Applied Soft Computing 109 (2021) 107516

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Collaborative filtering via heterogeneous neural networks

Wei Zeng [a,*,1], Ge Fan [a,c,1], Shan Sun [a], Biao Geng [b], Weiyi Wang [a], Jiacheng Li [a], Weibo Liu [a]

[a] Center for Artificial Intelligence and Smart Health, University of Electronic Science and Technology of China, Chengdu, China
[b] College of Engineering, Carnegie Mellon University, Pittsburgh, USA
[c] Lightspeed & Quantum Studios, Tencent Inc., Shenzhen, China

ARTICLE INFO

Article history:
Received 2 April 2020
Received in revised form 21 December 2020
Accepted 7 May 2021
Available online 27 May 2021

Keywords:
Collaborative filtering
Deep learning
Neural networks
Matrix factorization

ABSTRACT

Over the last few years, the deep neural network is utilized to solve the collaborative filtering problem, a method of which has achieved immense success on computer vision, speech recognition as well as natural language processing. On one hand, the deep neural network can be used to capture the side information of users and items. On the other hand, it is also capable of modeling interactions between users and items. Most of existing approaches exploit the neural network with solo structure to model user–item interactions such that the learning representation may be insufficient over the extremely sparse rating data. Recently, a large number of neural networks with mixed structures are devised for learning better representations. A carefully designed hybrid network is able to achieve considerable accuracy but only requires a small amount of extra computation. In order to model user–item interactions, we elaborate a hybrid neural network consisting of the global neural network and several local neural blocks. The multi-layer perceptron is adopted to build the global neural network and the residual network is used to form the local neural block which is inserted into two adjacent global layers. The hybrid network is further combined with the generalized matrix factorization to capture both the linear and nonlinear relationships between users and items. It is verified by experimental results on benchmark datasets that our method is superior to certain state-of-the-art approaches in terms of top-n item recommendation.

1. Introduction

Owing to the rapid growth and prevalence of Internet, users are confronted with abundant online contents (e.g. movies, books and music), which makes it very time-consuming to select the needed information. This is often referred to as the information overload problem. In order to fulfill the needs of personalized services, recommender systems are widely studied and applied to online systems [1] which have achieved success in plenty of famous companies such as Netflix, Amazon and YouTube [1–3]. It is reported that 80 percent of movies watched by Netflix users come from the recommendation engine [3] and more than 60 percent of video clicks are from home page recommendations in YouTube [2].

Improving the accuracy of the algorithm is always challenged in the field of collaborative filtering. The key to resolve this issue lies in establishing an accurate model to describe the interactions between users and items [4]. To accomplish such objective, the matrix factorization method jointly maps users and items into a latent space and calculates the predictive score by the inner product of the latent vector of user and item [5]. The matrix factorization can be considered as a linear model and a recent study reveals that both the linear and nonlinear relationships between users and items should be taken into account [6]. A natural way to capture these nonlinear relationships is employing the neural network which incorporates the activation function for learning the nonlinear representation.

Since the interaction matrix is fairly sparse, the embedding layer is applied to turn the sparse representation into low dimensional dense vector. Then the embedding vectors of users and items are concatenated and fed into a fully connected neural network. The deep neural network needs a large amount of data to train layer weights (edges in network) [7,8], but the rating data is extremely sparse, which may decline the representation learning ability of neural networks. In order to manifest this issue, we plot the weight of the last layer. The predictive scores of items can be calculated via the last layer of the network (the output layer), namely $\hat{y} = a_{out}(\mathbf{h}^T\mathbf{z}+b)$, where a_{out} is the activation function and $\mathbf{z}$ is the output vector, which is determined by the input and the structure of the network. $\mathbf{h}$ is the weight vector $w.r.t$ the neurons of the output layer, which is learnt from the data automatically.

* Corresponding author.
E-mail addresses: zwei504@uestc.edu.cn (W. Zeng), fange@std.uestc.edu.cn (G. Fan), sunshan0813@hotmail.com (S. Sun), bgeng@andrew.cmu.edu (B. Geng), wwy@std.uestc.edu.cn (W. Wang), jiachengli@std.uestc.edu.cn (J. Li), Weiboliu@std.uestc.edu.cn (W. Liu).

[1] Equal contribution.

https://doi.org/10.1016/j.asoc.2021.107516
1568-4946/

We plot the absolute value of **h** and the result is presented in Fig. 1 where the left subfigure represents a solo multi-layer perceptron (MLP) modeling interactions between users and items and the absolute value of **h** is given in the right. One can see that the weights are mainly centralized in the interval of 0 to 0.1. If weights are too centralized (close to 0), predictive scores of various items could be quite close, which may bring a negative impact on the accuracy of item recommendations.

The majority of existing recommendation algorithms adopt the plain neural network (as depicted in the left of Fig. 1) to calculate the final predictive scores and plenty of experiments have shown its superiority of representation learning ability [6,9–11]. However, recent researches indicate that the network-in-network architecture is beneficial to improving the representation learning ability [12–14]. For example, Qu et al. [14] made use of micro networks as kernel functions to augment the nonlinearity of the whole network. The hybrid network is one kind of the network-in-network architecture and used to predict the Click-Through Rate (CTR). The attention network can be considered as another kind of network-in-network architectures. Recently, it is used to determine the importance of parameters in the model [9,15]. In these method, the final predictive scores are obtained by the solo MLP. As discussed above, the predictive scores obtained by the plain neural network may be centralized, which could undermine the performance of the algorithm.

Encouraged by recent advances in collaborative filtering and inspired by recent successes in exploring mixed networks for representation learning, we propose a hybrid neural network which is one kind of "network-in-network" structure. Our objective is to enhance the representation learning ability of the network via mixed structures. The traditional MLP is adopted to construct the global neural network and the residual network is utilized to form the local neural block, which can be considered as remapping functions projecting the output of the global layer into a variety of dimensional space. The output of a local neural block is then re-fed into the next global layer. The local neural block is built by the residual neural network which learns how to change the input instead of learning what the output should be (using shortcut connections) [16]. Choosing the residual network is due to the following two reasons. Firstly, it is proved to have excellent generalization performance on recognition tasks. Secondly and more importantly, adopting a different structure from the global neural network (MLP) is capable of improving the diversity of the whole network.

In summary, the main contributions of our work are as follows:

1. We devise a hybrid neural network to model user–item interactions. The depth of the network is increased by inserting a few of neural blocks from the localization. Mixed structures are exploited such that the information uncovered by our method is more diverse than the information obtained by solo MLP.
2. We study the weight distribution of the last layer and visualize the output vector. The results demonstrate that the layer weights of our hybrid network distribute more diversely than solo MLP, which indicates that the representation learning ability of the network is enhanced by our method.
3. We compare our method with certain state-of-the-art recommendation approaches, and the experimental results verify that our method outperforms baseline approaches in terms of top-n item recommendation.

2. Related works

We employ a hybrid neural network to model user–item interactions. In this section, we firstly review some traditional deep learning based recommendation approaches which mainly adopt plain neural network to capture user–item interactions. Secondly, recent development on neural networks with mixed structures are summarized.

2.1. Collaborative filtering with the solo neural network

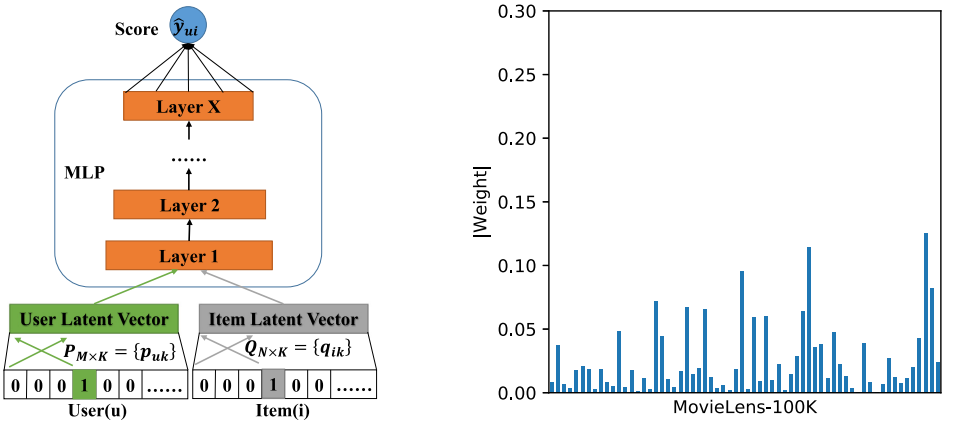
The application of deep learning methods to the task of collaborative filtering attracts enormous attention in recent years [7,17,18]. Salakhutdinov et al. [17] firstly introduced a class of two-layer undirected graphical models, Restricted Boltzmann Machines (RBM), to model the data of individual ratings. Authors further linearly combined multiple RBM models with multiple SVD models to enhance the performance of the separated model. Since the deep learning techniques achieve tremendous success in computer vision, speech recognition and textual analysis, they are exploited to learn features from audio and text content in the recommender system. Then the probabilistic matrix factorization is combined with these deep learning models to generate recommendation for users [18–20].

The recommendation can be considered to be the match between the user's preference and the item's feature. For instance, the matrix factorization maps users and items into a joint latent space and calculates their predictive scores by the inner product of the latent vector of user and item [5]. Thus, it is natural to build a dual network for modeling the two-way interaction between users and items. In many online systems, it is not so easy to gather individual explicit ratings and thus it might be more practical to infer users' preferences through their implicit feedbacks. With the implicit feedbacks, the neural networks can be used to model the nonlinear relationships between users and items and the linear relationship can be captured by the linear models such as linear regression and the matrix factorization [6,7]. Recent researches show that the neural networks are able to model both the explicit and implicit feedbacks simultaneously by combining the point-wise loss and the pair-wise loss [10,11]. For these approaches, the plain neural network is used to compute the final predictive scores. Despite these methods achieve considerable recommendation accuracy, their representation learning ability can be further improved by using networks with mixed structures [14].

2.2. Collaborative filtering via network-in-network architecture

The network-in-network architecture was initially introduced in Ref. [12], wherein the generalized linear model in convolutional neural network (CNN) is replaced by a "micro network". It is revealed by recent works that such hybrid structure is capable of learning better representations [13,21]. The network-in-network architectures have a variety of forms such as the inception networks and the attention networks [15,22–24]. The Inception model aggregates outputs of several convolutional layers, method of which is mainly applicable to computer vision [22]. The attention network is built from the localization which equips a neural network with the capability to focus on a subset of its inputs (or features). The attention network has diverse architectures due to the applications and it is recently introduced to solve the collaborative filtering task.

He et al. [15] took a multi-layer perception as the attention network to learn contributions of an item for the top-n item recommendation. Furthermore, Xue et al. [9] extends the method by accounting for the nonlinear and higher-order relationship



(a) MLP architecture

(b) $|Weight|$ of neural units in the output layer.

Fig. 1. Weight distribution of the output layer of MLP on MovieLens dataset.

among items. More specifically, the interactions among all interacted items are considered by such method, whereas a number of existing methods only take into account the second-order interactions (e.g. similarity) between two items. Tay et al. [24] took advantage of the attention mechanism over an augmented memory network to learn latent relation describing each user and item interaction. In recent, the inception model is combined with the attention fusion mechanism to improve the quality of news recommendations, method of which is capable of learning user features, item features as well as context features synchronously. The semantic information is learnt from the reviews and articles by the attention network to boost the accuracy of recommender systems [25,26].

