A collaborative filtering model with heterogeneous neural networks for recommender systems

Ge Fan, Wei Zeng, Shan Sun, Weiyi Wang, Weibo Liu

Abstract—In recent years, deep neural network is introduced in recommender systems to solve the collaborative filtering problem, which has achieved immense success on computer vision, speech recognition and natural language processing. On one hand, deep neural network can be used to model the auxiliary information in recommender systems. On the other hand, it is also capable of modeling nonlinear relationships between users and items. One advantage of deep neural network is that the performance of the algorithm can be easily enhanced by augmenting the depth of the neural network. However, two potential problems may emerge when the deep neural work is exploited to model relationships between users and items. The fundamental problem is that the complexity of the algorithm grows significantly with the increment in the depth of the neural network. The second one is that a deeper neural network may undermine the accuracy of the algorithm. In order to alleviate these problems, we propose a hybrid neural network that combines heterogeneous neural networks with different structures. The experimental results on real datasets reveal that our method is superior to the state-of-the-art methods in terms of the item ranking.

Index Terms—collaborative filtering, deep learning, neural networks, matrix factorization.

I. INSTRUCTION

Owing to the rapid growth and prevalence of Internet, people 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 the information overload problem. In order to solve it, recommender systems are widely studied and applied to real systems [1]. Successful application cases include many famous companies such as Netflix, Amazon and YouTube [2], [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].

One of the most important challenges in recommender systems is how to improve the accuracy of the algorithm. The key to solve the problem is to establish an accurate model to describe the relationship between users and items [4]. For instance, the matrix factorization method jointly maps users and items into a latent space and the predictive score is obtained by the inner product of the latent factor vector of user and item [5]. The inner product method is in fact a linear model. A recent study reveals that it is inadequate for the linear method to model the relationships between users and items, because their relationships may be nonlinear [6]. He et al. [6] proposed the neural collaborative filtering framework (NCF) which exploited a multi-layer perceptron (MLP) to learn nonlinear relationships between users and items.

NCF adopts a tower pattern MLP, in which each neural layer has as twice as many nodes than the next neural layer, making the complexity of the algorithm grow exponentially when the depth of the neural network increases. Therefore, the number of the neural layers can not be too large. One potential way to solve this problem is to apply a parallel MLP, in which every neural layer has the same number of nodes. However, a degradation problem may be exposed for such structure. That is, the accuracy of the algorithm does not continually increase with the depth of the neural network rising. Hence, a deeper neural network may jeopardize the accuracy of the algorithm.

In order to solve these problems, we propose a hybrid neural network which consists of the global neural network and the local neural block. The tower pattern MLP is adopted to build the global neural network and a few of stacked neural layers are inserted into two adjacent global layers. These stacked neural layers, termed as the local neural block, leads the total number of neural layers increasing while the complexity of the algorithm does not grow greatly since the scale of the whole neural network does not significantly change. The experimental results on real datasets show that our method outperforms baseline approaches in terms of the item ranking, which indicates that our method models relationships between users and items more precisely than existing methods. Contributions of our work can be summarized as follows:

1) A deeper neural network. We propose a new approach to augment the depth of the tower pattern MLP by inserting neural layers at the local level. With our method, the complexity of the algorithm is not significantly raised and the degradation problem is alleviated to some extent.
2) A better predictive performance. Our proposed hybrid neural network adopts more neural layers than existing methods, which makes the information learnt by our method more accurate. In addition, we employ heterogeneous neural networks with different structures. Therefore, the information learnt by our method is more diverse than that learnt by existing ones.

II. RELATED WORKS

Based on the neural collaborative filtering method, we propose an enhanced approach which incorporates both the
multi-layer perceptron and the deep residual network [7]. Additionally, we combine our method with the traditional matrix factorization. Therefore, our work is related with both the matrix factorization and deep network based approaches. In this section, the related works are reviewed.

A. Matrix factorization based approaches

Matrix factorization (MF) based approaches are becoming increasingly popular in recent years due to its superior scalability and predictive accuracy. It maps both users and items to a joint latent factor space with $k$ dimensions, such that the predictive score is obtained by the inner product between the latent feature vector of user and item. One advantage of the matrix factorization method is its adaptability to integrate other application-specific requirements [5]. For instance, biases of users and items can be added in the matrix factorization method, to indicate the observed deviations of users and items, respectively. The probabilistic matrix factorization is a probabilistic interpretation of the traditional matrix factorization method, which scales linearly with the number of ratings [8]. Furthermore, a Bayesian treatment is applied in the probabilistic matrix factorization which exploits the Markov Chain Monte Carlo (MCMC) to perform approximation inference [9].

In addition, the matrix factorization allows incorporation of auxiliary information such as social information [10], [11], geographical information [13], [14] and time factor [15], [16]. Both the factorization and regularization approach are capable of fusing the heterogeneous social relationships, and the factorization method is better than the regularization method when the social data is binary [11]. Shen et al. [10] exploited the mixture membership stochastic block model to extract the social factor vectors for each user and combined the social model with the latent factor model to provide item recommendations for users. Some online social networks introduce a new feature of "Friends Circles" which allows users to assign classmates, family members, colleagues and others to corresponding groups. The new factors of "Friends Circles" bring opportunities and challenges to recommender systems to solve the cold start and sparsity problem of datasets [12]. In paper [17], social relations are divided into two categories: local relations and global relations. The local relation refers to homophily which indicates that users with similar tastes are more likely to be socially connected. The global social relation reveals the reputation of a user in the whole social network. More recently, both the social matrix factorization and the topic matrix factorization are combined to jointly model three sources, namely ratings, item reviews and social relations [18].

Incorporated in the matrix factorization, the geographical information is another important additional resource to alleviate the sparsity problem of recommender systems. Lian et al. [13] combined the geographical information and the user activity data by the weighted matrix factorization. Liu et al. [19] proposed a general geographical probabilistic factor model which incorporated geographical influences, user mobility behaviors and user check-in count data. A recent study suggests that human mobility is highly regular and predictable [20]. Wang et al. [21] thus combine the regularity and conformity of human mobility as well as their mutual reinforcement to improve the mobility prediction. Moreover, the geographical information and the temporal information are jointly modeled to provide Point-of-Interest recommendation for users [22].

In general, the parameter inference of the matrix factorization is performed by the low-rank approximation which constructs a matrix that approximates the rating matrix at its observed entries. In recent years, a local low-rank approximation method is proposed which firstly divides the rating matrix into several sub-matrices and then each sub-matrix is constructed by a low-rank approximation [23]. Moreover, the squared loss reconstruction is replaced by a rank based loss minimization in this method which makes the method outperform state-of-the-art approaches in terms of the item ranking [24]. In order to capture the temporal behavior of users, the poisson matrix factorization is proposed to recommend the right item to the right user at the right time [25], [26]. The poisson matrix factorization treats time as a natural constituent of the model. Recently, the social network is incorporated in the poisson factorization approach to improve its accuracy.

