applied sciences

MDPI

Article

Deep Learning and Embedding Based Latent Factor Model for Collaborative Recommender Systems

Abebe Tegene [1,2], Qiao Liu [1,*], Yanglei Gan [1], Tingting Dai [1], Habte Leka [3] and Melak Ayenew [3]

[1] School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China
[2] CoE of Big Data Analytics and HPC, Addis Ababa Science and Technology University, Addis Ababa P.O. Box 16417, Ethiopia
[3] School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China
* Correspondence: qliu@uestc.edu.cn

Abstract: A collaborative recommender system based on a latent factor model has achieved significant success in the field of personalized recommender systems. However, the latent factor model suffers from sparsity problems. It is also limited in its ability to extract non-linear data features, resulting in poor recommendation performance. Inspired by the success of deep learning in different application areas, we incorporate deep learning into our proposed method to overcome the above problems. In this paper, we propose a dual deep learning and embedding-based latent factor model that considers dense user and item feature vectors. The model combines the existing deep learning and latent factor models to extract deep abstractions and non-linear feature representations of the data for rating prediction. The core idea is to map the dense user and item vectors generated by embedding techniques into dual, fully connected deep neural network architectures. In these two separate architectures, it learns the non-linear representation of the input data. The method then predicts the rating score by integrating the factors obtained from the two independent structures using the inner product. From the experimental result, we observe that the proposed model outperformed state-of-the-art existing models in real-world datasets (MovieLens ML-100K and ML-1M).

Keywords: deep learning; latent factors; collaborative filtering; recommender systems

Citation: Tegene, A.; Liu, Q.; Gan, Y.; Dai, T.; Leka, H.; Ayenew, M. Deep Learning and Embedding Based Latent Factor Model for Collaborative Recommender Systems. *Appl. Sci.* **2023**, *13*, 726. https://doi.org/10.3390/app13020726

Academic Editor: Yu-Dong Zhang

Received: 23 November 2022
Revised: 27 December 2022
Accepted: 27 December 2022
Published: 4 January 2023

1. Introduction

Nowadays, consumers have a large variety of possibilities, such as books, restaurants, movies, and other entertainment options, thanks to the rapid growth of internet services, which has resulted in a problem of information overload [1]. Recommender Systems (RSs) are algorithms that contribute to the resolution of the problem of information explosion. They are smart algorithms that use the past preferences of a user to suggest a similar product to other users [2]. Therefore, using efficient and accurate RSs is not questionable.

In a personalized recommendation system, recommendations are based on past interactions between the user and the item, either explicit or implicit feedback interactions called collaborative filtering (CF). This type of recommendation is often used since it effectively captures users' preferences. However, the collaborative filtering technique has data sparsity problems [3]. Additionally, due to their inability to use the item content, CF algorithms have a limited scope for producing explanations for the recommendations [4]. To overcome these, researchers have incorporated additional attributes into collaborative filtering. Ref. [5], for example, integrates user and item attributes, whereas [6] used contexts as additional information to improve performance. According to collaborative filtering methods, users may be curious about items selected by people with comparable interaction habits to them. To implement CF-based methods, we need interaction data from various sources, such as users and items, and then we form a user–item interaction matrix.

Memory- and model-based approaches are two commonly used collaborative filtering-based techniques [1]. Memory-based (also called neighborhood-based) CF makes predictions based on the nearest neighborhoods. Neighborhood-based methods are further classified into two types: user-based methods and item-based methods [7]. User-based CF will give a recommendation based on the user's past interaction history. On the other hand, item-based CF learns the relationships between items and recommends comparable items to a user based on their interaction history. In model-based CF, the latent factor model (LFM) is one of its variants, which captures the user and item's latent representations from high-dimensional data [8].

The matrix factorization (MF) technique is one of the successful and efficient methods in LFM. This method decomposes high-dimensional data into low-rank data [9], and the method projects the latent factors into a shared latent space. Then, it predicts users' preferences for items in this space by taking the inner product between user latent and item latent vectors [10,11].

To improve the efficiency of matrix factorization, several academics have conducted a range of studies. For example, the work given in Ref. [12] uses additional features like item content to address the problem of sparsity, whereas [13,14] combine, social matrix factorization with social relations to boost recommendation quality. Despite its high quality, it has low computational performance for large amount of data, scalability problems for sparse data, and information extraction problems for non-linear data. When there is a lack of rating data, matrix factorization frequently results in poor recommendation quality. Contrarily, a large amount of data will make computing more difficult. As a result, it is necessary to look for various alternative models to increase the quality of recommendations.

Deep learning (DL), a new discipline, has achieved astounding success in several domains, especially in natural language processing [15–17]. Recently, recommendation systems have benefited from the deployment of DL, which provides new possibilities for enhancing the effectiveness of RSs. One benefit of deep learning is its ability to handle non-linear data representations. It can boost computational power and handle massive amount of data [18]. It can also aid in solving sparsity problems, which results in higher recommendation quality and performance [19]. In general, the work done by [20] has mentioned several specific benefits of deep learning. It allows for the modeling of non-linearity in data using a variety of non-linear activation functions. Deep neural networks (DNNs) enable models to incorporate a variety of information while reducing the time required for feature generation.

In this article, we look at the advantages of deep learning architecture and provide a new approach for improving RS performance. We present such a dual DL and embedding-based latent factor model for recommender systems, (shortened as DELCR). To improve recommendation performance, DELCR employs embedding and deep neural network architectures to determine the latent factors. The model combines known deep learning and latent factor models to extract deep abstractions and non-linear feature representations of the data for rating prediction. The core idea is to map the dense user and item vectors generated by embedding techniques into two fully connected deep neural network (DNN) architectures. We used the concept from the work cited in [8]. For the proposed model, we embed the interaction rating matrix to obtain a dense vector that overcomes the sparsity problem, and we developed the model to predict the rating score.

