Hindawi
International Journal of Digital Multimedia Broadcasting
Volume 2019, Article ID 1326831, 10 pages
https://doi.org/10.1155/2019/1326831

Research Article

QoS Prediction for Neighbor Selection via Deep Transfer Collaborative Filtering in Video Streaming P2P Networks

Wenming Ma,[1] Qian Zhang,[1] Chunxiao Mu,[1] and Meng Zhang[2]

[1] *School of Computer and Control Engineering, Yantai University, Yantai 264005, China*
[2] *China National Nuclear Corporation, Beijing 100045, China*

Correspondence should be addressed to Wenming Ma; mwmytu@126.com

Received 21 September 2018; Accepted 25 October 2018; Published 1 January 2019

Guest Editor: Yuanlong Cao

To expand the server capacity and reduce the bandwidth, P2P technologies are widely used in video streaming systems in recent years. Each client in the P2P streaming network should select a group of neighbors by evaluating the QoS of the other nodes. Unfortunately, the size of video streaming P2P network is usually very large, and evaluating the QoS of all the other nodes is resource-consuming. An attractive way is that we can predict the QoS of a node by taking advantage of the past usage experiences of a small number of the other clients who have evaluated this node. Therefore, collaborative filtering (CF) methods could be used for QoS evaluation to select neighbors. However, we might use different QoS properties for different video streaming policies. If a new video steaming policy needs to evaluate a new QoS property, but the historical experiences include very few evaluation data for this QoS property, CF methods would incur severe overfitting issues, and the clients then might get unsatisfied recommendation results. In this paper, we proposed a novel neural collaborative filtering method based on transfer learning, which can evaluate the QoS with few historical data by evaluating the other different QoS properties with rich historical data. We conduct our experiments on a large real-world dataset, the QoS values of which are obtained from 339 clients evaluating on the other 5825 clients. The comprehensive experimental studies show that our approach offers higher prediction accuracy than the traditional collaborative filtering approaches.

1. Introduction

In recent years, video content accounts for a large proportion of global Internet consumption. Video steaming is gradually becoming the most attractive service [1–3]. However, Internet does not provide any quality of service guarantees to video content delivery. To expand the server capacity and reduce the video streaming bandwidth, P2P technologies are widely adopted by many content delivery systems [4–7]. In a P2P network, a peer not only downloads the media data from the network but also uploads the download data to other clients in the same network. To get a better user experience of watching videos, each client (or node) in the P2P network should select some other nodes as its neighbors in terms of the quality of service (QoS) for this client [8–10]. For example, a client might prefer to select nodes with high bandwidth. Due to the different locations and network conditions, different clients might have different QoS experience for the same node. To get the neighbors with the best QoS, one might want to evaluate the QoS of all the other nodes for each client. Unfortunately, the video streaming P2P network usually includes an extremely large number of users, and evaluating the QoS of all the nodes is time-consuming and resource-consuming.

An attractive way is that we can predict the QoS value of a node by taking advantage of the past usage experiences of a small number of the other clients who have evaluated this node. This refers to a famous technology, collaborative filtering (CF), which has been extremely studied in recommender systems [11–13]. With the help of CF method, each client only needs to know a small number of the real QoS values of the other nodes to select neighbors. The core idea is that if two clients have similar evaluation values of a specific QoS for some known nodes, they might also have similar QoS evaluation values for the other unknown nodes.

However, the neighbor selection policy might need to be changed to improve the quality of video content delivery. If the new policy uses the new QoS property to select neighbors, but the historical user experiences include very few data of this new QoS property, CF methods would incur severe overfitting issues, and then each client might get worse neighbor recommendation list. Transfer learning aims to adapt a model trained in a source domain with rich labeled data for use in a target domain with less labeled data, where the source and target domain are usually related but under different distributions [14–16]. Recently, deep neural networks have yielded remarkable success on many applications, especially on the computer vision, speech recognition, and natural language processing. Deep neural networks are powerful for learning general and transferable features. There are two major transfer learning scenarios, fine-tuning the pretrained network or treating the pretrained network as a fixed feature exactor. Instead of random initialization, we can initialize the network with a pretrained network, or we can freeze the weights of some layers of the network [17–19].

Unlike many supervised transfer learning tasks, we cannot simply fine-tune or freeze the weights of the network. The only information about the nodes in the video streaming P2P network is their identifiers (IDs) and the QoS evaluation historical experience. There is no raw feature for each node, and we need to lean abstract features for the nodes using embedding. Freezing the embedding features seems unreasonable. Furthermore, different QoS properties have different value ranges, and fine-tuning will make the final weights differ greatly from the initial weights pretrained in the source domain. Due to the sparsity of target domain labeled data, fine-tuning too much would incur severe overfitting problem.

In this paper we proposed a novel neural style collaborative filtering method, DTCF (Deep Transfer Collaborative Filtering). We can first train the model using the QoS evaluation data in the source domain and then adapt the model in the target domain with different QoS property. The core idea is that we only use the weights of first several layers to initialize the same layers of the model in the target domain, and randomly initialize the remaining layers. To control the degree of fine-tuning, we integrate the maximum mean discrepancy (MMD) measurement into the loss function [20–22]. The main contributions of our work are as follows:

(i) We propose a novel neural collaborative filtering model for QoS prediction using transfer learning technology.

(ii) We provide a novel interaction layer to represent the relationship between latent embedding factors of the nodes.

(iii) We adopt partial fine-tuning and MMD measurement to train the target domain model to implement domain adapting.

The remainder of this paper is organized as follows: We introduce the related work in Section 2. Section 3 presents the design details of our method. Section 4 describes our experiments and Section 5 concludes this paper.

2. Related Work

Distributed user-generated videos delivery poses a new challenge to large-scale streaming systems. To stream live videos generated by users, many existing video streaming systems rely on a centralized network architecture [23–25]. Even these streaming systems use Content Delivery Network (CDN) for video delivery, such a kind of solution is not cost-effective [26–28]. The unit price of content delivery over the Internet has dramatically decreased in recent years. However, there are higher requirements in terms of resolution, frame rate, or bitrate than before. Therefore, the amount of bandwidth consumed per user has grown at a faster rate. To reduce the bandwidth or the costs and improve the user experience, the P2P architectures can be adopted instead.

