Deep Field Relation Neural Network for Click-Through Rate Prediction

Dafang Zou[a], Zidong Wang[b], Leimin Zhang[c], Jinting Zou[d], Qi Li[a], Yun Chen[e], Weiguo Sheng[a,*]

[a]*Department of Computer Science, Hangzhou Normal University, Hangzhou, China*
[b]*Department of Computer Science, Brunel University London, Uxbridge, Middlesex, UK*
[c]*College of Computer and Information Engineering, Zhejiang Gongshang University, Hangzhou, China*
[d]*Wenzhou University Oujiang College, Wenzhou, China*
[e]*Institute of Information and Control, Hangzhou Dianzi University, Hangzhou 310018, China*

Abstract

Click-Through Rate (CTR) prediction is crucial in calculating advertisements and recommendation systems. To effectively predict CTR, it is important to properly model the interaction among features of data. This work tends to fully utilise the interaction information among features while employing deep neural networks for CTR prediction. To this end, we propose a Deep Field Relation Neural Network (DFRNN), which models feature interaction via a 3-dimensional relation tensor. The proposed method is evaluated on real data sets and compared with related methods. The results demonstrate that our method could be used to derive significant information contained in feature interaction and achieve an accurate CTR prediction.

Keywords: Click-through rate, Neural network, Feature interaction, Relation tensor.

1. Introduction

Many industrial applications, such as online advertising [31], recommendation system [23] and web search [3], are Cost Per Click (CPC) for most of

*Corresponding author
Email address: w.sheng@ieee.org (Weiguo Sheng)

Internet companies. In the CPC advertising system, the ranking sore of an advertisement is usually determined by the product's bid price and Click-Through Rate (CTR) [46]. Therefore, correctly predicting CTR of advertisements is a prerequisite for ensuring revenue and user experience.

The CTR prediction is a typical supervised machine learning problem, whose goal is to accurately predict the probability of user's behaviour under a given advertising context. Denoting features of the problem as x and the target as y, a large number of labeled samples (x_i, y_i) can be obtained from online advertisement's click logs for training purpose. By employing a parameter w to model the probability of click, the CTR can be written as:

$$CTR = Probability(click|AD, User, Query) = f(x, w). \tag{1}$$

Consequently, the problem can be transferred to an optimisation problem and solved by searching for a proper value of w, which minimises the objective loss function $L(y, f(x, w))$. The negative log-likelihood function is typically used as the loss function for CTR prediction. Many classical machine learning models, including Logistic Regression (LR) [34], Bayesian models [13], polynomial-2 (Poly2) [33], gradient boosting decision tree [4, 16], tensor-based models [24], and Factorization Machines (FM) [19, 20, 32], have been proposed to deal with the problem.

To build an effective machine learning model for CTR prediction, it is crucial to model interactions among features of the data. In feature engineering, first-order discrete features are often combined in pairs to form high-order combinatorial features to improve the fitting capability of complex relationships. For example, suppose there are two discrete features: language=$\{Chinese, English\}$ and type=$\{movie, teleplay\}$. They can be crossed to obtain a new feature: language_type=$\{Chinese-movie, Chinese-teleplay, English-movie, English-teleplay\}$. It has been proved in the Kaggle competition [20] that crafting combinatorial features is an effective way for CTR prediction [2, 28, 29, 35]. However, this approach is usually problem dependent and relies on manual feature engineering as well as domain knowledge.

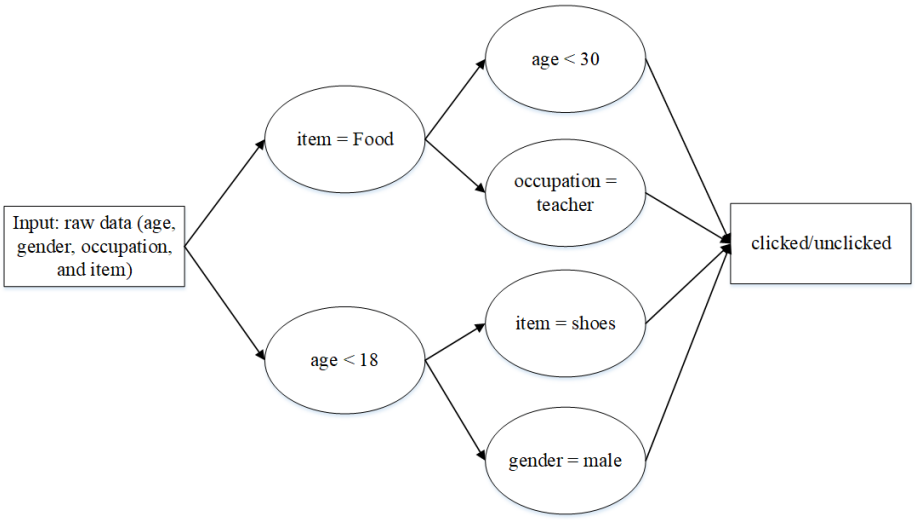

Figure 1: An example of learning feature interactions from the raw data.

Instead of designing new features manually, feature interactions can be learned from raw data by applying machine learning algorithms. Suppose the raw data contain information of age, gender, occupation and item. A decision tree can be constructed based on these data and tags (i.e., clicked/unclicked), as shown in Figure 1. In Figure 1, each path from the root node to the leaf node can be regarded as a way of feature interactions. By defining w_{ij} as the feature interaction coefficient of features i and j, then it can be learned from machine learning models such as LR. Factorization machines [32], proposed to solve the automatic feature combination problem via inner product of feature embedding vectors, is regarded as one of the most successful embedding models [43]. Deep Neural Networks (DNN) have been successfully applied in image classification [14, 25], Natural Language Processing (NLP) [8] and speech recognition [11] over the past years. DNN can automatically capture feature representations and dependencies for prediction purpose. As a result, several DNN based methods have also been proposed for CTR prediction. One challenging issue of applying DNN for CTR prediction is data sparsity [40]. Most of the data (e.g., user ID, gender and city) in CTR problems are non-contiguous

and discrete. These data are typically converted to a set of high-dimensional sparse features via the one-hot encoding [6, 35] for CTR prediction. In such a situation, they first need to be converted into dense feature embedding vectors before inputting to neural networks. Further, appropriate representations for feature interactions in the neural network structure are required. Intuitively, we can directly utilise a feature interaction vector to represent the interaction of two features. However, by employing such a representation, there are usually not enough data to estimate interactions between features. As a result, the parameters of feature interaction vector cannot be trained adequately. To deal with this issue, the methods of employing two feature embedding vectors, such as element-wise product [15, 39] and inner product [29, 32], could be used to calculate feature interactions.

