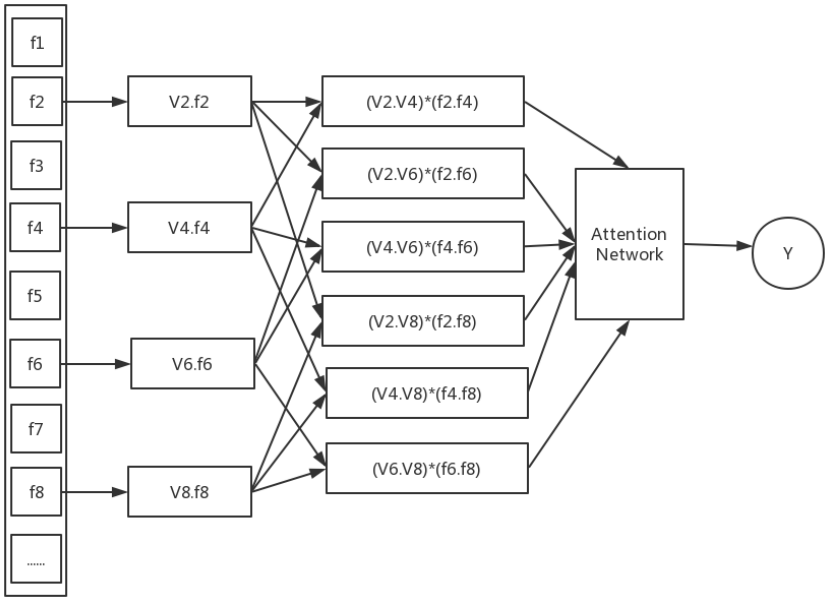

Figure4: Two_order feature interactions attention

Our attention network is defined as:

$$a_{ij}^{*} = h^{T} Leak\_reLu\big(W(v_i \circ v_j)x_i x_j + b\big) \tag{11}$$

$$a_{ij} = \frac{\exp(a_{ij}^{*})}{\sum_{(i,j)\in R_x} \exp(a_{ij}^{*})} \tag{12}$$

The attention scores are normalized through the soft-max function. We use the leak rectifier as the activation function, which empirically shows good performance. To summarize, we give the improved formulation of generalize linear model as:

$$y\_linear = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} a_{ij}(v_i \circ v_j)x_i x_j \tag{13}$$

where a_{ij} has been defined in Eq. (12). Compared with classic linear model, adding attention mechanism improves 1.5 percent AUC score.

4 Experiments

4.1 Data analysis

We need an evaluation function for learning algorithm, and the dataset is divided into training set, validation set, and test set. Training set is used to train models, validation set is used to adjust hyper-parameters, and test set is used to test the final quality of the overall model. Our model is 3 percent higher than the deep neural network model alone. We experiment six models in the same data set. The first model only applies DNN part. The second model only applies linear part. The third model uses both DNN part and linear part.

The fourth model uses both GBDT_CNN part, linear part, and DNN part. The fifth model adds attention based on the fourth model. The sixth model shares the embedding with first order feature value and second order feature value. Our experiment shows that the sixth model has better performance, and our empirical analysis is as follows:

Table 1: The AUC of model performance

Model	AUC	LogLoss	MSE	Accuracy
Logistic Regression	0.8320	0.15	0.63	0.8214
DNN	0.8351	0.14	0.62	0.8301
LR&DNN	0.8429	0.13	0.62	0.8534
LR_GBDT_CNN_DNN	0.8597	0.08	0.61	0.8567
LR_GBDT_CNN_DNN_v1	0.8599	0.08	0.58	0.8603
LR_GBDT_CNN_DNN_v2	0.8626	0.07	0.56	0.8762

Our proposed model contains convolution neural network based on gradient boosting decision tree and deep neural network. We use decision tree as base machine learner, and then employ convolutional neural network to mine the perception of local features. In this paper, we make model fusion by the characteristics of these two algorithms for the first time.

In Fig. 5, back propagation implements an iterative process. At the beginning of each iteration, we should select some train sample, which called batch data. By feed forward, the batch data can get its prediction result. Because train data has label, the neural network compute the loss between label and prediction result. Finally, back propagation updates the weights.