Collaborative filtering is a rational QoS prediction technology to select neighbors for each client in the P2P video streaming network [29–31]. To select the best neighbors with high delivery quality for the clients, CF should predict the QoS values between the clients and then select the top k best neighbors in terms of the QoS values. Each client only knows partial information about the QoS values for all the nodes in the network. Memory-based CF methods are some kinds of generalized k-nearest-neighbors (KNN) algorithms [32, 33], which have two types of models: user-based and item-based. Model-based CF methods are more popular, which act like generalized regression or classification algorithms, but they deal with abstract features not concrete or raw features. Among many model-based CF methods, matrix factorization has become the most popular technology to handle such kind of issues [34–40]. Probabilistic Matrix Factorization (PMF) model considers that the QoS values obey a normal Gaussian distribution, and the latent factors should be learned from zero-mean Gaussian distribution [41]. Nonnegative matrix factorization (NMF) can learn the nonnegative latent factors for the users or items, but it usually deals with the implicit feedback [42–44].

However, even if matrix factorization CF algorithms have obtained remarkable success, they have difficulty in dealing with cross-domain learning tasks if the output values of the source and target domain have different ranges. Deep neural networks can easily learn general and transferable features. More and more cross-domain applications adopt deep learning technologies and have yielded remarkable performance [45–47]. However, the exploration of deep neural networks on recommender systems or QoS prediction has received relatively less attraction. Recently, some studies have proposed some deep learning-based collaborative filtering models. Two impressive technologies are Google's Wide & Deep [48] and Microsoft's Deep Crossing [49]. The input of these models is side information, not the interaction between the users and items. Neural Collaborative Filtering (NCF) models are designed purely for user and item interactions [50]. However, none of them are designed for cross-domain QoS prediction.

3. DTCF Model

For the cross-domain QoS prediction in the video streaming P2P network, we are given a *source* domain $\mathscr{D}_s = \{\langle x_i^s, x_j^s, r_{ij}^s\rangle\}_{i\neq j}$ with n_s examples, which is characterized by the probability distribution p and a *target* domain $\mathscr{D}_t = \{\langle x_k^t, x_l^t, r_{kl}^t\rangle\}_{k\neq l}$ with n_t examples which is characterized by the probability q. Usually the size of examples in the target domain is extremely small, $n_s \succ n_t$. Our work aims to build a deep neural network to learn transferable features that bridge these two domains' discrepancy.

3.1. Model Architecture Overview. We propose a novel neural architecture, outlined in Figure 1. The source domain and the target domain share the same network architecture. The input of the model is the identifier number of the nodes. For example, if size of nodes in the P2P network is n, the ID of each node is an integer number from 1 to n. The output of the mode is the QoS value that the node x_i evaluates on the node x_j.

Since we do not use any concrete feature for each node, we need to learn abstract features for them. Here, we use embedding layer to learn a continuous latent vector/factor $\mathbf{u}$ for each node. The details of designing embedding layers are described in Section 3.1.

If we get two latent vectors for x_i and x_j, $\mathbf{u}_i$ and $\mathbf{u}_j$, one might expect that we should concatenate these two vectors and then use affine function $\mathbf{W}\left[\mathbf{u}_i \;\; \mathbf{u}_j\right]^{\mathrm{T}} + \mathbf{b}$ to transform the latent vectors into the input of the other hidden layer above. However, there is no interactive action between the latent factors, but only weighted summation of elements of the vectors. Some studies use the dot product of the vectors to represent the interaction, which is described as follows.

$$\mathbf{u}_i \odot \mathbf{u}_j = \left\langle u_{i,1} \times u_{j,1}, u_{i,2} \times u_{j,2}, \cdots, u_{i,d} \times u_{j,d} \right\rangle \tag{1}$$

Unfortunately, it is too simple to completely represent the complex interaction between nodes. In this paper, we propose a novel interaction layer to tackle this problem, which has powerful representation capacity. We will give the design details in Section 3.2.

Above the interaction layer, we use ReLU as the hidden layer. We might need multiple ReLU layers. The ReLU activation function is as follows.

$$\begin{aligned} \mathbf{z}_o &= \mathbf{W}\mathbf{a}_i + \mathbf{b} \\ \mathbf{a}_o &= \left(\mathbf{0}, \mathbf{z}_o\right) \end{aligned} \tag{2}$$

Finally, we use a fully connected layer to generate the output. When training the model in the source domain, we use the regression loss. We then use the all the layers of the pretrained model but the last FC layer to construct the model for target domain. The weights of these layers are kept as the initialized weights of the target domain model, but the final FC layer is initialized randomly. To avoid the overadaptation problem, we use both the domain loss and regression loss to train the target domain model. We will describe how to design the domain loss in Section 3.3.

3.2. Embedding Layer. Since we can assign a unique integer number as the identifier for each node in the network, we can use a one-hot vector to represent the identifier. If we have at most n nodes in the network, the ith nodes can be expressed as follows.

$$\begin{gathered} \mathbf{v}_i = \left[v_{i,1}, v_{i,2}, \ldots, v_{i,n}\right] \\ v_{i,r} = \begin{cases} 1, & if\ r = i,\ 1 \le r \le n \\ 0, & if\ r \neq i,\ r \le l \le n \end{cases} \end{gathered} \tag{3}$$

Our embedding layer is defined as follows:

$$\mathbf{u}_i = \mathbf{W}\mathbf{v}_i \tag{4}$$

where $\mathbf{W}$ is a $d \times n$ matrix. Expanding the formulas $\mathbf{W}\mathbf{v}_i$, we can see the following.

$$\mathbf{W}\mathbf{v}_i = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ \vdots & \vdots & \cdots & \vdots \\ w_{k1} & w_{k2} & \cdots & w_{km} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ \vdots \\ 0 \end{bmatrix} \begin{bmatrix} w_{1i} \\ w_{2i} \\ \vdots \\ w_{ii} \\ \vdots \\ w_{ki} \end{bmatrix} \tag{5}$$

Therefore, $\mathbf{u}_i$ is the ith column of matrix $\mathbf{W}$. Since the node identifier number is transformed to a one-hot vector, the result of matrix multiplication is exactly a specific latent vector for each node. This weight matrix is jointly trained with the other parameters of the whole network.

3.3. Interaction Layer. There are two inputs of the interaction layer, $\mathbf{u}_i$ and $\mathbf{u}_j$. Suppose any single vector is a column vector, and concatenating the two inputs will get a longer vector. This concatenation vector will be transformed to another vector, encoding interactive information between these two inputs. The transformation process is outlined in Figure 2.