In this work, we propose a Deep Field Relation Neural Network (DFRNN) for CTR prediction. The proposed method tends to enhance DNNs by modelling 2-order feature interactions after feature embedding. Such a feature interaction operation is devised to improve the capability of neural network to learn feature cross information. Further, we deepen the shallow model by combining it with a classical DNN component, thus effectively modelling high-order and nonlinear feature interactions. Comparing with traditional methods, which perform inner product or element-wise on embedding vectors to model feature interaction in low levels, the proposed feature interaction operation is able to encode more informative feature interactions, thus facilitating the deep layers in our model to learn meaningful information. The proposed method has been evaluated on two real-world datasets and compared with related methods. The results show our method is able to achieve an accurate CTR prediction and outperform related methods, including classical models of FM and recently proposed deep models such as PNN and DeepFM.

The remainder of the paper proceeds as follows. Firstly, a brief review of related work is given in Section 2. Then, the details of proposed method are described in Section 3. Section 4 provides experiments and analysis. Finally, Section 5 concludes the paper with a summary and future work.

2. Related Work

Many methods have been proposed to process high dimensional sparse data. In this section, we will briefly review shallow and neural network based methods, which are related to our work.

2.1. Shallow Methods

The CTR task can be transformed into an optimisation problem, which is unconstrained convex with a unique optimal solution. To deal with this problem, LR [22], which is a linear model, could be employed. LR can be applied on problems with large-scale features and is able to quickly converge to the optimal solution through the commonly used gradient descent method. The LR model also possesses a good interpretability. By employing this model, the weights corresponding to features as well as the importance of each feature and its influence on the click rate can be analysed. However, the feature expression capability of LR is generally weak since it relies on manual feature engineering or feature selection techniques.

FM [32], which is proposed to learn feature interactions using the inner product, is one of the most successful CTR models. The FM is defined as:

$$\hat{y}_{FM}(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle v_i, v_j \rangle \cdot x_i x_j, \tag{2}$$

where $w_0 \in R$ is a global bias, $w_i \in R$ is the weight of i-th feature, $v_i \in R^k$ is a k-dimensional vector and $\langle v_i, v_j \rangle$ defines the inner product. FM performs well on large sparse data and has a low time complexity. By introducing the concept of field, Lin et al. proposed a Field-aware Factorization Machine (FFM) [19, 20]. Comparing with FM, in FFM, each feature is allowed to use different vectors to interact with other features of different fields, thus improving the feature expressiveness. However, FFM requires a large memory and, thus, is not easily applicable to address real CTR tasks. In addition, both FM and FFM tend to learn all interactions between each pair of features, which is not able to model high-order feature interactions.

2.2. Neural Network based Methods

Recently, many deep learning models have been developed for CTR [17, 27, 48]. For these models, effectively modelling the feature interactions is crucial for their performance. In traditional DNN models, the input of network is usually required to be dense and numerical. For the CTR task, the dimension of data could be over one million after one-hot encoding. In this case, these models are typically not applicable. To deal with such an issue, Factorization machine supported Neural Networks (FNN) have been proposed. In this method, an embedding layer pretrained by FM, which is used to convert sparse features to low-dimensional dense data, is combined with a DNN component for capturing high-order feature interactions [43]. FNN can be thought as a deepened version of FM. While, Operation-aware Neural Networks (ONN) [41], which utilises a new embedding method named operation-aware embedding for learning feature representations, is a deepened version of FFM. To strengthen the capacity of feature interactions, Product-based Neural Network (PNN) [29] and its extension Product-network In Network (PIN) [30] introduced product operations, which are performed on the embedding layer before applying full-connected DNN. Wide & Deep's model [7] tended to train both shallow and deep components at the same time. The shallow component in this method is based on a linear model such as LR, which carries the benefit of memorisation of low order features while the deep component is based on DNN, which can be used to improve the generalisation capability of the model. However, to ensure a good performance, the features of linear component in this model, which are used directly for final prediction, require to be designed manually. To alleviate this issue, DeepFM [12] tried to replace the linear part of Wide & Deep model with FM to learn feature interactions, thus avoiding manual feature engineering. By jointly training the FM and DNN parts, DeepFM is deemed to be one of the best performing models. In [15], a model called Neural Factorization Machines (NFM), which employs DNN to improve FM, has also been proposed. Since the contribution of each feature interaction to the CTR prediction result could be different, Xiao et al. [39] proposed an Attention neural Factorization Machines

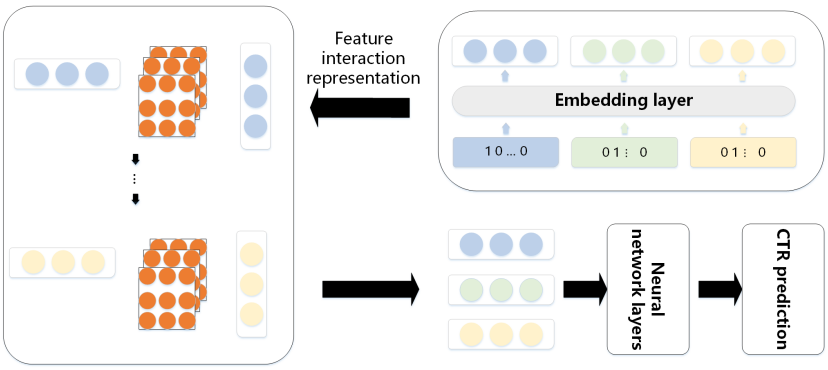

Figure 2: The architecture of our proposed model. The input raw multi-field feature vector is first converted to field embedding vectors via an embedding layer and then fed into the feature interaction layer to model feature interactions. A neural network layer is applied on the output of feature interaction layer to predict the click through rate $\hat{y}$.