B. Deep learning based recommendation approaches

The application of deep learning methods to the task of collaborative filtering attracts enormous attention in recent years [6], [27], [28], [29], [30]. Salakhutdinov et al. [27] introduced a class of two-layer undirected graphical models, Restricted Boltzmann Machines (RBM), to model the data of individual ratings. Furthermore, authors linearly combined multiple RBM models with multiple SVD model and the experimental results on the Netflix dataset showed that their model outperformed baseline methods. Truyen et al. [31] explored Boltzmann Machines (BM) to integrate the latent aspects of user’s preference as well as the correlations between users and between items. In the past few years, autoencoder is exploited to solve the task of collaborative filtering [28], [32], [33], [34]. Autoencoder is an unsupervised model which attempts to reconstruct its input data in the output layer. For instance, Li et al. [28] combined the probabilistic matrix factorization with marginalized denoising stacked autoencoders to learn latent representation of users and items.

Since the rating data is normally very sparse, quite a number of recommender systems suffer from the sparsity problem [35], [56]. In order to alleviate the problem, certain deep learning models are thus employed to learn latent factors from side information such as raw features of audio, text information of articles and reviews of items [37], [38]. Wang et al. [37] made use of the stacked denoising autoencoder to learn latent factor from the text information which is extracted from titles and abstracts of articles. Strub et al. [33] exploited the stacked denoising autoencoder to learn a nonlinear representation of users and items and to alleviate the cold start problem by integrating tags of items. Some other side information such as the user’s age, gender and occupation can also be modeled by the stacked denoising autoencoder [34].

The Convolutional Neural Network (CNN) is another important way to model the side information in recommender systems [38], [39]. Kim et al. [39] integrated convolutional
neural network into probabilistic matrix factorization to capture contextual information of documents. Zheng et al. [38] proposed a deep model to learn item properties and user behaviors jointly from review text. The model adopted two convolutional neural networks which are coupled in the last layers. Wang et al. [40] proposed a Point-of-Interest recommender system with enhanced visual content, which adopted convolutional neural network to extract image features. He et al. [41] incorporated visual features into the Bayesian Personalized Ranking approach for the task of personalized ranking on implicit feedback datasets. He et al. [42] further extended their method by combining high-level visual features extracted from a deep convolutional neural network, users’ historical feedbacks, as well as evolving trends within the community.

In order to model the temporal dynamics of ratings and sequential features of datasets, the Recurrent Neural Network (RNN) is introduced in recommender systems [43], [44], [45], [46]. Hidasi et al. [43] made use of Gated Recurrent Unit (GRU) for session-based recommendation. The GRU is a more elaborate model of an RNN unit aiming at dealing with the vanishing gradient problem. The input of the network is the actual state of the session and the output is the item of the next event in the session. Moreover, the performance of the basic RNN model is improved by using proper data augmentation techniques and by accounting for temporal shifts in user behavior [44]. Wu et al. [45] built a deep recurrent neural network to track how users browse the website by using multiple hidden layers. Each hidden layer models how the combinations of webpages are accessed and in what order. Wu et al. [46] made use of recurrent neural network to capture temporal dependencies for both users and movies, and incorporated past observations to predict future trajectories in an integrated manner.

In most cases, recommendation is considered to be a two-way interaction between the user’s preferences and the item’s feature. For instance, the matrix factorization maps users and items into a joint latent space and computes 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. He et al. [6] proposed the Neural Collaborative Filtering (NCF) method which utilized a multi-layer perceptron (MLP) to capture the nonlinear relationships between users and items. Additionally, Wang et al. [47] extended the NCF model to cross-domain social recommendations, which links the user-item interactions and the user-user connections in Social Networking Services (SNSs). Lian et al. [48] incorporated the content information into the neural collaborative filtering approach.

C. Limitations of related works

Although the application of matrix factorization in the recommender system is very successful, it fails to capture the nonlinear relationships between users and items. Plenty of deep learning based approaches are proposed to the task of collaborative filtering, but few of them are designed to directly model the interactions between users and items. The neural collaborative filtering is proposed to model the nonlinear relationships between users and items, method of which utilizes a multi-layer perceptron with a tower pattern [6]. However, each neural layer has nodes as twice as the next neural layer in the NCF approach, which leads to a poor efficiency when the number of neural layers is significantly large.

Therefore, given the limitations of related works, we design a hybrid neural network which consists of the global neural network and the local neural block. The experimental results on real datasets show that our method outperforms the NCF method, indicating that our method models the nonlinear relationships better than the NCF method.

III. PRELIMINARIES

In general, there are two types of datasets in recommender systems, i.e. explicit ratings and implicit ratings. Lots of recommendation approaches being proposed so far 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’ clickings and interaction records [49].

Given the implicit feedback of users, a recommender system can be naturally represented by an adjacent matrix, \( Y_{M \times N} \), where the element \( y_{ui} = 1 \) if user \( u \) has an interaction with item \( i \) and \( y_{ui} = 0 \) if the interaction between \( u \) and \( i \) is unobserved. \( M \) and \( N \) represent the number of users and items, respectively.

A. Matrix factorization

With the adjacent matrix, one can make use of observed elements to train a model and predict those missing values in the matrix. The matrix factorization approach is such technique that divides \( Y \) into the user latent factor matrix \( P \) and the item latent factor matrix \( Q \). The predictive score \( \hat{y}_{ui} \) between user \( u \) and item \( i \) is based on the inner product of the user latent factor vector \( p_u \) and the item latent factor vector \( q_i \):

\[
\hat{y}_{ui} = f(u, i; p_u, q_i) = p_u^T q_i = \sum_{k=1}^{K} p_{uk} q_{ik},
\]

where \( K \) is the dimension of \( p_u \) and \( q_i \). As shown in equation [1] the matrix factorization computes the predictive score in a linear way, which indicates that it might be difficult for this method to capture nonlinear relationships between users and items. Thus, He et al. [6] employed deep neural networks to model the nonlinear relationships between users and items.

B. Neural collaborative filtering

The NCF approach adopts a multi-layer perceptron to model a user-item interaction \( y_{uis} \), where the output of one layer serves as the input of the next layer. The bottom input layer consists of two feature vectors \( v_u \) and \( o_i \), which describe user \( u \) and item \( i \), respectively. Above the input layer, specifically the embedding layer, is a fully connected layer that projects the sparse representation to a dense vector. The obtained user (item) embedding can be seen as the latent vector for users.
(items) in the context of the latent factor model. The user embedding and item embedding are then fed into a multi-layer perceptron to map the latent vector to prediction scores. Each layer of the neural layers can be customized to discover certain latent structures of user-item interactions. The dimension of the last hidden layer $X$ determines the model’s capability. The final output layer 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 structure of the NCF method is illustrated in figure [1].

Fig. 1. The architecture of NCF.

The NCF’s predictive model can be formulated as

$$\hat{y}_{ui} = f(P^T v_u, Q^T o_i | P, Q, \Theta_f),$$  

(2)

where $P \in \mathbb{R}^{M \times K}$ and $Q \in \mathbb{R}^{N \times K}$ represent the latent factor matrix for users and items, respectively. $\Theta_f$ denotes the model parameters of the interaction function $f$, defined as a multi-layer neural network and formulated as

$$f(P^T v_u, Q^T o_i) = \phi_{out}(\phi_X (\ldots \phi_1 (P^T v_u, Q^T o_i)) \ldots),$$  

(3)

where $\phi_{out}$ and $\phi_x$ respectively denote the mapping function for the output layer and $x$-th neural layer, and there are $X$ neural layers in total.