Suppose the output of interaction layer is a vector $\mathbf{h}$, the length of which is k. The sth element of the vector $\mathbf{h}$ is defined as follows.

$$h_c = \left[\mathbf{u}_i^{\mathrm{T}} \;\; \mathbf{u}_j^{\mathrm{T}}\right] \mathbf{W}_s \begin{bmatrix} \mathbf{u}_i \\ \mathbf{u}_j \end{bmatrix} + b_s \tag{6}$$

If the length of $\mathbf{u}_i$ is d, $\mathbf{W}_s$ is a $2d \times 2d$ square matrix. h_c is a scalar, the value of which is determined by the matrix $\mathbf{W}_s$ and the bias b_s. If the length of $\mathbf{h}$ is k, we need k weight matrices and biases.

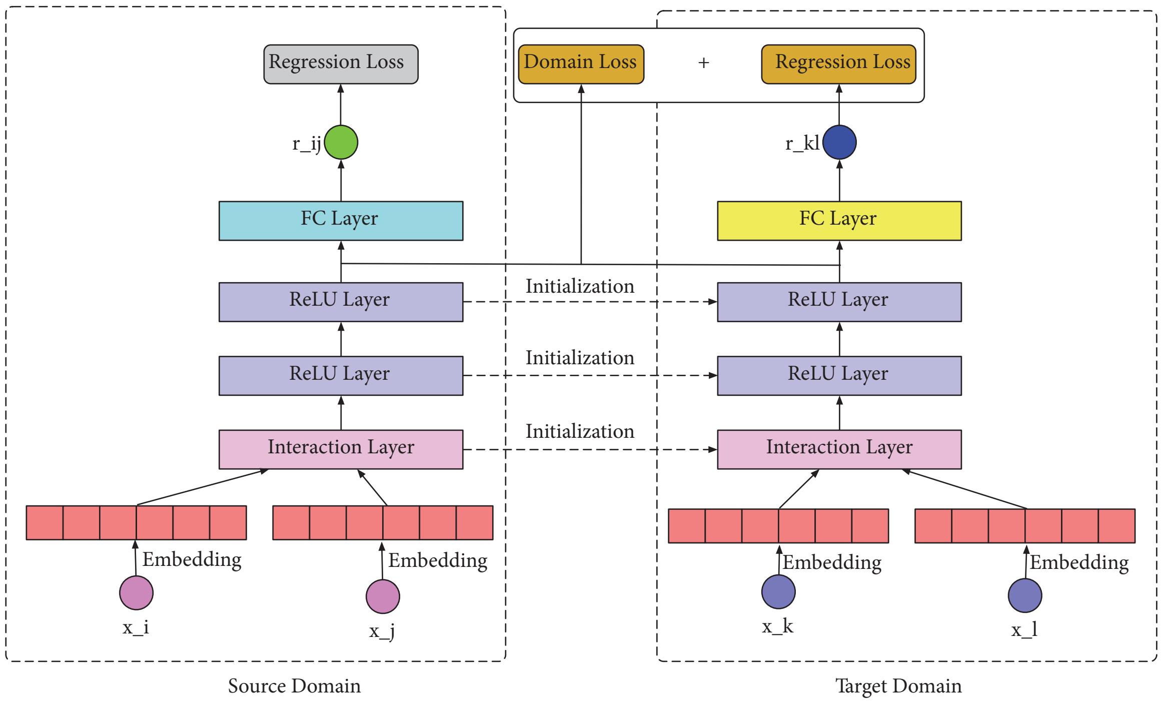


Figure 1: DTCF architecture.

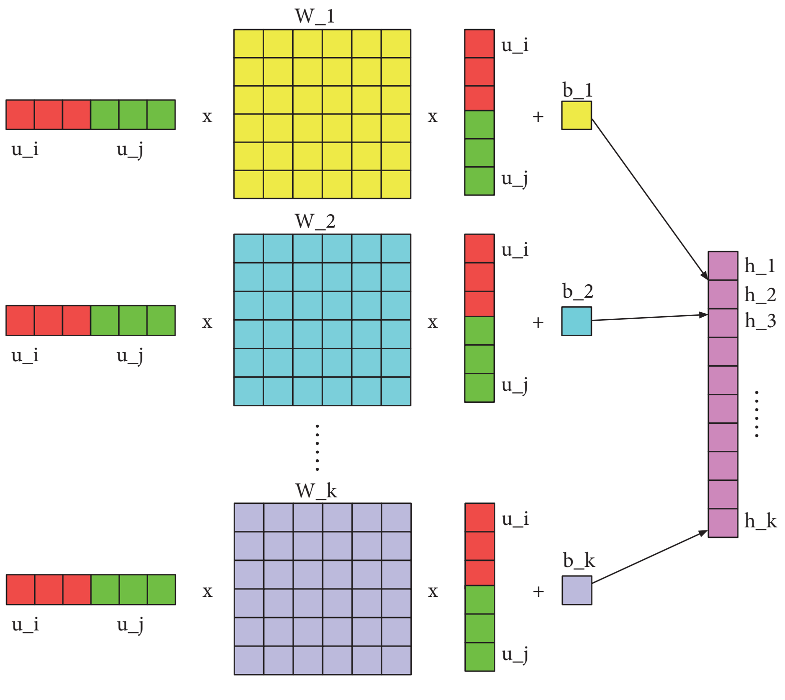


Figure 2: Interaction layer.

h_c include all the possible interaction relationships between $\mathbf{u}_i$ and $\mathbf{u}_j$. Denote $\mathbb{U}^{\mathrm{T}} = \left[\mathbf{u}_i^{\mathrm{T}}\ \mathbf{u}_j^{\mathrm{T}}\right] = [\mathbb{U}_1, \mathbb{U}_2, \ldots, \mathbb{U}_{2k}]$, and we can see that $h_c = \sum w_s^{ij} \mathbb{U}_i \mathbb{U}_j + b_s$, where w_s^{ij} is the element at the ith row and the jth column in the matrix $\mathbf{W}_s$

3.4. Domain Loss. The output of the last ReLU layer of the model in the source domain is denoted as $\mathbf{h}^s$, and the output of the last ReLU layer of the model in the target domain is denoted as $\mathbf{h}^t$. If we want to avoid the overadaptation problem, one possible way is to minimize the differences between the distributions $p = \Pr(x_i^s, x_j^s, \mathbf{h}^s(x_i^s, x_j^s))$ and $q = \Pr(x_i^t, x_j^t, \mathbf{h}^t(x_i^t, x_j^t))$, where $x_i^s = x_i^t$ and $x_j^s = x_j^t$.

