IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 24 May 2024, accepted 12 June 2024, date of publication 21 June 2024, date of current version 12 August 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3417422

RESEARCH ARTICLE

Improved xDeepFM With Single Value Decomposition and Attention Mechanism

YIWAN ZHANG, ZHAN WANG, AND INWHEE JOE
Department of Computer Science, Hanyang University, Seoul 04763, South Korea
Corresponding author: Inwhee Joe (iwjoe@hanyang.ac.kr)

ABSTRACT Due to the sheer volume and variety of data in industrial manufacturing, manually creating features can be costly. Therefore, appropriate feature processing methods become crucial. Most existing feature processing methods abstract feature engineering as a feature search problem, i.e., finding feature transformations that optimize model performance. However, for automated feature engineering, the number of searches and the number of transformation combinations are huge. Therefore, we use a factorization-based model that measures interactions in terms of vector products, automatically learns patterns of combined features, and generalizes them to unseen features. Prior to this paper, the DeepFM algorithm (which combines an FM model with a deep neural network model) and the xDeepFM algorithm (which proposes a novel Compressed Interaction Network (CIN) designed to make feature interactions explicit) were available. The LRCIN proposed in this paper focuses on improving the CIN network in the xDeepFM method, by introducing a low-rank approximation method in the CIN network to reduce the number of parameters, and adding an attention mechanism after the CIN to ensure the accuracy of the model. The experimental results show that our method can effectively reduce the time complexity of the model and improve the model accuracy to some extent.

INDEX TERMS Automatic feature engineering, compressed interaction network, attention mechanism.

I. INTRODUCTION

A feature is actually an abstract representation of information about a particular behavior. Because a behavior needs to be transformed into a mathematical form in order to be learned by a machine learning model [1], and in order to accomplish this transformation, we need to extract the information from these behaviors in the form of features [2]. In recommendation systems, the model in training and learning has the following characteristics:

(1) Contains a large number of discrete features, such as demographic attributes, device attributes, user classification interests, etc [3]; Contains a large number of sparse features with high latitude;

(2) Feature overlap or combination is very critical to the prediction effect [4]. For a recommendation system based on CTR prediction, the most important thing is to learn the feature crossover behind the user's click behavior. In different recommendation scenarios, low order feature crossover or high order feature crossover can have an impact on the final CTR [5].

The Factorization Machine (FM) [6] algorithm extracts feature combinations, i.e., feature crossovers, by performing inner product operations on the hidden variables of the features. Although FM can theoretically model higher-order feature crossovers, in practice it generally uses only second-order feature crossovers due to computational complexity. Years of research have shown that both higher- and lower-order cross-features are very important, and that learning both cross-features at the same time is better than considering only one of them. The key question now is how to extract these cross-features efficiently. To extract these cross-features efficiently, many methods have been proposed in academia and industry: Wide&Deep [7], DeepFM [8], and xDeepFM [9]. In the Wide&Deep model, the Wide and Deep parts are used to ensure the ‘‘memory capacity’’ of the

The associate editor coordinating the review of this manuscript and approving it for publication was Mu-Yen Chen.

model, i.e., the retention of low-order features, and the LR is used directly here, while the Deep part is used to ensure the ''generalization capacity'' of the model, i.e., the cross-over of high-order features, and the MLP is used directly here.

DeepFM uses the FM model on the wide side, which is able to preserve not only the original features (non-feature crossover), but also the second-order feature crossover compared to low-rank (LR) [10]. xDeepFM is an improved version of Wide Deep with the addition of a CIN layer to explicitly construct finite-order feature combinations [11].

This paper focuses on some improvements to xDeepFM. Since the xDeepFM model combines FM and deep neural networks, its internal structure is more complex, so the model is less interpretable and it is difficult to explain the model's contribution to the prediction results. We will first introduce a low-rank approximation method [12] to simplify the CIN network and thus reduce its time complexity, and then add an attention mechanism [13] after the CIN output part of xDeepFM to dynamically learn the weights between features to better capture the interactions between features. In this way, the model will be able to predict the target variable more accurately and generalize well to new and unseen data. Incorporating the attention mechanism also improves the interpretability of the model, allowing us to more intuitively understand the importance of the model's ranking of features, which in turn helps in feature selection and model optimization. Our main contributions are summarized below:

We propose a new model that adds an attention mechanism to the original xDeepFM for learning the weights between features.

We simplify the number of parameters of the CIN network, which reduces the time complexity of CIN, lowers the computational requirements of the model, and drastically reduces the running cost so that the model can survive in resource-constrained environments.

We take the output of the CIN part as the input of the attention mechanism, and then use the attention mechanism to compute the weights of the cross-features to obtain the weighted cross-feature representation. Finally, the weighted cross-features are combined with the output of the DNN and the output of the linear part. By introducing the attention mechanism, the model not only improves its prediction performance, but also improves its interpretability, which is especially important for complex models.