2.3. Limitations of related works

During the past few years, the network-in-network architecture was introduced in recommender systems. However, from the reviewed literatures, it can be seen that most of the efforts are devoted to exploit the attention network to extract features from the data. Qu et al. [14] used a micro-network as the kernel function to manipulate the embedding vectors and utilized a plain neural network as the classifier. Moreover, such method resolve the task of CTR estimation and therefore it is different from our method. A lot of neural networks with mixed structures are proposed to learn better representations in recent years. A carefully designed hybrid network is able to significantly enhance the accuracy while only a little of extra computation is required [13,22,23,27]. These network-in-network architectures have achieved tremendous success in both computer vision and collaborative filtering [14,15,25]. Inspired by recent successes in exploring mixed neural networks for representation learning, we propose a hybrid network to solve the task of top-n item recommendations.

3. Preliminaries

Lots of recommendation approaches stand on the assumption that users' explicit ratings are available [5]. Actually it is not so easy to obtain explicit ratings from users. Thus, it might be more practical to derive user preferences from their implicit feedbacks, such as users' clicking and interaction record [28]. It is worth noting that the individual implicit feedback may be not binary. For example, a user purchases an item multiple times. In this paper, we define the interaction matrix $\mathbf{Y}_{M\times N}$ as: $y_{ui} = 1$ if user u has at least one interaction with item i and $y_{ui} = 0$ indicates that there is no observational link between user u and item i. M and N are the number of users and items, respectively.

If the recommendation is considered as a two-way interaction between the user's preferences and the item's properties, a neural network can be built to model their interactions. In respect that the interaction matrix is extremely sparse, the embedding layer is taken into consideration to transform the sparse representation into a dense vector. Thereby, the bottom of the network is constituted of input layer and embedding layer. The input layer is composed of two feature vector $\mathbf{v}_u$ and $\mathbf{o}_i$, which respectively represent user u and item i. The embedding layer is a fully connected layer and the obtained embedding vector can be regarded as the latent vector in the context of the latent factor model [6]. The user and item embedding vector are then concatenated and fed into a multi-layer perceptron to map the user's and item's feature to predictive score. The final output is the predictive score $\hat{y}_{ui}$ and the training process is performed by minimizing the pointwise loss between $\hat{y}_{ui}$ and its target value y_{ui}.

The above procedure can be formulated as:

$$\hat{y}_{ui} = \phi_{out}(\phi_X(...\phi_2(\phi_1(\mathbf{P}^T\mathbf{v}_u, \mathbf{Q}^T\mathbf{o}_i))...)), \tag{1}$$

where $\mathbf{P} \in \mathbf{R}^{M\times K}$ and $\mathbf{Q} \in \mathbf{R}^{N\times K}$ represent the latent factor matrix for users and items, respectively. We define the embedding vector $\mathbf{p}_u = \mathbf{P}^T\mathbf{v}_u$ $w.r.t$ user u and $\mathbf{q}_i = \mathbf{Q}^T\mathbf{o}_i$ for item i which are optimized by the back propagation method. ϕ_{out} and ϕ_x separately denote the mapping function for the output layer and x-th neural layer, and there are X hidden layers in total.

4. Our proposed model

In this section, we mainly discuss our method, the Heterogeneous Neural Collaborative Filtering (HNCF for short), wherein the neural network is constructed by heterogeneous networks.

4.1. Heterogeneous neural collaborative filtering

The solo MLP has considerable learning representation and its feature learning ability can be further improved by the network-in-network architecture indicated by recent researches. The hybrid architecture brings extra computations and hence the structure should not be too complex. In addition, the mixed structures should improve the representation learning power to enhance the accuracy of item recommendation. Due to these concerns, we elaborate the neural network similar to the approach in [12]. Our

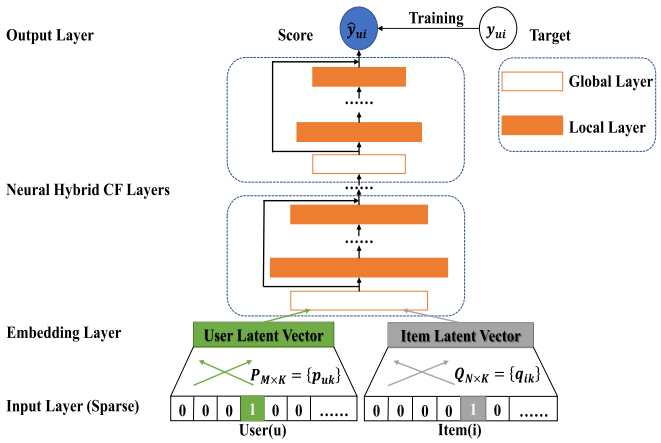


Fig. 2. The architecture of our method. The deep residual neural network is chosen for illustration.

hybrid neural network is comprised of a global neural network and several local neural blocks. The global neural network is constructed by the traditional MLP and we adopt two structures to build the local neural block: traditional MLP and deep residual neural network [16]. In the model, the global layer and local block are placed alternatively. More specifically, the local neural block is inserted into two adjacent global layers. The output of a global layer is transformed to the input of a local neural block and the output of the local block is then re-fed to the next global layer. On one hand, the local neural block is capable of augmenting nonlinearity of the whole network [12]. On the other hand, heterogeneous networks are able to increase the diversity of the network.

We make use of the method in [6] to train the network. The bottom of the network is composed of input layer and embedding layer. The user and the item embedding vector are then concatenated and fed into the hybrid neural network. In a similar way, the output is the predictive score $\hat{y}_{ui}$. By calculating the pointwise loss between $\hat{y}_{ui}$ and its target value y_{ui}, we can train the parameters in the network through the back propagation.

By combining the idea of Ref. [12] and [6], the predictive score can be formulated as follow:

$$\hat{y}_{ui} = \phi_{out}(\Phi_X^L(\phi_X^G ... (\Phi_1^L(\phi_1^G(\mathbf{P}^T\mathbf{v}_u, \mathbf{Q}^T\mathbf{o}_i))) ...)), \tag{2}$$

where ϕ_{out} is the mapping function for the output layer and ϕ_x^G denotes the mapping function for the x-th global neural layer. Φ_x^L represents a series of mapping functions in the x-th local neural block, which can be formulated as:

$$\Phi_x^L(y_x) = \phi_{x,Z}^L(...\phi_{x,2}^L(\phi_{x,1}^L(y_x))...), \tag{3}$$

where $\phi_{x,z}^L$ represents the mapping function for the z-th layer in the x-th local neural block, and there are Z hidden layers in the block. y_x in Eq. (3) is the output of the x-th global layer and it is directly fed into the first layer of the x-th local block. In Ref. [12], the generalized linear model is replaced by a MLP (the micro network). The input data is directly fed into the micro network. In our method, the input is firstly fed into the global network, and then the output of each global layer is taken as the input for the local block. Finally, the output of the local block is fed into the next global layer. Our method is different from methods in Ref. [12] and [6]. The architecture of our network is given in Fig. 2.

Intuitively, the computation of the predictive score is a regression problem. The data we used is implicit feedback of users and we changed the data into binary forms. Thus, we address the item recommendation with implicit feedback as a binary classification problem. For classification, if it is a binary (2-class) problem, then the cross-entropy error function often does better [29].

The predictive score $\hat{y}_{ui}$ gives the probability that user u will purchase item i. Consequently, we utilize a probabilistic function (e.g. the *Logistic* or *Probit* function) as the activation function for the output layer ϕ_{out} to constrain the predictive score $\hat{y}_{ui}$ to be in the range of [0, 1]. With the above setting, we define the likelihood function as follow:

$$p(\mathcal{Y}, \mathcal{Y}^-|\mathbf{P}, \mathbf{Q}) = \prod_{(u,i)\in\mathcal{Y}} \hat{y}_{ui} \prod_{(u,i)\in\mathcal{Y}^-} (1-\hat{y}_{ui}), \tag{4}$$

where $\mathcal{Y}$ denotes the set of observed entries in the adjacent matrix Y, and $\mathcal{Y}^-$ is the set of negative instances, which can be sampled from missing values since it is quite difficult to infer the negative preference from these miss values. It is worth mentioning that a missing value does not mean a negative feedback.

By taking the negative logarithm of the likelihood, we can obtain the object function of our method:

$$\begin{aligned} \Phi &= -\sum_{(u,i)\in\mathcal{Y}} \log \hat{y}_{ui} - \sum_{(u,i)\in\mathcal{Y}^-} \log(1-\hat{y}_{ui}) \\ &= -\sum_{(u,i)\in\mathcal{Y}\bigcup\mathcal{Y}^-} \hat{y}_{ui}\log\hat{y}_{ui} + (1-\hat{y}_{ui})\log(1-\hat{y}_{ui}). \end{aligned} \tag{5}$$

The optimization can be done by performing stochastic gradient descent (SGD). For the negative instances $\mathcal{Y}^-$, we uniformly sample them from unobserved interactions in each iteration and correlate the sampling ratio $w.r.t$ the number of observed interactions.

4.2. Local neural block

The local neural block is inserted into two adjacent global layers. Two structures are picked to construct the local neural block: traditional MLP and deep residual neural network.

4.2.1. MLP

This neural network has the same structure with the global neural network, in which the number of nodes in the n-th neural layer is as twice as that of the $(n-1)$-th layer. The architecture contained such structure is illustrated in Fig. 3 and the formulation is defined as:

$$\begin{aligned} \mathbf{z}_1^G &= \phi_1^G(\mathbf{p}_u, \mathbf{q}_i) = \begin{bmatrix} \mathbf{p}_u \\ \mathbf{q}_i \end{bmatrix}, \\ \mathbf{z}_{1,1}^L &= \phi_{1,1}^L(\mathbf{z}_1^G) = a_{1,1}^L(\mathbf{W}_{1,1}'^T\mathbf{z}_1^G + \mathbf{b}_{1,1}'), \\ \mathbf{z}_{1,2}^L &= \phi_{1,2}^L(\mathbf{z}_{1,1}^L) = a_{1,2}^L(\mathbf{W}_{1,2}'^T\mathbf{z}_{1,1}^L + \mathbf{b}_{1,2}'), \\ &\cdots\cdots, \\ \mathbf{z}_{1,Z}^L &= \phi_{1,Z}^L(\mathbf{z}_{1,Z-1}^L) = a_{1,Z}^L(\mathbf{W}_{1,Z}'^T\mathbf{z}_{1,Z-1}^L + \mathbf{b}_{1,Z}'), \\ \mathbf{z}_2^G &= \phi_2^G(\mathbf{z}_{1,Z}^G) = a_2^L(\mathbf{W}_2^T\mathbf{z}_{1,Z}^L + \mathbf{b}_2), \\ \mathbf{z}_{2,1}^L &= \phi_{2,1}^L(\mathbf{z}_2^G) = a_{2,1}^L(\mathbf{W}_{2,1}'^T\mathbf{z}_2^G + \mathbf{b}_{2,1}'), \\ &\cdots\cdots, \\ \mathbf{z}_{X,Z}^L &= \phi_{X,Z}^L(\mathbf{z}_{X,Z-1}^L) = a_{X,Z}^L(\mathbf{W}_{X,Z}'^T\mathbf{z}_{X,Z-1}^L + \mathbf{b}_{X,Z}'), \\ \hat{y}_{ui} &= \phi_{out}(\mathbf{z}_{X,Z}^L) = a_{out}(\mathbf{h}^T\mathbf{z}_{X,Z}^L + \mathbf{b}), \end{aligned} \tag{6}$$

where $\mathbf{p}_u$ and $\mathbf{q}_i$ are the embedding vector for user u and item i. $\mathbf{W}_x$, $\mathbf{b}_x$ and a_x^G denote the weight matrix, the bias vector and the activation function for the x-th global layer, respectively. $\mathbf{W}_{x,z}'$, $\mathbf{b}_{x,z}'$ and $a_{x,z}^L$ denote the weight matrix, the bias vector and the activation function for the z-th layer in the x-th local neural block. The size of $\mathbf{W}_x$ changes with the layer's position in the global neural network shifting. The row number of $\mathbf{W}_x$ is $D \cdot 2^{X-x+1}$ and the column number of $\mathbf{W}_x$ is $D \cdot 2^{X-x}$, where D is the length of the