IV. OUR PROPOSED MODEL

In this section, we mainly discuss our method, the Hybrid Neural Collaborative Filtering (HNCF for short). The NCF approach exploits MLP to model the nonlinear relationships between users and items while our method adopts a hybrid deep neural network to learn the nonlinear relationships.

A. Hybrid neural collaborative filtering

As mentioned above, NCF utilizes a multi-layer perceptron with a tower pattern, making the algorithm’s complexity grow exponentially with the increment of the depth of the neural network. If every MLP layer has the same number of nodes, a degradation problem might be exposed (more details see Experiments section). In order to solve the problem, we exploit a hybrid deep neural network to capture nonlinear relationships between users and items. Our method has the same input layer, embedding layer and output layer with the NCF method, but the neural CF layer of NCF is replaced by a hybrid neural CF layer. The hybrid neural CF layer has two parts: the global neural network and the local neural block. We choose the tower pattern MLP to build the global neural network and insert a few of stacked neural layers into two adjacent global layers. Those stacked neural layers are termed as the local neural block. In this paper, three neural structures are adopted to build the local neural block: (1) parallel MLP; (2) tower pattern MLP; (3) deep residual neural network [7]. The architecture of our method is demonstrated in figure [2].

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

The formulation of our method is given as follow:

$$\hat{y}_{ui} = f(P^T v_u, Q^T o_i | P, Q, \Theta_f^G, \Theta_f^L),$$  

(4)

where $\Theta_f^G$ and $\Theta_f^L$ denote parameters of the global neural network and the local neural block. The function $f$ can be formulated as:

$$f(P^T v_u, Q^T o_i) = \phi_{out}(\Phi^L_X (\phi^G_X (\ldots \phi^G_1 (P^T v_u, Q^T o_i)) \ldots)),$$  

(5)

where $\phi_{out}$ is the mapping function for the output layer. $\phi^G_X$ denotes the mapping function for the $x$-th global neural layer. $\Phi^L_x$ denotes a series of mapping functions in the $x$-th local neural block, which can be formulated as:

$$\Phi^L_x(y) = \phi^L_{x,z} (\ldots \phi^L_{x,2}(\phi^L_{x,1}(y)) \ldots),$$  

(6)

where $\phi^L_{x,z}$ represents the mapping function for the $z$-th layer in the $x$-th local neural block, and there are $Z$ layers in total in the block.

In order to learn parameters of the model, existing pointwise methods normally perform a regression with squared loss:

$$L_{sqr} = \sum_{(u,i) \in \mathcal{Y} \cup \mathcal{Y}^-} w_{ui}(y_{ui} - \hat{y}_{ui})^2,$$  

(7)

where $\mathcal{Y}$ denotes the set of observed entries in the adjacent matrix $Y$, and $\mathcal{Y}^-$ denotes the set of negative instances, which can be all (or sampled from) missing values. $w_{ui}$ is a hyperparameter which represents the weight of training instance $(u, i)$. 

$$\end{document}$$
Since our collected data is implicit feedback, we view the value of $y_{ui}$ as a label: 1 indicates that user $u$ has chosen item $i$ and 0 otherwise. Then the prediction score $\hat{y}_{ui}$ represents the probability that $u$ will choose $i$. To endow our method with such a probabilistic explanation, we constrain the output $\hat{y}_{ui}$ in the range of $[0,1]$, which can be easily achieved by utilizing a probabilistic function (e.g., the Logistic or Probit function) as the activation function for the output layer $\phi_{out}$. With the above settings, we then define the likelihood function as

$$p(Y|P, Q, \Theta^G, \Theta^L) = \prod_{(u,i) \in Y} \hat{y}_{ui} \prod_{(u,i) \in Y^c} (1 - \hat{y}_{ui}).$$

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

$$L = - \sum_{(u,i) \in Y} \log \hat{y}_{ui} - \sum_{(u,i) \in Y^c} \log(1 - \hat{y}_{ui})$$

$$= - \sum_{(u,i) \in Y \cup Y^c} \hat{y}_{ui} \log \hat{y}_{ui} + (1 - \hat{y}_{ui}) \log(1 - \hat{y}_{ui}).$$

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

B. Local neural block

The local neural block is inserted into two adjacent global layers. We adopt three neural structures to build the local neural block: (1) parallel MLP; (2) tower pattern MLP; (3) deep residual neural network.

1) Parallel MLP: In this structure, every neural layer has the same number of nodes and the nodes in one layer are fully connected with the nodes in the next layer one by one. The corresponding architecture is demonstrated in figure 5.

The formulation of this architecture is defined as:

$$z^G_1 = \phi^G_1(p_u, q_i) = [p_u],$$

$$z^L_1 = \phi^L_{1,1}(z^G_1) = a^L_{1,1}(W^{L}_{1,1}z^G_1 + b^L_{1,1}),$$

$$z^L_{2} = \phi^L_{1,2}(z^L_1) = a^L_{1,2}(W^{L}_{1,2}z^L_1 + b^L_{1,2}),$$

$$\ldots$$

$$z^L_{z,1} = \phi^L_{1,z}(z^L_{z-1}) = a^L_{1,z}(W^{L}_{1,z}z^L_{z-1} + b^L_{1,z}).$$

$$z^L_2 = \phi^G_2(z^L_1, x) = a^G_2(W^{G}_{2}z^L_1 + b_2),$$

$$z^L_{2,1} = \phi^L_{2,1}(z^L_2) = a^L_{2,1}(W^{L}_{2,1}z^L_2 + b^L_{2,1}),$$

$$\ldots$$

$$z^L_{X,Z} = \phi^L_{X,Z}(z^L_{X,Z-1}) = a^L_{X,Z}(W^{L}_{X,Z}z^L_{X,Z-1} + b^L_{X,Z}),$$

where $W_x, b_x$ and $a^G_2$ denote the weight matrix, the bias vector and the activation function for the $x$-th global layer, respectively. $W^L_{x,z}, b^L_{x,z}$ and $a^L_{x,z}$ denote the weight matrix, the bias vector and the activation function for the $x$-th local neural block. The size of $W_x$ changes with the layer’s position in the global neural network shifting. The row number of $W_x$ is $D \cdot 2^{X-x+1}$ and the column number of $W_x$ is $D \cdot 2^{X-x}$, where $D$ is the length of the last global layer $X$. The size of $W^L_{x,z}$ is closely related to the size of its previous global neural network. Since every neural layer has the same number of nodes for this architecture, both the row number and the column number of $W^L_{x,z}$ are $D \cdot 2^{X-x}$.