The method combines the low-rank approximation and the attention mechanism, and the experiments demonstrating the application of the model to different datasets show that the new model has good generalization ability, and we find that the accuracy of this new model is significantly better than that of the original xDeepFM model.

II. RELATED WORK

Extracting higher-order and lower-order cross-features is critical in business. And people are looking for good solutions again.

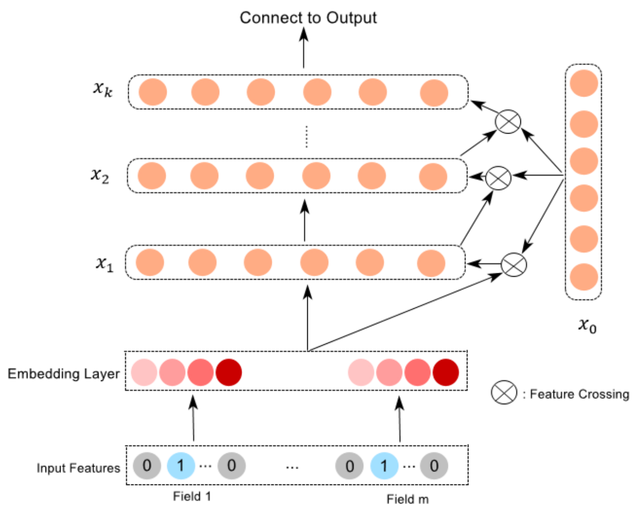

FIGURE 1. The architecture of Cross Network. The original feature vector of the output and input of the fully connected layer is used as the input of the cross-layer to perform feature cross and obtain cross features. Then the cross and continuous features are stitched together and used as the input of the second fully connected layer. The output is a vector.

A. CROSS NETWORK

In many competitions on Kaggle, most winning solutions use manual feature engineering to construct low-order combinatorial features that are meaningful and efficient [14]. In contrast, the features learned by DNN are highly non-linear, and high-order combinatorial features were very difficult to interpret in a meaningful way [15]. The DCN (Deep Cross Network) [16] using cross networks is then designed to improve the efficiency of learning features. Figure 1 shows only the structure of the cross network, and the output of the deep cross network is concatenated with the output of the cross network and the deep network.

The original feature vector of the output and input of the fully connected layer is used as the input of the crossover layer to perform feature crossover and obtain crossover features. Then, the crossover and continuous features are merged and used as the input of the second fully connected layer. The output is a vector [17].

Each cross layer is crossed using the following formula:

$$x_{l+1} = x_0 x_l^T w_l + b_l + x_l = f(x_l, w_l, b_l) + x_l \tag{1}$$

where $x_l, x_{l+1} \in R^d$ are column vectors denoting the outputs of the l-th and (l+1)-th cross layers, respectively, and $W_l, b_l \in R^d$ are the weight and bias parameters of the l-th layer. Each cross layer adds back its input after a feature crossing operation f.

B. EMBEDDING LAYER

The embedding method represents an object as a low-dimensional vector (a word, a product, a movie, etc.). If one-hot is used to encode the category, id-type features, it leads to extremely sparse sample feature vectors, and the structural features of deep learning make it unfavorable for processing sparse feature vectors. Here, embedding plays

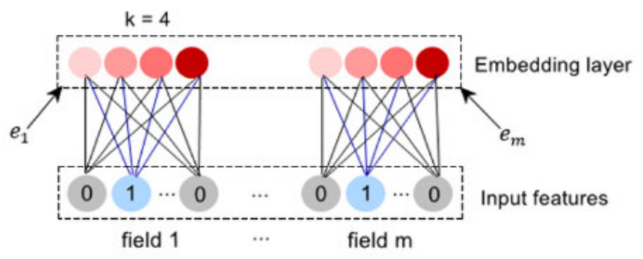

FIGURE 2. The structure of the embedding layer, k is the dimension of the embedding.

a very good role by converting numerical features into dense vectors, and the distance between vectors reflects the similarity between objects. The distance between objects with low similarity is generally larger.

The input features in a recommendation system are usually in the form of strings, and we first use field-aware one-hot encoding to convert the raw features into a high-dimensional sparse feature [18]. Figure 2 shows the structure of the embedding layer. The output of the embedding layer can be described as:

$$a_0 = [e_0, e_1, \ldots e_m] \tag{2}$$

where e_i is the embedding of the i-th patch and m is the number of patches. In this model, the implicit high-order interaction part and the explicit high-order interaction part share the same feature embedding. All inputs are mapped to the same embedding space, which allows the model to better understand the semantics and features of the input data. The number of parameters to be trained is reduced, reducing the time and computational resources required to train the model.

C. DEEP COMPONENT

This part uses Deep Neural Networks (DNN) to capture the implicit higher order interactions (e.g. second and third order interactions). The embedding vectors are combined using feedforward neural networks to generate higher order interaction features. The feedforward process is expressed as:

$$a_{(l+1)} = f(W^l a^l + b^l) \tag{3}$$

where l is the layer depth and σ is an activation function. a^l is the output of the l-th layer, W^l is the weight of the l-th layer, b^l is the bias of the l-th layer. f is often rectified linear units (ReLUs).