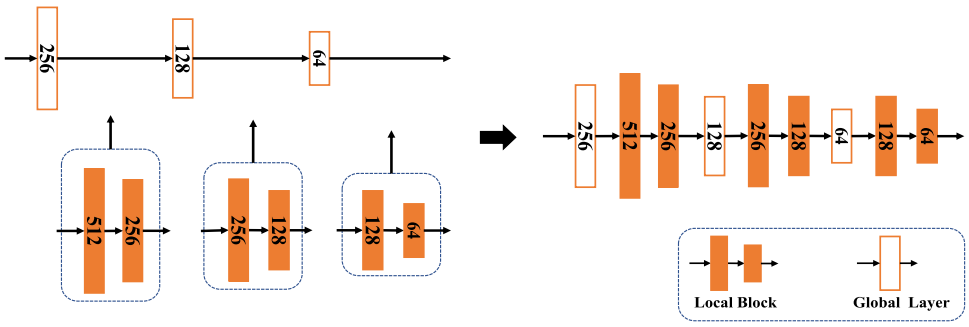


Fig. 3. The architecture of the local neural block built by the traditional MLP. Numbers in the rectangles denote the size of the layer.

last global layer X. The length of the first layer in the x-*th* local neural block is determined by both the length of the x-*th* global layer and the depth of the local neural block, so the row number and column number of $\mathbf{W}'_{x,1}$ are $D \cdot 2^{(X-x)}$ and $D \cdot 2^{(X-x)+(Z-1)}$, respectively. The length of the z-*th* ($z \neq 1$) layer is determined by the length of the $(z-1)$-*th* layer. Thus the row number and the column number of $\mathbf{W}'_{x,z}$ are $D \cdot 2^{(X-x)+(Z-z+1)}$ and $D \cdot 2^{(X-x)+(Z-z)}$, respectively. In Fig. 3, the global network is a MLP with the architecture of $256 \rightarrow 128 \rightarrow 64$. There are accordingly three local blocks: $512 \rightarrow 256$, $256 \rightarrow 128$ and $128 \rightarrow 64$. By combining the global network and local blocks, there are nine layers in the network.

4.2.2. Deep residual neural network

This structure is proposed by He et al. [16] in 2016 to handle the degradation problem in the field of the computer vision. The residual neural network adds shortcut connections among disconnected layers, and outputs of those shortcut connections are combined with outputs of a few stacked layers. Suppose $\mathcal{H}(x)$ is an underlying mapping to be fit by a few stacked layers, where x denotes the inputs to the first of these layers. These stacked layers approximate another residual function $\mathcal{F}(x) := \mathcal{H}(x) - x$. With the residual neural network, our model is defined as:

$$
\begin{aligned}
\mathbf{z}_1^G &= \phi_1^G(\mathbf{p}_u, \mathbf{q}_i) = \begin{bmatrix} \mathbf{p}_u \\ \mathbf{q}_i \end{bmatrix}, \\
\mathbf{z}_{1,1}^L &= \phi_{1,1}^L(\mathbf{z}_1^G) = a_{1,1}^L(\mathbf{W}_{1,1}'^T \mathbf{z}_1^G + \mathbf{b}'_{1,1}), \\
\mathbf{z}_{1,2}^L &= \phi_{1,2}^L(\mathbf{z}_{1,1}^L) = a_{1,2}^L(\mathbf{W}_{1,2}'^T \mathbf{z}_{1,1}^L + \mathbf{b}'_{1,2}), \\
&\cdots\cdots, \\
\mathbf{z}_{1,Z}^L &= \phi_{1,Z}^L(\mathbf{z}_{1,Z}^L) = a_{1,Z}^L(\mathbf{W}_{1,Z}'^T \mathbf{z}_{1,Z-1}^L + \mathbf{b}'_{1,Z}), \\
\mathbf{z}_1 &= \mathbf{z}_1^G + \mathcal{F}(\mathbf{z}_1^G, \mathcal{W}_1) = \mathbf{z}_1^G + \mathbf{z}_{1,Z}^L \\
\mathbf{z}_2^G &= \phi_2^G(\mathbf{z}_1) = a_2^G(\mathbf{W}_2^T z_1 + \mathbf{b}_2), \\
\mathbf{z}_{2,1}^L &= \phi_{2,1}^L(\mathbf{z}_2^G) = a_{2,1}^L(\mathbf{W}_{2,1}'^T \mathbf{z}_2^G + \mathbf{b}'_{2,1}), \\
&\cdots\cdots, \\
\mathbf{z}_{X,Z}^L &= \phi_{L,Z}^L(\mathbf{z}_{X,Z-1}^L) = a_{X,Z}^L(\mathbf{W}_{X,Z}'^T \mathbf{z}_{X,Z-1}^L + \mathbf{b}'_{X,Z}), \\
\mathbf{z}_X &= \mathbf{z}_X^G + \mathcal{F}(\mathbf{z}_X^G, \mathcal{W}_X) = \mathbf{z}_X^G + \mathbf{z}_{X,Z}^L \\
\hat{y}_{ui} &= \phi_{out}(\mathbf{z}_X) = a_{out}(\mathbf{h}^T \mathbf{z}_X + \mathbf{b}).
\end{aligned} \tag{7}
$$

We choose the tower pattern structure for the residual neural network. The combination of the residual neural network and the global neural network is presented in Fig. 4.

For the activation function of each neural layer, we choose ReLU which yields a better performance than tanh and sigmoid [6]. The sigmoid function may suffer from saturation where neurons stop learning when their output is near either 0 or 1. Tanh only alleviates the issues of sigmoid to a certain extent since it can be considered as a rescaled version of sigmoid [30]. ReLU is proven to be non-saturated and more suitable for sparse data, making the model less likely to be overfitting [31]. The sigmoid function $\sigma(x) = 1/(1+e^{-x})$ is selected as the activation function for the output layer ϕ_{out} whose range is in [0, 1]. The pseudo code of our method is presented in Algorithm 1.

4.3. Fusion of MF and our method

The previous work indicated that the traditional MF model can be easily extended to the neural collaborative filtering framework, called *Generalized Matrix Factorization* (GMF for short) [6]. The fusion of GMF and MLP can further improve the accuracy of the algorithm. In a similar manner, we combine GMF and our method by concatenating their last hidden layer. Fig. 5 illustrates this proposal and the formulation is given as follows:

$$
\begin{aligned}
\phi^{GMF} &= \mathbf{p}_u^{GMF} \odot \mathbf{q}_i^{GMF} \\
\mathbf{z}_1^G &= \begin{bmatrix} \mathbf{p}_u^{Hybrid} \\ \mathbf{q}_i^{Hybrid} \end{bmatrix}, \\
\mathbf{z}_{1,1}^L &= a_{1,1}^L(\mathbf{W}_{1,1}'^T \mathbf{z}_1^G + \mathbf{b}_{1,1}), \\
\mathbf{z}_{1,2}^L &= a_{1,2}^L(\mathbf{W}_{1,2}'^T \mathbf{z}_{1,1}^L + \mathbf{b}_{1,2}), \\
&\cdots\cdots, \\
\mathbf{z}_{1,Z}^L &= a_{1,Z}^L(\mathbf{W}_{1,Z}'^T \mathbf{z}_{1,Z-1}^L + \mathbf{b}_{1,Z}), \\
\mathbf{z}_1 &= \mathbf{z}_1^G + \mathbf{z}_{1,Z}^L \\
\mathbf{z}_2^G &= a_2^G(\mathbf{W}_2^T \mathbf{z}_1 + \mathbf{b}_2), \\
\mathbf{z}_{2,1}^L &= a_{2,1}^L(\mathbf{W}_{2,1}'^T \mathbf{z}_2^G + \mathbf{b}_{2,1}), \\
&\cdots\cdots, \\
\mathbf{z}_{X,Z}^L &= a_{X,Z}^L(\mathbf{W}_{X,Z}'^T \mathbf{z}_{X,Z-1}^L + \mathbf{b}_{X,Z}), \\
\phi^{Hybrid} &= \mathbf{z}_X^G + \mathbf{z}_{X,Z}^L \\
\hat{y}_{ui} &= \sigma(\mathbf{h}^T \begin{bmatrix} \phi^{GMF} \\ \phi^{Hybrid} \end{bmatrix})
\end{aligned} \tag{8}
$$

Eq. (8) gives the formulation when the residual network is used to build the local neural network (e.g. $\mathbf{z}_1 = \mathbf{z}_1^G + \mathbf{z}_{1,Z}^L$). If the shortcut connections are taken away, the output of the xth local block is directly fed into the next global layer, which means $\mathbf{z}_x = \mathbf{z}_{x,Z}^L$.

Likewise, we pre-train GMF and hybrid neural network with random initializations until convergence and then use their parameters as the initialization for the hybrid model's parameters. We concatenate these two models on the output layer as follow:

$$
\mathbf{h} \leftarrow \begin{bmatrix} \alpha \mathbf{h}^{GMF} \\ (1-\alpha)\mathbf{h}^{Hybrid} \end{bmatrix}, \tag{9}
$$

where $\mathbf{h}^{GMF}$ and $\mathbf{h}^{Hybrid}$ denote the $\mathbf{h}$ vector of the pre-trained GMF and the hybrid neural network, respectively. α is a hyper-parameter which determines the trade-off between these two pre-trained models.

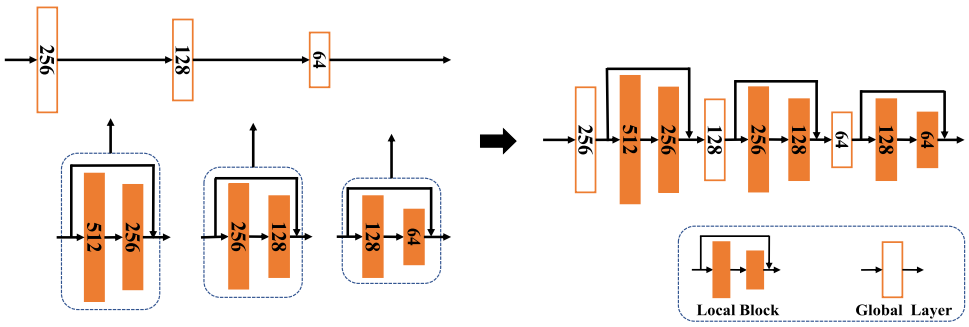


Fig. 4. The architecture of the local neural block built by the deep residual network. Numbers in the rectangles denote the size of the layer.