In general, in most existing deep learning-based recommender system designs, input feature vectors are concatenated and projected into a common DNN architecture to learn the abstraction of the important features. According to our argument, the approach is insufficiently resilient to extract the required information from the data. Because users have naturally complex abstractions, we must understand these phenomena separately to obtain the appropriate information. We argue that such a strategy is insufficiently robust to learn the required features from the data. Existing approaches did not employ separate architectures for these two different elements. However, the proposed model transforms the dense feature vector obtained from embedding techniques into separate deep neural

networks. In addition, we will integrate the inner product into the network's output layers for the sake of obtaining the interaction rating results. The newly designed model, DELCR, efficiently learns the sparse data set in the embedding layer and the non-linear feature of the data in their corresponding DNN structure, which results in the quality of the recommendation. Thus, we can infer that the use of these two separate architectures will make a significant difference in the performance of the model.

The main contributions of our work are as follows:

- We develop a dual DNN structure to extract the non-linear representation of users' and items' latent factors in their respective spaces separately.
- We construct a robust DL and embedding-based latent factor model. The method effectively alleviates sparsity problems by combining deep learning and embedding techniques.
- We conduct an extensive experiment on two real-world datasets to predict the rating score. The result demonstrates that DELCR improves the performance of the recommendation to the state-of-the-art level.

The remainder of the paper is organized as follows: Section 2 includes preliminary work, while Section 3 includes related work. Section 4 presents the proposed model. Then, the experimental result is described in Section 5. Finally, the conclusion is mentioned in Section 6.

2. Preliminaries

2.1. Problem Statement

In this section, we develop a user-item historical interaction rating matrix Y from the user's explicit feedback as follows:

Let set of users $U = \{u_1, u_2, u_3, \ldots, u_m\}$ and set of items $V = \{v_1, v_2, v_3, \ldots, v_n\}$ denote m and n total number of users and items in the matrix, respectively. Following [8], we set $R \in \mathbb{R}^{m \times n}$ to be the interaction matrix between user and item, where $y_{ij} \in R$ represent the preference score of user i for item j. Then, we define matrix Y using Equation (1)

$$r_{ij} = \begin{cases} y_{ij} & \text{if } y_{ij} \in R. \\ unknown & otherwise \end{cases} \tag{1}$$

2.2. Learning the Model

Existing parameter estimation methods are typically based on optimizing an objective function. Pointwise loss and pairwise loss are the two forms of objective functions that are most frequently employed in the literature [21]. Pointwise learning techniques are often used as a regression framework by minimizing the Frobineous norm. As a result, we optimized Equation (2) in the embedding space to determine user latent vectors in matrix P and item latent vectors in matrix Q.

$$\min_{P,Q} \|R - PQ^T\|_F^2. \tag{2}$$

Any recommendation task must have a robust objective function for model optimization. Thus, in the proposed method, we incorporate the loss function in Equation (3) along with (2) to learn the model parameters for optimization because it performs very well with pointwise explicit data types [22].

$$L = \min_{p,q} \left[\sum_{i,j} (\hat{r}_{ij} - r_{ij})^2 + \lambda W^2 + \alpha b^2 \right]. \tag{3}$$

where $\hat{r}_{ij}$ is the user i's rating for the item j estimated by the model. We use L_2 regularization terms to avoid over-fitting and enhance the model's predictive performance for unseen data. The regularization values related to hidden-layer's weight W and bias b are represented by the positive numbers λ and α, respectively.

3. Related Work

3.1. Latent Factor Model for Recommender Systems

The matrix factorization model is one of the well-known techniques in the latent factor model. It is also a popular research topic in collaborative filtering since it frequently produces a good low-dimensional latent feature representation of users and items. The models generate a latent space from each user and item vector. Finally, it predicts the elements that were missed using the inner product. To demonstrate this concept, consider the latent factors of a user p and an item q, respectively. Then, the method estimates the approximate rating score, $\hat{r}_{ij}$, by applying Equation (4) [11].

$$\hat{r}_{ij} = p_i q_j^T = \sum_{k=1}^{K} (p_{ik} q_{jk}), \tag{4}$$

where K represents the embedding dimension that must be taken into account.

The latent factors p_i and q_j in most MF-based models are calculated using gradient descent methods to minimize the square error between the true value r_{ij} and the estimated values $\hat{r}_{ij}$ [11,14,23] on Equation (5).

$$\sum_{i,j} e_{ij}^2 = \min_{p_i, q_j} \sum (r_{ij} - \hat{r}_{ij})^2. \tag{5}$$

According to [14], SVD is one of the most effective matrix factorization approaches for enhancing reliability and scalability issues. To improve rating prediction, biased SVD introduced biasing features to the model [14]. By decomposing the rating matrix, the SVD++ algorithm achieved excellent prediction performance [23]. However, these methods are not robust because they require too many iterations to discover the proper latent features. The author of [24] suggested using large values of latent dimension K to overcome this. However, in the case of a sparse rating matrix, this also has an impact on the recommendation's quality. As a result, we use a dual deep learning network architecture to extract these latent features in our proposed method, which is well suited to tackling those problems.

3.2. Deep Learning-Based Recommender Systems

The main problem of the recommender task is the non-linear interaction and large amount of data between users and items, making feature learning a difficult task. This in turn makes the learning process slow, as it requires much more time to train. In addition, it affects the computational power. Therefore, we will employ deep learning frameworks that can handle the above-stated challenges and improve the quality performance of the RSs. For instance, the author of [21], in Neural Collaborative Filtering (NCF), showed that the inner product could be automatically learned using the multi-layer perceptron (MLP). This implicitly learns the non-linear feature representation between users and items, which is the MF-based models' main weakness. DeepFM [25] integrated factorization machine (FM) and MLP to build an end-to-end framework for learning both low and high-dimensional feature data. Due to this, many companies employ RSs to improve the user experience and boost sales on their websites. For example, Ref. [26] developed a video recommendation algorithm based on deep neural networks on YouTube. The authors of [27] stated that Netflix RS has an 80% impact on consumer decisions. Most of these approaches outperformed traditional recommendation models significantly. As a result, we can observe that deep learning has sparked a significant change in the recommender sector. To solve the aforementioned shortcomings, we introduce a dual deep learning architecture to determine the latent factors. Like that of NCF, our approach does not learn the interaction record automatically; instead, we use dual deep learning and embedding-based techniques to extract the features of users and items separately. We then predict the ratings using the inner product.

4. The Proposed Model

In this section, first, we will go over the model that motivated us, called DMF. Then, we will present an embedding technique that reveals the behavior of the latent factor features. A dual deep learning-based recommender system that integrates embedding techniques is also discussed.