The structure of the deep neural network (DNN) in this paper is shown in Figure 3. Deep neural networks (DNNs) have a wide range of applications in both image recognition and CTR prediction, but the applications in these two areas are very different. The most important is the difference in data type and network structure. The input data type for CTR prediction is usually sparse data (e.g. user's gender, age, historical click behavior, etc.); the main input for image recognition is continuous data (e.g. pixel values, RGB values, etc.). The second is the difference in network structure. CTR prediction usually uses the structure of a fully connected network or embedding and fully connected network, while image recognition uses the structure of a convolutional neural network (CNN). This is because image data has spatial continuity, and CNN is more effective at extracting features from images.

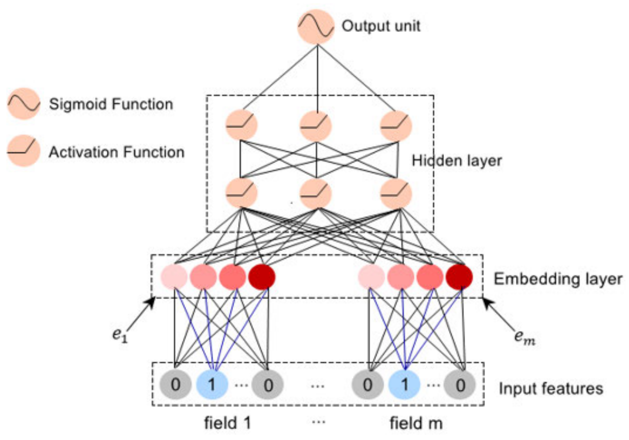

FIGURE 3. The architecture of DNN, each hidden layer is calculated as equation 3.

III. METHOD

The LRCIN model mainly improves the Compressed Interaction Network by introducing low-rank approximation and attention mechanism based on the xDeepFM model. The number of parameters and computational complexity are reduced by introducing low-rank approximation in the CIN network. The approximation is performed using two weight matrices W[i] and V[i], which optimizes the computational efficiency of the CIN layer. And the attention mechanism unit is added after the output part of CIN. The attention mechanism is used to dynamically learn the weights between features to better capture the interactions between features. The final output is a combination of the attention-weighted cross-feature representation combined with the outputs of the deep neural network and the linear part.

The low-rank approximation reduces model parameters and computational complexity and is suitable for large-scale data processing. The attention mechanism enhances the model's ability to automatically recognize the importance of features and improves the model's interpretability and accuracy. To perform the feature selection process, Field-aware One-hot Encoding is used to transform the original features into high-dimensional sparse vectors, and Embedding Layer transforms the sparse vectors into dense embeddings that capture the implicit relationships between the features. In the experiments, all numerical features are first normalized to ensure the numerical stability of the model training. And the outliers and missing values in the data are handled to ensure the integrity of the data using appropriate padding strategies.

A. COMPRESSED INTERACTION NETWORK WITH LOW-RANK APPROXIMATION (LRCIN)

This section uses a low-rank approximation of the original Compressed Interaction Network (CIN) to reduce its computational complexity. Algorithm 1 describes the entire process.

Algorithm 1 Compressed Interaction Network With Low-Rank approximation(LRCIN)

```
Require: inputs with shape $[None, n, k]$, cin_size
Ensure: output with shape $[None, \sum_{i=1}^{n} field\_num[i], k]$
 1: Initialize CIN Layer with cin_size
 2: Build method:
 3: Set input_shape: $[None, n, k]$
 4: Set field_num: $[input\_shape[1]] + cin\_size$
 5: for $i$ in range(len(field_num)-1) do
 6:     Initialize W[$i$] with shape $(1, field\_num[0] * field\_num[i], field\_num[i+1])$
 7:     Initialize V[$i$] with shape $(1, field\_num[0] * field\_num[i], field\_num[i + 1])$ (for low-rank approximation)
 8: end for
 9: Call method:
10: Split inputs along the last dimension into $k$ parts: $X0$
11: for each layer $i$ in range(len(field_num[1:])) do
12:     Split res_list[-1] along last dimension into $k$ parts: $Xi$
13:     Perform matrix multiplication between $X0$ and $Xi$, transpose $Xi$ before multiplication
14:     Reshape the result and transpose it
15:     Apply low-rank approximation using W[$i$] * V[$i$] and perform 1D convolution with input $x$
16:     Transpose the result and append it to res_list
17: end for
18: Remove the first element ($X0$) from res_list
19: Concatenate res_list along axis 1
20: Set output as the concatenated result
```

Algorithm 1 describes this process of improving cross-interaction networks using low-rank approximations, and I'll explain each step in detail below.