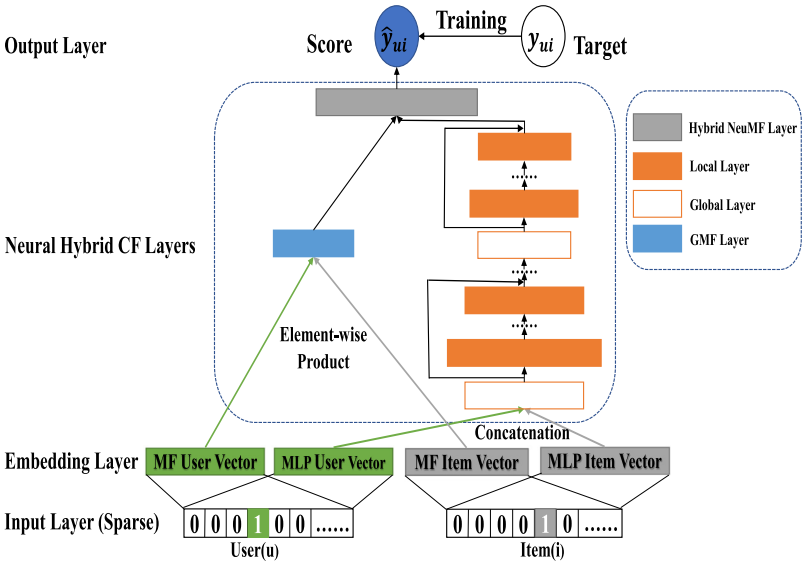


Fig. 5. The combination of the hybrid neural network and GMF.

The Adaptive Moment Estimation (Adam) is adopted to learn parameters in models, computing adaptive learning rates for parameters by estimating the first and the second moments of the gradients [32]. After pre-training parameters of GMF and the hybrid neural network, we change the approximation approach from Adam to vanilla SGD for it is unsuitable to optimize the combined model with momentum-based methods [6].

4.4. Summary of our methods

To model interactions between users and items, we propose a hybrid neural network consisting of the global neural network and the local neural block. Two architectures are accordingly used to build the local neural block and the hybrid neural network is combined with the GMF model. To give a clear explanation of our method, abbreviations and descriptions of our methods are presented in Table 1.

5. Experiments

5.1. Experimental setup

5.1.1. Datasets

In order to evaluate the accuracy of our method, five benchmark datasets are selected, namely *MovieLens-100K*, *MovieLens-1M*, *Douban Book*, *Amazon Movies* and *Amazon Games*. MovieLens is a movie recommendation website, which employs individual ratings to generate personalized recommendations.[2] The *MovieLens-100K* dataset consists of 943 users and 1682 movies and the *MovieLens-1M* dataset has 6040 users and 3706 movies. Douban, launched on March 6, 2005, is a Chinese Web 2.0 website which provides user reviews and recommendation services of movies, books, and music [33]. The raw data contains user activities before Aug 2010 and we filter out those users who have rated fewer than 20 movies because it is difficult to accurately recommend items for inactive users. *Amazon.com* is a multinational e-commerce company and the world's largest online retailer. The raw Amazon dataset contains product reviews and product metadata spanning from May, 1996 to July, 2014 [34].[3] We select individual ratings on movies and games to evaluate our method and filter out those users who have rated fewer than 20 products. For explicit ratings, we transform them into binary forms. If the user has rated the item, the corresponding entry is marked as 1. If the explicit rating is unobserved, the entry is marked as 0. The statistics of datasets is presented in Table 2.

5.1.2. Evaluation and metrics

In order to measure the performance of recommendation methods, the *leave-one-out* evaluation is taken into consideration. For each user, her/his latest interaction is held-out as the test set and the remaining interactions are used as the training set.

[2] https://grouplens.org/datasets/movielens/
[3] https://jmcauley.ucsd.edu/data/amazon/

Table 1
A summary of our methods.

Abbreviation	Description
$HybridNN_{MLP}$	The local neural block is built by MLP.
$HybridNN_{res}$	The local neural block is built by the residual neural network.
$HybridNNMF_{MLP}$	The hybrid neural network is combined with GMF and the local neural block is built by MLP.
$HybridNNMF_{res}$	The hybrid neural network is combined with GMF and the local neural block is built by the residual neural network.

ALGORITHM 1: Heterogeneous neural collaborative filtering.

Input:
Iter: training iterations.
neg: the number of negative samples.
γ: learning rate.
Y: the adjacent matrix.
Output:
P: latent matrix for user;
Q: latent matrix for item;
Θ_f^G: parameters of the global neural network;
Θ_f^L : parameters of the local neural block;

```
// Initialisation
```

randomly initialize **P**, **Q**, Θ_f^G and Θ_f^L;
$\mathcal{Y} \leftarrow$ observed interaction set;
$\mathcal{Y}^- \leftarrow$ unobserved interaction set;
for it *from 1 to* Iter **do**
  set $\mathcal{Y}^-_{sampled} \leftarrow$ sampling neg^* | $\mathcal{Y}$ | unobserved interactions from $\mathcal{Y}^-$;
  set $\mathcal{T} \leftarrow$ from $\mathcal{Y} \cup \mathcal{Y}^-_{sampled}$;
  for *each interaction of user u and item i in* $\mathcal{T}$ **do**
    // Global neural network
    $\mathbf{p}_u \leftarrow \mathbf{P}^T \mathbf{v}_u$;
    $\mathbf{q}_i \leftarrow \mathbf{Q}^T \mathbf{o}_i$;
    for *global layer x from 1 to X* **do**
      if *x==1* **then**
        set $\mathbf{z}_1^G \leftarrow$ with the input of $\mathbf{v_u}$, $\mathbf{o_i}$;
      else
        set $\mathbf{z}_x^G \leftarrow$ with the input of $\mathbf{z}^L_{x-1,Z}, \mathbf{z}^G_{x-1}$;
      end
      // Local neural block
      for *local layer z from 1 to Z* **do**
        if *z==1* **then**
          set $\mathbf{z}^L_{x,1} \leftarrow$ with the input of $\mathbf{z}^G_x$;
        else
          set $\mathbf{z}^L_{x,z} \leftarrow$ with the input of $\mathbf{z}^L_{x,z-1}$;
        end
      end
    end
    set $\hat{y}_{ui} \leftarrow$ use the input $\mathbf{z}^L_{X,Z}$;
    set $L \leftarrow$ use Eq. (5) with input of $\hat{y}_{ui}$, y_{ui};
    updating model parameters with back propagation:
    set $\Theta_f^G \leftarrow \Theta_f^G - \gamma \frac{\partial \Phi}{\partial \Theta_f^G}$;
    set $\Theta_f^L \leftarrow \Theta_f^L - \gamma \frac{\partial \Phi}{\partial \Theta_f^L}$.
    set $\mathbf{p}_u \leftarrow \mathbf{p}_u - \gamma \frac{\partial \Phi}{\partial \mathbf{p}_u}$;
    set $\mathbf{q}_i \leftarrow \mathbf{q}_i - \gamma \frac{\partial \Phi}{\partial \mathbf{q}_i}$;
  end
end

The leave-one-out evaluation method is adopted mainly based on two considerations. Firstly, for the benchmark methods, authors also utilize leave-one-out method to measure the performance of recommendation algorithms [6,9]. To be fair, we use the same evaluation method and process. The leave-out-out method is widely used to evaluate the performance of methods by previous literature [35,36]. Secondly, the latest item of a user is selected as the test set. The main purpose of designing this evaluation process lies in simulating a user's selection on items in the recommender system, namely using a user's historical data to predict the user's future preferences or choices.

Table 2
The statistics of benchmark datasets.

Dataset	Interaction#	Item#	User#	Sparsity
MoiveLens-100K	100000	1682	943	93.70%
MoiveLens-1M	1000209	3706	6040	95.53%
Douban Book	696669	30687	8144	99.72%
Amazon Movies	887233	67975	15063	99.91%
Amazon Games	62203	9364	1637	99.94%

Since it is quite time-consuming to rank all items for every user during evaluation, we randomly sample 100 items which are not interacted by the target user and rank the test item among these 100 items [6]. *Hit Ratio* (HR) as well as *Normalized Discounted Cumulative Gain* (NDCG) are accordingly selected to measure the ranked list of recommendation approaches. We truncate the ranked list at 10 for both metrics. Since there is only one test item in each user's test set, $HR_u = 1$ indicates that the test item is presented on the top-10 list for user u, and 0 otherwise. We average all user's HR_u as the final metric: $HR = \frac{1}{M}\sum_{u=1}^{M} HR_u$. The NDCG measures the position of the test item in the ranked list, which is defined as: $NDCG_u = \mathcal{Z}\sum_{j=1}^{10} \frac{2^{r_j}-1}{log(j+1)}$, where $\mathcal{Z}$ is the normalizer to ensure the perfect ranking has a value of 1; r_j is the graded relevance of item at position j. We use the simple binary relevance for our work: $r_j = 1$ if the item is in the test set, and 0 otherwise. In a similar way, we average $NDCG_u$ over all users: $NDCG = \frac{1}{M}\sum_{u=1}^{M} NDCG_u$.

5.1.3. Baseline methods

We compare our method with the following approaches:

1. **ItemPop** [1]. Items are ranked according to their popularity which is reflected by the number of interactions. This is a non-personalized method selected as the benchmark approach.
2. **ItemKNN** [37]. This is the standard item-based collaborative filtering method with the assumption that a user tends to collect similar items. We make use of implicit feedbacks to calculate the item similarity.
3. **BPR** [38]. The bayesian personalized ranking (BPR) is a generic optimization method for personalized ranking. We apply this method to matrix factorization model by utilizing implicit feedbacks of users.
4. **SoloMLP** [6]. This method exploited the traditional MLP to model the nonlinear relationships between users and items. Furthermore, this method is combined with the matrix factorization. The combined method is termed as NeuMF. Both SoloMLP and NeuMF are compared with out method.
5. **DeepICF** [9]. Such method captures both second-order and higher order interactions among items. In the method, the attention network is used for adaptively learning the relative importance of these interactions.

5.1.4. Parameter settings

All parameters are randomly initialized (including baseline methods) with a Gaussian distribution (with a mean of 0 and a standard deviation of 0.01) [39]. We follow the evaluation process and parameter setting of [6]. For our method, the depth of the local neural block is set to 2 and the pre-training parameter α is set to 0.5. For ItemKNN, we test the number of neighbors of [20,50,100]. SoloMLP, NCF and our proposed methods contain some hyperparameters, including learning rate, negative sampling ratio and the number of global layers. We test learning rate of [0.001,0.05,0.01,0.05,0.1] and the number of global layers in [1,3,6]. The negative sampling ratio is from 1 to 10. The architecture of global network and local blocks is shown in Table 3 and their combinations are illustrated in Figs. 3 and 4.