4.1. DMF Method

The authors of [8] introduced DMF for top-N item recommendations. To modify the representation of the user and item, the model first maps users' and items' input vectors into latently structured spaces. In this space, users and items are represented by low-dimensional vectors. DMF uses latent features to compute the rank of items for specified users based on similarity results. DMF uses both explicit and implicit feedback as input features. On the other hand, DELCR uses dense latent factors as input features, and we developed the method to predict the ratings rather than predict the rank. For model optimization, we use distinct loss functions from DMF. In DMF, they filled the missing values in the rating matrix with zeros. We believed that this resulted in biases, which in turn affected the performance of recommendations. In our case, we learned these missed values from existing rating values using embedding techniques. In addition, we used an embedding layer to create a dense vector representation that addresses the problem of sparsity in our scenario. In particular, our approach maps the input data into a low-dimensional dense vector before mapping the features into a dual deep neural network structure. In general, our model is a pure collaborative filtering method because, unlike DMF, it predicts user preference for an item based on previous interactions.

4.2. Overview of the Architecture

The proposed model follows the following procedures:

- To handle sparsity problems, the method first learns the unknown rating values from the user–item interaction matrix R.
- We use embedding techniques to learn those missed values from existing ratings.
- The ultimate goal of the model is to predict the ratings using two fully connected deep neural networks.

4.3. Latent Factor Model

The latent factor (LF) model is one of the variants of the CF-based approach that is responsible for capturing users' and items' latent vectors from high-dimensional data. The method uses the inner product to determine the interactions [28]. In our proposed model, we follow the procedure of the LF model to predict the ratings for the sake of overcoming the sparsity problem. In most traditional LF models, they follow the concepts used in SVD as an overall rating prediction. However, here we used the factors as input to the method. According to Ref. [29], the missing values are first filled by taking the average values of the existing ratings, and the latent factors are obtained using SVD techniques. In contrast to this method, we learned the missed values from existing ratings by minimizing the loss function in Equation (8). In addition, Ref. [29] has multiple user latent feature matrix and one-item latent matrix, which increase computational complexity. In this regard, the proposed method has only one user latent factor matrix P and one item latent factor matrix Q that are used as feature input vectors for the DNN architectures. The architecture of our proposed scheme, DELCR, is shown in, Figure 1. The details of the proposed method are discussed layer by layer as follows:

Input Layer: The input layer of our model takes explicit feature vectors obtained from past user interactions for items called "historical interactions". The rating could be taken as it is since it shows the level of a user's liking for a given item. In this step, we used only known user ratings to determine the dense user and item latent factors as a feature. The proposed method follows the procedure shown in [30]. In their work, they embed user ID to obtain a dense continuous-valued vector. However, in our case, we embed user rating

values to obtain the embedding vectors. Here, we initialized the latent factors with random values during the model training to adjust the values.

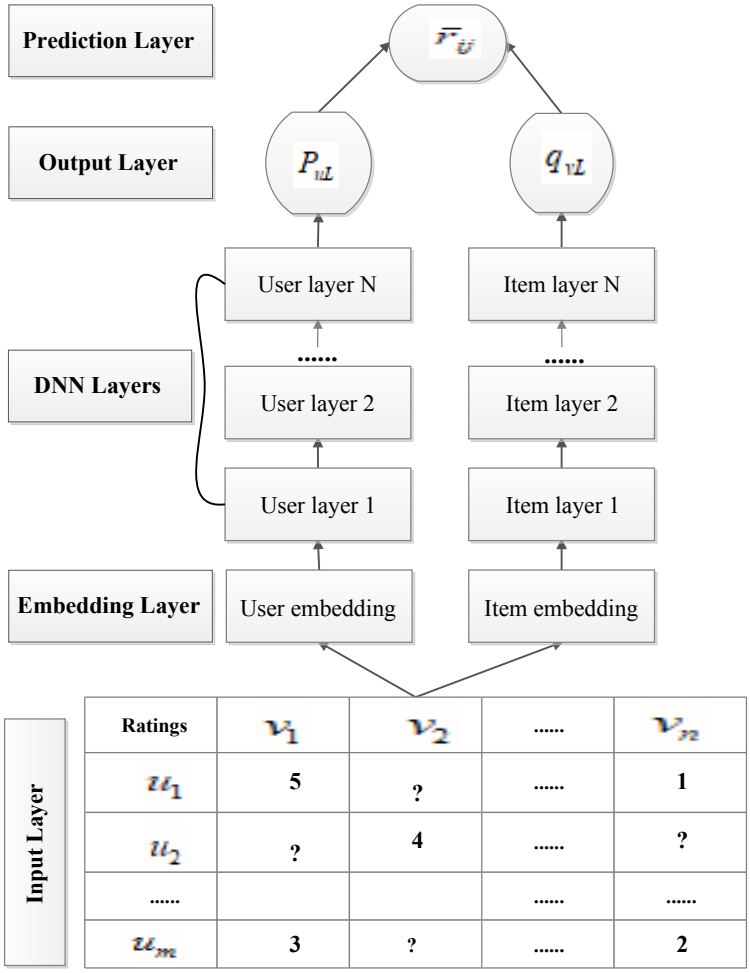


Figure 1. General scheme of the proposed model. The architecture consists of an embedding layer with a dense vector obtained from the input matrix, followed by deep neural network layers. In the output layer, the method generates the latent factors that are used for rating prediction in the prediction layer.