The first step is to initialize the CIN layer with inputs of the form [N,n,k], where N is the number of samples, n is the number of fields, and k is the embedding dimension. And cin_size is the number of output fields for each CIN layer. Then comes the build method section, which sets the input shape 'input_shape' to [N,n,k]. Set 'field_num' to the number of fields in the input layer n plus 'cin_size'. The next step is to initialize the weights. For each CIN layer i, two weight matrices W[i] and V[i] are initialized for the low-rank approximation.

Then comes the implementation section. First, the input is split into k parts along the last dimension to obtain $X0$, and then the following operations are performed for each CIN layer i Split the output of the previous layer 'res_list[-1]' into k parts along the last dimension to obtain Xi, perform matrix multiplication of X0 and Xi, where Xi is transposed before multiplication, after which the result of the multiplication is reshaped and transposed, and a low-rank approximation is applied to the input x using $W[i] \cdot V[i]$ to perform 1D convolution, and finally the convolution result is transposed and added to 'res_list'.

Finally, the output result part, starting from the first element $X0$ in a place in 'res_list', joins all elements of 'res_list' along the first dimension (axis 1), and the output is the result after joining.

B. ATTENTION MECHANISM

We add an attention activation unit after the LACIN, get the attention weights, sum the output of the CIN to get the attention vector, and then splice the attention vector with the output of the CIN to get the final attention output. Since the introduction of the attention mechanism in neural network modeling, it has been widely used in many tasks such as recommendation, information retrieval, and computer vision [19]. Motivated by the drawbacks of CIN, we decided to obtain the output by weighting the crossing features with attention, thus selecting the more important features for crossing. Specifically, attention weights can be computed separately for each feature cross term in the CIN network, and then the attention-weighted cross terms are accumulated. Meanwhile, the computational complexity of the model can be reduced by using the low-rank approximation method. The attention mechanism in this method is explained below. The formula of the attention mechanism is:

$$O = (A^T \cdot X) = [softmax(h \cdot ReLU(W \cdot X))]^T \cdot X \tag{4}$$

X is the input feature matrix (the form is [None, Field, k]), W is the attention weight matrix (the form is [None, Field, Field]), h is the matrix for calculating the feature weights (the form is [None, Field, 1]), A is the attention score matrix (the form is [None, Field, 1]), O is the output feature vector (the form is [None, k]), A^T is the transposed attention score matrix (the form is [None, 1, Field]). First, the ReLU function is applied to the input features X to perform a weighting operation, and then the weights are computed and entered to compute the attention score. The softmax($\cdot$) function normalizes the weights between 0 and 1, so that the sum of the weights is 1. Finally, the attention score is multiplied by the input features to obtain the weighted feature vector O.

C. COMBINATION WITH ALL LAYERS

In this section, we will connect all the layers mentioned above to form a new model, the overall structure of which is shown in Figure 5. In this model, we connect all the outputs together, as can be seen in the figure, our LRCIN method evolves from the original CIN method and takes the outputs as inputs to the attention unit, which is finally connected with the output of the DNN and the linear outputs to form the final output unit:

$$y = \sigma(y_{LR} + y_{ATT} + y_{DNN}) \tag{5}$$

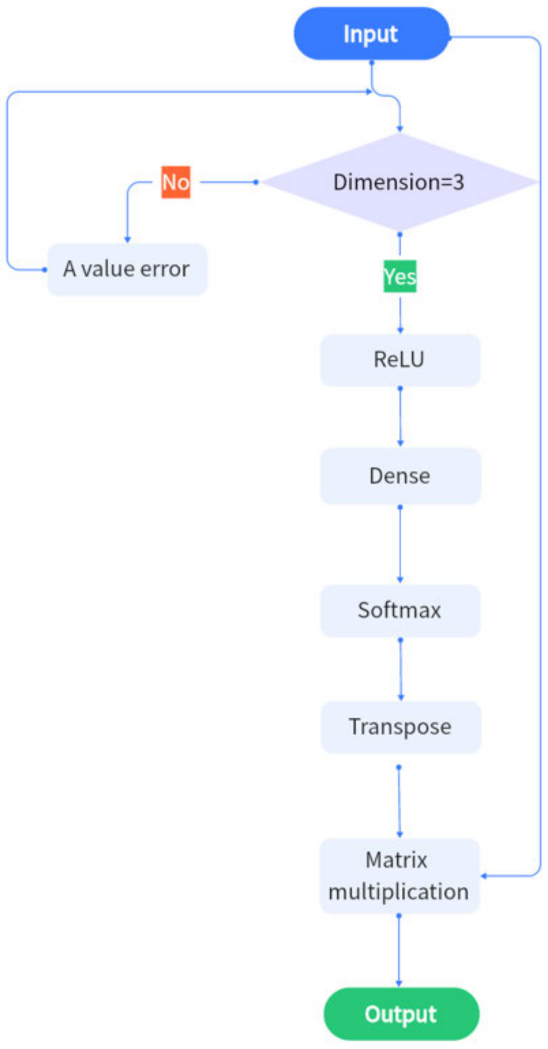

FIGURE 4. This figure shows the flowchart of the attention part, which includes weighting the input features, calculating the feature weights, normalizing the feature weights using the softmax function to obtain the attention score, and matrix multiplying the attention score with the input features to obtain the weighted feature representation. Finally, the tensor shape is fitted to obtain the output tensor.