2) Tower pattern MLP: This neural network has the same structure with the global neural network, in which the $n$-th neural layer has the number of nodes as twice as the $(n-1)$-th neural layer. Accordingly, the number of layers in the network should not be too large, neither. The architecture is illustrated in figure 4 and the formulation is defined as:

$$z^G_1 = \phi^G_1(p_u, q_i) = [p_u],$$

$$z^L_{1,1} = \phi^L_{1,1}(z^G_1) = a^L_{1,1}(W^{L}_{1,1}z^G_1 + b^L_{1,1}),$$

$$z^L_{1,2} = \phi^L_{1,2}(z^L_{1,1}) = a^L_{1,2}(W^{L}_{1,2}z^L_{1,1} + b^L_{1,2}),$$

$$\ldots$$

$$z^L_{1,z} = \phi^L_{1,z}(z^L_{1,z-1}) = a^L_{1,z}(W^{L}_{1,z}z^L_{1,z-1} + b^L_{1,z}),$$

$$z^L_2 = \phi^G_2(z^L_{1,z}, x) = a^G_2(W^{G}_{2}z^L_{1,z} + b_2),$$

$$z^L_{2,1} = \phi^L_{2,1}(z^L_2) = a^L_{2,1}(W^{L}_{2,1}z^L_2 + b^L_{2,1}),$$

$$\ldots$$

$$z^L_{X,Z} = \phi^L_{X,Z}(z^L_{X,Z-1}) = a^L_{X,Z}(W^{L}_{X,Z}z^L_{X,Z-1} + b^L_{X,Z}),$$

$$\hat{y}_{ui} = \phi_{out}(z^L_{X,Z}) = a_{out}(h^T z^L_{X,Z} + b),$$

where $W_x, b_x$ and $a^G_2$ denote the weight matrix, the bias vector and the activation function for the $x$-th global layer, respectively. $W^L_{x,z}, b^L_{x,z}$ and $a^L_{x,z}$ denote the weight matrix, the bias vector and the activation function for the $x$-th local neural block. The size of $W_x$ changes with the layer’s position in the global neural network shifting. The row number of $W_x$ is $D \cdot 2^{X-x+1}$ and the column number of $W_x$ is $D \cdot 2^{X-x}$, where $D$ is the length of the last global layer $X$. The size of $W^L_{x,z}$ is closely related to the size of its previous global neural network. Since every neural layer has the same number of nodes for this architecture, both the row number and the column number of $W^L_{x,z}$ are $D \cdot 2^{X-x}$. 
network adds shortcut connections among disconnected layers, and outputs of those shortcut connections are combined with outputs of a few stacked layers. Suppose $H(x)$ is an underlying mapping to be fit by a few stacked layers, where $x$ denotes the inputs to the first of these layers. A shortcut connection is then added. The head of the shortcut connection is at the beginning of the first layer and the tail of the shortcut connection is at the end of the last layer. Therefore, these stacked layers approximate another residual function $F(x) := H(x) - x$. A demonstration of the residual neural network is presented in figure 5. With the residual neural network, our model is defined as:

$$z_i^G = \phi_i^G(p_{ui}, q_i) = \begin{bmatrix} p_u \\ q_i \end{bmatrix},$$

$$z_{l+1}^L = \phi_{l+1}^L(z_l^G) = a_{l+1}^L(W_{l+1}^Tz_l^G + b_{l+1}^L),$$

$$z_{l+1}^L = \phi_{l+1}^L(z_l^G) = a_{l+1}^L(W_{l+1}^Tz_l^G + b_{l+1}^L),$$

$$z_1^G = \phi_1^G(z_0) = a_1^L(W_0^Tz_0 + b_0),$$

$$z_{l+1}^G = \phi_{l+1}^G(z_l) = a_{l+1}^L(W_{l+1}^Tz_l + b_{l+1}),$$

$$z_{l+1}^G = \phi_{l+1}^G(z_l) = a_{l+1}^L(W_{l+1}^Tz_l + b_{l+1}),$$

$$z^G_{X,Z} = \phi_{X, Z}^G(z_{X,Z-1}) = a_{X,Z}^L(W_{X,Z}^Tz_{X,Z-1} + b_{X,Z}),$$

$$z^G_{X,Z} = \phi_{X, Z}^G(z_{X,Z-1}) = a_{X,Z}^L(W_{X,Z}^Tz_{X,Z-1} + b_{X,Z}),$$

$$\hat{y}_{ui} = \phi_{out}(z_X) = a_{out}(h^Tz_X + b).$$

We also 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 figure 6.

![Fig. 5. The architecture of the residual network.](image1)

For the activation function of each neural layer, we choose Rectifier which yields a better performance than tanh and sigmoid [6]. The sigmoid function $\sigma(x) = 1/(1 + e^{-x})$ is chosen as the activation function for the output layer $\phi_{out}$ to map the prediction score $\hat{y}_{ui}$ in the range of $[0, 1]$. The pseudo code of our method is presented in Algorithm 1.

**C. 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. Therefore, we also combine our method with GMF. In a similar manner, we combine GMF and our method by concatenating their last hidden layer. Figure 7 illustrates this proposal and the formulation is given as follows:

$$\phi_{GMF}^{\text{Hybrid}} = \begin{bmatrix} p_u^{GMF} \\ q_i^{GMF} \end{bmatrix},$$

$$z_1 = \begin{bmatrix} p_u^{\text{Hybrid}} \\ q_i^{\text{Hybrid}} \end{bmatrix},$$

$$z_{l+1}^L = a_{l+1}^L(W_{l+1}^Tz_l^G + b_{l+1}),$$

$$z_{l+1}^L = a_{l+1}^L(W_{l+1}^Tz_l^G + b_{l+1}),$$

$$z_{l+1}^G = \phi_{l+1}^G(z_l) = a_{l+1}^L(W_{l+1}^Tz_l + b_{l+1}),$$

$$z_{l+1}^G = \phi_{l+1}^G(z_l) = a_{l+1}^L(W_{l+1}^Tz_l + b_{l+1}),$$

$$z^G_{X,Z} = \phi_{X, Z}^G(z_{X,Z-1}) = a_{X,Z}^L(W_{X,Z}^Tz_{X,Z-1} + b_{X,Z}),$$

$$z^G_{X,Z} = \phi_{X, Z}^G(z_{X,Z-1}) = a_{X,Z}^L(W_{X,Z}^Tz_{X,Z-1} + b_{X,Z}),$$

$$\hat{y}_{ui} = a_{out}(h^Tz_X + b).$$

In order to save the space, we use a matrix $W$ to distinguish MLP and residual neural network in the local neural block in equation 13. Taking $z_1 = Wz_0 + z_{l,Z}^G$ for instance, when $W = 0$, the formulation represents MLP and when $W = I$, the formulation denotes the residual neural network.

![Fig. 7. The combination of the hybrid neural network and GMF.](image2)

Similar to NCF method, we also pre-train GMF and hybrid neural network with random initializations until convergence and then use their parameters as the initialization for the corresponding parts of the hybrid model’s parameters. We concatenate these two models on the output layer as follow:

$$h \leftarrow \frac{\alpha \hat{h}^{\text{GMF}}}{(1 - \alpha) \hat{h}^{\text{Hybrid}}},$$

(14)
where $h^{\text{GMF}}$ and $h^{\text{Hybrid}}$ denote the $h$ vector of the pre-trained GMF and the hybrid neural network, respectively. $\alpha$ is a hyper-parameter which determines the trade-off between these two pre-trained models.