(AFM), which employs an attention mechanism originated from Neural Machine Translation (NMT) field [1], to learn the weights of feature interactions. While in [46], Zhou et al. devised a Deep Interest Network (DIN), in which a local activation unit structure is designed to adaptively capture diverse interests of the users from historical behaviours.

One of the key limitations of existing CTR models, which use inner product and/or element-wise product schemes for interaction representation, is that they generally have difficulty to effectively calculate the interactions of feature vectors. To improve the prediction accuracy, it is useful to provide representations for feature interactions after the raw feature embedding layer. Here, we therefore propose a feature interaction representation scheme to support the neural network by learning informative feature interactions at the low level. Further, we deepen the shallow model by combining a classical deep neural network component to appropriately model high-order and nonlinear feature interactions.

3. Proposed Method

In this section, we propose a deep field relation neural network model, which can effectively model the feature interaction, for CTR prediction. In the pro-

posed method, DNNs are enhanced by modelling 2-order feature interactions after feature embedding. This feature interaction operation is devised to improve the capability of neural networks to learn feature cross information. Specifically, our proposed feature interaction operation employs two embedding vectors to calculate the interaction vector. In this operation, rather than calculating the feature interactions using element-wise product or inner product, we tend to support the neural network by learning informative feature interactions at the low level. Further, in our method, the shallow model is deepened by combining a classical DNN component, thus effectively modelling high-order and nonlinear feature interactions. In the proposed method, an embedding operation is used to map high-dimensional sparse input into low-dimensional dense real-valued vectors. Then, several operations (including relation matrix definition and interaction vector calculation) are applied on embedding vectors in the feature layer to model 2-order feature interactions. A multiple hidden layer network is finally employed on the combination vector to learn nonlinear relations among features. These components can be used to appropriately process the data at different stages thus forming an inherent structure to properly deal with CTR prediction tasks. An overall architecture of the proposed DFRNN is illustrated in Figure 2. In the following subsections, we should present the details of main components (including sparse input, embedding, feature interaction, combination, multiple hidden and output layers) of the proposed model.

3.1. Sparse Input and Embedding Layers

Unlike image classification or speech recognition, the data of CTR tasks are usually non-contiguous and categorical. To represent raw input features, they are usually converted to high-dimensional sparse features via the one-hot encoding. For example, for the features of user_id = $\{001, 002, \dots\}$, goods = $\{book, basketball, \dots\}$ and gender = $\{male, female\}$, after the one-hot encoding, an input instance can be written as:

$$\underbrace{[1,0,0,0,...,0]}_{user\_id=001} \quad \underbrace{[0,1,0,0,...,0]}_{goods=book} \quad \underbrace{[0,1]}_{gender=female}$$

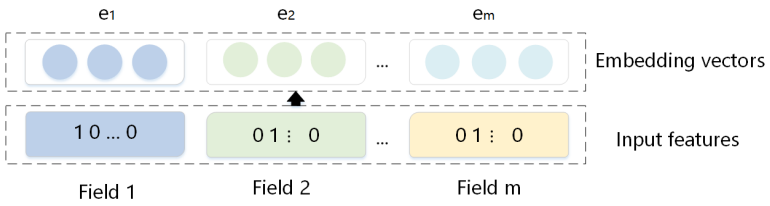

Figure 3: The architecture of embedding layer.

The dimension of features could become huge after encoding. For example, to encode 550 goods, the dimension of features for "goods" will become 550 after coding and only one of them is effective. In this case, DNNs are usually not directly applicable. Here, we propose to transform sparse features into a continuous, dense real-valued vector with a low dimension. The architecture of embedding layer is illustrated in Figure 3. The result of embedding layer is a wide concatenated field embedding vector, $E = [e_1, e_2, \ldots, e_i, \ldots, e_m]$. Here, m denotes the number of fields and $e_i \in R^k$ is the embedding vector of i-th field, where k denotes dimension of the vector.

3.2. Feature Interaction Layer

To improve prediction accuracy of CTR task, it is useful to provide representations for the feature interactions after raw feature embedding layer [41]. The feature interaction layer aims to model the second order feature relations in a precise and effective way. Intuitively, we can directly utilise the feature interaction vector p_{ij} to represent interaction between i-th and j-th features. The number of feature interaction vectors is therefore $n * (n-1)/2$, where n denotes the number of coded features. However, it is difficult to adequately train the vector p_{ij} in real application scenarios where data sparsity is common. The reason is that training each parameter p_{ij} requires a large number of samples with non-zero x_i and x_j. Since the data are inherently sparse, the samples, which satisfy the above requirement could be very few, thus leading to insufficient training samples. This, in turn, results in an inaccurate calculation of parameter p_{ij}, which will ultimately affect the performance of the model. To

deal with such an issue, a possible solution is to employ the embedding vector x to calculate the interaction vector p. Inner product and element-wise product are perhaps the most popular methods for calculating feature interaction. The inner product and element-wise product can be defined as:

$$f_{inner}(\varepsilon) = \{(v_i \cdot v_j)x_i x_j\}_{(i,j)\in\mathcal{R}_\mathcal{X}}, \tag{3}$$

$$f_{element-wise}(\varepsilon) = \{(v_i \odot v_j)x_i x_j\}_{(i,j)\in\mathcal{R}_\mathcal{X}}, \tag{4}$$

respectively. Here, $\mathcal{R}_\mathcal{X} = \{(i,j)\}_{i\in\mathcal{X},j\in\mathcal{X},j>i}\}$, v_i denotes the i-th embedding vector, the symbol " $\cdot$ " means inner product, $\odot$ defines element-wise product and $()_k$ denotes k-th dimension value of the vector, i.e., $(v_i \odot v_j)_k = v_{ik}v_{jk}$.