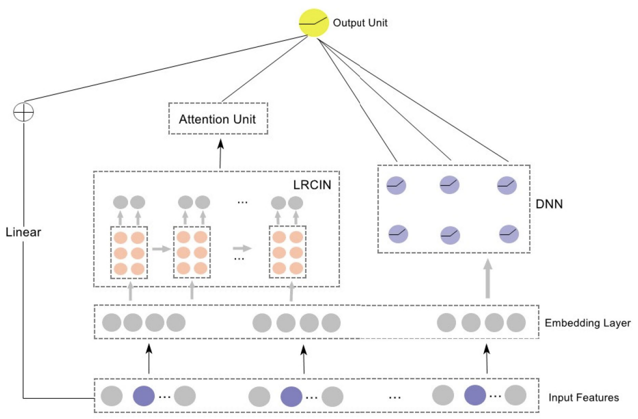

FIGURE 5. This illustration shows the entire process of model.

where σ is the sigmoid function, y_{LR}, y_{ATT}, y_{DNN} are the outputs of the linear, attention, and DNN layers, respectively.

IV. EXPERIMENT AND RESULTS

In this section, the parameter selection and results of our experimental method are described in detail.

A. DATASETS

The Criteo Dataset is a famous industry benchmark dataset used to develop models for predicting ad click-through rates and is publicly available [20]. Given a user and the page they are visiting, the goal is to predict the probability that they will click on a given ad. The dataset contains click records, each record consisting of feature values and labels (clicked or not). Data cleaning is performed first: processing missing values, removing duplicate records and outliers. Most of the feature values are categorical and need to be coded. The click rate in the dataset is typically around 0.1%, the distribution of certain categorical features is extremely uneven, the number of occurrences of some categories is extremely high, and the distribution of most numerical features is concentrated in a small range with a long-tailed distribution.

The MovieLens dataset contains rating data from multiple users for multiple movies, as well as movie metadata and user attribute information [21]. Data cleaning operations are performed to ensure that the user ID, movie ID, and rating of each record are valid. Ratings are typically between 3-4, with a small number of extremely high and low ratings.

The Avazu dataset, provided by Avazu Corporation, including information on whether a user clicked on an ad, and is a widely used dataset in the ad tech industry for predicting click-through rates. Click-through rates are typically around 0.02%, with a relatively even distribution of attributes across most categories, but some attribute values are more common. Certain feature values are significantly correlated with higher click-through rates, suggesting an important influence on click-through behavior.

Tables 1 and 2 provide basic information about the statistical analysis of Criteo, MovieLens, and Avazu datasets.Criteo and Avazu datasets are mainly used for click-through-rate prediction, with a large number of features and high sparsity, and the main feature type is categorical. MovieLens dataset is mainly used for movie recommendation, with a small number of features, medium sparsity, and contains both categorical and continuous features. Category-type and continuous-type features. The Criteo dataset has the highest number of instances at 45.8 million, while the MovieLens dataset has the lowest number of instances at 10 million.

B. EVALUATION

The experiments use AUC and LogLoss as metrics to evaluate the model. AUC is the area under the ROC curve, which measures the probability that the positive cases predicted

TABLE 1. Specific information about the datasets.

Datasets	Features	Fields	Instances
Cretio	39	39	45.8M
MovieLens	3	7	10.0M
Avazu	24	24	40.4M

TABLE 2. Summary of Key Characteristics of Criteo, MovieLens, and Avazu Datasets.

Dataset	**Purpose**	**Sparsity**	**Feature Types**
Criteo	Predict CTR	High	Categorical, some continuous
MovieLens	Movie recommendations	Medium	Categorical and continuous
Avazu	Predict CTR	Very high	Mainly categorical

TABLE 3. The AUC results between this model and other traditional CTR models on Criteo, Movielens and Avazu datasets.

Datasets	Criteo	Movielens	Avazu
FM	0.7312	0.7160	0.7089
PNN	0.7432	0.7210	0.7129
Wide&Deep	0.7580	0.7643	0.7602
DeepFM	0.7740	0.7790	0.7659
xDeepFM	0.8136	0.7970	0.7820
LRCIN	0.8230	0.8078	0.8013

by the model are in front of the negative cases, and the higher the value of AUC, the better the binary classification performance [22]. LogLoss, as logarithmic loss, is also a widely used evaluation metric in the binary classification task, which can measure the gap between the predicted results of the CTR and the true value, and the lower the value of LogLoss, the better the performance of the model.