Embedding layer: The embedding layer of DELCR projects each high-dimensional input feature vector and sparse data matrix into two dense low-rank matrices. To obtain the user u and item v primitive feature vectors in the embedding space, perform the following steps: Let $P \in \mathbb{R}^{m \times k}$ represent the user embedding matrix and $Q \in \mathbb{R}^{n \times k}$ represent the item embedding matrix, where k represents the embedding dimension. The user latent vector p_i and the item latent vector q_j are then computed using Equations (6) and (7), respectively [31].

$$p_i = P^T u_i. \tag{6}$$

$$q_j = Q^T v_j. \tag{7}$$

To demonstrate the concepts discussed above, the method first decomposes the rating matrix R into two low-rank matrices, P (the user latent matrix) and Q (the item latent matrix), that satisfy $R \approx PQ^T$. To find the matrices, we used the embedding loss function

(L_{EM}) using Equation (8) to minimize the difference between the true and approximate values [32].

$$L_{EM} = \min_{P,Q} \left[\sum_{(i,j) \in R_{known}} (r_{ij} - p_i {q_j}^T)^2 \right]. \tag{8}$$

A Dual deep learning layer: To capture the data's non-linear phenomena and intrinsic abstraction, we map embedded user latent p_i and item latent q_j into the dual deep learning structures independently. This can be illustrated as follows:

After obtaining the dense feature vectors via the embedding equation, we trained the deep neural network to minimize the loss function (L_{DL}).

$$L_{DL} = \min_{\theta} \left[\sum_{i,j} (\hat{r}_{ij} - r_{ij})^2 + \lambda W^2 + \alpha b^2 \right]. \tag{9}$$

We conclude that the proposed method learns the latent features in their corresponding deep neural network architecture by minimizing the two loss functions specified at Equations (8) and (9) at the same time using the Adaptive Moment Estimation (Adam) algorithm [33].

In contrast to many other models, such as [21,25,26], DELCR does not combine latent vectors to flow via the shared DNN. We constructed a dual deep learning model that learns users' and items' latent features separately for the proposed technique. Here, user and item input vectors are projected into the next connected layer until they reach the output layers. To extract non-linearity features in the data, we used the activation function tanh in each hidden-layer of the network and ReLu in the output layer of the network. To illustrate this concept, let p_0 represent the initial user latent feature vector and q_0 represent the initial item feature vectors. Then, for the user fully connected DNN layer, we apply Equation (10).

$$\begin{aligned} x_{u1} &= \sigma(w_{u1}^T p_0 + b_{u1}), \\ x_{u2} &= \sigma(w_{u2}^T x_{u1} + b_{u2}), \\ &\vdots \\ p_{uL} &= \sigma(w_{uL}^T x_{uL-1} + b_{uL}), \end{aligned} \tag{10}$$

where $w_{u1}, w_{u2}, \ldots, w_{uL}$ and $b_{u1}, b_{u2}, \ldots, b_{uL}$ indicate weights and biases of users, respectively. In this case, σ stands for an activation function. From each layer we have $x_{u1}, x_{u2}, \ldots, p_{uL}$ as an output. Likewise, completely connected item layers can be expressed using Equation (11).

$$\begin{aligned} x_{v1} &= \sigma(w_{v1}^T q_0 + b_{v1}), \\ x_{v2} &= \sigma(w_{v2}^T x_{v1} + b_{v2}), \\ &\vdots \\ q_{vL} &= \sigma(w_{vL}^T x_{vL-1} + b_{vL}), \end{aligned} \tag{11}$$

where q_{vL} denotes the output latent factor from item layers.

prediction layer: After obtaining the latent features, p_{uL} and q_{vL} of users and items, respectively, we apply Equation (12) to estimate the desired rating score, $\hat{r}_{ij}$.

$$\hat{r}_{ij} = p_{uL} \odot q_{vL}, \tag{12}$$

where $\odot$ represents the dot product.

5. Experiments

In this part, we undertake an extended experiment to answer the undermentioned research questions:

- **RQ1** Does our method outperform the baseline models?
- **RQ2** How does the use of the embedding techniques improve the recommendation performance?
- **RQ3** How does the performance of DELCR vary with different hyper-parameter values?

5.1. Experimental Datasets

This study makes use of freely available MovieLens datasets. The GroupLens research team put together these datasets. It is a collection of various-sized movie ratings. It is a popular dataset for testing collaborative filtering algorithms [21]. For model comparison, we chose data comprising ML-100K and ML-1M. We summarized the details of the data in Table 1.

Table 1. Details of the datasets used.

Datasets	Users	Items	Ratings	Rating Density
ML-100K	944	1683	100,000	6.3%
ML-1M	6040	3706	1,000,208	4.5%

5.2. Evaluation Protocol

To ensure the accuracy of the recommendation result, the predicted performance of the models is evaluated using the root mean squared error (RMSE) and mean absolute error (MAE). The MAE measures the absolute error between the expected and actual values, and the RMSE determines the difference between the true and predicted values. In most recommender system literature, these are extensively used evaluation metrics [34]. In this case, the lower the MAE and RMSE values, the better the accuracy is [23]. This is depicted in Equations (13) and (14).

$$MAE = \sum_{(i,j)\in D_{Test}} \frac{|\hat{r}_{ij} - r_{ij}|}{|D_{Test}|}, \tag{13}$$

$$RMSE = \sqrt{\sum_{(i,j)\in D_{Test}} \frac{(\hat{r}_{ij} - r_{ij})^2}{|D_{Test}|}}, \tag{14}$$

where r_{ij} indicates user i's preference for item j, $\hat{r}_{ij}$ is the associated estimated result of user i to item j, and $|D_{Test}|$ is the test set's size in the data.

5.3. Baseline Methods

To evaluate the performance of the proposed model, DELCR, we compared it with the state-of-the-art models for MAE and RMSE evaluation metrics provided in [29].

- For collaborative filtering, MDA employs an autoencoder architecture with MF for model prediction [35].
- CDL provides a hierarchical Bayesian model by combining two methods. It enables the acquisition of an appropriate latent feature to address the sparsity problem [12].
- RMF model uses Manhattan distance to minimize the distances within matrices. It successfully solves the problem of data sparsity [36].
- DMF is a Deep MF model that uses MLP to transform the representation of the user and item [8].
- HLFM is one of the MF-based algorithms that investigates the underlying hierarchical characteristics of users and items. To increase the effectiveness of RSs, the approach integrates the hierarchy of items as well as user preferences [37].

- DLFCF is a CF model based on latent factors. The model uses deep factorization on users and items to determine their latent representation vectors [29].
- DELCR is the model presented in this paper.

5.4. Parameter Settings

We randomly divided the data into training with 80%, and the rest went to test and validation data. Each time, we execute five different pieces of training and pick the average value as the final result to acquire representative prediction results as depicted in Table 2. Our tests were run on the Ubuntu 18.04 operating system, with a CPU memory of 16GB and a GPU memory of 6GB (NVIDIA). Tensorflow (https://www.tensorflow.org accessed on 27 February 2021). was utilized to simulate the model, which will be released publicly upon acceptance. For model optimization, we used Adam [33], with a batch size of 512, a learning rate of 0.0001, and a regularizer coefficient λ of $1 \times e^{-6}$. We employed two hidden-layers for the dual neural network with several factors on the hidden-layers as $\{80, 40\}$ and set the number of embedding dimensions to 8.