5.2. Results and analysis

5.2.1. The weight of the last layer

The final predictive scores are obtained via the layer weight $\mathbf{h}$ and the output vector $\mathbf{z}$: $\hat{y} = a_{out}(\mathbf{h}^T\mathbf{z} + b)$. The layer weight refers to the weights of neurons in the last layer. For the sake of better demonstration, we directly plot the layer weights with their absolute values (not the probability distribution) and the result is given in Fig. 6, wherein the blue histogram represents the layer weight distribution of MLP and the orange histogram gives the layer weight distribution of $HybridNN_{res}$ network. All the weights are obtained when the network achieves the best predictive accuracy. From the figure, it can be seen that layer weights of $HybridNN_{res}$ distribute more diversely than layer weights of MLP. Taking *MovieLens-100K* dataset for instance, the layer weights of MLP are mainly concentrated in the interval of 0 to 0.1 while the range of the weight is enlarged nearly three times by $HybridNN_{res}$ network. However, for *MovieLens-1M* and *Amazon Movies* datasets, the weight distribution of $HybridNN_{res}$ is not significantly different from the distribution of SoloMLP. On the *Amazon Movies* dataset, the layer weights of SoloMLP distribute more diversely than $HybridNN_{res}$. For most recommender systems, individual preferences are quite complex and the deep network may be better than the shallow network to learn features of users. However, for certain recommender systems (e.g. *Amazon Movies*), their users may have relatively simple preferences. As a result, the shallow network may have a better feature learning capability than the deep network in these systems.

In order to quantify the results of Fig. 6, we compute the information entropy of the probability distribution of the layer weights. We unify the range in [-0.5,0.5] for all datasets, dividing a total of 20 intervals, and then count the probability of the weight value in each interval. The information entropy is computed by $H = \sum_{i=1}^{20}(p_i log(p_i))$, where p_i is the probability of the weight value in the *ith* interval. The higher the entropy is, the more uncertain the variable is. The entropy results are presented in Table 4 where one can see that entropy of $HybridNN_{res}$ network is higher than the entropy of SoloMLP network except on the *Amazon Movies* dataset. As mentioned before, the shallow network may be more suitable for this dataset. These results provide a potential way to choose the proper network. In other words, one can choose the shallow network or deep network by analyzing the weight distribution of the last layer.

5.2.2. The visualization of output vector

In addition, we visualize the output vector. Firstly, we randomly sample 1000 positive instances (positive user–item pairs) and 1000 negative instances. Their embedding vectors are fed into the *SoloMLP* and $HybridNN_{res}$ network. Secondly, we compute the output vector $\mathbf{z}_{MLP}$ and $\mathbf{z}_{hybrid}$ whose dimensions are both 64. Finally, the T-SNE method is applied to map the output vectors into two dimensions [40], and the Fig. 7 gives the visualization results. It can be seen that most of those instances can be classified by *SoloMLP* and $HybridNN_{res}$ network and $HybridNN_{res}$'s classification result seems better than the result of *SoloMLP*. In order to quantify the results, we make use of the method in Ref. [41] to compute the Kullback–Leibler (KL) divergence and the results are shown in Table 5. The smaller the KL divergence is, the better the classification represents. From these results, $HybridNN_{res}$'s KL divergence is smaller than the KL divergence of *SoloMLP* for all datasets. These results imply that $HybridNN_{res}$ has a better feature learning ability than the *SoloMLP*.

5.2.3. The performance of our method

We compare our proposed methods with baseline approaches on several datasets and results are presented in Table 6. From the table, four conclusions can be obtained:

1. Without considering the combination with the GMF, the hybrid neural network $HybridNN_{res}$ is superior to SoloMLP, indicated by the results that the $HybridNN_{res}$ has the higher *HR*@10 and *NDCG*@10 than the soloMLP. On certain datasets such as *Amazon Movies*, the performance of $HybridNN_{res}$ is even close to the performance of NeuMF which incorporates both linear and nonlinear model. Such result implies our proposed mixed structures can learn better high-level representations than SoloMLP. One possible reason may be that the local neural block increases the nonlinearity of the whole network [12] such that the recommendation accuracy is augmented.
2. When the hybrid neural network is integrated with GMF ($HybridNNMF_{MLP}$ and $HybridNNMF_{res}$), the accuracy of algorithm can be further improved. This result indicates that both linear and nonlinear relationships may exist simultaneously between users and items. It is inadequate to consider only the linear (or nonlinear) relationship to build a predictive model.
3. From Table 6, it can be seen that $HybridNNMF_{res}$ enjoys the best predictive accuracy on all datasets, the method of which takes advantage of both MLP and residual neural network to construct the hybrid neural network. Although $HybridNNMF_{MLP}$ builds the collaborative filtering network with a "network-in-network" architecture, its accuracy is worse than $HybridNNMF_{res}$. One possible reason may be that the learning ability of the residual network is superior to the traditional MLP. The residual network is proved to have excellent generalization performance on recognition tasks and previous works also reveal that both the accuracy and efficiency of traditional networks can be greatly enhanced by residual connections [22]. Another possible reason is that $HybridNNMF_{res}$ employs heterogeneous networks to build the collaborative filtering network, which makes the information acquired by the network more diverse.
4. The DeepICF makes use of the attention network to capture the relative importance of higher order interactions of items. Broadly speaking, this network can be regarded as one kind of the network-in-network architecture. It can be seen that $HybridNNMF_{res}$ outperforms DeepICF in terms of top-n item recommendations. This result suggests that our method is better than DeepICF in modeling user–item interactions.

Table 6 only shows the comparative results when the length of the recommendation list is 10. We give the performance of algorithms in Table 7 when the length of the recommendation list ranges from 2 to 10. From the table, it can be seen that the accuracy (*HR* and *NDCG*) of $HybridNNMF_{res}$ and *NeuMF* is better

Table 3
The architecture of global network and local blocks.

Global network	local blocks
$\rightarrow 32 \rightarrow$ $128 \rightarrow 64 \rightarrow 32$	$64 \rightarrow 32$ $256 \rightarrow 128,\ 128 \rightarrow 64,\ 64 \rightarrow 32$
$1024 \rightarrow 512 \rightarrow 256 \rightarrow 128 \rightarrow 64 \rightarrow 32$	$2048 \rightarrow 1024,\ 1024 \rightarrow 512,\ 512 \rightarrow 256,$ $256 \rightarrow 128,\ 128 \rightarrow 64,\ 64 \rightarrow 32$

Table 4
The information entropy of the probability distribution of the layer weights.

	MovieLens-100K	*MovieLens-1M*	*Douban Book*	*Amazon Movies*	*Amazon Games*
SoloMLP	1.73	2.09	1.96	2.25	1.63
HybridNN$_{res}$	3.53	2.11	3.11	1.95	3.32

Table 5
The Kullback–Leibler divergence.

	MovieLens-100K	*MovieLens-1M*	*Douban Book*	*Amazon Movies*	*Amazon Games*
SoloMLP	0.242	0.216	0.220	0.206	0.333
HybridNN$_{res}$	0.228	0.200	0.208	0.194	0.198

Table 6
The performance of different models in five datasets. Five independent experiments are executed and different initial parameters are used in each experiment. The standard error is shown in the brackets and bold values indicate the best results.

	MovieLens-100K		*MovieLens-1M*		*Douban Book*		*Amazon Movies*		*Amazon Games*	
	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10
ItemPOP	0.426 (0)	0.240 (0)	0.453 (0)	0.254 (0)	0.524 (0)	0.326 (0)	0.244 (0)	0.130 (0)	0.353 (0)	0.196 (0)
ItemKNN	0.643 (0)	0.364 (0)	0.668 (0)	0.398 (0)	0.617 (0)	0.454 (0)	0.433 (0)	0.339 (0)	0.502 (0)	0.304 (0)
BPR	0.679 (.0016)	0.382 (.0008)	0.674 (.0002)	0.399 (.0010)	0.664 (.0010)	0.445 (.0020)	0.695 (.0024)	0.457 (.0027)	0.511 (.0040)	0.302 (.0028)
SoloMLP	0.681 (.0020)	0.396 (.0040)	0.704 (.0010)	0.422 (.0010)	0.678 (.0030)	0.458 (.0040)	0.674 (.0010)	0.441 (.0030)	0.504 (.0030)	0.288 (.0030)
NeuMF	0.705 (.0010)	0.410 (.0030)	0.729 (.0010)	0.449 (.0020)	0.702 (.0010)	0.475 (.0020)	0.703 (.0020)	0.471 (.0020)	0.530 (.0020)	0.327 (.0030)
DeepICF	0.691 (.0013)	0.400 (.0070)	0.695 (.0009)	0.423 (.0015)	0.669 (.0015)	0.463 (.0005)	0.525 (.0034)	0.316 (.0025)	0.516 (.0045)	0.312 (.0039)
HybridNN$_{MLP}$	0.684 (.0017)	0.397 (.0022)	0.721 (.0011)	0.435 (.0028)	0.704 (.0013)	0.471 (.0016)	0.701 (.0006)	0.469 (.0009)	0.503 (.0019)	0.290 (.0030)
HybridNN$_{res}$	0.698 (.0006)	0.401 (.0042)	0.719 (.0002)	0.437 (.0027)	0.694 (.0002)	0.465 (.0029)	0.700 (.0023)	0.466 (.0016)	0.515 (.0001)	0.310 (.0029)
HybridNNMF$_{MLP}$	0.707 (.0007)	0.411 (.0292)	0.737 (.0002)	0.453 (.0013)	0.712 (.0019)	0.487 (.0013)	0.713 (.0001)	0.478 (.0021)	0.547 (.0009)	0.334 (.0006)
HybridNNMF$_{res}$	**0.726** (.0006)	**0.420** (.0018)	**0.738** (.0008)	**0.456** (.0013)	**0.713** (.0006)	**0.487** (.0029)	**0.719** (.0002)	**0.483** (.0032)	**0.562** (.0007)	**0.343** (.0017)

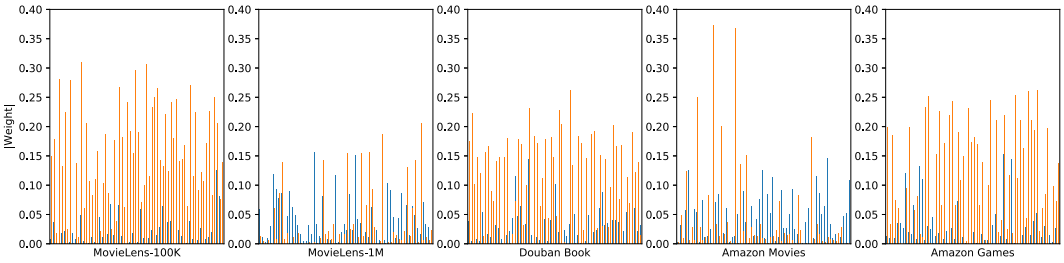


Fig. 6. The layer weights (absolute value) of the last layer. The blue histogram represents the layer weight distribution of MLP and the orange histogram gives the layer weight distribution of our hybrid network.