Let $(\mathscr{X}, D)$ be a metric space, and $\langle x_i^s, x_j^s, \mathbf{h}^s(x_i^s, x_j^s)\rangle \in \mathscr{X}$, $\langle x_i^t, x_j^t, \mathbf{h}^t(x_i^t, x_j^t)\rangle \in \mathscr{X}$. Let $\mathscr{F}$ be a class of functions $f : \mathscr{X} \longrightarrow \mathbb{R}$, and the Maximum Mean Discrepancy (MMD) is as [22]

$$\mathrm{MMD}[\mathscr{F}, p, q] = \sup_{f\in\mathscr{F}} \left(\mathbb{E}_{\mathbf{s}\sim p}[f(\mathbf{s})] - \mathbb{E}_{\mathbf{t}\sim q}[f(\mathbf{t})]\right) \tag{7}$$

where $\mathbf{s} = \langle x_i^s, x_j^s, \mathbf{h}^s(x_i^s, x_j^s)\rangle \in \mathbf{S}$ and $\mathbf{t} = \langle x_i^t, x_j^t, \mathbf{h}^t(x_i^t, x_j^t)\rangle \in \mathbf{T}$.

Denote $\langle x_i^s, x_j^s\rangle \in \mathbf{S}_x$ and $\langle x_i^t, x_j^t\rangle \in \mathbf{T}_x$. The biased empirical estimated of the MMD is defined as follows.

$$\begin{aligned}&\mathrm{MMD}[\mathscr{F}, \mathbf{S}, \mathbf{T}]\\ &\quad= \sup_{f\in\mathscr{F}} \frac{1}{|\mathbf{S}_x \cap \mathbf{T}_x|} \sum_{\mathbf{s}_i, \mathbf{t}_i \in \mathbf{S}_x\cap\mathbf{T}_x} ([f(\mathbf{s}_i)] - [f(\mathbf{t}_i)])\end{aligned} \tag{8}$$

If the function class $\mathscr{F}$ is too large, it is not practical to work with this rich function class in the finite sample setting. A rational choice of the function class is a universal reproducing kernel Hilbert space $\mathscr{H}$, named *universal RKHS*. Therefore, we have that $f(x) = \langle f, \phi(x)\rangle_{\mathscr{H}}$, where $f \in \mathscr{H}$. The kernel function $k(x, y)$ is equal to $\langle \phi(x), \phi(y)\rangle$.

Denote $f = \phi(a) = k(a, \cdot)$, and we can get the mean embedding of the distribution p; that is, $\mu_p(a) = \langle \mu_p, k(a, \cdot)\rangle = \mathbb{E}_x k(a, x)$. From [22], we can obtain the following.

$$\begin{aligned}&\mathrm{MMD}[\mathscr{F}, p, q]^2\\ &\quad= \left[\sup_{\|f\|_{\mathscr{H}}\le 1} \left(\mathbb{E}_{\mathbf{s}\sim p}[f(\mathbf{s})] - \mathbb{E}_{\mathbf{t}\sim q}[f(\mathbf{t})]\right)\right]^2\\ &\quad= \left[\sup_{\|f\|_{\mathscr{H}}\le 1} \left\langle \mu_p - \mu_q, f\right\rangle_{\mathscr{H}}\right]^2 = \left\|\mu_p - \mu_q\right\|_{\mathscr{H}}^2\\ &\quad= \mathbb{E}_{\mathbf{s},\mathbf{s}'}\left[k\left(\mathbf{s}, \mathbf{s}'\right)\right] - 2\mathbb{E}_{\mathbf{s},\mathbf{t}}[k(\mathbf{s}, \mathbf{t})] + \mathbb{E}_{\mathbf{t},\mathbf{t}'}\left[k\left(\mathbf{t}, \mathbf{t}'\right)\right]\end{aligned} \tag{9}$$

Similarly, the empirical estimate can be defined now as follows.

$$\begin{aligned}\mathrm{MMD}[\mathscr{F}, \mathbf{S}, \mathbf{T}]^2 &= \frac{1}{L^2}\sum_{i=1}^{L}\sum_{j\ne i}^{L} k\left(\mathbf{s}_i, \mathbf{s}_j\right)\\ &\quad+ \frac{1}{L^2}\sum_{i=1}^{L}\sum_{j\ne i}^{L} k\left(\mathbf{t}_i, \mathbf{t}_j\right)\\ &\quad- \frac{2}{L^2}\sum_{i=1}^{L}\sum_{j=1}^{L} k\left(\mathbf{s}_i, \mathbf{t}_j\right)\end{aligned} \tag{10}$$

In this paper, we use the empirical estimate of MMD^2 as the domain loss. What we need to do is to select suitable universal kernel function. Here, we adopt Gaussian kernel function, which is defined as follows.

$$k(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\delta^2}\right) \tag{11}$$

3.5. Algorithm. The total loss of target domain includes regression loss and MMD loss. We use the minibatch to train the model. Only a small group of examples are used to compute the loss per training iteration. Denote the set of the minibatch examples in the source domain $\mathscr{M}_s$ and the set of the minibatch examples in the target domain $\mathscr{M}_t$. The loss function of the model in the source domain is defined as follows.

$$\mathscr{L}(\mathscr{M}_s) = \frac{1}{|\mathscr{M}_s|}\sum\left(\hat{r}_{ij} - r_{ij}\right)^2 \tag{12}$$

However, the loss function of the model in the target domain is defined as

$$\mathscr{L}(\mathscr{M}_t) = \frac{1}{|\mathscr{M}_t|}\sum\left(\hat{r}_{ij} - r_{ij}\right)^2 + \mathrm{MMD}[\mathscr{F}, \mathbf{S}, \mathbf{T}]^2 \tag{13}$$

where $\mathbf{S}_x \cap \mathbf{T}_x = \mathscr{M}_t$.