The Adaptive Moment Estimation (Adam) is adopted to learn parameters in models, computing individual adaptive learning rates for various parameters from estimation of the first and the second moments of the gradients [50]. 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].

**ALGORITHM 1:** Hybrid neural collaborative filtering.

**Input:**
- \textit{Iter:} training iterations.
- \textit{neg:} the number of negative samples.
- $Y$: the adjacent matrix.

**Output:**
- $P$: latent matrix for user;
- $Q$: latent matrix for item;
- $\Theta^G$: parameters of the global neural network;
- $\Theta^L$: parameters of the local neural network;
- $\gamma$: observed interaction set;
- $\gamma^-$: unobserved interaction set;
- for $i$ from 1 to \textit{Iter} do

\begin{itemize}
  \item set $\gamma^\text{sampled}$ ← sampling \textit{neg} $| Y |$ unobserved interactions from $\gamma^-$;
  \item set $T$ ← from $Y \cup \gamma^\text{sampled}$:
  \item for each interaction of user $u$ and item $i$ in $T$ do
    \begin{itemize}
      \item if $x = \text{1}$ then
        \begin{itemize}
          \item set $z^G_u$ ← with the input of $v_u$, $o_i$;
          \item set $z^G_u$ ← with the input of $z^L_{u-1}, z^G_{u-1}$;
        \end{itemize}
      \end{itemize}
      \begin{itemize}
        \item if $z = \text{1}$ then
          \begin{itemize}
            \item set $z^L_{i,1}$ ← with the input of $z^G_i$;
            \item set $z^L_{i,z}$ ← with the input of $z^L_{i,z-1}$;
          \end{itemize}
        \end{itemize}
    \end{itemize}
  \end{itemize}

\begin{itemize}
  \item set $\hat{y}_{ui}$ ← use the input $z^L_{ui}$;
  \item set $L$ ← use Equation 2 with input of $\hat{y}_{ui}$, $y_{ui}$;
  \item use back propagation to optimize model parameters.
\end{itemize}

\end{itemize}

**D. Summary of our methods**

To learn nonlinear relationships between users and items, we employ a hybrid neural network, which consists of the global neural network and the local neural block. Three architectures are used to build the local neural block. Moreover, we combine the hybrid neural network with the GMF model. To give a clear explanation of our method, abbreviations and descriptions of our methods are presented in table I.

**V. EXPERIMENTS**

**A. Experimental setup**

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 users’ ratings to generate personalized recommendations. The MovieLens-100K dataset consists of 943 users and 1682 movies and the MovieLens-1M dataset has 6000 users and 4000 movies. Douban, launched on March 6, 2005, is a Chinese Web 2.0 web site which provides user reviews and recommendation services of movies, books, and music [51]. 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 [42]. 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 implicit feedbacks. 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 II.

\begin{itemize}
  \item [2] Metrics: In order to measure the performance of recommendation methods, the \textit{leave-one-out} evaluation is adopted. For each user, her latest interaction is held-out as the test set and the remaining interactions are used as the training set. Since it is considerably time-consuming to rank all items for every user during evaluation, we randomly sample 100
\end{itemize}
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 as metrics to measure the ranked list of recommendation approaches. Without special mention, 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: \( NDCG_u = \frac{1}{M} \sum_{u=1}^{M} \frac{2^{-r_j}}{\log(j+1)} \), where \( Z \) 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. \( L \) is the length of the ranked list. In a similar manner, we average \( NDCG_u \) over all users: \( NDCG = \frac{1}{M} \sum_{u=1}^{M} NDCG_u \).

3) Baseline methods: We compare our method with the following approaches:

1) ItemPop. 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 [52]. 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 [53]. The bayesian personalized ranking (BPR) is a generic optimization method for personalized ranking. We apply this generic method to matrix factorization model by utilizing implicit feedbacks of users.

4) SoloMLP [6]. He et al. proposed the neural collaborative filtering framework that exploited a tower pattern MLP to model the nonlinear relationships between users and items. Furthermore, authors combined this method with the matrix factorization model to improve the accuracy. The combined method is termed as NeuMF. Both SoloMLP and NeuMF are compared with our method.

4) Parameter settings: Without special mention, we randomly initialize model parameters (including baseline methods) with a Gaussian distribution (with a mean of 0 and a standard deviation of 0.01) [8]. 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 \( \alpha \) is set to 0.5.

B. Results and analysis

1) Possible limitations of SoloMLP: Two possible problems may emerge as neural CF layers (as shown in figure [1]) are built by solo MLP. If neural CF layers are built by MLP with a tower pattern, the complexity of the algorithm grows exponentially with the augmentation of the depth of the neural network, thus constraining the number of neural layers to be within certain range. In figure [8] we present the training time measured by seconds as a tower pattern MLP is taken into account. It can be seen that the algorithm’s training time grows exponentially for all datasets. One potential way to raise the depth of the neural network is to choose the parallel MLP, in which every neural layer has the same number of nodes. However, a degradation problem may be exposed for this structure, i.e the accuracy of the algorithm declines as the number of the neural layers increases. The relationship between the algorithm’s accuracy and the number of the neural layers is given in table [III]. From the table, it can be seen that the accuracy of the algorithm reaches a peak with the number of neural layers rising to certain extent. Taking the MovieLens-100K dataset for instance, the algorithm achieves the best \( HR@10 \) and \( NDCG@10 \) when the depth of neural network is 15. The accuracy of the algorithm drops when the number of neural layers increases to 30.

![Fig. 8. Training time of SoloMLP with a tower pattern.](image)

| TABLE III |
| --- |
| THE RELATIONSHIP BETWEEN THE ACCURACY OF PARALLEL MLP AND THE NUMBER OF NEURAL LAYERS. BOLD VALUES INDICATE THE BEST RESULTS. |
| | 3 layers | 6 layers | 9 layers | 15 layers | 30 layers |
| | HR@10 |
| MovieLens-100K | 0.6702 | 0.6617 | 0.6617 | 0.6702 | 0.6638 |
| MovieLens-1M | 0.6856 | 0.6854 | 0.6815 | 0.6858 | 0.6825 |
| Douban Book | 0.6611 | 0.6567 | 0.6548 | 0.6539 | 0.6449 |
| Amazon Movies | 0.6512 | 0.6521 | 0.6633 | 0.6566 | 0.6471 |
| Amazon Games | 0.4765 | 0.4661 | 0.452 | 0.4594 | 0.4569 |
| MovieLens-100K | 0.3694 | 0.3805 | 0.3847 | 0.3944 | 0.3774 |
| MovieLens-1M | 0.4082 | 0.4079 | 0.4104 | 0.4123 | 0.4104 |
| Douban Book | 0.4395 | 0.4347 | 0.4301 | 0.4293 | 0.4189 |
| Amazon Movies | 0.4275 | 0.4255 | 0.4342 | 0.4333 | 0.4247 |
| Amazon Games | 0.2711 | 0.2641 | 0.2583 | 0.2605 | 0.2672 |

2) The performance of our method: We compare our proposed methods with baseline approaches on several datasets and results are presented in table [IV]. From the table, three conclusions can be obtained:

1) Without considering the combination with the GMF, our proposals which exploit the hybrid neural network (\( HybridNN_{MLP-tower} \) and \( HybridNN_{NMF} \)) are generally superior to SoloMLP. This result indicates that these two approaches learn relationships between users and items more precisely than SoloMLP. One reason may be that these two methods adopt a deeper neural network than SoloMLP. Supposing the original SoloMLP has 6 neural layers, yet the depth of the neural network is increased to 18 by our method (the depth of the local neural block is set to 2). For \( HybridNN_{MLP-same} \), it underperforms the SoloMLP on MovieLens-1M and Amazon Games datasets (\( HR@10 \)). One possible reason
may be that the degradation problem is exposed on these two datasets.