C. PARAMETER SETTING

Each user's click time is used to form a user click sequence of items and item categories, where the last bit of the sequence is the target to be predicted. The sequence of items not clicked by the user in each sample is randomly selected in a positive to negative ratio of 1:1, forming a pair of positive and negative samples. The length of the historical click sequence is too short, resulting in the loss of effective feature information, and too long, introducing too much noise. Experiments were run in increments of 10 to find the best length setting of 100, with the missing portion filled with 0 and the excess portion intercepted. The data is randomly divided into two parts, 80% for training and 20% for testing, the epoch is set to 100, and the learning rate is set to 0.01. Optimization is done with the Adam optimizer. A mini-batchware regularization was added to avoid overfitting.

D. RESULTS AND ANALYSIS

Table 3 shows the results of the AUC comparison between this model and other traditional CTR models on the Criteo, Movielens and Avazu datasets. We can see that the difference between the AUC indexes of BaseModel and PNN is not too big, but the difference is quite big when we go to

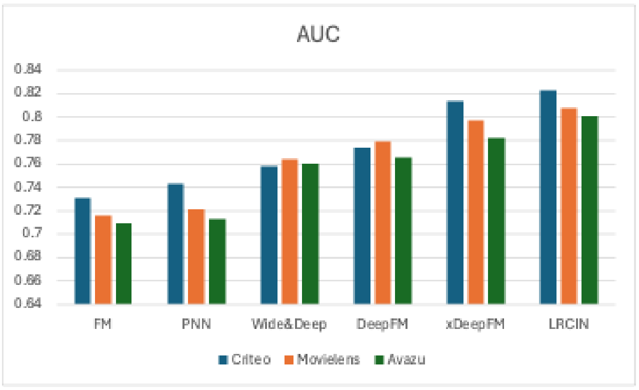

FIGURE 6. AUC results.

TABLE 4. The Logloss results between this model and other traditional CTR models on Criteo, Movielens and Avazu datasets.

Datasets	Criteo	Movielens	Avazu
FM	0.5638	0.5071	0.4782
PNN	0.5490	0.4930	0.4603
Wide&Deep	0.5700	0.4752	0.4056
DeepFM	0.5248	0.4686	0.4039
xDeepFM	0.5032	0.4408	0.3958
LRCIN	0.4851	0.4258	0.3971

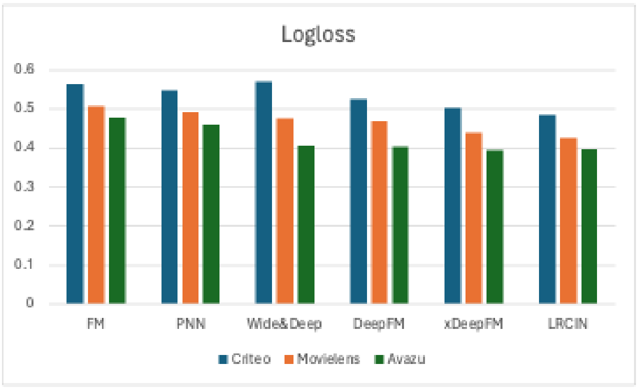

FIGURE 7. Logloss results.

Wide& Deep, and then DeepFM changes the manual feature engineering in Wide & Deep to automatic, which improves the effect to some extent, and then xDeepFM optimizes the feature explicit crossover by adding several groups of different order explicit crossover, which improves the AUC score significantly. We can see that the effect of this method on the Criteo dataset is higher than the AUC score on the Movielens dataset, mainly due to the fact that the Criteo dataset has a large number of sparse features as well as a much larger scale than the Movielens dataset, which makes it very suitable for deep neural network model learning. We can see that the Wide & Deep model performs better on Movielens than on Criteo, mainly because the Wide & Deep model is closer to its original design for the recommendation task. While on the Criteo dataset, it may face more challenges due to its high sparsity and size.

TABLE 5. Accuracy, Recall, F1 Score using Wide & Deep, xDeepFM, LRCIN on Criteo dataset.

Model	Accuracy	Recall	F1-score
Wide&Deep	0.685	0.702	0.693
xDeepFM	0.732	0.749	0.740
LRCIN	0.745	0.762	0.753

Table 4 shows the results of LogLoss comparison between this paper's model and other traditional CTR models, and it can be seen that the LogLoss of this paper's model is significantly smaller than other models, which proves the superiority of this paper's model.

Table 5 shows the precision, recall and F1 score using Wide & Deep, xDeepFM and LRCIN on the Criteo dataset. It can be seen that our model performs better.