Table 2. Results of the DELCR from five-fold validation for both datasets. The values show the metrics' average and a 95% confidence interval in a five-fold validation.

Folds	For ML-100K		For ML-1M	
	MAE	**RMSE**	**MAE**	**RMSE**
Fold 1	0.672 ± 0.005	0.855 ± 0.005	0.643 ± 0.007	0.818 ± 0.004
Fold 2	0.674 ± 0.003	0.857 ± 0.007	0.642 ± 0.005	0.817 ± 0.006
Fold 3	0.671 ± 0.004	0.853 ± 0.002	0.644 ± 0.003	0.817 ± 0.001
Fold 4	0.673 ± 0.007	0.855 ± 0.007	0.642 ± 0.001	0.818 ± 0.004
Fold 5	0.675 ± 0.007	0.858 ± 0.003	0.641 ± 0.003	0.818 ± 0.003

5.5. Experimental Results

Table 2 lists MAE and RMSE metrics used in the experiments. The results in Table 2 are the averages for five different pieces of training, along with marginal error estimates for each metric and a 95% confidence interval. Our analysis of the results shows that their coefficient of variance is less than 1%. These findings indicated that outcomes are reproducible and that the proposed approach generates reproducible results.

The performance result of our proposed method, DELCR, is summarized in Table 3. The bold result was achieved using our method. When compared to the methods depicted in Table 4, the percentage improvement result shows a significantly improved performance overall.

Table 3. Performance comparison result of DELCR for both datasets. The bold result is obtained using the proposed method.

Methods	For ML-100K		For ML-1M	
	MAE	**RMSE**	**MAE**	**RMSE**
MDA	0.758	0.981	0.686	0.879
CDL	0.742	0.953	0.689	0.871
RMF	0.732	0.938	0.689	0.876
DMF	0.735	0.940	0.691	0.878
HLFM	0.750	0.962	0.698	0.880
DLFCF	0.717	0.901	0.678	0.854
DELCR	**0.674**	**0.856**	**0.643**	**0.818**

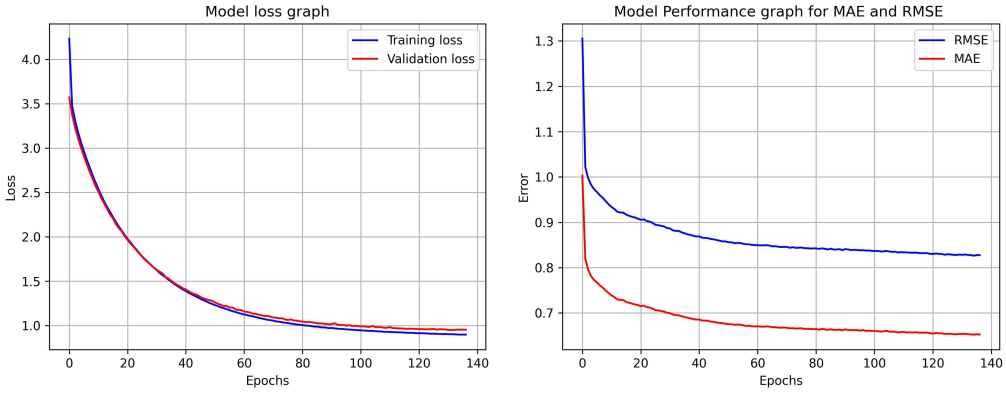
Table 4. Performance improvement results in DELCR for both datasets.

Methods	For ML-100k		For ML-1M	
	MAE	**RMSE**	**MAE**	**RMSE**
MDA	11.08%	12.74%	6.27%	6.94%
CDL	9.16%	10.18%	6.68%	6.08%
RMF	7.92%	8.74%	6.68%	6.62%
DMF	8.30%	8.94%	6.95%	6.83%
HLFM	10.13%	11.02%	7.88%	7.05%
DLFCF	6.00%	5.00%	5.16%	4.22%

6. Discussion

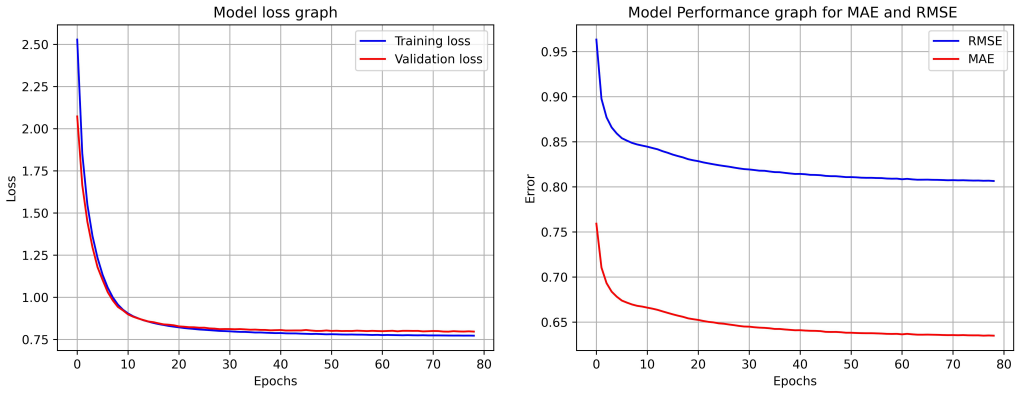
6.1. Performance Comparison (RQ1)

Figures 2a and 3a, show that our model converges for both datasets. The rate of convergence of our method is faster for ML-1M than for ML-100K because ML-1M has 10 times more data points than ML-100K. We can also observe that a two-hidden-layer model delivers the greatest performance result for both datasets. The following is a summary of the main outcomes of the experiments:



(**a**) Loss graph (**b**) Performance curve for both metrics

Figure 2. A model with 2 hidden-layer for the ML-100K dataset. In Part (**a**), we have the training and validation loss curve versus the number of training epochs, and in Part (**b**), we have the performance results. As observed, the method converges.