To optimize our model, we need to compute the gradient of each weight. For any weigh related to both of the regression and domain loss, its gradient is computed as follows.

$$\begin{aligned}&\frac{\partial \mathscr{L}(\mathscr{M}_t)}{\partial w}\\ &\quad= \frac{1}{|\mathscr{M}_t|}\sum \frac{\partial\left(\hat{r}_{ij} - r_{ij}\right)^2}{\partial w} + \frac{\partial \mathrm{MMD}[\mathscr{F}, \mathbf{S}, \mathbf{T}]^2}{\partial w}\\ &\quad= \frac{2}{|\mathscr{M}_t|}\sum \frac{\left(\hat{r}_{ij} - r_{ij}\right)\partial \hat{r}_{ij}}{\partial w}\\ &\qquad+ 2\mathrm{MMD}[\mathscr{F}, \mathbf{S}, \mathbf{T}]\frac{\partial \mathrm{MMD}[\mathscr{F}, \mathbf{S}, \mathbf{T}]}{\partial w}\end{aligned} \tag{14}$$

$$\begin{aligned}&\frac{\partial \mathrm{MMD}[\mathscr{F}, \mathbf{S}, \mathbf{T}]}{\partial w}\\ &\quad= -\frac{2}{|\mathscr{M}_t|}\sum_{i\ne j}\exp\left(-\frac{\left\|\mathbf{h}_i^t - \mathbf{h}_j^t\right\|^2}{2\delta^2}\right)\left\|\mathbf{h}_i^t - \mathbf{h}_j^s\right\|\frac{\partial \mathbf{h}_i^t}{\partial w}\\ &\qquad- \frac{2}{|\mathscr{M}_t|}\sum_{i\ne j}\exp\left(-\frac{\left\|\mathbf{h}_i^t - \mathbf{h}_j^t\right\|^2}{2\delta^2}\right)\left\|\mathbf{h}_i^t - \mathbf{h}_j^s\right\|\frac{\partial \mathbf{h}_j^t}{\partial w}\\ &\qquad+ \frac{4}{|\mathscr{M}_t|}\sum\exp\left(-\frac{\left\|\mathbf{h}_i^s - \mathbf{h}_j^t\right\|^2}{2\delta^2}\right)\left\|\mathbf{h}_i^s - \mathbf{h}_j^t\right\|\frac{\partial \mathbf{h}_j^t}{\partial w}\end{aligned} \tag{15}$$

Note that $\mathbf{h}_i^s$ is not used for computing gradients, because we only train the target domain model after pretraining in the source domain. The training process is described as follows:

(i) We first train the model of the source domain using the loss function $\mathscr{L}(\mathscr{M}_s)$. The gradient of each weigh is computed according to formula (13).

(ii) After training, we use the weights of this model to initialize the model in the target domain except the weights of the last FC layer. The last FC layer of the model of the target domain is initialized randomly.

(iii) While training the model of the target domain, we use the loss function $\mathscr{L}(\mathscr{M}_t)$.

(iv) For each training iteration, we randomly select examples in the dataset, and compute the gradient according to formulas (14) and (15).

(v) We use ADAM (Adaptive Moment Estimation) as the optimizer.

4. Experimental Results

4.1. Dataset and Evaluation Metrics. We conduct our experiments on a publicly large accessible dataset, WS-DREAM dataset#1, obtained from 339 hosts doing QoS evaluation on the other 5825 hosts. There are two types of QoS properties in this dataset: response time and throughput. Here, we use the response time as the source domain, and the throughput as the target domain.

For the source domain, we randomly extract 30% (density) of the data as the source training set. For the target domain, we construct 5 different training sets with different density of 0.5%, 1%, 1.5%, 2%, 2.5%, and 3%. Consequently, the remaining data is the test set.

We adopt a common evaluation metric: Mean Absolute Error (MAE), which is widely employed to measure the QoS prediction quality.

$$MAE = \frac{\sum_{(i,j,r_{i,j})\in \mathbf{Q}_E} \left| r_{i,j} - \hat{r}_{i,j} \right|}{\left| \mathbf{Q}_E \right|} \tag{16}$$

4.2. Performance Comparison. We compare our methods with some traditional collaborative filtering methods: UPCC, IPCC, UIPCC [34], and matrix factorization (MF). UPCC is a user-based CF method, which uses PCC (Pearson Correlation Coefficient) to calculate the similarity between users. IPCC is similar to UPCC, except that it calculates the similarity between items. UIPCC combines the advantages of these two methods by balancing the proportions of them in the final results. For UPCC, IPCC, and UIPCC, different tradeoff parameters $k = 5, 10, 15, 20, 25, 30$ (the parameters of top k similar users or services) are tried, and finally we choose $k = 10$. For MF and DTCF, the sizes of latent factors are also set to 10. For DTCF, different hidden ReLU layers and different hidden unit sizes are tried. Here, the maximum number of hidden layers is limited to 5. We tested the batch size of [128, 256, 512, 1024], the learning rate of [0.0001,0.0005, 0.001, 0.005], and the training epoch of [10, 20, 30, 40, 50, 60, 70, 80]. The bandwidth δ is set to the median pairwise distance on the source training data.

We conduct 10 experiments for each model and each sparsity level and then average the prediction accuracy values.

The results are reported in Figures 3 and 4. We can make the following observations:

(i) As the sparsity level increases, the MAEs of all the models decrease.

(ii) Our DTCF methods outperform the other traditional collaborative filtering methods, especially when the training set is extremely sparse.

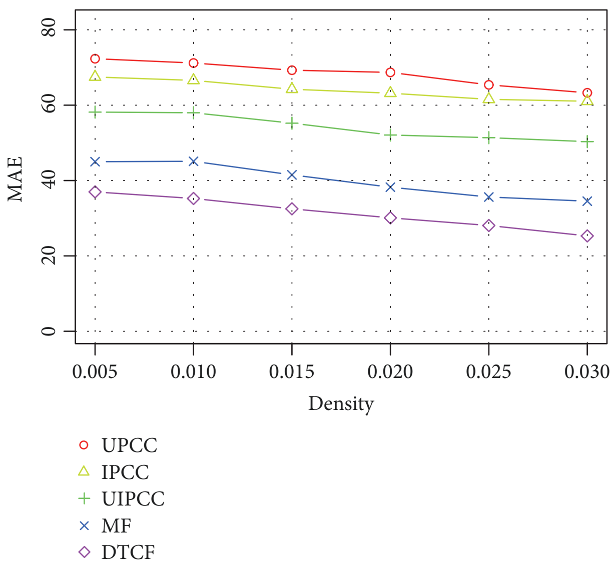


FIGURE 3: MAE with respect to density.

(iii) DTCF model has more weights that need to be trained than the other models, but it gets the best performance, which indicates that the relationship between nodes is very complex, and shallow models cannot capture these structures.

Although shallow models are not easily overfitting when the target domain training dataset is extremely sparse, they cannot transfer rich information from the source domain. The deep models might easily incur overfitting problem, but they can learn common latent features from the source domain. To balance this dilemma, we need to control the degree of fine-tuning the deep model. This experiment shows that MMD domain loss is an efficient way of controlling the adapting degree.