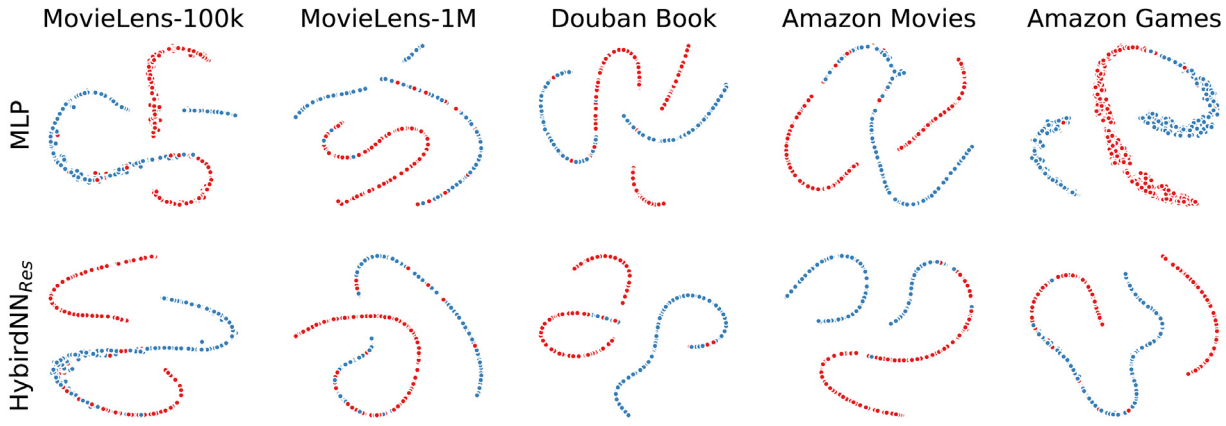


Fig. 7. The visualization of output vectors. The red dots denotes the mapped output vectors $w.r.t$ positive instances and the blue dots indicate the mapped output vectors of negative instances.

than the accuracy of *ItemKNN* and *BPR* since the two former methods take both linear and nonlinear models into account. It is worth mentioning that the simple algorithm *ItemKNN* achieves competitive performance to certain complex methods such as

Table 7
Evaluation of Top-K item recommendation where K ranges from 2 to 10.

	HR@10				*NDCG*@10			
MovieLens-100K								
K	*itemKNN*	*BPR*	*NeuMF*	*HybridNNMF$_{res}$*	*itemKNN*	*BPR*	*NeuMF*	*HybridNNMF$_{res}$*
2	0.243	0.261	**0.291**	0.287	0.211	0.223	0.249	**0.252**
4	0.393	0.404	0.445	**0.455**	0.282	0.290	0.322	**0.327**
6	0.479	0.513	**0.559**	0.557	0.298	0.314	0.346	**0.350**
8	0.567	0.606	0.639	**0.660**	0.342	0.361	0.390	**0.396**
10	0.643	0.679	0.705	**0.726**	0.364	0.382	0.410	**0.420**
MovieLens-1M								
K	*itemKNN*	*BPR*	*NeuMF*	*HybridNNMF$_{res}$*	*itemKNN*	*BPR*	*NeuMF*	*HybridNNMF$_{res}$*
2	0.284	0.293	0.343	**0.351**	0.243	0.251	0.297	**0.303**
4	0.432	0.442	0.502	**0.513**	0.313	0.321	0.372	**0.379**
6	0.531	0.543	0.604	**0.612**	0.334	0.343	0.393	**0.401**
8	0.605	0.617	0.675	**0.687**	0.374	0.382	0.433	**0.441**
10	0.662	0.674	0.729	**0.738**	0.390	0.399	0.449	**0.456**
Douban Book								
2	0.415	0.369	0.402	**0.417**	0.371	0.328	0.355	**0.369**
4	0.524	0.500	0.539	**0.554**	0.423	0.389	0.419	**0.433**
6	0.570	0.579	0.616	**0.629**	0.432	0.406	0.437	**0.449**
8	0.597	0.629	0.664	**0.677**	0.449	0.435	0.464	**0.476**
10	0.617	0.664	0.701	**0.713**	0.454	0.445	0.475	**0.487**
Amazon Movies								
2	0.322	0.374	0.394	**0.402**	0.293	0.330	0.347	**0.357**
4	0.383	0.511	0.528	**0.543**	0.322	0.394	0.411	**0.423**
6	0.406	0.594	0.612	**0.622**	0.327	0.412	0.429	**0.441**
8	0.421	0.653	0.666	**0.679**	0.335	0.444	0.459	**0.471**
10	0.433	0.695	0.703	**0.719**	0.339	0.457	0.471	**0.483**
Amazon Games								
2	0.229	0.220	0.246	**0.257**	0.199	0.188	0.217	**0.223**
4	0.330	0.336	0.360	**0.389**	0.247	0.243	0.271	**0.285**
6	0.403	0.407	0.434	**0.460**	0.264	0.259	0.286	**0.300**
8	0.455	0.461	0.488	**0.520**	0.290	0.287	0.316	**0.331**
10	0.502	0.511	0.530	**0.562**	0.304	0.302	0.328	**0.343**

Table 8
The result of the significance analysis.

	HR@10			*NDCG*@10		
	ItemKNN	BPR	NeuMF	ItemKNN	BPR	NeuMF
$n_A/(n_A+n_B)*100\%$						
MovieLens-100K	74.53%	67.44%	61.36%	64.32%	62.24%	52.35%
MovieLens-1M	71.56%	70.76%	54.72%	63.07%	63.84%	52.54%
Douban Book	70.37%	66.25%	55.11%	55.55%	60.74%	53.55%
Amazon Movies	86.31%	59.29%	56.39%	71.03%	56.81%	53.44%
Amazon Games	60.12%	62.24%	60.00%	57.14%	59.53%	53.95%
z^**score*						
MovieLens-100K	6.23	3.96	2.13	6.75	5.66	0.97
MovieLens-1M	13.51	12.59	2.24	15.54	16.42	2.75
Douban Book	17.84	11.32	3.05	7.29	13.75	4.36
Amazon Movies	55.87	8.06	5.46	40.05	11.86	5.94
Amazon Games	4.48	4.51	3.22	4.39	5.44	2.12

NeuMF. Taking the Douban Book dataset for example, *ItemKNN* has a higher *NDCG* than *NeuMF* when the recommendation list is below 4. The assumption behind *ItemKNN* is that users tend to collect similar items. For items like books, users are more inclined to read similar books. After reading an author's book, the user is likely to read another book written by the same author. This may be the potential reason that *ItemKNN*'s *NDCG@K* ($K < 4$) is higher than *NeuMF*'s *NDCG*. However, the limitations of *ItemKNN* are also apparent. Firstly, the traditional *ItemKNN* takes the user list who have purchased the book to compute similarities among items, ignoring the textual information of books. As a result, the similarities obtained by *ItemKNN* may be inaccurate. Secondly, the preference pattern of users is diverse, not just choosing similar items, and *ItemKNN* method fails to capture the complexity of the preference pattern. The performance of *ItemKNN* is therefore worse than that of *NeuMF* when the recommendation list is increased.

In general, *HybridNNMF$_{res}$* outperforms *NeuMF* while the gap between these two methods is not significant on some datasets (e.g. *MovieLens-100K*). In order to perform the significance analysis, we apply the sign test which requires few assumptions about the distributional form of the data [42]. Suppose n_A is the number of users for whom our method is superior to the baseline algorithm and n_B denotes the number of users for whom the baseline algorithm is better than our method. We then calculate the z^* score by $z^* = \frac{n_A - 0.5n}{\sqrt{n/4}}$, where $n = n_A + n_B$. $|z^*| < 1.96$ indicates that our method and the baseline method are not significantly different with at least 95% confidence. If $z^* > 0$, there are at least half users for whom our method is superior to the baseline method.

Table 9
Performance of our model with different number of layers.

	HR@10			NDCG@10		
Number of layers	3	9	18	3	9	18
MovieLens-100K						
$HybridNN_{MLP}$	0.680	0.672	0.673	0.383	0.394	0.400
$HybridNN_{res}$	0.670	0.685	0.698	0.396	0.399	0.401
MovieLens-1M						
$HybridNN_{MLP}$	0.691	0.700	0.721	0.412	0.426	0.435
$HybridNN_{res}$	0.692	0.707	0.719	0.418	0.422	0.437
Douban Book						
$HybridNN_{MLP}$	0.660	0.672	0.704	0.432	0.445	0.471
$HybridNN_{res}$	0.676	0.690	0.693	0.442	0.459	0.465
Amazon Movies						
$HybridNN_{MLP}$	0.663	0.677	0.701	0.425	0.447	0.469
$HybridNN_{res}$	0.680	0.694	0.700	0.440	0.464	0.466
Amazon Games						
$HybridNN_{MLP}$	0.460	0.491	0.503	0.267	0.284	0.290
$HybridNN_{res}$	0.487	0.509	0.515	0.287	0.299	0.310

We present the result of $n_A/(n_A + n_B) * 100\%$ and z^* scores in Table 8. Firstly, the improvement of our method is significant comparing to the traditional method such as *ItemKNN*. On the Amazon Movies dataset, nearly 86.31 percent of users' recommendation accuracy is enhanced by our method. In other words, the improvement of our method is not just for a small group of users. Some users may be only fond of similar items, but most of them have diverse and complicated preferences. Therefore, it needs complex model to learn their preferences. Secondly, the gaps between $HybridNNMF_{res}$ and baseline methods (*ItemKNN*, *BPR* and *NeuMF*) are significant ($|z^*| > 1.96$) for all datasets when *HR*@10 is chosen as the metric. For *NDCG*@10, the difference between $HybridNNMF_{res}$ and *NeuMF* is not significant ($|z^*| < 1.96$) on *MovieLens-100K* dataset. The results imply that the feature learning ability may be not improved by our method on the dataset, which reflects the limitation of our method.

5.2.4. The depth of the hybrid neural network

Prior works reveal that the algorithm's accuracy is beneficial from the growth of the number of neural layers [6]. From a theoretical point of view, the result provides a potential way to improve the accuracy of the algorithm. Likewise, we also study the influence of the depth of the hybrid neural network on the algorithm's accuracy and the result is presented in Table 9.

In general, the accuracy of the hybrid network is beneficial from the augmentation of the network's depth. However, the increasing trend of the accuracy slows down as the growth of the network. For $HybridNN_{MLP}$, the accuracy of 3-layer network is better than the accuracy of 18-layer network. We devise a mixed structure to enhance the feature learning ability of the neural network, but in the real recommender system, individual preferences are extremely complicated and change over time. It is quite difficult and impractical to learn all their preference patterns by a single network. As mentioned before, some users may have relatively simple preferences and the shallow network may be more suitable to recommend items for them than the deep network.

5.2.5. The number of negative samples

When the pairwise object function in Eq. (5) is optimized, a common way is to randomly sample a certain number of negative instances for each positive instance. Table 10 shows the performance of $HybridNNMF_{res}$ when each positive instance is related to a portion of negative samples. When the ratio of negative samples ranges from 1 to 10, it is found that the optimal ratio is around 4 to 6. In consequence, we only present the result when the ratio are 1, 4, 5, 6 and 10. As a matter of fact, the algorithm achieves a competitive performance when the sample ratio equals to 1, which demonstrates the advantages of pointwise log loss in terms of the item ranking. In general, more negative samples are beneficial to the accuracy of the algorithm. However, when the sample ratio is greater than 6, the performance of $HybridNNMF_{res}$ starts to drop. In addition, the performance of the algorithm stays relatively stable when the sample ratio takes different values.