(**a**) Loss graph (**b**) Performance curve for both metrics

Figure 3. A model with 2 hidden-layer for the ML-1M dataset. In Part (**a**), we have the training and validation loss curve versus the number of training epochs, and in Part (**b**), we have the performance results. As observed, the method converges.

- According to Table 4, the proposed method, DELCR, outperforms all other methods for both datasets. Particularly when contrasted to our highly motivated DMF model, our DELCR approach with ML-100K results in a relative improvement of 8.30% of MAE and 8.94% of RMSE. In addition, the method delivers a relative improvement with MAE of 6.95% and RMSE of 6.83% for the ML-1M dataset. Furthermore, when contrasted to the DL-based models, CDL and MDA, the advances made by our model are significant. For these two particular models, there is an average relative improvement of 10.12% of MAE and 11.46% of RMSE for the ML-100K dataset whereas, for ML-1M dataset, there is an average relative improvement of 6.48% of MAE and 6.51% of RMSE. The reasons for this best performance achievement are: (i) the use of embedded ratings will effectively alleviate the problem of the sparse dataset and (ii) the separate mapping of these dense embedded vectors to our proposed architecture helps to discover the complex and abstract latent features of these two different entities called users and items effectively.
- Once more, the discrepancy between the DELCR and RMF methods is significant. When we see the comparison with the ML-100K dataset, we got a 7.92% and 8.74% MAE and RMSE relative improvement, respectively. For the ML-1M dataset, the method gives a relative improvement with 6.68% of MAE and 6.62% of RMSE. This means that using a deep learning architecture allows us to successfully extract the latent variables in our data, which was previously the fundamental constraint of most classic matrix factorization-based models.
- When we evaluate DELCR, with the state-of-the-art latent factor models HLFM and DLFCF, DELCR produces relatively minimal improvement results for MAE and RMSE as compared to the other methods. This is expected as the model follows the latent factor approach. In spite of this, the proposed model improvement result is due to the effectiveness of the architecture in mining the latent features in a robust way. In addition, when we see improvements made by the proposed method among the two used datasets, our method yields a better result for the ML-100K dataset than ML-1M. This achievement demonstrates that DELCR effectively alleviates the problem of data sparsity since ML-100K is relatively sparser than ML-1M. In general, our method outperforms all other methods in terms of improving the quality of recommendations. This performance improvement shows the effectiveness of an embedding technique and the use of a dual deep-learning architecture in the proposed method.

6.2. Effect of Embedding (RQ2)

To look into the impact of embedding, we compared the performance of our proposed method, DELCR (with embedding), with that of DCF (without embedding), which was our previously published proceeding. For a fair comparison, we made the number of hidden-layers for DCF [19] the same as DELCR. We observe that DELCR with embedding outperforms DCF on both datasets. Specifically, our proposed method outperforms DCF with a relative improvement of 3.56% and 3.49% of MAE and RMSE, respectively, for the ML-100K dataset. It also shows a relative improvement for the ML-1M dataset with a MAE of 2.87% and a RMSE of 2.85% as shown in Table 5. From this result, we can conclude that the use of an embedding technique improves the performance of recommendation quality.

Table 5. Embedding effect for both datasets.

Models	ML-100K		ML-1M	
	MAE	**RMSE**	**MAE**	**RMSE**
DCF	0.698	0.887	0.662	0.842
DELCR	0.674	0.856	0.643	0.818

6.3. Sensitivity to Hyper-Parameters (RQ3)

In this subsection, we look at how different hyperparameter values affect the performance of the methods we have developed.

6.3.1. Depth of Layers in the Network

The most important parameters are the number of layers utilized in the network and the number of neurons taken into account in each layer. To determine the ideal number of layers for the network, we carried out an extensive experiment. The results depicted in Table 6 show that performance decreases as we go deeper. This result demonstrates that our system performs best with two hidden-layers in the network, so we chose the two hidden-layer model outcomes given in Figures 2b and 3b since they provide the smallest MAE and RMSE results. The reason for these performance decrements is that, as we move deeper, we have to train more parameters that contribute to over-fitting for the limited amount of available data, resulting in poor recommendation quality.

Table 6. Models performance result with different number of layers using both datasets.

Models	For ML-100k		For ML-1M	
	MAE	**RMSE**	**MAE**	**RMSE**
Two layer	0.674	0.856	0.643	0.818
Three layer	0.685	0.874	0.659	0.835
Four layer	0.707	0.896	0.677	0.860

6.3.2. Number of Nodes in Each Layer

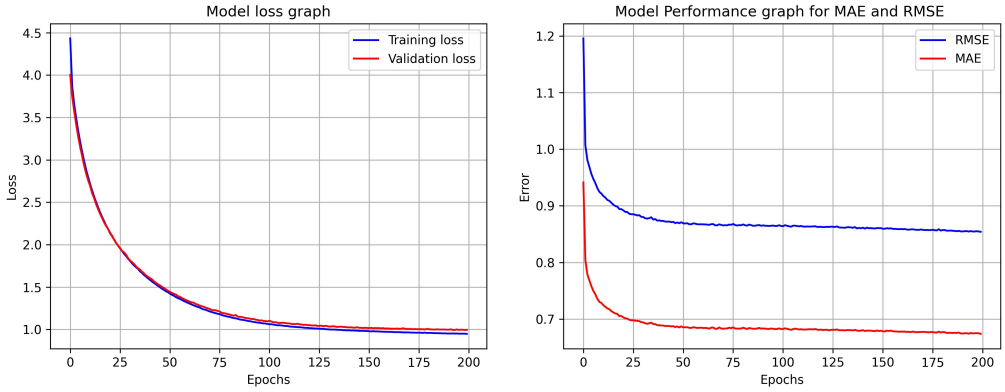
The number of nodes also plays a big role in our method. We experimented to determine the right number of nodes to take into account for each layer, and we came up with 80 neurons for the first outer layer. By taking half of the nodes from the layer before, we may compute the subsequent number of nodes for the subsequent inner layer. Thus, the two hidden-layer models require 80 and 40 neurons on successive layers, the three hidden-layer models require 80, 40, 20, and the four hidden-layer models require 80, 40, 20, 10 neurons on successive layers. The results obtained from the models are depicted in Table 6.