2) When the hybrid neural network is combined with GMF (HybridNNMF\textsubscript{MLP−same}, HybridNNMF\textsubscript{MLP−tower} and HybridNNMF\textsubscript{res}), the accuracy of algorithm is further improved. This result implies that both linear and nonlinear relationships may exist simultaneously between users and items. It may be inadequate only to consider the linear (or nonlinear) relationship to build a predictive model.

3) From table IV it can be seen that HybridNNMF\textsubscript{res} enjoys the best predictive accuracy on all datasets, the method of which exploits both the tower patter MLP and the residual neural network to build the hybrid neural network. The residual neural network links disconnected neural layers by shortcut connections. He et al. [7] proposed the residual neural network to solve the degradation problem in the field of image processing. We apply this method in collaborative filtering and achieve a better result than baseline methods, which indicates that the residual neural network may be better than MLP in modeling relationships between users and items.

Table V only shows the comparative results when the length of the recommendation list is 10. We give the performance of algorithms in figure 9 when the length of the recommendation list ranges from 1 to 10. To make the figure clearer, we present the comparative result of HybridNNMF\textsubscript{res} rather than all hybrid neural network methods. From the figure, it can be seen that the accuracy (HR and NDCG) of HybridNNMF\textsubscript{res} and NeuMF is better than the accuracy of ItemKNN and BPR since the former two methods take both linear and nonlinear relationships into account.

In general, HybridNNMF\textsubscript{res} outperforms NeuMF while the gap between these two methods is not significant on some datasets (e.g. MovieLens-100K). In order to analyze the differences between HybridNNMF\textsubscript{res} and baselines approaches, we employ the sign test which requires few assumptions about the distributional form of the data [54]. In the sign test, we count the number of users for whom our proposed algorithm outperforms the baseline algorithm ($n_A$) and the number of users for whom the baseline algorithm outperforms our algorithm ($n_B$). The significance level is the probability that our method is not truly better than the baseline method, and is estimated as the probability of at least $n_A$ out of $n = n_A + n_B$ 0.5-probability Binomial trials succeeding, and is given by $p = (0.5)^n \sum_{i=n_A}^{n} \binom{n}{i}$. When $n$ is large, we can take advantage of Normal distribution to approximate the Binomial. We then compute $z^*$ score by $z^* = \frac{n_A - 0.5n}{\sqrt{n}/4}$. If $|z^*| < 1.96$, 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 outperforms the baseline method.

The result of $n_A/(n_A + n_B) \times 100\%$ and $z^*$ scores are given in table V. From the table, it can be seen that more users’ HR@10 have been improved by HybridNNMF\textsubscript{res} comparing to the result of NDCG@10. The NDCG metric not only considers whether the test item is in the recommendation list or not, but also measures the position of the test item. Since HybridNNMF\textsubscript{res} is an extension of NeuMF, positions of recommended items may not be changed significantly. However, from the result of HR@10, we can see that more test items are presented in the top-$K$ item list of individual users. From the result of $z^*$ scores in table V, it is shown that the differences between HybridNNMF\textsubscript{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\textsubscript{res} and NeuMF is not significant ($|z^*| < 1.96$) on MovieLens-100K dataset. As mention before, the improvements of test items’ positions are limited since HybridNNMF\textsubscript{res} and NeuMF are homogeneous.

3) The depth of the hybrid neural network: In [6], authors discussed the influence of the depth of MLP on the algorithm’s accuracy. Their results indicated that the algorithm’s accuracy rose with the growth of the number of neural layers. From a theoretical point of view, the result provides a potential way to improve the accuracy of the algorithm. In a similar manner, 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 VIII From the table, it can be seen that the accuracy of the HybridNN\textsubscript{MLP−same} does not increase as the number of neural layers reaches a certain level. This result may be caused by the degradation problem when we choose the parallel MLP to build the local neural block which is similar to the result of table III.

For HybridNN\textsubscript{MLP−tower} and HybridNN\textsubscript{res}, their accuracy generally increases with the augmentation of the number of neural layers. It is worth mentioning that the depth of our hybrid neural network is 2 times bigger than the depth of the SoloMLP. Therefore, if one expects to enhance the accuracy of the algorithm by adding more neural layers, a potential way is to insert neural layers at the local level. It is too costly to add neural layers at the global level for the tower pattern neural network.

4) The number of negative samples: When the pairwise object function in equation 9 is trained, a commonly used way is to randomly sample a certain number of negative instances for each positive instance. Table VII shows the performance of HybridNNMF\textsubscript{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. Therefore, we only present the result when the ratio are 1, 4, 5, 6 and 10. As a matter of fact, the algorithm achieves a considerable performance when the sample ratio equals to 1. This result demonstrates the advantages of pointwise log loss in terms of the item ranking. When the sample ratio is augmented, the accuracy of algorithm can be further improved. However, when the sample ratio is greater than 6, the performance of HybridNNMF\textsubscript{res} starts to drop. In addition, the performance of the algorithm stays relatively stable when the sample ratio takes different values.

5) The dimension of latent factors: Table IV gives the performance of algorithms when dimensions of their latent vectors are optimal. In figure 10 we present the performance of algorithms with different dimensions of latent vectors.
To make the figure clearer, we only present HybridNNMFRres rather than all hybrid neural network methods. We only show the performance of NeuMF with three layers and HybridNNMFRres(HybridNNMFRres) with nine layers. The figure illustrates that HybridNNMFRres outperforms the remaining approaches when the latent vector's dimension ranges from 16 to 128. One possible reason may be that HybridNNMFRres adopts more neural layers. Therefore, the information learnt by HybridNNMFRres is more accurate than the information learnt by baseline methods. Another possible reason may be that HybridNNMFRres employs heterogeneous neural networks with varied structures, which makes the information learnt by this method more diverse comparing to existing ones.