Table 10
The performance of $HybridNNMF_{res}$ $w.r.t$ the number of negative samples per positive instance. Bold values indicate the best results.

Negative ratio	1	4	5	6	10
HR@10					
MovieLens-100K	0.701	**0.722**	0.711	0.706	0.704
MovieLens-1M	0.720	0.732	**0.733**	0.732	0.725
Douban Book	0.700	0.704	0.712	**0.716**	0.714
Amazon Movies	0.712	**0.719**	0.716	0.712	0.706
Amazon Games	0.543	0.552	**0.563**	0.561	0.552
NDCG@10					
MovieLens-100K	0.397	**0.419**	0.404	0.411	0.409
MovieLens-1M	0.443	0.448	0.450	**0.452**	0.445
Douban Book	0.476	0.488	0.488	**0.492**	0.492
Amazon Movies	0.476	**0.483**	0.478	0.479	0.472
Amazon Games	0.333	0.342	**0.347**	0.348	0.340

5.2.6. The dimension of latent factors

Table 6 gives the performance of algorithms when dimensions of their embedding vectors are optimal. In Fig. 8, we present the performance of algorithms with a variety of embedding vectors' dimensions. To make the figure clearer, we only present $HybridNN_{res}$ and $HybridNNMF_{res}$ rather than all hybrid neural network methods. We only show the performance of *NeuMF* with three layers and $HybridNN_{res}$($HybridNNMF_{res}$) with nine layers. The figure illustrates that $HybridNNMF_{res}$ outperforms the remaining approaches when the embedding vector's dimension ranges from 16 to 128. One possible reason may be that $HybridNNMF_{res}$ adopts more neural layers and the information learnt by $HybridNNMF_{res}$ is more accurate than the information uncovered by baseline methods. Another possible reason may be that $HybridNNMF_{res}$ employs heterogeneous neural networks with various structures such that the information obtained by our method is more diverse than existing ones.

For $HybridNNMF_{res}$, the optimal dimension of embedding vector arranges from 32 to 64. If the dimension is too small, the information expressed by the embedding vector may be insufficient. Redundant information may appear in the embedding vector on condition that the dimension is set too large, which may hurt the accuracy of the algorithm. Moreover, Fig. 8 shows that $HybridNN_{res}$ has unstable performance on different datasets. For example, $HybridNN_{res}$ outperforms *BPR* on *MovieLens-1M* and *Douban Book* datasets, while $HybridNN_{res}$ achieves the worse accuracy than *BPR* on *Amazon Games* dataset. This result indicates that relationships between users and items are not all nonlinear. For example, some users may only be fond of comedy movies. Strong linear correlations accordingly exist between these users and comedy movies, which enables the linear model to be superior to the nonlinear method. In general, $HybridNNMF_{res}$ holds the best predictive accuracy because it integrates both linear and nonlinear models.

5.2.7. The utility of pre-training

There are plenty of ways to initialize model parameters. A commonly used approach is random initialization and another

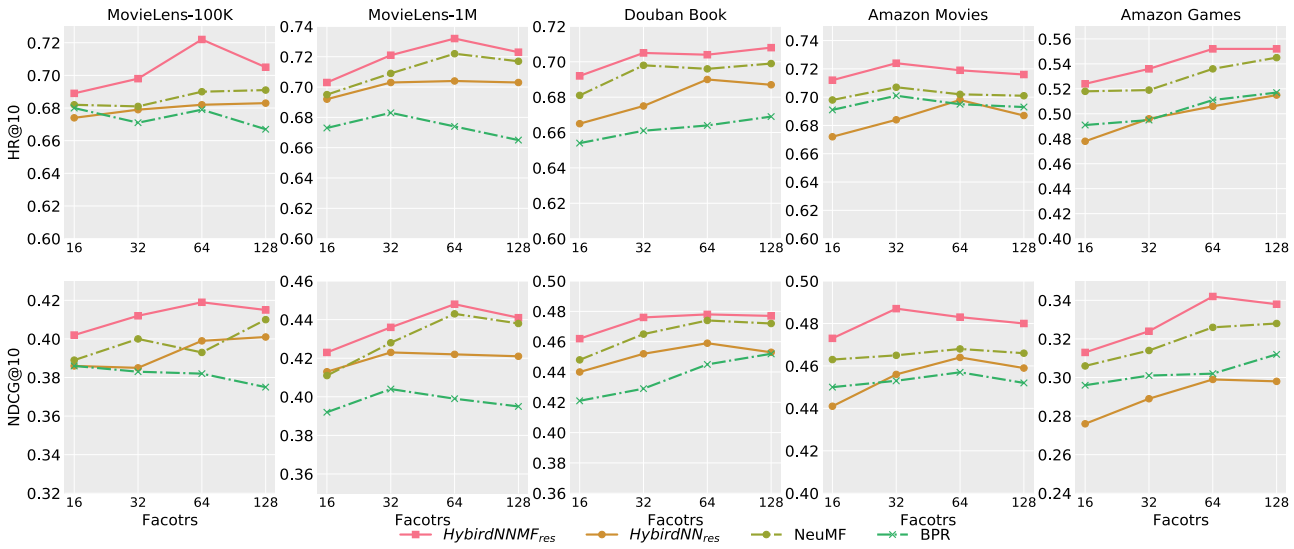


Fig. 8. The performance of HR@10 and NDCG@10 $w.r.t.$ the dimension of latent factors.

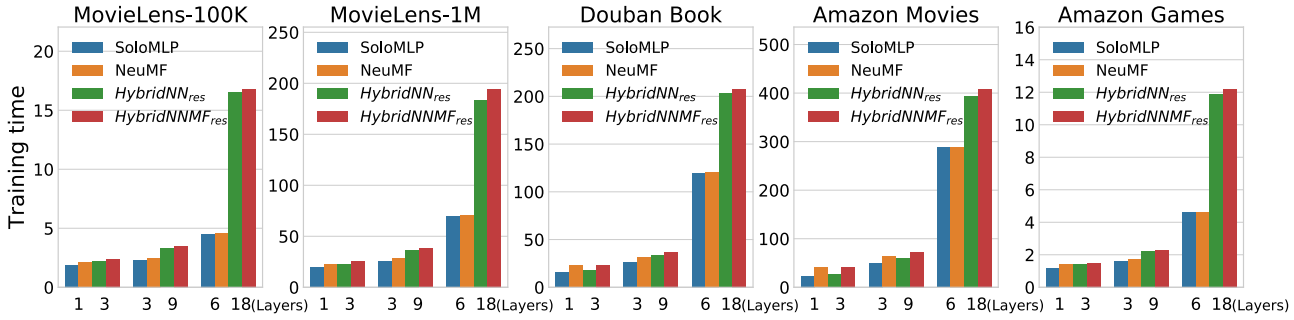


Fig. 9. The training time of methods.

Table 11
The performance of the *HybridNNMF*$_{res}$ with and without pre-training. Bold values indicate the best results.

	With pre-training		Without pre-training	
datasets	HR@10	NDCG@10	HR@10	NDCG@10
MovieLens-100K	**0.722**	**0.419**	0.699	0.400
MovieLens-1M	**0.732**	**0.448**	0.707	0.423
Douban Book	**0.704**	**0.478**	0.691	0.460
Amazon Movies	**0.719**	**0.483**	0.699	0.463
Amazon Games	0.552	**0.342**	**0.564**	0.339

one is to utilize parameters of a pre-trained model. We compare these two initialization methods and the result is given in Table 11. We make use of parameters which are pre-trained by GMF and the hybrid neural network to initialize the embedding vector (as shown in Fig. 5). This method is termed as "with pre-training". The embedding vector can also be initialized randomly, which is termed as "without pre-training". From the table, it can be seen that the algorithm with pre-training normally enjoys a better performance than the algorithm without pre-training. The possible reason may be that the pre-training process preserves particular information of the trained GMF and the hybrid neural network, and such information is beneficial to the model's accuracy.

5.2.8. The complexity analysis

Since our methods adopt more neural layers than baseline algorithms, it takes more time to train our models. We thus compare the training time of *SoloMLP*, *NeuMF*, *HybridNN*$_{res}$ and *HybridNNMF*$_{res}$. All these methods are implemented by Keras,[4] which is a high-level neural network API, written in Python and capable of running on top of Tensorflow. TensorFlow[5] is a widely used open-source machine learning framework [10]. All methods run on the same machine (Intel I7 6800K CPU and Nvidia GTX 1080 Ti GPU). The result is demonstrated in Fig. 9 where the batch size is set to 256 and the number of negative samples is 1. The training time is measured by seconds. For *SoloMLP* and *NeuMF*, the depth of the neural layer are 1, 3 and 6. For *HybridNN*$_{res}$ and *HybridNNMF*$_{res}$, the corresponding depth of the neural network are 3, 9 and 18, respectively. From the figure, it can be seen that the algorithm combined with *GMF* has small difference of training time with the algorithm uncombined with *GMF* (*SoloMLP* vs. *NeuMF*, *HybridNN*$_{res}$ vs. *HybridNNMF*$_{res}$). It is because *GMF* has one neural layer and the training cost is mostly taken by MLP or the hybrid neural network.

In addition, when the number of neural layers is relatively small (e.g. 1 layer and 3 layers for *SoloMLP*), the difference of training time between *HybridNN*$_{res}$ (*HybridNNMF*$_{res}$) and *SoloMLP* (*HybridNNMF*$_{res}$) is inapparent. When the depth of the neural network is augmented to 6, the total training time grows significantly on those relatively small datasets (e.g. *MovieLens-100K*). When the amount of the dataset is relatively small, the time taken by non-computational processes (e.g. sampling process) cannot be ignored when it is compared with the time taken by the computational processes (e.g computation of the weight

[4] https://keras.io/
[5] https://www.tensorflow.org/

matrix W). Consequently, when the depth of the neural network grows, the non-computational time is enormously increased, making the total training time increase rapidly. For those relatively large datasets (e.g. *Amazon Movies*), the cost of those non-computational processes can be ignored and the computational processes dominate the training time. As a result, the running time of $HybridNN_{res}$ ($HybridNNMF_{res}$) is not greatly raised on those relatively large datasets when the depth of the neural network is augmented. In the case of *Amazon Movies* dataset, $HybridNN_{res}$ has 36.7% more training time than *SoloMLP* and $HybridNNMF_{res}$ has 41.2% more training time than *NeuMF*. This ratio is far smaller than the ratio of these two neural networks' depth.

6. Conclusion and future work

In order to model user–item interactions, we design a hybrid neural network where the MLP is used to build the global network and the residual network is applied to form the local neural blocks. Two top-n accuracy metrics are selected to measure the performance of algorithms since the accuracy of top-n item recommendation are the main concern for users in the real world applications.

On one hand, the hybrid network with residual connections outperforms the network without them. Such result implies that the residual network has excellent representation learning ability on collaborative filtering tasks. The conclusion is also verified on the recognition tasks by the previous research [22]. On the other hand, the performance of the hybrid network can be further improved by combining the generalized matrix factorization, which means both the linear and nonlinear relationships are important to build the model. Furthermore, we study the weight distribution of the output layer and visualize the output vector. It is revealed by the results that the weight vector obtained by our method distributes more diversely comparing to the solo MLP and our method also has a smaller Kullback–Leibler divergence than the solo MLP when the output vector is visualized. However, on certain datasets (e.g. *Amazon Movies*), the feature learning ability of the network is indeed not improved by our method. On these datasets, other networks such as memory networks can be candidates to build the local block [24,43]. Thereby, there are still many open issues.