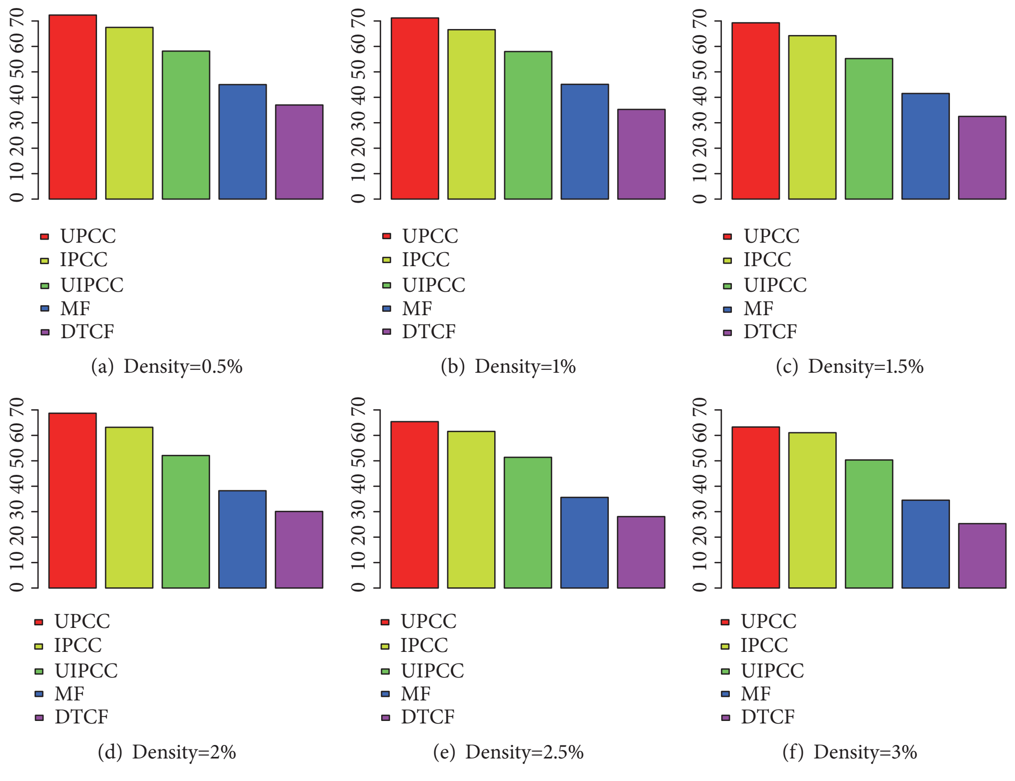
4.3. Impact of the Network Depth. The network depth usually has important impact on the prediction performance. Here, the number of neurons of each ReLU is set to 128, and we add the number of ReLU layers from 1 to 6 to see how the MAE values change. The experimental result is outlined in Figure 5, from which we can see the following:

(i) Adding more ReLU layers can get better prediction performance, but when the depth exceeds a limited value, the performance starts to become worse.

(ii) Although adding more ReLU layers can improve the performance, it seems that enlarging the size of the training data would be more helpful.

(iii) Sometimes, adding more layers would not improve the performance anymore, but it also does not get worse prediction performance. This indicates that deep neural network has some kind of regularization property.

Actually, if the training dataset is very large, adding more layers usually does not incur overfitting problems, but for the



(a) Density=0.5%

(b) Density=1%

(c) Density=1.5%

(d) Density=2%

(e) Density=2.5%

(f) Density=3%

FIGURE 4: MAE comparison for each density.

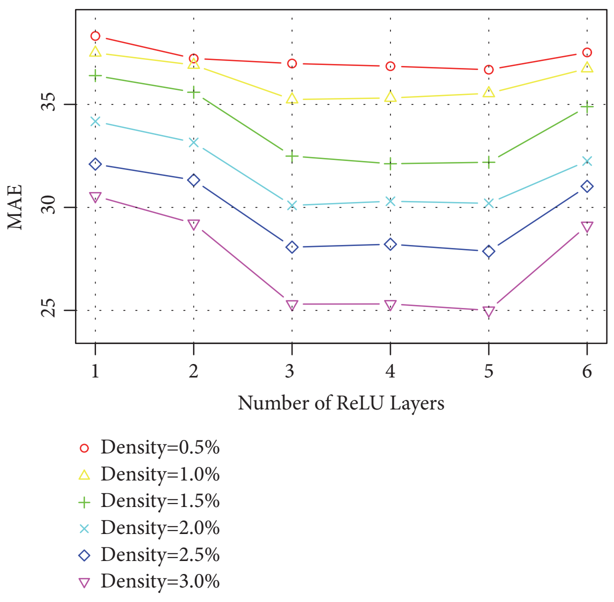


FIGURE 5: MAE with respect to number of ReLU layers.

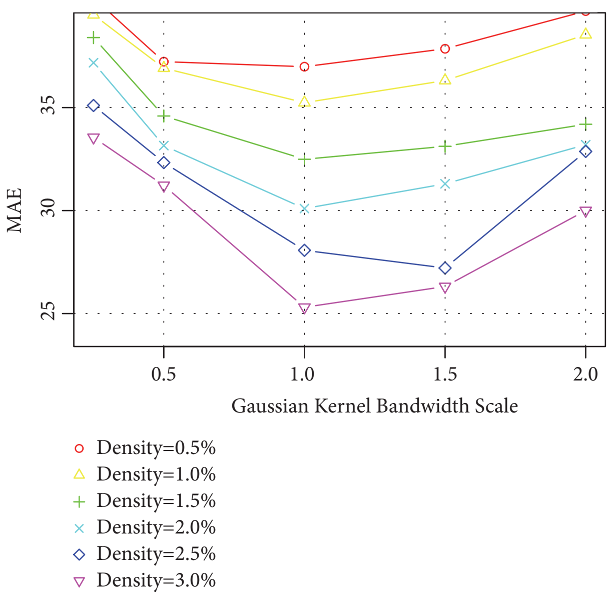


FIGURE 6: MAE with respect to Gaussian kernel bandwidth scale.

cross-domain learning, the target domain has very little data, so the network depth needs control.

4.4. Impact of the Gaussian Kernel Bandwidth. Another hyperparameter that we need to determine is the Gaussian kernel bandwidth. By default, it is set to the median pairwise distance on the source training data. We scale the default value from 0.25 to 2.0, and the experimental result is outlined in Figure 6.

(i) Obviously, the default value is a rational choice, and scaling too small or too large would get worse prediction performance.