For HybridNNMFRres, the optimal dimension of latent vector arranges from 32 to 64. If the dimension is too small, the information expressed by the latent vector may be insufficient. There may be redundant information if the dimension is set too large, which may hurt the accuracy of the algorithm. Moreover, figure 10 illustrates that HybridNNMFRres has unstable performance on different datasets. For instance, HybridNNMFRres outperforms BPR on MovieLens-1M and Douban Book datasets, while HybridNNMFRres underperforms BPR on Amazon Games dataset. This result indicates that relationships between users and items are not all nonlinear. For instance, some users may only be fond of comedy movies. Therefore, there is a strong linear correlation between

### TABLE IV

| Model       | 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  | 0.24    | 0.453  | 0.254    | 0.524  | 0.326    | 0.244  | 0.13     | 0.353  | 0.196    |
| ItemKNN     | 0.643  | 0.364   | 0.668  | 0.398    | 0.617  | 0.454    | 0.433  | 0.339    | 0.502  | 0.304    |
| BPR         | 0.679  | 0.382   | 0.674  | 0.399    | 0.664  | 0.445    | 0.695  | 0.457    | 0.511  | 0.302    |
| SoloMLP     | 0.681  | 0.396   | 0.704  | 0.422    | 0.678  | 0.458    | 0.674  | 0.441    | 0.504  | 0.288    |
| NeuMF       | 0.705  | 0.410   | 0.729  | 0.449    | 0.702  | 0.475    | 0.703  | 0.471    | 0.530  | 0.327    |

### TABLE V

| Model       | MovieLens-100K | MovieLens-1M | Douban Book | Amazon Movies | Amazon Games |
|-------------|----------------|--------------|-------------|---------------|--------------|
|             | HR@K | NDCG@K | HR@K | NDCG@K | HR@K | NDCG@K | HR@K | NDCG@K | HR@K | NDCG@K |
| ItemKNN     | 0.688  | 0.394   | 0.700  | 0.425    | 0.681  | 0.442    | 0.687  | 0.461    | 0.495  | 0.283    |
| BPR         | 0.684  | 0.397   | 0.721  | 0.435    | 0.704  | 0.471    | 0.701  | 0.469    | 0.503  | 0.290    |
| NeuMF       | 0.698  | 0.401   | 0.719  | 0.437    | 0.694  | 0.465    | 0.700  | 0.466    | 0.515  | 0.310    |
| HybridNNMFRres_same | 0.698 | 0.402   | 0.724  | 0.445    | 0.706  | 0.46    | 0.715  | 0.481    | 0.543  | 0.320    |
| HybridNNMFRres_tower | 0.707 | 0.411   | 0.737  | 0.453    | 0.712  | 0.487    | 0.713  | 0.478    | 0.547  | 0.334    |
| HybridNNMFRres | 0.726  | 0.420   | 0.738  | 0.456    | 0.713  | 0.487    | 0.719  | 0.483    | 0.562  | 0.343    |

Fig. 9. Evaluation of Top-K item recommendation where K ranges from 1 to 10.
these users and comedy movies, which enables the linear method to be superior to the nonlinear method. In general, HybridNNMFres holds the best predictive accuracy because it incorporates both linear and nonlinear relationships between users and items.

6) The utility of Pre-training: There are varied ways to initialize model parameters. A commonly used approach is random initialization and another one is to utilize a pre-trained model. We compare these two initialization methods and the result is given in table VIII. We make use of parameters which are pre-trained by GMF and the hybrid neural network to initialize the latent vector in the embedding layer (as shown in figure 7). This method is termed as with pre-training. The latent vector in the embedding layer 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 achieves a better performance than the algorithm without pre-training. The possible reason may be that the pre-training process preserves the particularity of GMF and the hybrid neural network, and the particularity is beneficial to enhance the model’s accuracy.

7) The complexity analysis: Since our methods adopt more neural layers, the training time of our method is more than the training time of original NCF approaches (SoloMLP and NeuMF). We compare the training time of SoloMLP, NeuMF, HybridNNres and HybridNNMFres. These methods are implemented by Keras\(^1\,\) which is a high-level neural network API, written in Python and capable of running on top of Tensorflow. TensorFlow\(^2\) is a widely used open-source machine learning framework\(^{[29]}\). All methods run on the same machine (Intel I7 6800K CPU and Nvidia GTX 1080 Ti GPU). The result is demonstrated in figure 11 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 HybridNNres and HybridNNMFres, 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, HybridNNres vs. HybridNNMFres). It is because GMF has one neural layer and the main cost is 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 HybridNNres (HybridNNMFres) and SoloMLP (HybridNNMFres) is not so obvious, neither. When the depth of the neural network is augmented to 6, the total training time is significantly raised for those relatively small datasets (e.g. MovieLens-100K). On those relatively small datasets, the time taken by non-computational processes (e.g. sampling process) can not be ignored when it is compared with the time taken by the computational processes (e.g. computation of the weight matrix W). Therefore, when the depth of the neural network grows, the non-computational time is enormously increased, making the total training time grow by multiplying. For those relatively large datasets (e.g. Amazon Movies), the non-computational time can be ignored. The increment of the total training time is mainly caused by the augmentation of the computational time as the depth of the neural network grows. The training time of HybridNNres (HybridNNMFres) is thus not greatly raised on those rela-

---

**TABLE VI**

| number of layers | HR@10 | NDCG@10 |
|------------------|-------|---------|
| 3                | 1     |         |
| 9                | 2     |         |
| 18               | 3     |         |

**TABLE VII**

| Negative ratio | HR@10 |
|----------------|-------|
| 1              | 0.701 |
| 4              | 0.722 |
| 5              | 0.711 |
| 6              | 0.706 |
| 10             | 0.704 |

**TABLE VIII**

| datasets       | With pre-training | Without pre-training |
|----------------|-------------------|----------------------|
| MovieLens-100K|                   |                      |
| MovieLens-1M   |                   |                      |
| Douban Book    |                   |                      |
| Amazon Movies  |                   |                      |
| Amazon Games   |                   |                      |

---

1. https://keras.io/
2. https://www.tensorflow.org/
In this paper, we propose a hybrid neural network that consists of the global neural network and the local neural block to learn the nonlinear relationships between users and items. More neural layers are adopted, which makes the information learnt by our method more accurate than that learnt by existing ones. Moreover, we exploit heterogenous neural networks with various structures, which makes the information learnt by our method more diverse.

Although only three neural structures are adopted to build the local neural block, the construction method is far more than these three methods. There are still many open issues. Similarly, the global neural network can also be replaced by other neural networks instead of the tower pattern MLP. In general, the tower pattern neural network may be better than the parallel neural network in modeling relationships between users and items because the degradation problem may emerge for the latter structure.

VI. CONCLUSION AND FUTURE WORK

In this paper, we propose a hybrid neural network that consists of the global neural network and the local neural block to learn the nonlinear relationships between users and items. More neural layers are adopted, which makes the information learnt by our method more accurate than that learnt by existing ones. Moreover, we exploit heterogenous neural networks with various structures, which makes the information learnt by our method more diverse.

Although only three neural structures are adopted to build the local neural block, the construction method is far more than these three methods. There are still many open issues. Similarly, the global neural network can also be replaced by other neural networks instead of the tower pattern MLP. In general, the tower pattern neural network may be better than the parallel neural network in modeling relationships between users and items because the degradation problem may emerge for the latter structure.