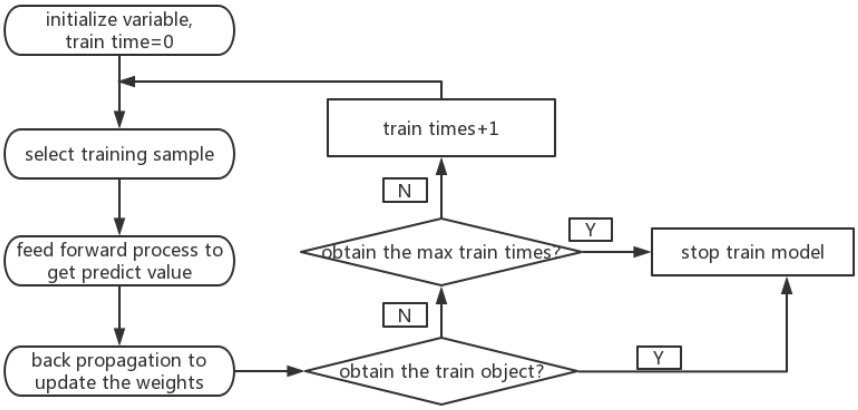

Figure 5: Neural network optimize process in our model

As the training process shown in Fig. 5, at the beginning we initialize the parameters to start training, and then adjust parameter by the error, which is based on the objective function, finally we obtain a better model to test performance. From a perspective of

serving, our model has fewer parameters than other models, which contain wide part using factorization machine or using logistic regression.

4.2 Parameter skills and service performance

In order to make the loss function continue declining in the training set, we clip the gradient [Bottou (2012)], and the mathematic formula is as follows:

$$g_i = \frac{\partial J(w)}{w_i} \tag{14}$$

We set clipped threshold c, and $||g|| = \sqrt[2]{\sum_{i=1}^{n} g_i^2}$, when the $||g|| > c$, we set $g = \frac{c}{||g||}.g$. By this way our model avoids gradient disappearance or gradient explosion [Johnson and Zhang (2013)]. In our model, generalized linear part shares the one order embedding and two order embedding, which reduces the number of parameters and improves the service prediction speed.

Tab. 2 shows the training time and evaluation index about different model. According to Occam's razor theory, when the model has a similar effect, fewer the parameters are, better the performance is. In summary, the efficiency of our model is comparable to the most efficient deep model in the state-of-the-art.

Table 2: FM part share embedding experiment

Experiment ID	Training time	AUC	LogLoss	Accuracy
Not share embedding	125 min	0.7152	0.08	0.8603
Sharing embedding	114 min	0.7274	0.07	0.8762

5 Conclusion

In this paper, we propose GBDT_CNN_DNN model, an end-to-end TREE&DEEP learning framework for the recommendation system. In the data of Kaggle competition, our model has the state-of-the-art performance. GBDT_CNN_DNN trains a deep component and tree component jointly. It has the following advatages:1) it does not need any pre-training; 2) it learns high-order, middle-order and low-order feature interactions. Linear part can learn the low-order feature, gradient boosting decision tree part can learn the middle-order feature and deep neural network can learn the high-order feature; 3) it introduces a sharing strategy of feature embedding to reduce manual feature engineering. There are two directions for future work. First, currently, we simply employ GBDT leaf node for embedding features. We can explore the usage of the Gate Recurrent Unit (GRU) mechanism [Dey and Salemt (2017)] to capture the related trees according to the candidate item. Second, in order to reduce the time complexity, we are interested in developing a distributed version of GBDT_CNN_DNN which can be trained efficiently on a GPU cluster.

Acknowledgment: This work is supported by the National Key Research, Development Program of China under Grant 2017YFB1002304 and the National Natural Science Foundation of china (No. 61672178). The authors would like to thank the anonymous reviewers for their insightful reviews, which are very helpful on the revision of this paper.

References

Akopyan, M.; Khashba, E. (2017): Large-scale YouTube-8M video understanding with deep neural networks. arXiv:1706.04488v1.

Bottou, L. (2012): Stochastic gradient descent tricks. *Neural Networks: Tricks of the Trade*, pp. 421-436.