The AUC results of the LRCIN model on all three datasets outperform those of the other models, demonstrating its effectiveness in improving the accuracy of CTR prediction. Specifically, LRCIN achieves an AUC of 0.8230 on the Criteo dataset, which is significantly higher than other comparative models such as xDeepFM and DeepFM, which are 0.8136 and 0.7740, respectively. The LogLoss results also show the superiority of the LRCIN model. On the Criteo dataset, the LogLoss of LRCIN is 0.4851, which is lower than that of xDeepFM at 0.5032 and DeepFM at 0.5248, indicating that LRCIN performs better in terms of both prediction accuracy and error rate. LRCIN also shows better performance on the Movielens dataset, especially in the recommender system task, where its design allows it to better understand the complex relationship between user preferences and movie attributes. In the field of advertising technology, especially when dealing with ad click data such as Avazu, LRCIN is able to effectively handle large and high-dimensional datasets, which are crucial for the real-world application of ad placement and optimization strategies.

These results suggest that by introducing the low-rank approximation and attention mechanism, the LRCIN model is not only more effective in handling feature interactions, but also demonstrates higher operational efficiency and prediction accuracy in resource-constrained environments. The design of LRCIN allows it to better capture and exploit the information in sparse data, which is difficult to achieve with traditional models such as FM and PNN.

E. TIME COMPLEXITY ANALYSIS

Let us first analyze the time complexity of the original CIN. In CIN, the operations in each layer are roughly matrix multiplication. If we consider the worst case (i.e., we have the same number of fields in each layer), then the complexity of each layer is O(n^2k), where n is the number of fields and k is the dimension of the embedding. The key to the low-rank approximation is to reduce the number of multiplications required. If there are L layers, the total time complexity is O(Ln^2k). In LRCIN, the purpose of a low-rank approximation is to reduce the number of multiplications required in each layer. By using an approximation of rank r instead of the original embedding dimension k. For L layers, the total time complexity is O($Ln(r^2 + nrk)$).

TABLE 6. Comparison of Time Complexity between CIN and LRCIN.

Model	Description	Time Complexity
CIN	Traditional approach with full embeddings	$O(Ln^2k)$
LRCIN	Low-rank approximation applied	$O(Ln(r^2 + nrk))$

In the following, we compare the time complexity of the two methods. The complexity of LRCIN is significantly lower than that of normal CIN when the approximation rank r is much smaller than the number of fields n and the embedding dimension k. And in LRCIN, since the r^2 terms and the rk terms are usually smaller compared to n^2k, this reduces the size of the matrix operation and improves computational efficiency.

As show in table 6, for CIN, the time complexity is quadratic in the number of fields and linear in the number of layers and the embedding dimension. For LRCIN, it is shown that a low-rank approximation reduces the complexity and is more computationally efficient. Here, n represents the number of fields, k the dimension of embedding, L the number of layers, and r the rank of approximation.

V. CONCLUSION

In this paper, we propose an innovative algorithm called Low-Rank Approximation for Compressed Interaction Networks (LRCIN), which focuses on the CIN layer and uses low-rank approximation techniques to compress it efficiently and capture the key feature interactions to reduce the computational complexity. Specifically, by performing a series of accurate and optimized matrix operations on the input data, such as transposition, matrix multiplication, and convolution, not only is the size of the model reduced, but the performance of the model is maintained at the same time. In addition, by using low-order approximations of the W[i] and V[i] matrices, we further optimize the memory and computational efficiency of the model.LRCIN not only provides a new perspective on deep learning model compression, but also lays a solid foundation for future research, especially in application domains that target large datasets and require high computational performance. However, the efficiency and accuracy of LRCIN still need to be improved when dealing with extremely sparse datasets. In the future, the combination of other compression techniques can be explored to further reduce the model's memory requirements and improve the model's execution speed.

Although LRCIN effectively reduces computational complexity and model size, the method relies on appropriate rank selection; choosing too low a rank may weaken the learning ability of the model, while choosing too high a rank may not effectively reduce the consumption of computational resources. Therefore, how to balance approximate rank and model performance is a key issue. One can consider introducing a mechanism that dynamically feeds back the size of the rank based on the error in the training process and adjusts it at any time.

REFERENCES

[1] L. Yu, P. Cui, F. Wang, C. Song, and S. Yang, "From micro to macro: Uncovering and predicting information cascading process with behavioral dynamics," in *Proc. IEEE Int. Conf. Data Mining*, Nov. 2015, pp. 559–568.

[2] A. Wang, V. V. Ramaswamy, and O. Russakovsky, "Towards intersectionality in machine learning: Including more identities, handling underrepresentation, and performing evaluation," in *Proc. ACM Conf. Fairness, Accountability, Transparency*, Jun. 2022, pp. 336–349.

[3] D. Liang, C.-F. Tsai, and H.-T. Wu, "The effect of feature selection on financial distress prediction," *Knowledge-Based Syst.*, vol. 73, pp. 289–297, Jan. 2015.

[4] Y. Xiao, W. He, Y. Zhu, and J. Zhu, "A click-through rate model of e-commerce based on user interest and temporal behavior," *Expert Syst. Appl.*, vol. 207, Nov. 2022, Art. no. 117896.

