HSA-Net: Hidden-State-Aware Networks for High-Precision QoS Prediction

Ziliang Wang, Xiaohong Zhang, Meng Yan, Ling Xu and Dan Yang

Abstract—The high-precision QoS (quality of service) prediction is based on the comprehensive perception of state information of users and services. However, the current QoS prediction approaches have limited accuracy, for most state information of users and services (i.e. network speed, latency, network type, and more) are hidden due to privacy protection. Therefore, this paper proposes a hidden-state-aware network (HSA-Net) that includes three steps called hidden state initialization, hidden state perception, and QoS prediction. A hidden state initialization approach is developed first based on the latent dirichlet allocation (LDA). After that, a hidden-state perception approach is proposed to abstract the initialized hidden state by fusing the known information (e.g. service ID and user location). The perception approach consists of four hidden-state perception (HSP) modes (i.e. known mode, object mode, hybrid mode and overall mode) implemented to generate explainable and fused features through four adaptive convolutional kernels. Finally, the relationship between the fused features and the QoS is discovered through a fully connected network to complete the high-precision QoS prediction process. The proposed HSA-Net is evaluated on two real-world datasets. According to the results, the HSA-Net's mean absolute error (MAE) index reduced by 3.67% and 28.84%, whereas the root mean squared error (RMSE) index decreased by 3.07% and 7.14% compared with ten baselines on average in the two datasets.

Index Terms—Service recommendation, Convolutional neural network, QoS prediction, Hidden states, Feature fusion.

1 Introduction

With the growing number of Web services, it is essential to determine how to recommend high-quality services to users. The main idea of service recommendation is to automatically predict the QoS values of the Web services that the user does not invoke and then recommend the most suitable Web services to the user through these predicted QoS values. Therefore, QoS prediction is a key step in service recommendation. QoS (e.g. response time and throughput) [1] [2] is considered to be an important criterion for evaluating the quality of Web services [3] [4]. In fact, reliable QoS data can effectively recommend optimal services to potential users. A recommended low-quality service (e.g., no response for a long time) may result in the loss of potential users. For example, service 0-5 is a public service used by a large number of users. The predicted response times between user 0 and five services were reported 5.982, 2.13, 0.854, 0.693, 0.866, and 1.833 seconds. We will give priority to recommend high-quality services 3 and 4 to the user 0. Therefore, it is important to predict the unknown QoS data based on historical data. This area has attracted a great deal of research interest [5] [6] [7].

Many challenges limit the accuracy of QoS prediction. For example, data sparsity is a major challenge in QoS prediction. Although there is a large number of services, users have access to a limited number of services which makes the user-service matrix sparse. According to the research literature, a common approach to the reliable QoS prediction is collaborative filtering (CF) [8] [9] [10]. In fact, CF is mainly based on similarity calculation performed by similar users and similar services to predict unknown QoS values. Due to the simplicity and availability of CF, it is considered a widely used QoS prediction approach. However, the sparsity and imbalance of QoS data make CF suffer from the limited prediction accuracy and weak scalability [11].

To overcome CF limitations, researchers proposed integrating deep learning approaches with the CF technology to improve the QoS prediction accuracy [12] [13] [14]. Among the existing approaches based on deep learning, obtaining as much information as possible is still the core idea for accuracy improvement. For example, Zhang et al. used location information as features and considered the user–service distance correlation to realize the QoS prediction [14]. The deep learning technology improves the prediction accuracy to a certain extent but requires effective information known as prediction conditions, such as location and temporal information. According to the previous studies, the QoS is often closely related to a user's state of the user and service itself when the service is used [15] [16]. For example, a user's network speed and the service load rate will directly affect the QoS [17]. Compared with algorithms that do not consider known information, the models that consider more information can lead to more accurate QoS predictions.

- *This work was supported in part by the National Key Research and Development Project of China (No. 2018YFB2101200), Special Funds for the Central Government to Guide Local Scientific and Technological Development (No. YDZX20195000004725), National Natural Science Foundation of China (No. 61772093), the Key Project of Technology Innovation and Application Development of Chongqing under (No. cstc2019jscx-mbdxX0020), the Chongqing Science and Technology Plan Projects (No. cstc2018jszx-cyztzxX0037), and the Chongqing Technology Innovation and Application Development Project (No. cstc2019jszx- mbdxX0064).*
- *Ziliang Wang, Xiaohong Zhang, Meng Yan, Ling Xu, Dan Yang are with Key Laboratory of Dependable Service Computing in Cyber Physical Society (Chongqing University), Ministry of Education, China and School of Big Data and Software Engineering, Chongqing University, Chongqing 401331, China.*
- *Xiaohong Zhang and Meng Yan are the corresponding authors. E-mail: xhongz,mengy@cqu.edu.cn*

For example, *Chen et al.* [18] further improve improved the accuracy of prediction through an extended version of the approach proposed by Wu *et al.*. [19] to embed the service location and the user location information in vectors. However, most current methods benefit from only the existing information and rarely consider the latent relationship between different information. In summary, there are still two challenges to the high-precision QoS prediction:

1) Limited information: The known information is limited, for it is difficult to collect much state information on services and users due to cost and privacy issues.
2) Independent features: Most studies use the known information on the QoS as independent features. In fact, just a few studies analyzed how to determine the relationship between different information on the QoS.