The complexity is increased by our method since we utilize a deeper network. However, the incremental ratio of the training time is far smaller than the incremental ratio of the network's depth. The complexity problem can be alleviated by certain network compression methods which are capable of reducing parameters in the network [44,45]. We will try these methods in the future work.

CRediT authorship contribution statement

Wei Zeng: Conceptualization, Methodology, Writing - original draft, Writing - review & editing. **Ge Fan:** Methodology, Software, Writing - original draft. **Shan Sun:** Writing - original draft. **Biao Geng:** Methodology. **Weiyi Wang:** Data process. **Jiacheng Li:** Software. **Weibo Liu:** Software.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The work is supported by the National Natural Science Foundation of China (61806045, 61806043 and 61872062), and Fundamental Research Funds for the Central Universities, China (2672018ZYGX2018J050).

References

[1] G. Adomavicius, A. Tuzhilin, Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions, IEEE Trans. Knowl. Data Eng. 17 (6) (2005) 734–749, http://dx.doi.org/10.1109/TKDE.2005.99.

[2] J. Davidson, B. Liebald, J. Liu, P. Nandy, T. Van Vleet, U. Gargi, S. Gupta, Y. He, M. Lambert, B. Livingston, D. Sampath, The youtube video recommendation system, in: Proceedings of the Fourth ACM Conference on Recommender Systems, in: RecSys '10, ACM, New York, NY, 2010, pp. 293–296, http://dx.doi.org/10.1145/1864708.1864770.

[3] C.A. Gomez-Uribe, N. Hunt, The netflix recommender system: Algorithms, business value, and innovation, ACM Trans. Manag. Inf. Syst. 6 (4) (2015) 13:1–13:19, http://dx.doi.org/10.1145/2843948.

[4] Z. Kang, C. Peng, Q. Cheng, Kernel-driven similarity learning, Neurocomputing 267 (C) (2017) 210–219, http://dx.doi.org/10.1016/j.neucom.2017.06.005.

[5] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (8) (2009) 30–37, http://dx.doi.org/10.1109/MC.2009.263.

[6] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, T.-S. Chua, Neural collaborative filtering, in: Proceedings of the 26th International Conference on World Wide Web, WWW '17, 2017, pp. 173–182, http://dx.doi.org/10.1145/3038912.3052569.

[7] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, R. Anil, Z. Haque, L. Hong, V. Jain, X. Liu, H. Shah, Wide & deep learning for recommender systems, in: Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, in: DLRS 2016, ACM, New York, NY, 2016, pp. 7–10, http://dx.doi.org/10.1145/2988450.2988454.

[8] G. Zhou, X. Zhu, C. Song, Y. Fan, H. Zhu, X. Ma, Y. Yan, J. Jin, H. Li, K. Gai, Deep interest network for click-through rate prediction, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, in: KDD '18, ACM, New York, NY, 2018, pp. 1059–1068, http://dx.doi.org/10.1145/3219819.3219823.

[9] F. Xue, X. He, X. Wang, J. Xu, K. Liu, R. Hong, Deep item-based collaborative filtering for top-n recommendation, ACM Trans. Inf. Syst. 37 (3) (2019) 33:1–33:25, http://dx.doi.org/10.1145/3314578.

[10] H.-J. Xue, X.-Y. Dai, J. Zhang, S. Huang, J. Chen, Deep matrix factorization models for recommender systems, in: Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI'17, 2017, pp. 3203–3209.

[11] W. Chen, F. Cai, H. Chen, M.D. Rijke, Joint neural collaborative filtering for recommender systems, ACM Trans. Inf. Syst. (TOIS) 37 (4) (2019) 1–30.

[12] M. Lin, Q. Chen, S. Yan, Network in network, in: Proceedings of the International Conference on Learning Representations, ICLR '14, 2014, pp. 1–10.

[13] J. Hu, L. Shen, G. Sun, Squeeze-and-excitation networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, in: CVPR '18, IEEE, Washington, DC, 2018, pp. 7132–7141, http://dx.doi.org/10.1109/CVPR.2018.00745.

[14] Y. Qu, B. Fang, W. Zhang, R. Tang, M. Niu, H. Guo, Y. Yu, X. He, Product-based neural networks for user response prediction over multi-field categorical data, ACM Trans. Inf. Syst. 37 (1) (2018) 5:1–5:35, http://dx.doi.org/10.1145/3233770.

[15] X. He, Z. He, J. Song, Z. Liu, Y.-G. Jiang, T.-S. Chua, NAIS: Neural attentive item similarity model for recommendation, IEEE Trans. Knowl. Data Eng. 30 (12) (2018) 2354–2366, http://dx.doi.org/10.1109/TKDE.2018.2831682.

[16] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit., in: CVPR '16, IEEE, Washington, DC, 2016, pp. 770–778, http://dx.doi.org/10.1109/CVPR.2016.90.

[17] R. Salakhutdinov, A. Mnih, G. Hinton, Restricted Boltzmann machines for collaborative filtering, in: Proceedings of the 24th International Conference on Machine Learning, in: ICML '07, ACM, New York, NY, 2007, pp. 791–798, http://dx.doi.org/10.1145/1273496.1273596.

[18] S. Li, J. Kawale, Y. Fu, Deep collaborative filtering via marginalized denoising auto-encoder, in: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, in: CIKM '15, ACM, New York, NY, 2015, pp. 811–820, http://dx.doi.org/10.1145/2806416.2806527.

[19] S. Sedhain, A.K. Menon, S. Sanner, L. Xie, Autorec: Autoencoders meet collaborative filtering, in: Proceedings of the 24th International Conference on World Wide Web, in: WWW '15 Companion, ACM, New York, NY, 2015, pp. 111–112, http://dx.doi.org/10.1145/2740908.2742726.
[20] X. Dong, L. Yu, Z. Wu, Y. Sun, L. Yuan, F. Zhang, A hybrid collaborative filtering model with deep structure for recommender systems, in: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, in: AAAI'17, AAAI Press, 2017, pp. 1309–1315.
[21] M. Jaderberg, K. Simonyan, A. Zisserman, K. Kavukcuoglu, Spatial transformer networks, in: Proceedings of the 28th International Conference on Neural Information Processing Systems - vol. 2, NIPS'15, 2015, pp. 2017–2025.
[22] C. Szegedy, S. Ioffe, V. Vanhoucke, A.A. Alemi, Inception-v4, inception-resnet and the impact of residual connections on learning, in: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, in: AAAI'17, AAAI Press, 2017, pp. 4278–4284.
[23] J. Lian, F. Zhang, X. Xie, G. Sun, Towards better representation learning for personalized news recommendations: A multi-channel deep fusion approach, in: Proceedings of the 27th International Joint Conference on Artificial Intelligence, IJCAI'18, 2018, pp. 3805–3811.
[24] Y. Tay, L. Anh Tuan, S.C. Hui, Latent relational metric learning via memory-based attention for collaborative ranking, in: Proceedings of the 2018 World Wide Web Conference, WWW '18, 2018, pp. 729–739, http://dx.doi.org/10.1145/3178876.3186154.
[25] D. Liu, J. Li, B. Du, J. Chang, R. Gao, Daml: Dual attention mutual learning between ratings and reviews for item recommendation, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 344–352.
[26] C. Wu, F. Wu, M. An, J. Huang, Y. Huang, X. Xie, NPA: neural news recommendation with personalized attention, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 2576–2584.
[27] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, in: CVPR '15, IEEE, Washington, DC, 2015, pp. 1–9, http://dx.doi.org/10.1109/CVPR.2015.7298594.
[28] Y. Hu, Y. Koren, C. Volinsky, Collaborative filtering for implicit feedback datasets, in: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, in: ICDM '08, IEEE, Washington, DC, 2008, pp. 263–272, http://dx.doi.org/10.1109/ICDM.2008.22.
[29] I. Goodfellow, Y. Bengio, A. Courville, Y. Bengio, Deep learning, vol. 1, MIT press Cambridge, 2016.
[30] A.M. Elkahky, Y. Song, X. He, A multi-view deep learning approach for cross domain user modeling in recommendation systems, in: Proceedings of the 24th International Conference on World Wide Web, 2015, pp. 278–288.
[31] X. Glorot, A. Bordes, Y. Bengio, Deep sparse rectifier neural networks, in: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, 2011, pp. 315–323.
[32] D. Kingma, J. Ba, Adam: A method for stochastic optimization, in: Proceedings of the International Conference on Learning Representations, ICLR '14, 2014, pp. 1–15.
[33] W. Zeng, M. Fang, J. Shao, M. Shang, Uncovering the essential links in online commercial networks, Sci. Rep. 6 (2016) 34292, http://dx.doi.org/10.1038/srep34292.
[34] R. He, J. McAuley, Ups and downs: modeling the visual evolution of fashion trends with one-class collaborative filtering, in: Proceedings of the 25th International Conference on World Wide Web, WWW '16, 2016, pp. 507–517, http://dx.doi.org/10.1145/2872427.2883037.
[35] X. He, H. Zhang, M.-Y. Kan, T.-S. Chua, 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, 2016, pp. 549–558.
[36] I. Bayer, X. He, B. Kanagal, S. Rendle, A generic coordinate descent framework for learning from implicit feedback, in: Proceedings of the 26th International Conference on World Wide Web, 2017, pp. 1341–1350.
[37] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Item-based collaborative filtering recommendation algorithms, in: Proceedings of the 10th International Conference on World Wide Web, in: WWW '01, ACM, New York, NY, 2001, pp. 285–295, http://dx.doi.org/10.1145/371920.372071.
[38] S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidt-Thieme, BPR: Bayesian personalized ranking from implicit feedback, in: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, in: UAI '09, AUAI Press, Arlington, Virginia, 2009, pp. 452–461.
[39] R. Salakhutdinov, A. Mnih, Probabilistic matrix factorization, in: Proceedings of the 20th International Conference on Neural Information Processing Systems, NIPS'07, 2007, pp. 1257–1264.
[40] L.v.d. Maaten, G. Hinton, Visualizing data using t-SNE, J. Mach. Learn. Res. 9 (Nov) (2008) 2579–2605.
[41] D. Wang, P. Cui, W. Zhu, Structural deep network embedding, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 1225–1234.
[42] F. Ricci, L. Rokach, B. Shapira, P.B. Kantor, Recommender Systems Handbook, first ed., Springer-Verlag, Berlin, Heidelberg, 2010.
[43] J. Weston, S. Chopra, A. Bordes, Memory networks, in: Proceedings of the International Conference on Learning Representations, ICLR '14, 2015, pp. 1–15.
[44] J. Ye, L. Wang, G. Li, D. Chen, S. Zhe, X. Chu, Z. Xu, Learning compact recurrent neural networks with block-term tensor decomposition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 9378–9387.
[45] A. Novikov, D. Podoprikhin, A. Osokin, D.P. Vetrov, Tensorizing neural networks, in: Advances in Neural Information Processing Systems, 2015, pp. 442–450.