A key issue with inner product and element-wise product methods in interaction representation is that they are not able to effectively calculate the interactions of feature vectors. To improve the situation, here, we propose a new scheme to represent the feature interaction vector. Specifically, in our representation, feature interaction vector p_{ij} of two feature vectors is defined as:

$$p_{ij} = [p_{ij}^1, ..., p_{ij}^u, ..., p_{ij}^l]. \tag{5}$$

Here, p_{ij}^u is the u-th dimension value of interaction vector, l is the dimension of interaction vector p_{ij}^u, which can be expressed as:

$$p_{ij}^u = v_i \cdot W_u \cdot v_j^T. \tag{6}$$

Here, $W \in R^{k\times k\times l}$ is a 3-dimensional tensor. Each slice $W_{u,u\in\{1,2,...,l\}}$ of the tensor W represents the i-th relation matrix. Figure 3 shows the representations of different feature interaction methods. By employing the above representation scheme based on the original embedding E, we can obtain the results of feature interaction layer, denoted as $\{p_1, ..., p_i, ..., p_n\}$. It should be noted that, computing p_{ij} has a complexity of $O(lk^2)$, where l and k are the dimension of interaction vector p_{ij}^u and feature vector, respectively. The feature interaction layer, thus, has a complexity of $O(nlk^2)$, where n is the number of field interactions.

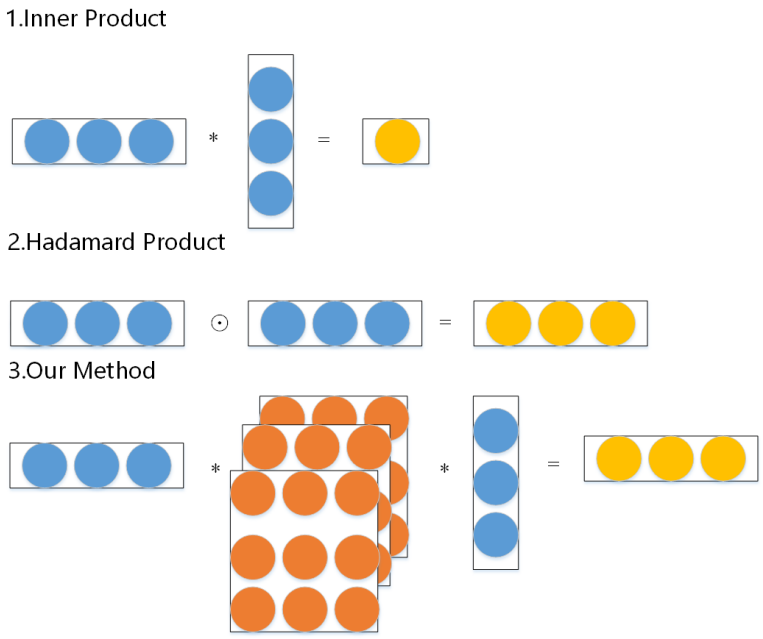

Figure 4: An illustration of different methods for calculating feature interaction.

3.3. Deep Network

The interaction vector p obtained at the interaction layer will be concatenated and fed into the deep component, which is a feed-forward neural network. The result of combination layer is defined as:

$$F_{concat}(p_1, ..., p_i, ..., p_n) = [c_1, ..., c_i, ..., c_k]. \tag{7}$$

The deep network is used to capture high-order interaction among features and generate the model result. Let $h^{(0)} = [c_1, c_2, ..., c_n]$ denote inputs of the deep network, where n is the total size of interaction vectors. Each fully-connected neural network layer is defined as:

$$h^{(l)} = \sigma(W^{(l)} h^{(l-1)} + b^{(l)}), \tag{8}$$

where l is the layer number of deep network and σ denotes the activation function. Here, $W_l \in R^{D_{l+1} \times D_l}$, $b^{(l)}$ and $h_l \in R^{D_l}$ are the model's weight, bias and output, respectively, of the l-th layer. The deep network is allowed to capture

high-order feature interactions by employing non-linear activation functions, such as sigmoid, tanh and ReLU. The output vector of the final neural network layer, which is used to calculate the final CTR prediction, is generated as:

$$y_d = \sigma(W^{|L|+1} h^{|L|} + b^{|L|+1}). \tag{9}$$

Here, $|L|$ denotes the depth of DNN and σ represents the sigmoid function, which is defined as $\sigma(x) = 1/(1 + e^{-x})$. It should be noted that our model is equivalent to FM if the neural network part of the model is removed and the relationship dimension u in relationship tensor is set to 1 while the relationship matrix W is set to be:

$$\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \in R^{k \times k},$$

where k is the dimension of feature vector. On the other hand, a shallow CTR model can be obtained by summing each element in the vector c in our proposed model and employing a sigmoid function to generate the prediction value.

3.4. Learning

Based on the above components, the DFRNN model's output can be written as:

$$\hat{y} = \sigma(w_0 + \sum_{i=1}^{n} w_i x_i + y_d), \tag{10}$$

where w_0 is a global bias, which is used to control the activation state, $\hat{y} \in (0, 1)$ denotes the value of CTR prediction, σ represents the sigmoid function, n is the total size of features, x_i is the i-th feature value and w_i is the weight of i-th feature. The proposed model will be used to address CTR tasks. The objective function is a negative log-likelihood function. In experiments, we aim to minimise the following loss function:

$$loss = -\frac{1}{N} \sum_{i=1}^{N} (y_i \log(\hat{y}) + (1 - y_i) \log(1 - \hat{y}_i)), \tag{11}$$

where y_i is the ground truth of instance x, $\hat{y}_i$ is the prediction value of CTR and N denotes the total size of instances for training.

In practice, a large size of training data is usually used to train deep neural networks, especially for CTR tasks. In this case, it could be very expensive to calculate the gradient on entire training data during the training process. To improve the efficiency of training, mini-batch gradient descent is often employed. However, gradient descent methods depend critically on the value of learning rate, which is an important hyper-parameter in neural network. It has been shown that adaptively adjusting the learning rate could be a viable choice and various methods such as AdaGrad [9], RMSprop [37], AdaDelta [42] and Adam [21] have been proposed. Here, the Adam algorithm [21], which is a combination of RMSProp and momentum methods, has been employed as the optimiser to obtain the learning rate. The algorithm is defined as:

$$M_t = \beta_1 M_{t-1} + (1 - \beta_1) g_t, \tag{12}$$

$$G_t = \beta_2 G_{t-1} + (1 - \beta_2) g_t \odot g_t, \tag{13}$$

$$\Delta\theta_t = -\alpha \frac{M_t/(1 - \beta_1^t)}{\sqrt{(G_t/(1 - \beta_2^t))} + \varepsilon}. \tag{14}$$

Here, β_1 and β_2, which denote the decay rates of two moving averages, are set to be 0.9 and 0.99, respectively. The symbol of ε denotes a small constant, which is used for numerical stability and is empirically set to be 10^{-8}. The M_t and G_t represent the first and second moment, respectively, while g_t is a real gradient at training step t. The learning rate of Adam is selected via a grid search among values of [0.0001, 0.0005, 0.001, 0.005, 0.01] by employing cross validation.