Comparing HybridNNMF res and NeuMF, we find that positions of test items are not significantly changed by HybridNNMF res. One reason may be that HybridNNMF res and NeuMF are homogeneous. Therefore, one potential way to improve our method is combining the neural network with different recommendation methods such as Poisson Factorization [25]. We will try this method in our future work.

When HybridNN res is compared with BPR, a contrary result is obtained on different datasets. For instance, HybridNN res outperforms BPR on MovieLens-1M dataset while HybridNN res underperforms BPR on Amazon Games dataset. One possible reason is that relationships between users and items are very diverse. Some users may have more linear relationships than nonlinear relationships with items, which makes the linear method better than the nonlinear method (e.g., Amazon Games). Thus there arises a question, that is, which kinds of users tend to have more linear (nonlinear) relationships with items? A more personalized recommendation algorithm can be designed on the base of the solution to the
problem. In our method, the tower pattern MLP is adopted to build the global neural network. As a result, the number of the global neural layers is under restraint. However, it does not prevent us from augmenting the depth of the whole neural network, because one can insert a local neural block not only in two adjacent layers of the global network but also in two adjacent layers of a local neural block. Therefore, plenty of neural layers can be added from a theoretical point of view.

We randomly sample a certain number of negative instances for each positive instance, but there are varied sample methods. In addition, for each positive instance, we sample the same number of negative instances. Thus there is still a lot of room to improve our algorithm. One potential way is to sample different number of negative instances for various positive instances. Another potential way is to assign varied weights for negative instances. We will make attempts on these methods in our future work.

REFERENCES

[1] G. Adomavicius and 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., vol. 17, no. 6, pp. 734–749, 2005.
[2] J. Davidson, B. Liebald, J. Liu, P. Nandy, T. Van Vleet, U. Gargi, S. Gupta, Y. He, M. Lambert, B. Livingston, and D. Sampath, “The youtube video recommendation system,” in Proc. ACM 4th conf. Recommen-der syst., 2010, pp. 293–296.
[3] C. A. Gomez-Urribe and N. Hunt, “The netflix recommender system: Algorithms, business value, and innovation,” ACM Trans. Manage. Inf. Syst., vol. 6, no. 4, pp. 13:1–13:19, 2015.
[4] Z. Kang, C. Peng, and Q. Cheng, “Kernel-driven similarity learning,” Neurocomputing, vol. 267, pp. 210–219, 2017.
[5] Y. Koren, R. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” Computer, vol. 42, no. 8, pp. 30–37, 2009.
[6] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, “Neural collaborative filtering,” in Proc. 26th Int. Conf. World Wide Web, 2017, pp. 173–182.
[7] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 770–778.
[8] A. Mnih and R. R. Salakhutdinov, “Probabilistic matrix factorization,” in Adv. Neural Inf. Process. Syst., 2008, pp. 1257–1264.
[9] R. Salakhutdinov and A. Mnih, “Bayesian probabilistic matrix factorization using markov chain monte carlo,” in Proc. Int. Conf. Mach. Learn., 2008, pp. 880–887.
[10] Y. Shen and R. Jin, “Learning personal+ social latent factor model for social recommendation,” in Proc. 18th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min., 2012, pp. 1303–1311.
[11] L. Chen, W. Zeng, and Q. Yuan, “A unified framework for recommend- ing items, groups and friends in social media environment via mutual resource fusion,” Expert. Syst. Appl., vol. 40, no. X, pp. 2889–2903, 2013.
[12] X. Qian, H. Feng, G. Zhao, and T. Mei, “Personalized recommendation combining user interest and social circle,” IEEE Trans. Knowl. Data Eng., vol. 26, no. 7, pp. 1763–1777, 2014.
[13] D. Lian, C. Zhao, X. Xie, G. Sun, E. Chen, and Y. Rui, “Geomf: joint geographical modeling and matrix factorization for point-of-interest recommendation,” in Proc. 20th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min., 2014, pp. 831–840.
[14] H. Gao, J. Tang, X. Hu, and H. Liu, “Content-aware point of interest recommendation on location-based social networks,” in AAAI Conf. Artif. Intell., 2015, pp. 1721–1727.
[15] Y. Koren, “Collaborative filtering with temporal dynamics,” Commun. ACM, vol. 53, no. 4, pp. 89–97, 2010.
[16] C. Zhang, K. Wang, H. Lu, J. Sun, and E.-P. Lim, “Latent factor transition for dynamic collaborative filtering,” in Proc. SIAM Int. Conf. Data Min., 2014, pp. 452–460.
[41] R. He and J. McAuley, “Vbpr: Visual bayesian personalized ranking from implicit feedback,” in AAAI Conf. Artif. Intell., 2016, pp. 144–150.

[42] R. He and J. McAuley, “Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering,” in Proc. 25th Int. Conf. World Wide Web, 2016, pp. 507–517.

[43] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, “Session-based recommendations with recurrent neural networks,” in ICLR, 2016, pp. 1–10.

[44] Y. K. Tan, X. Xu, and Y. Liu, “Improved recurrent neural networks for session-based recommendations,” in Proc. 1st Workshop Deep Learn. Recommender syst., 2016, pp. 17–22.

[45] S. Wu, W. Ren, C. Yu, G. Chen, D. Zhang, and J. Zhu, “Personal recommendation using deep recurrent neural networks in netease,” in Proc. IEEE 32nd Int. Conf. Data Eng., 2016, pp. 1218–1229.

[46] C.-Y. Wu, A. Ahmed, A. Beutel, A. J. Smola, and H. Jing, “Recurrent recommender networks,” in Proc. 10th ACM Int. Conf. Web Search Data Min., 2017, pp. 495–503.

[47] X. Wang, X. He, L. Nie, and T.-S. Chua, “Item silk road: Recommending items from information domains to social users,” in Proc. 40th Int. ACM SIGIR Conf. Research Dev. Info. Retrieval, 2017, pp. 185–194.

[48] J. Lian, F. Zhang, X. Xie, and G. Sun, “Cccnet: A content-boosted collaborative filtering neural network for cross domain recommender systems,” in Proc. 26th Int. Conf. World Wide Web, 2017, pp. 817–818.

[49] Y. Hu, Y. Koren, and C. Volinsky, “Collaborative filtering for implicit feedback datasets,” in Proc. IEEE Int. Conf. Data Min., 2008, pp. 263–272.

[50] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in ICLR, 2014, pp. 1–15.

[51] W. Zeng, M. Fang, J. Shao, and M. Shang, “Uncovering the essential links in online commercial networks,” Sci. Rep., vol. 6, pp. 34292, 2016.

[52] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, “Item-based collaborative filtering recommendation algorithms,” in Proc. 10th Int. Conf. World Wide Web, 2001, pp. 285–295.

[53] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, “Bpr: Bayesian personalized ranking from implicit feedback,” in Proc. 25th Conf. Uncertainty Artif. Intell., 2009, pp. 452–461.

[54] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Recommender Systems Handbook, 1st ed. Berlin, Heidelberg: Springer-Verlag, 2010.