(ii) If the bandwidth is too large, the kernel will be approximately equal to 1, and the nodes would look the same. We cannot propose personal recommendation for them.

(iii) If the bandwidth is too small, the kernel will be approximately equal to 0, and the nodes cannot find similar neighbors to follow their past experiences.

5. Conclusion

Selecting neighbors in terms of the QoS is an effective way of providing high quality contents in video streaming P2P networks. Due to the heterogeneous network conditions, the QoS between any pairs of nodes is different. However, evaluating the QoS of all the nodes for each user is resource-consuming. An attractive way is to adopt collaborative filtering technologies, which use only a small amount of past usage experience.

Unfortunately, the video content providers might often choose different QoS properties to select neighbors. Traditional CF methods cannot solve the cross-domain QoS prediction problem. This paper proposed a novel neural style CF method based on transfer learning. We first outlined our model architecture and then introduced the details of important parts of this model. To avoid the overadaptation problem, we combined domain loss and prediction loss together to train the model of the target domain. We adopted MMD distance as our domain loss, and we also provide its principle and how to compute the gradient. Finally, we conducted our experiments on a real-world public dataset. The experimental results show that our DTCF model can outperform the other models for cross-domain QoS prediction.

Data Availability

The WS-Dream data used to support the finding of this study is owned by a third party, which is an open dataset and is deposited in "https://github.com/wsdream/wsdream-dataset".

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work are supported by the National Nature Science Foundation of China (No. 61602399 and No. 61502410).

References

[1] B. E. Mada, M. Bagaa, and T. Taleb, "Efficient Transcoding and Streaming Mechanism in Multiple Cloud Domains," in *Proceedings of the 2017 IEEE Global Communications Conference (GLOBECOM 2017)*, pp. 1–6, Singapore, December 2017.

[2] H. Du, Q. Zheng, W. Zhang, and X. Gao, "A Bandwidth Variation Pattern-Differentiated Rate Adaptation for HTTP Adaptive Streaming over an LTE Cellular Network," *IEEE Access*, vol. 6, pp. 9554–9569, 2017.

[3] J. Li, W. Chen, M. Xiao, F. Shu, and X. Liu, "Efficient Video Pricing and Caching in Heterogeneous Networks," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 10, pp. 8744–8751, 2016.

[4] Y. He, H. Li, X. Cheng, Y. Liu, C. Yang, and L. Sun, "A Blockchain based Truthful Incentive Mechanism for Distributed P2P Applications," *IEEE Access*, vol. 6, pp. 27324–27335, 2018.

[5] S.-H. Lin, R. Pal, B.-C. Wang, and L. Golubchik, "On market-driven hybrid-P2P video streaming," *IEEE Transactions on Multimedia*, vol. 19, no. 5, pp. 984–998, 2017.

[6] G. Huang, Y. Gao, L. Kong, and K. Wu, "An incentive scheme based on bitrate adaptation for cloud-assisted P2P video-on-demand streaming systems," in *Proceedings of the 2018 IEEE 3rd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*, pp. 404–408, Chengdu, April 2018.

[7] J. Junchen, S. Shijie, V. Sekar, and H. Zhang, "Pytheas: Enabling data-driven quality of experience optimization using group based exploration-exploitation," in *Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation*, pp. 393–406, 2017.

[8] L. Wang, D. Zhang, and H. Yang, "Qos-Awareness variable neighbor selection for mesh-based P2P live streaming system," in *Proceedings of the 2013 IEEE 3rd International Conference on Information Science and Technology, ICIST 2013*, pp. 1197–1201, China, March 2013.

[9] A. T. Liem, I.-S. Hwang, A. Nikoukar, C.-Z. Yang, M. S. Ab-Rahman, and C.-H. Lu, "P2P Live-Streaming Application-Aware Architecture for QoS Enhancement in the EPON," *IEEE Systems Journal*, vol. 12, no. 1, pp. 648–658, 2018.

[10] J. Li, Y. Bai, N. Zaman, and V. C. M. Leung, "A Decentralized Trustworthy Context and QoS-Aware Service Discovery Framework for the Internet of Things," *IEEE Access*, vol. 5, pp. 19154–19166, 2017.

[11] K. Wang, H. Yin, W. Quan, and G. Min, "Enabling collaborative edge computing for software defined vehicular networks," *IEEE Network*, no. 99, pp. 1–6, 2018.

[12] X. Yang, Y. Guo, Y. Liu, and H. Steck, "A survey of collaborative filtering based social recommender systems," *Computer Communications*, vol. 41, pp. 1–10, 2014.

[13] J. Wei, J. He, K. Chen, Y. Zhou, and Z. Tang, "Collaborative filtering and deep learning based recommendation system for cold start items," *Expert Systems with Applications*, vol. 69, pp. 29–39, 2017.

[14] P. Peng, Y. Tian, T. Xiang, Y. Wang, M. Pontil, and T. Huang, "Joint Semantic and Latent Attribute Modelling for Cross-Class Transfer Learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 7, pp. 1625–1638, 2018.

[15] H. Chang, J. Han, C. Zhong, A. M. Snijders, and J.-H. Mao, "Unsupervised Transfer Learning via Multi-Scale Convolutional Sparse Coding for Biomedical Applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 5, pp. 1182–1194, 2018.

[16] Y. Guo, G. Ding, J. Han, and Y. Gao, "Zero-shot learning with transferred samples," *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3277–3290, 2017.

[17] G. Wang, W. Li, M. A. Zuluaga et al., "Interactive Medical Image Segmentation Using Deep Learning With Image-Specific Fine Tuning," *IEEE Transactions on Medical Imaging*, vol. 37, no. 7, pp. 1562–1573, 2018.

[18] X. An, X. Zhou, X. Lü, F. Lin, and L. Yang, "Sample Selected Extreme Learning Machine Based Intrusion Detection in Fog Computing and MEC," *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 7472095, 10 pages, 2018.

[19] K. Yanai and Y. Kawano, "Food image recognition using deep convolutional network with pre-training and fine-tuning," in *Proceedings of the 2015 IEEE International Conference on Multimedia and Expo Workshops, ICMEW 2015*, pp. 1–6, Turin, Italy, July 2015.

[20] M. J. Gangeh, H. Tadayyon, L. Sannachi, A. Sadeghi-Naini, W. T. Tran, and G. J. Czarnota, "Computer Aided Theragnosis Using Quantitative Ultrasound Spectroscopy and Maximum Mean Discrepancy in Locally Advanced Breast Cancer," *IEEE Transactions on Medical Imaging*, vol. 35, no. 3, pp. 778–790, 2016.