3.5. Regularisation

Sparse L2 Regularisation. To avoid feature vector representations overfitting the data, L2 regularisation will generally be employed. This regularisation method, however, could result in an intensive computation especially when the input data is sparse. To deal with this issue, instead of L2 regularisation, sparse L2 regularisation has been employed in our method. Rather than all sparse

input x, the sparse L2 regularisation penalises only the embedding feature $v = \text{embed}(x)$ [23].

Dropout. The dropout approach [36], which randomly discards a part of neurons (as well as their corresponding connected edges), has been widely employed to avoid overfitting. In our DFRNN model, the dropout will be applied in the feature interaction layer to investigate its impact on feature interaction representations. Specifically, after performing the feature interaction layer, we randomly drop the concatenated vector with a probability of p. Additionally, the dropout has also been employed to train neural network in the proposed method. In experiments, six different dropout rates (i.e., 0.0, 0.1, 0.2, 0.3, 0.4 and 0.5) have been evaluated on the validation dataset and the one with the best performance has been chosen as the dropout rate for our method. As a result, we set the dropout rate as 0.0 and 0.5 on Criteo and Avazu dataset, respectively.

Batch Normalisation. During the training of deep neural network, the input of a middle layer is the output of previous neural layer. Therefore, changes in the parameters of the neural layer will cause a large difference in the distribution of its output. From the perspective of machine learning, if the input distribution of a neural layer changes, then its parameters need to be relearned. This phenomenon is called internal covariate shift. To deal with this problem, it is necessary to make the distribution of the input of each neural layer to be consistent during the training process. The simplest approach is to normalise each neural layer to make its distribution stable. For this purpose, the Batch Normalisation (BN) [18] method has been adopted in our method. The BN is an effective layer-by-layer normalisation method, which can normalise any intermediate layer in the neural network.

4. Experiments

In this section, we evaluate our model and compare it with related methods. In the following subsections, we will provide the details of datasets, methods to

Table 1: Dataset statistics.

Dataset	#instance	#categories	#fields	pos ratio
Criteo	1×10^8	1×10^6	39	0.5
Avazu	4×10^7	6×10^5	24	0.17

be compared, evaluation metrics, data processing, experimental settings, performance comparison and relevant analysis.

4.1. Datasets

Criteo[1] data consists of 98 millions of click records. It is a widely used industry benchmarking dataset for evaluating CTR models. The goal is to predict the probability that a user will click on a given advertising item. The data has 13 continuous and 26 categorical features, with no feature description available. The dataset will be split into two parts: "day6-12" for training and "day13" for testing. For numerical features, they are discretised using bucketing. The bucket label is used to replace the numerical value. For categorical features, we set the categories appeared less than 20 times as "other". Due to enormous data volume and extremely unbalanced labels (only 3% samples are positive), negative sampling is used to obtain a positive sample ratio of about 0.5. After one-hot encoding, the feature space could reach approximately 1M.

Avazu[2] is published in Avazu click-through rate prediction contest. The FFM [19, 20] achieves the best performance in this contest. Avazu data contains several days of click-through data (40 million click records). In this data set, each click instance has 24 data fields. We randomly split the dataset into two parts: 80% for training and 20% for testing, and remove categories appeared less than 20 times to reduce the dimension.

[1] Criteo http://labs.criteo.com/downloads/download-terabyte-click-logs/

[2] Avazu http://www.kaggle.com/c/avazu-ctr-prediction

4.2. Methods to be Compared

The following six classical and state-of-the-art models have been included for comparison. These models are implemented with Tensorflow and trained using the Adam optimisation algorithm.

LR [34]: LR is a classical model for CTR task, which treats the recommendation as a classification problem and ranks items by predicting the probability of positive samples.

FM [32]: FM learns a feature vector for each feature, and the inner product of two feature vectors is used as feature interactions.

FNN [43]: FNN initialises an embedding layer with latent vector of FM as the input of neural network.

PNN [29]: In this method, embedding vectors of different features employ product operations to perform pairwise feature interactions to obtain the interaction information.

Wide & Deep [7]: This method is based on a hybrid model consisting of a single layer of wide part and multiple layers of deep part.

DeepFM [12]: This method improves the Wide & Deep model by replacing the wide part with FM.

4.3. Evaluation Metrics

The area under curve (AUC) and log loss have been used as the evaluation metrics. AUC refers to area under ROC curve, which has been widely used in binary classification. This metric is insensitive to positive ratio and classification threshold. AUC can quantitatively reflect the model's performance based on the ROC curve. Generally, a larger value of AUC means a better classification performance. Log loss can be used to measure the distance between two distributions, which is another widely used metric for binary classification. The lower bound of log loss is 0, indicating the two distributions perfectly match, and a smaller value indicates a better performance.

Table 2: Comparing the performance of various methods.

Method	Criteo		Avazu	
	AUC	Log loss	AUC	Log loss
LR	0.7742	0.5742	0.7545	0.3996
FM	0.7922	0.5509	0.7765	0.3820
FNN	0.7987	0.5431	0.7802	0.3801
PNN	0.7994	0.5425	0.7807	0.3797
Wide & Deep	0.7986	0.5432	0.7806	0.3800
DeepFM	0.7986	0.5428	0.7804	0.3797
DFRNN	0.8020	0.5409	0.7834	0.3780

4.4. Performance Comparisons

In this section, we will evaluate our proposed method by comparing it with related methods. In experiments, all models are implemented using Tensorflow[3]. To make the comparison fair, the size of embedding vector is set to be 50 and 30 for Avazu and Criteo data, respectively, for all methods. The Adam [21], which has been adopted in our model, is configured with a mini-batch size of 500 and 1000 on Criteo and Avazu data, respectively, along with a learning rate of 1.0E-4. For all deep network based methods, the layer depth is configured to be 5 while the number of neurons per layer is set to be 700 and 500 for Criteo and Avazu data, respectively. The RELU is used as the activation function. For initialisation, DNN's hidden layers are initialised with xavier [10] while the embedding vectors are initialised using uniform distributions (the ranges of which are selected from $\{\sqrt{\frac{c}{Nk}}, \sqrt{\frac{c}{nk}}, \sqrt{\frac{c}{k}}\}$), where c=$\{1, 3, 6\}$, N is the input dimension, n denotes the number of fields and k is the embedding size). All experiments are conducted on a machine with 2 GTX 1080Ti GPUs.