[5] X. Xia and R. Tong, "Click-through rate prediction based on feature importance and feature interaction," in *Proc. 2nd Int. Symp. Comput. Appl. Inf. Syst. (ISCAIS)*, Jun. 2023, pp. 86–91.

[6] S. Rendle, "Factorization machines," in *Proc. IEEE Int. Conf. Data Mining*, Dec. 2010, pp. 995–1000.

[7] H. T. Cheng, L. Koc, and J. Harmsen, "Wide & deep learning for recommender systems," in *Proc. 1st Workshop Deep Learn. Recommender Syst.*, 2016, pp. 7–10.

[8] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "DeepFM: A factorization-machine based neural network for CTR prediction," 2017, *arXiv:1703.04247*.

[9] J. Lian, X. Zhou, F. Zhang, Z. Chen, X. Xie, and G. Sun, "XDeepFM: Combining explicit and implicit feature interactions for recommender systems," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 1754–1763.

[10] N. Kishore Kumar and J. Schneider, "Literature survey on low rank approximation of matrices," *Linear Multilinear Algebra*, vol. 65, no. 11, pp. 2212–2244, Nov. 2017.

[11] G. Brauwers and F. Frasincar, "A general survey on attention mechanisms in deep learning," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 4, pp. 3279–3298, Apr. 2023.

[12] W. Song, C. Shi, Z. Xiao, Z. Duan, Y. Xu, M. Zhang, and J. Tang, "AutoInt: Automatic feature interaction learning via self-attentive neural networks," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, Nov. 2019, pp. 1161–1170.

[13] Z. Li, W. Cheng, Y. Chen, H. Chen, and W. Wang, "Interpretable click-through rate prediction through hierarchical attention," in *Proc. 13th Int. Conf. Web Search Data Mining*, Jan. 2020, pp. 313–321.

[14] R. Wang, B. Fu, G. Fu, and M. Wang, "Deep & cross network for ad click predictions," in *Proc. ADKDD*, Aug. 2017, pp. 1–7.

[15] L. Guo, Y. Yu, H. Gao, T. Feng, and Y. Liu, "Online remaining useful life prediction of milling cutters based on multisource data and feature learning," *IEEE Trans. Ind. Informat.*, vol. 18, no. 8, pp. 5199–5208, Aug. 2022.

[16] J. Yu, W. Liu, M. Zhou, Y. Chen, D. Ji, and N. Sai, "Recommendation ranking method combining graph convolutional network and factorization machine," in *Proc. 10th Int. Conf. Intell. Comput. Wireless Opt. Commun. (ICWOC)*, Jun. 2022, pp. 55–62.

[17] F. Khawar, X. Hang, and R. Tang, "Autofeature: Searching for feature interactions and their architectures for click-through rate prediction," in *Proc. 29th ACM Int. Conf. Inf. Knowl. Manag.*, 2020, pp. 625–634.

[18] P. P. K. Chan, X. Hu, and L. Zhao, "Convolutional neural networks based click-through rate prediction with multiple feature Sequences," in *Proc. IJCAI*, 2018, pp. 2007–2013.

[19] J. Xiao, H. Ye, X. He, H. Zhang, F. Wu, and T.-S. Chua, "Attentional factorization machines: Learning the weight of feature interactions via attention networks," 2017, *arXiv:1708.04617*.

[20] J. Zhu, J. Liu, and S. Yang, "Open benchmarking for click-through rate prediction," in *Proc. 30th ACM Int. Conf. Inf. Knowl. Manag.*, 2021, pp. 2759–2769.

[21] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Trans. Interact. Intell. Syst.*, 2015, vol. 5, no. 4, pp. 1–19.

[22] D. Chicco and G. Jurman, "The Matthews correlation coefficient (MCC) should replace the ROC AUC as the standard metric for assessing binary classification," *BioData Mining*, vol. 16, no. 1, pp. 1–23, Feb. 2023.

YIWAN ZHANG received the B.S. degree in software engineering from Nanyang Institute of Technology, China, in 2022. She is currently pursuing the master's degree in computer science with Hanyang University, South Korea. Her research interests include recommendation systems and explainable AI.

ZHAN WANG received the B.S. degree in computer science and technology from Zhengzhou University of Light Industry, in 2020, China. She is currently pursuing the integrated master's and Ph.D. degree in computer science with Hanyang University, South Korea. Her research interests include machine learning, deep learning, computer vision, and explainable artificial intelligence.

INWHEE JOE received the B.S. degree in electronics engineering from Hanyang University, Seoul, South Korea, and the Ph.D. degree in electrical and computer engineering from Georgia Institute of Technology, Atlanta, GA, USA, in 1998. Since 2002, he has been a Faculty Member with the Division of Computer Science and Engineering, Hanyang University. His current research interests include the Internet of Things, cellular systems, wireless power communication networks, embedded systems, network security, machine learning, and performance evaluation.

• • •