6.3.3. Number of Nodes in the Final Hidden Layers

Referring to the results shown in Table 6, which are values extracted from Figures 2b, 3b, 4b and 5b, we subsequently conclude that our proposed model will have two hidden-layers in the network. In the final hidden-layer, we played around with the number of nodes. According to [8], the number of neurons placed on the last layer is used as a prediction factor. As a result, we programmed it to range between 20 to 80, with a step size of 10. The best results were obtained whenever we had 40 nodes in the final hidden-layer, as shown in Table 7.

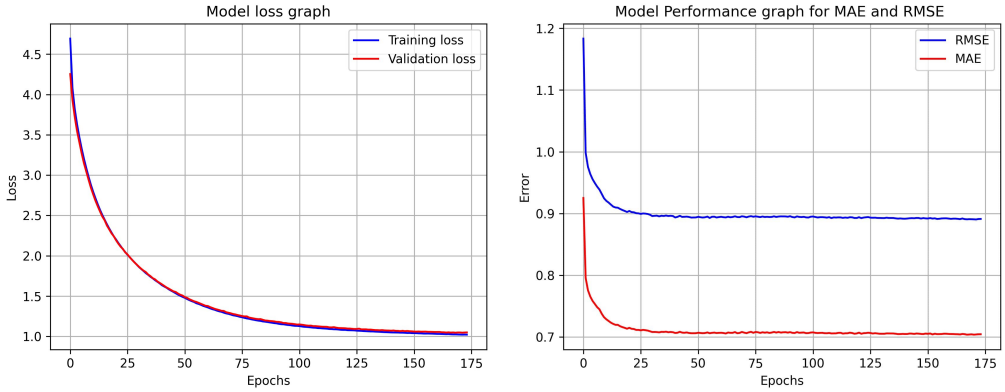
Table 7. Performance of DELCR with different numbers of neurons on the last layer for ML-1M dataset.

Metrics	Number of Neurons					
	20	**30**	**40**	**50**	**60**	**70**
MAE	0.650	0.647	0.643	0.645	0.644	0.643
RMSE	0.825	0.823	0.818	0.819	0.820	0.818



(**a**) Loss graph (**b**) Performance curve for both metrics

Figure 4. A three-hidden-layer model for the ML-100K dataset. In Part (**a**), we have the training and validation loss curve versus a number of training epochs, and in Part (**b**), we have the performance results. As observed, the method converges.



(**a**) Loss graph (**b**) Performance curve for both metrics

Figure 5. A four-hidden-layer model for the ML-100K dataset. In Part (**a**), we have the training and validation loss curve versus several training epochs, and in Part (**b**), we have the performance results. As observed, the method converges.

7. Conclusions

In this paper, we explored a dual DL and embedding-based latent factor method for recommender systems. The method combines deep learning and embedding techniques to overcome the data sparsity problem faced by latent factor models. To overcome this problem, the design first embeds users' and items' input vectors to build a dense low-dimensional representation of data using embedding techniques. Then, it projects these dense vectors separately into two well-structured deep neural network architectures to discover the abstract and non-linear representation of the data. Finally, we use the inner product to estimate the rating score in the network's output layers. By significantly improving performance, this framework effectively alleviates problems observed in latent factor-based models. We performed extensive experiments on two real-world datasets, and the resulting experimental findings revealed that our proposed model outperformed existing state-of-the-art techniques for the rating prediction task on all datasets. As we discussed in our approach, particularly effective results were obtained for the sparse dataset ML-100K rather than ML-1M. We can conclude from this result that deep learning approaches are one of the most effective technical means of improving the performance of recommendation systems.

In the future, we plan to incorporate other deep learning architectures for recommender systems to improve the quality of their performance in advance. We also intend to look into the effect of dual embedding on dual deep learning architectures.

Author Contributions: A.T. and Q.L. built the framework of the whole paper. A.T. and Y.G. carried out preprocessing of the data. A.T. designed the whole experiment and method and implemented the experiment. Q.L. provided analytical and experimental tools. H.L. and M.A. wrote the manuscript. T.D. revised the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the National Natural Science Foundation of China (No. U19B2028, No. U22B2061).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: This study makes use of publicly available MovieLens datasets. This data can be found at https://files.grouplens.org/datasets/movielens/, accessed on 26 February 2021.