[3]Tensorflow: https://www.tensorflow.org/

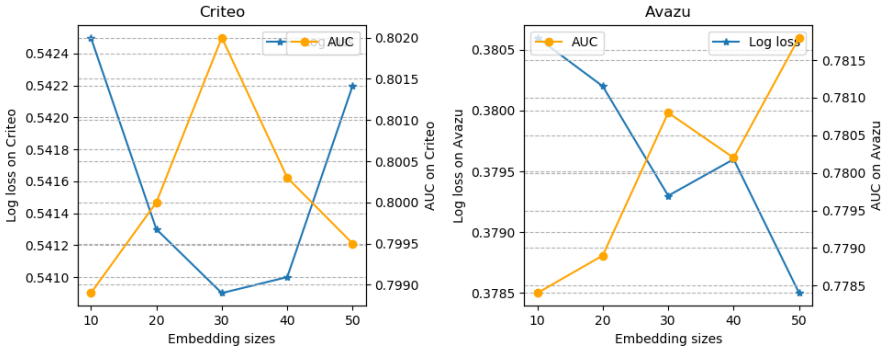

Figure 5: Comparing the performance of proposed method with different embedding sizes.

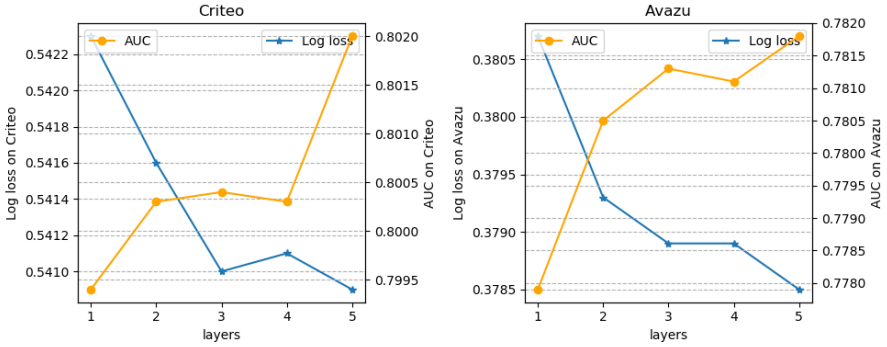

Figure 6: Comparing the performance of proposed method with different network depths in DNN.

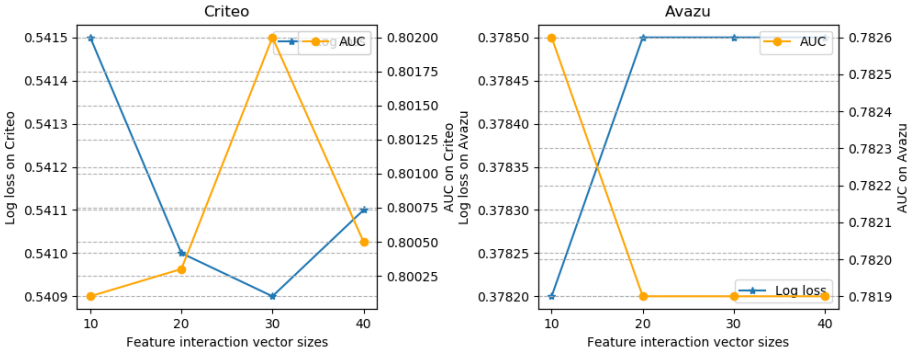

Figure 7: Comparing the results of the proposed method with different sizes of feature interaction vector.

Table 3: The impact of BN and dropout on the proposed method.

Method	Criteo		Avazu	
	AUC	Log loss	AUC	Log loss
DFRNN	0.8020	0.5409	0.7818	0.3785
DFRNN+drop	0.7983	0.5435	0.7834	0.3780
DFRNN+BN	0.8003	0.5412	0.7796	0.3812

Table 2 shows the performance of various methods on Avazu and Criteo datasets. The results clearly show that our proposed DFRNN achieves the best performance among the methods to be compared on both datasets. For example, DFRNN outperforms FNN by 0.413% and 0.405% in terms of AUC and log loss, respectively, on the Criteo data. While, comparing to DeepFM, our method outperforms it by 0.384% and 0.448% in terms of AUC and log loss, respectively, on the Avazu data. These results thus reveal that our proposed feature interaction mechanism is viable to model the feature interaction. We can also find that, comparing FM with LR in terms of AUC and log loss, FM outperforms LR on both datasets. Looking at NN based models, all of them achieve a better performance than FM, which only models two-order feature patterns. This could demonstrate the importance of modelling high-order feature interactions in a nonlinear manner. The results show that both PNN and DFRNN outperform FNN as well as DeepFM. A possible reason is that, by concatenating feature vectors as the input of NNs, FNN has the difficulty to explore all possible feature interactions. While for DeepFM, which directly concatenates feature vectors as the input of NNs in the deep part of model, it lacks the interaction representation of feature vector. This may confirm that providing a representation of feature interaction at the bottom layer helps the NN based model to gain expressive power, thus facilitating the model to learn feature interactions. The above results also indicate that combining NNs with

our proposed feature interaction operation is effective for CTR prediction. This is due to, in our method, the shallow model is deepened by combining a classical deep neural network component, which helps to boost the capability of modelling high-order and nonlinear feature interactions. On the other hand, an effective feature interaction operation has been designed to support the NN to learn informative feature interactions at the low level.