[21] H. Yan, Y. Ding, P. Li, Q. Wang, Y. Xu, and W. Zuo, "Mind the class weight bias: Weighted maximum mean discrepancy for unsupervised domain adaptation," in *Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, pp. 945–954, USA, July 2017.

[22] A. Gretton, K. M. Borgwardt, M. J. Rasch et al., "A kernel two-sample test," *Journal of Machine Learning Research (JMLR)*, vol. 13, pp. 723–773, 2012.

[23] Online. 2018. http://www.twitch.tv/.

[24] Online. 2018. http://www.ustream.tv.

[25] Online. 2018. http://www.livestream.com/.

[26] A. O. Al-Abbasi and V. Aggarwal, "EdgeCache: An optimized algorithm for CDN-based over-the-top video streaming services," in *Proceedings of the IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 202–207, Honolulu, HI, USA, April 2018.

[27] R. Denis, S. Matias, R. Juergen et al., "Service migration from cloud to multi-tier fog nodes for multimedia dissemination with QoE support," *Sensors*, vol. 18, no. 2, 2018.

[28] S. Dernbach, N. Taft, J. Kurose, U. Weinsberg, C. Diot, and A. Ashkan, "Cache content-selection policies for streaming video services," in *Proceedings of the 35th Annual IEEE International Conference on Computer Communications, IEEE INFOCOM 2016*, USA, April 2016.

[29] X. Wu, B. Cheng, and J. Chen, "Collaborative Filtering Service Recommendation Based on a Novel Similarity Computation Method," *IEEE Transactions on Services Computing*, vol. 10, no. 3, pp. 352–365, 2017.

[30] J. Liu, M. Tang, Z. Zheng, X. Liu, and S. Lyu, "Location-aware and personalized collaborative filtering for web service recommendation," *IEEE Transactions on Services Computing*, vol. 10, no. 3, pp. 686–699, 2016.

[31] D. Margaris, C. Vassilakis, and P. Georgiadis, "An integrated framework for adapting WS-BPEL scenario execution using QoS and collaborative filtering techniques," *Science of Computer Programming*, vol. 98, pp. 707–734, 2015.

[32] A. Bellogín, P. Castells, and I. Cantador, "Neighbor Selection and Weighting in User-Based Collaborative Filtering: A Performance Prediction Approach," *ACM Transactions on the Web (TWEB)*, vol. 8, no. 2, pp. 1–30, 2014.

[33] Z. Jia, Y. Yang, W. Gao, and X. Chen, "User-based collaborative filtering for tourist attraction recommendations," in *Proceedings of the IEEE International Conference on Computational Intelligence and Communication Technology (CICT '15)*, pp. 22–25, February 2015.

[34] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Collaborative web service Qos prediction via neighborhood integrated matrix factorization," *IEEE Transactions on Services Computing*, vol. 6, no. 3, pp. 289–299, 2013.

[35] J. Zhu, P. He, Z. Zheng, and M. R. Lyu, "Online QoS Prediction for Runtime Service Adaptation via Adaptive Matrix Factorization," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 10, pp. 2911–2924, 2017.

[36] R. Zhu, D. Niu, and Z. Li, "Robust web service recommendation via quantile matrix factorization," in *Proceedings of the 2017 IEEE Conference on Computer Communications, INFOCOM 2017*, USA, May 2017.

[37] Y. Zhang, Z. Zheng, and M. R. Lyu, "Exploring latent features for memory-based QoS prediction in cloud computing," in *Proceedings of the 30th IEEE International Symposium on Reliable Distributed Systems (SRDS '11)*, pp. 1–10, IEEE, Madrid, Spain, October 2011.

[38] D. Yu, Y. Liu, Y. Xu, and Y. Yin, "Personalized QoS prediction for web services using latent factor models," in *Proceedings of the 11th IEEE International Conference on Services Computing, (SCC '14)*, pp. 107–114, July 2014.

[39] Y. Koren, "Factorization meets the neighborhood: a multi-faceted collaborative filtering model," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '08)*, pp. 426–434, New York, NY, USA, August 2008.

[40] W. Lo, J. Yin, S. Deng, Y. Li, and Z. Wu, "An Extended Matrix Factorization Approach for QoS Prediction in Service Selection," in *Proceedings of the 2012 9th IEEE International Conference on Services Computing (SCC)*, pp. 162–169, Honolulu, HI, USA, June 2012.

[41] R. Salakhutdinov and A. Mnih, "Probabilistic Matrix Factorization," in *Proceedings of the Advances in Neural Information Processing Systems (NIPS'07)*, pp. 1257–1264, 2007.

[42] D. Daniel, H. Lee, and S. Sebastian, "Algorithms for Non-negative Matrix Factorization," in *Proceedings of the Advances in Neural Information Processing Systems (NIPS'00)*, pp. 556–562, 2000.

[43] X. Luo, M. Zhou, Y. Xia, and Q. Zhu, "An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1273–1284, 2014.

[44] A. Hernando, J. Bobadilla, and F. Ortega, "A non negative matrix factorization for collaborative filtering recommender systems based on a Bayesian probabilistic model," *Knowledge-Based Systems*, vol. 97, pp. 188–202, 2016.

[45] H. Shin, H. R. Roth, M. Gao et al., "Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning," *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1285–1298, 2016.

[46] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *Proceedings of the 27th IEEE Conference on Computer Vision and Pattern Recognition (CVPR '14)*, pp. 1717–1724, IEEE, Columbus, OH, USA, 2015.

[47] H. Chen, Q. Dou, D. Ni et al., "Automatic fetal ultrasound standard plane detection using knowledge transferred recurrent neural networks," in *Proceedings of the International Conference*

On Medical Image Computing & Computer Assisted Intervention, pp. 507–514, 2015.

[48] H.-T. Cheng, K. Levent, H. Jeremiah et al., “Wide & Deep Learning for Recommender Systems,” in *Proceedings of the1st Workshop on Deep Learning for Recommender Systems*, pp. 7–20, 2016.

[49] Y. Shan, T. R. Hoens, J. Jiao, H. Wang, D. Yu, and J. C. Mao, “Deep crossing: Web-scale modeling without manually crafted combinatorial features,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2016*, pp. 255–262, USA, August 2016.

[50] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. Chua, “Neural Collaborative Filtering,” in *Proceedings of the 26th International Conference*, pp. 173–182, Perth, Australia, April 2017.