Conflicts of Interest: The authors declare no conflict of interest

References

1. Pawlicka, A.; Pawlicki, M.; Kozik, R.; Choraś, R.S. A systematic review of recommender systems and their applications in cybersecurity. *Sensors* **2021**, *21*, 5248. [CrossRef] [PubMed]
2. Ricci, F.; Rokach, L.; Shapira, B. Recommender systems: Introduction and challenges. In *Recommender Systems Handbook*; Springer: Boston, MA, USA, 2015; pp. 1–34.
3. Sun, Z.; Guo, Q.; Yang, J.; Fang, H.; Guo, G.; Zhang, J.; Burke, R. Research commentary on recommendations with side information: A survey and research directions. *Electron. Commer. Res. Appl.* **2019**, *37*, 100879. [CrossRef]
4. Catherine, R.; Mazaitis, K.; Eskenazi, M.; Cohen, W. Explainable entity-based recommendations with knowledge graphs. *arXiv* **2017**, arXiv:1707.05254.
5. Wang, H.; Zhang, F.; Hou, M.; Xie, X.; Guo, M.; Liu, Q. Shine: Signed heterogeneous information network embedding for sentiment link prediction. In Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, Marina Del Rey, CA, USA, 5–9 February 2018; pp. 592–600.
6. Sun, Y.; Yuan, N.J.; Xie, X.; McDonald, K.; Zhang, R. Collaborative intent prediction with real-time contextual data. *ACM Trans. Inf. Syst. (TOIS)* **2017**, *35*, 1–33. [CrossRef]
7. Wang, Y.; Deng, J.; Gao, J.; Zhang, P. A hybrid user similarity model for collaborative filtering. *Inf. Sci.* **2017**, *418*, 102–118. [CrossRef]
8. Xue, H.J.; Dai, X.; Zhang, J.; Huang, S.; Chen, J. Deep matrix factorization models for recommender systems. In Proceedings of the 26th International Joint Conference on Artificial Intelligence, Melbourne, Australia, 19–25 August 2017; Volume 17, pp. 3203–3209.
9. Yagci, A.M.; Aytekin, T.; Gurgen, F.S. A Meta-Algorithm for Improving Top-N Prediction Efficiency of Matrix Factorization Models in Collaborative Filtering. *Int. J. Pattern Recognit. Artif. Intell.* **2020**, *34*, 2059007. [CrossRef]
10. He, X.; Zhang, H.; Kan, M.Y.; Chua, T.S. Fast matrix factorization for online recommendation with implicit feedback. In Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, Pisa Italy, 17–21 July 2016; pp. 549–558.
11. Koren, Y.; Bell, R.; Volinsky, C. Matrix factorization techniques for recommender systems. *Computer* **2009**, *42*, 30–37. [CrossRef]
12. Wang, H.; Wang, N.; Yeung, D.Y. Collaborative deep learning for recommender systems. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, Australia, 10–13 August 2015; pp. 1235–1244.
13. Guo, L.; Ma, J.; Chen, Z.; Zhong, H. Learning to recommend with social contextual information from implicit feedback. *Soft Comput.* **2015**, *19*, 1351–1362. [CrossRef]
14. Gu, Y.; Yang, X.; Peng, M.; Lin, G. Robust weighted SVD-type latent factor models for rating prediction. *Expert Syst. Appl.* **2020**, *141*, 112885. [CrossRef]
15. Bengio, Y. Learning deep architectures for AI. *Found. Trends® Mach. Learn.* **2009**, *2*, 1–127. [CrossRef]
16. Hinton, G.E. Deep belief networks. *Scholarpedia* **2009**, *4*, 5947. [CrossRef]
17. Adnan, M.; Habib, A.; Ashraf, J.; Mussadiq, S.; ALI, A. Deep neural network based m-learning model for predicting mobile learners' performance. *Turk. J. Electr. Eng. Comput. Sci.* **2020**, *28*, 1422–1441. [CrossRef]
18. Dang, D.; Chen, C.; Li, H.; Yan, R.; Guo, Z.; Wang, X. Deep knowledge-aware framework for web service recommendation. *J. Supercomput.* **2021**, *77*, 14280–14304. [CrossRef]
19. Tegene, A.T.; Liu, Q.; Muhammed, S.B.; Leka, H.L. Deep Learning Based Matrix Factorization For Collaborative Filtering. In Proceedings of the 2021 18th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), Chengdu, China, 17–19 December 2021; pp. 165–170.

20. Zhang, S.; Yao, L.; Sun, A.; Tay, Y. Deep learning based recommender system: A survey and new perspectives. *ACM Comput. Surv. (CSUR)* **2019**, *52*, 1–38. [CrossRef]
21. He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; Chua, T.S. Neural collaborative filtering. In Proceedings of the 26th International Conference on World Wide Web, Perth, Australia, 3–7 April 2017; pp. 173–182.
22. Salakhutdinov, R.; Mnih, A.; Hinton, G. Restricted Boltzmann machines for collaborative filtering. In Proceedings of the 24th International Conference on Machine Learning, Corvalis, OR, USA, 20–24 June 2007; pp. 791–798.
23. Wang, S.; Sun, G.; Li, Y. SVD++ recommendation algorithm based on backtracking. *Information* **2020**, *11*, 369. [CrossRef]
24. Rendle, S. Factorization machines with libfm. *ACM Trans. Intell. Syst. Technol. (TIST)* **2012**, *3*, 1–22. [CrossRef]
25. Guo, H.; Tang, R.; Ye, Y.; Li, Z.; He, X. DeepFM: A factorization-machine based neural network for CTR prediction. *arXiv* **2017**, arXiv:1703.04247.
26. Covington, P.; Adams, J.; Sargin, E. Deep neural networks for youtube recommendations. In Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, 15–19 September 2016; pp. 191–198.
27. Gomez-Uribe, C.A.; Hunt, N. The netflix recommender system: Algorithms, business value, and innovation. *ACM Trans. Manag. Inf. Syst. (TMIS)* **2015**, *6*, 1–19. [CrossRef]
28. Wu, D.; Luo, X.; Shang, M.; He, Y.; Wang, G.; Zhou, M. A deep latent factor model for high-dimensional and sparse matrices in recommender systems. *IEEE Trans. Syst. Man Cybern. Syst.* **2019**, *51*, 4285–4296. [CrossRef]
29. Mongia, A.; Jhamb, N.; Chouzenoux, E.; Majumdar, A. Deep latent factor model for collaborative filtering. *Signal Process.* **2020**, *169*, 107366. [CrossRef]
30. Nassar, N.; Jafar, A.; Rahhal, Y. A novel deep multi-criteria collaborative filtering model for recommendation system. *Knowl.-Based Syst.* **2020**, *187*, 104811. [CrossRef]
31. Cheng, W.; Shen, Y.; Zhu, Y.; Huang, L. DELF: A Dual-Embedding based Deep Latent Factor Model for Recommendation. In Proceedings of the IJCAI, Stockholm, Sweden, 13–19 July 2018; Volume 18, pp. 3329–3335.
32. He, G.; Zhao, D.; Ding, L. Dual-embedding based Neural Collaborative Filtering for Recommender Systems. *arXiv* **2021**, arXiv:2102.02549.
33. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
34. Shi, C.; Hu, B.; Zhao, W.X.; Philip, S.Y. Heterogeneous information network embedding for recommendation. *IEEE Trans. Knowl. Data Eng.* **2018**, *31*, 357–370. [CrossRef]
35. Li, S.; Kawale, J.; Fu, Y. Deep collaborative filtering via marginalized denoising auto-encoder. In Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, Melbourne, Australia, 18–23 October 2015; pp. 811–820.
36. Liu, T.; Tao, D. On the performance of manhattan nonnegative matrix factorization. *IEEE Trans. Neural Netw. Learn. Syst.* **2015**, *27*, 1851–1863. [CrossRef] [PubMed]
37. Wang, S.; Tang, J.; Wang, Y.; Liu, H. Exploring hierarchical structures for recommender systems. *IEEE Trans. Knowl. Data Eng.* **2018**, *30*, 1022–1035. [CrossRef]