4.5. Parameter Study

In this section, we will investigate the impact of hyper-parameters as well as the BN and dropout technique in our model. Three key hyper-parameters (i.e., the size of embedding vector, the depth of DNN and the size of feature interaction vector) have been considered and evaluated.

First, the impact of different embedding sizes is investigated. For this purpose, we evaluate our method using the embedding sizes = {10, 20, 30, 40, 50} on Criteo and Avazu data. The results are shown in Figure 5. It can be seen from the results that, on Avazu, the performance of our model generally improves along with the increase of embedding size from 10 to 50 in term of the loss. While on Criteo data, it is difficult to fit the parameters in memory and the model could become over-fit when the size is large. The results show that the best performance is obtained with an embedding size of 30 on Criteo data. A larger embedding size means more parameters in the model. Generally, it could be more difficult for our model to optimise on Criteo data than Avazu data. This is due to Criteo data contain much more features than Avazu data.

Then, the impact of different network depths in our model is evaluated. To this end, we test the model with network depths = {1, 2, 3, 4, 5}. The results are shown in Figure 6. From the results, we can see that the performance of our model generally improves along with the increase of network depth on both datasets. It should be noted that increasing the depth of network will lead to an increased complexity of the model and affect its capability to learn high-order features.

Subsequently, we evaluate the impact of the feature interaction vector size

by changing it from 10 to 40 in the feature-interaction layer. The results are shown in Figure 7. From the results, we can find that the values of 30 and 10 are the best feature interaction vector sizes for Criteo and Avazu datasets, respectively. In addition, the results show that, on Avazu data, the performance of our method is generally stable along with the increase of feature interaction vector size.

Finally, we evaluate the impact of BN and dropout technique in DNN structure. Dropout [36] is a technique developed for DNN, which randomly discards a part of neurons to avoid overfitting. BN [18] is used to deal with internal covariate shift and accelerate training of DNN model. In order to show the impact of these two techniques, a BN layer and/or a dropout scheme with a rate of 0.5 is added to DNN layers for comparison. The results are shown in Table 3. The results show that DFRNN without BN outperforms DFRNN with BN on both Criteo and Avazu data in terms of AUC and log loss. The failure of BN for CTR prediction tasks is mainly due to the input data is sparse, as BN relying on statistics of a mini-batch [30]. While, for the dropout scheme, the results show that it has little impact on the performance of DFRNN.

5. Conclusions

In this work, we propose a novel neural network model called DFRNN, which brings together the effectiveness of feature interaction machines with a strong representation capability of non-linear neural networks for CTR prediction. In the proposed model, a new feature interaction operation is designed to support the neural network to learn informative feature interactions at the low level. Further, we deepen the shallow model by combining a classical deep neural network component to effectively model high-order and nonlinear feature interactions. Experiments on two real-world datasets show that, with only one hidden layer, our method could achieve accurate CTR prediction and significantly outperform LR, FM and state-of-the-art deep learning methods including Wide & Deep and DeepFM. The proposed work can be extended in a few directions such

as investigating the graph attention networks [38] for improving the expressive capability of learning useful high-order feature interactions, employing recurrent neural networks to deal with dynamic CTR prediction [44, 47], exploring the interpretability of deep model for CRT prediction as well as employing filtering [45, 49] and Markovian [5, 26] models to deal with noise and unlabelled data, which are widely existed in real CTR problems.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant No. 61873082, Grant No. 62003121 and Grant No. 61973102, the Zhejiang Provincial Natural Science Foundation of China under Grant No. LQ20F030014, and the Royal Society of the UK.

References

[1] Bahdanau, D., Cho, K., Bengio, Y.. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:14090473 2014;.

[2] Chapelle, O., Manavoglu, E., Rosales, R.. Simple and scalable response prediction for display advertising. ACM Transactions on Intelligent Systems and Technology (TIST) 2015;5(4):61.

[3] Chapelle, O., Zhang, Y.. A dynamic bayesian network click model for web search ranking. In: Proceedings of the 18th international conference on World wide web. 2009. p. 1–10.

[4] Chen, T., Guestrin, C.. Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. ACM; 2016. p. 785–794.

[5] Chen, Y., Chen, Z., Chen, Z., Xue, A.. Observer-based passive control of non-homogeneous markov jump systems with random communication delays. International Journal of Systems Science 2020;51(6):1133–1147.

[6] Cheng, H.T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., Anderson, G., Corrado, G., Chai, W., Ispir, M., et al. Wide & deep learning for recommender systems. In: Proceedings of the 1st Workshop on Deep Learning for Recommender Systems. ACM; 2016. p. 7–10.

[7] Cheng, H.T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., Anderson, G., Corrado, G., Chai, W., Ispir, M., et al. Wide & deep learning for recommender systems. In: Proceedings of the 1st Workshop on Deep Learning for Recommender Systems. ACM; 2016. p. 7–10.

[8] Devlin, J., Chang, M.W., Lee, K., Toutanova, K.. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:181004805 2018;.

[9] Duchi, J., Hazan, E., Singer, Y.. Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research 2011;12(Jul):2121–2159.

[10] Glorot, X., Bengio, Y.. Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the thirteenth international conference on artificial intelligence and statistics. 2010. p. 249–256.

[11] Graves, A., Mohamed, A.r., Hinton, G.. Speech recognition with deep recurrent neural networks. In: Acoustics, speech and signal processing (icassp), 2013 ieee international conference on. IEEE; 2013. p. 6645–6649.

[12] Guo, H., Tang, R., Ye, Y., Li, Z., He, X.. Deepfm: a factorization-machine based neural network for ctr prediction. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence. AAAI Press; 2017. p. 1725–1731.

[13] Hand, D.J., Yu, K.. Idiot's bayes—not so stupid after all? International statistical review 2001;69(3):385–398.

[14] He, K., Zhang, X., Ren, S., Sun, J.. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. p. 770–778.

[15] He, X., Chua, T.S.. Neural factorization machines for sparse predictive analytics. In: Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval. ACM; 2017. p. 355–364.

[16] He, X., Pan, J., Jin, O., Xu, T., Liu, B., Xu, T., Shi, Y., Atallah, A., Herbrich, R., Bowers, S., et al. Practical lessons from predicting clicks on ads at facebook. In: Proceedings of the Eighth International Workshop on Data Mining for Online Advertising. ACM; 2014. p. 1–9.