Chen, T.; Guestrin, C. (2016): Xgboost: a scalable tree boosting system. *Proceedings of the 22nd ACM Sigkdd International Conference on Knowledge Discovery and Data Mining*, pp. 785-794.

Cheng, H. T.; Koc, L.; Harmsen, J.; Shaked, T.; Chandra, T. et al. (2016): Wide & deep learning for recommender systems. *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pp. 7-10.

Dey, R.; Salemt, F. M. (2017): Gate-variants of gated recurrent unit (GRU) neural networks. *IEEE 60th International Midwest Symposium on Circuits and System*, pp. 1597-1600.

Frome, A.; Corrado, G. S.; Shlens, J.; Bengio, S.; Dean, J. et al. (2013): Devise: a deep visual-semantic embedding model. *Advances in Neural Information Processing Systems*, pp. 2121-2129.

Han, J.; Bhanu, B. (2004): Statistical feature fusion for gait-based human recognition. *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, no. 2, pp. 842-847.

Hsiao, S. W.; Huang, H. C. (2002): A neural network based approach for product form design. *Design Studies*, vol. 23, no. 1, pp. 67-84.

Johnson, R.; Zhang, T. (2013): Accelerating stochastic gradient descent using predictive variance reduction. *Advances in Neural Information Processing Systems*, pp. 315-323.

Juan, Y.; Zhuang, Y.; Chin, W. S.; Lin, C. J. (2016): Field-aware factorization machines for CTR prediction. *Proceedings of the 10th ACM Conference on Recommender Systems*, pp. 43-50.

LeCun, Y.; Bengio, Y.; Hinton, G. (2015): Deep learning. *Nature*, vol. 521, no. 7553, pp. 436-444.

Lian, J.; Zhou, X.; Zhang, F.; Chen, Z.; Xie, X. et al. (2018): xDeepFM: combining explicit and implicit feature interactions for recommender systems. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1754-1763.

Liu, B.; Tang, R.; Chen, Y.; Yu, J.; Guo, H. et al. (2019): Feature generation by convolutional neural network for click-through rate prediction. *World Wide Web Conference*, pp. 1119-1129.

Liu, Q.; Yu, F.; Wu, S.; Wang, L. (2015): A convolutional click prediction model. *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pp. 1743-1746.

McMahan, H. B.; Holt, G.; Sculley, D.; Young, M.; Ebner, D. et al. (2013): Ad click prediction: a view from the trenches. *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1222-1230.

Peng, C. Y. J.; Lee, K. L.; Ingersoll, G. M. (2002): An introduction to logistic regression analysis and reporting. *Journal of Educational Research*, vol. 96, no. 1, pp. 3-14.

Rendle, S. (2010): Factorization machines. *IEEE International Conference on Data Mining*, pp. 995-1000.

Rendle, S. (2012): Factorization machines with LIBFM. *ACM Transactions on Intelligent Systems and Technology*, vol. 3, no. 57, pp. 1-22.

Schmidt-Hieber, J. (2017): Nonparametric regression using deep neural networks with ReLU activation function. arXiv:1708.06633v4.

Sutskever, I.; Hinton, G. E.; Krizhevsky, A. (2012): ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, pp. 1097-1105.

Wang, R.; Fu, B.; Fu, G.; Wang, M. (2017): Deep & cross network for ad click predictions. *Proceedings of the ADKDD'17*, pp. 1-7.

Zhang, Y.; Dai, H.; Xu, C.; Feng, J.; Wang, T. et al. (2014): Sequential click prediction for sponsored search with recurrent neural networks. *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pp. 1369-1375.

Zhang, Z.; Liu, Y.; Zhang, Z. (2018): Field-aware matrix factorization for recommender systems. *IEEE Access*, vol. 6, no. 45, pp. 690-698.

Zhou, G.; Mou, N.; Fan, Y.; Pi, Q.; Bian, W. et al. (2018): Deep interest evolution network for click-through rate prediction. arXiv:1809.03672v5.

Zhou, G.; Zhu, X.; Song, C.; Fan, Y.; Zhu, H. et al. (2018): Deep interest network for click-through rate prediction. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1059-1068.