[17] Huang, T., Zhang, Z., Zhang, J.. Fibinet: combining feature importance and bilinear feature interaction for click-through rate prediction. In: Proceedings of the 13th ACM Conference on Recommender Systems. 2019. p. 169–177.

[18] Ioffe, S., Szegedy, C.. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:150203167 2015;.

[19] Juan, Y., Lefortier, D., Chapelle, O.. Field-aware factorization machines in a real-world online advertising system. In: Proceedings of the 26th International Conference on World Wide Web Companion. International World Wide Web Conferences Steering Committee; 2017. p. 680–688.

[20] Juan, Y., Zhuang, Y., Chin, W.S., Lin, C.J.. Field-aware factorization machines for ctr prediction. In: Proceedings of the 10th ACM Conference on Recommender Systems. ACM; 2016. p. 43–50.

[21] Kingma, D.P., Ba, J.. Adam: A method for stochastic optimization. arXiv preprint arXiv:14126980 2014;.

[22] Kleinbaum, D.G., Dietz, K., Gail, M., Klein, M., Klein, M.. Logistic regression. Springer, 2002.

[23] Koren, Y., Bell, R., Volinsky, C.. Matrix factorization techniques for recommender systems. Computer 2009;42(8):30–37.

[24] Koren, Y., Bell, R., Volinsky, C.. Matrix factorization techniques for recommender systems. Computer 2009;42(8):30–37.

[25] Krizhevsky, A., Sutskever, I., Hinton, G.E.. Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. 2012. p. 1097–1105.

[26] Li, Q., Liang, J.. Dissipativity of the stochastic markovian switching cvnns with randomly occurring uncertainties and general uncertain transition rates. International Journal of Systems Science 2020;51(6):1102–1118.

[27] Lian, J., Zhou, X., Zhang, F., Chen, Z., Xie, X., Sun, G.. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2018. p. 1754–1763.

[28] Menon, A.K., Chitrapura, K.P., Garg, S., Agarwal, D., Kota, N.. Response prediction using collaborative filtering with hierarchies and side-information. In: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM; 2011. p. 141–149.

[29] Qu, Y., Cai, H., Ren, K., Zhang, W., Yu, Y., Wen, Y., Wang, J.. Product-based neural networks for user response prediction. In: Data Mining (ICDM), 2016 IEEE 16th International Conference on. IEEE; 2016. p. 1149–1154.

[30] Qu, Y., Fang, B., Zhang, W., Tang, R., Niu, M., Guo, H., Yu, Y., He, X.. Product-based neural networks for user response prediction over

multi-field categorical data. ACM Transactions on Information Systems (TOIS) 2018;37(1):5.

[31] Ren, K., Zhang, W., Rong, Y., Zhang, H., Yu, Y., Wang, J.. User response learning for directly optimizing campaign performance in display advertising. In: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management. 2016. p. 679–688.

[32] Rendle, S.. Factorization machines. In: Data Mining (ICDM), 2010 IEEE 10th International Conference on. IEEE; 2010. p. 995–1000.

[33] Rendle, S., Gantner, Z., Freudenthaler, C., Schmidt-Thieme, L.. Fast context-aware recommendations with factorization machines. In: Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval. ACM; 2011. p. 635–644.

[34] Richardson, M., Dominowska, E., Ragno, R.. Predicting clicks: estimating the click-through rate for new ads. In: Proceedings of the 16th international conference on World Wide Web. ACM; 2007. p. 521–530.

[35] Shan, Y., Hoens, T.R., Jiao, J., Wang, H., Yu, D., Mao, J.. Deep crossing: Web-scale modeling without manually crafted combinatorial features. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM; 2016. p. 255–262.

[36] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.. Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research 2014;15(1):1929–1958.

[37] Tieleman, T., Hinton, G.. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural networks for machine learning 2012;4(2):26–31.

[38] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.. Graph attention networks. arXiv preprint arXiv:171010903 2017;.

[39] Xiao, J., Ye, H., He, X., Zhang, H., Wu, F., Chua, T.S.. Attentional factorization machines: Learning the weight of feature interactions via attention networks. arXiv preprint arXiv:170804617 2017;.

[40] Xie, Y., Jiang, D., Wang, X., Xu, R.. Robust transfer integrated locally kernel embedding for click-through rate prediction. Information Sciences 2019;491:190–203.

[41] Yang, Y., Xu, B., Shen, S., Shen, F., Zhao, J.. Operation-aware neural networks for user response prediction. Neural Networks 2020;121:161–168.

[42] Zeiler, M.D.. Adadelta: an adaptive learning rate method. arXiv preprint arXiv:12125701 2012;.

[43] Zhang, W., Du, T., Wang, J.. Deep learning over multi-field categorical data. In: European conference on information retrieval. Springer; 2016. p. 45–57.

[44] Zhang, X.M., Han, Q.L., Ge, X., Zhang, B.L.. Passivity analysis of delayed neural networks based on lyapunov–krasovskii functionals with delay-dependent matrices. IEEE transactions on cybernetics 2018;50(3):946–956.

[45] Zhao, Z., Wang, Z., Zou, L., Guo, J.. Set-membership filtering for time-varying complex networks with uniform quantisations over randomly delayed redundant channels. International Journal of Systems Science 2020;:1–14.

[46] Zhou, G., Song, C., Zhu, X., Fan, Y., Zhu, H., Ma, X., Yan, Y., Jin, J., Li, H., Gai, K.. Deep interest network for click-through rate prediction. arXiv preprint arXiv:170606978 2017;.

[47] Zhou, L., She, J., Zhang, X.M., Zhang, Z.. Improving disturbance-rejection performance in a modified repetitive-control system based on equivalent-input-disturbance approach. International Journal of Systems Science 2020;51(1):49–60.

[48] Zou, D., Sheng, M., Yu, H., Mao, J., Chen, S., Sheng, W.. Factorized weight interaction neural networks for sparse feature prediction. Neural Computing and Applications 2019;:1–13.

[49] Zou, L., Wang, Z., Hu, J., Liu, Y., Liu, X.. Communication-protocol-based analysis and synthesis of networked systems: progress, prospects and challenges. International Journal of Systems Science 2021;:1–22.