To tackle the above challenges, this paper proposes a hidden-state-aware networks (HSA-Net), which is mainly based on the idea of initializing the hidden-state information on users and services in the first step. As a result, it is possible to overcome the challenge of limited information. In the second step, the information is divided into four categories, i.e. known user information, hidden-state information of users, known service information, and hidden-state information of services. The perception approach is then proposed. It includes four hidden-state perception (HSP) modes to obtain the fused features by abstracting and fusing different information. Each HSP mode achieves a comprehensive perception of information based on different classifications through an adaptive convolution kernel. This approach can overcome the challenge of independent features and enhance the compatibility to different datasets. In the third step, fully connected networks are employed to learn the relationship between the fused features and the QoS in order to achieve the high-precision QoS prediction.

In summary, the main contributions of this paper are as follows:

a) We proposed a novel approach based on Hidden State Aware Network (HSA-Net) with three steps, i.e., hidden state initialization, hidden state perception, and QoS prediction. HSA-Net provided a novel and effective solution for addressing the data incompleteness problem in the field of QoS prediction.
b) A hidden-state perception approach is also proposed. It includes four HSP modes (i.e. known mode, object mode, hybrid mode, and overall mode) based on different convolutional kernels. Such modes encode the information of a specific category to obtain fused features (i.e. known features, object features, hybrid features, and overall features) where the convolutional adaptive kernels are adjusted dramatically according to the amount of features.
c) A few large-scale experiments are conducted to evaluate the effectiveness of the proposed HSA-Net by comparison with ten state-of-the-art baselines on two real-world datasets. According to the experimental results, the HSA-Net outperforms all the baselines in terms of the QoS prediction precision [1].

The remainder of this paper is organized as follows. In Section 2, related work is described. Section 3 provides the description of the proposed HSA-Net. Section 4 discusses the study set includes datasets, evaluation criteria, and more. Section 5 presents the experimental results and analysis in detail. The last section, concludes the paper and introduces the direction of future work.

2 Related work

To predict QoS, two review studies have recently been presented in [20] and [21]. Roughly speaking, the QoS prediction approaches can be classified into two categories: collaborative filtering-based QoS prediction approaches and deep learning-based QoS prediction approaches.

2.1 Collaborative Filtering-Based QoS Prediction Approaches

In this approach, a User-Service QoS matrix is the basic data source. This paper uses the User-Service QoS matrix definition that is similar in most studies [22], [23], [24], [25], [26], [27], [28]. CF-based Qos prediction approaches can be divided into two categories: memory-based and model-based. Memory-based approaches consider response time, throughput, or location as a similarity measure between users or services. The approach includes user-based [29], service-based [30], and a combination of the user and service [31]. The key step is to perform similarity calculations on users or services. For example, Zheng *et al.* introduced a QoS prediction approach with a similarity computation approach of similar neighbors [32]. Model-based approaches, on the other hand, apply techniques like matrix factorization (MF). This is an effective technique to implement recommender systems [33] [34] [35] [36], which has been extensively utilized to implement the QoS predictors in recent years. Ren *et al.* proposed a support vector machine-based CF approach to found the services that be truly preferred by a user [37]. The advantage of the QoS prediction approach based on collaborative filtering is that little data is required, but due to the lack of information and data sparsity, satisfactory prediction accuracy is often not achieved by a related approach.

To further improve the accuracy of collaborative filtering, many researchers have begun to focus on additional information (such as time and locations). For example, Chen *et al.* considered the user's trust degree and location information for QoS prediction [38]. Tang *et al.* and Liu *et al.* used the locations of users and services to improve the accuracy of QoS prediction effectively [39] [40]. Among various models with additional information, latent factors (LF)-based QoS predictors are prevalent due to their scalability and high prediction accuracy [41] [42] [43] [44] [45]. Luo *et al.* proposed a highly accurate prediction approach for missing QoS data by ensembling nonnegative latent factor (NLF) models [46]. Wu *et al*. propose a posterior-neighborhood-regularized latent factors (PLF) model for highly accurate

1. Our replication package:https://github.com/HotFrom/HSA-Net

QoS prediction [47]. The main idea of the latent factor (LF)-based approach is to improve the accuracy of prediction by discovering the hidden state of users and services. Similarly, Luo proposed a latent ability model (LAM) that can initialize the hidden capabilities of employees and services [48] but lacks an effective prediction approach. Although the additional information contributes to the accuracy of the CF method, the prediction accuracy still does not reach the expected because it can only learn the linear and low-dimensional relationship.

The hidden state initialization step is closely related to the latent factor-based approaches because it also considers the hidden information of user and service [15]. Compared with the MF-based approaches that only obtain a single user latent matrix and service latent matrix, the proposed initialization approach in this paper can set any number of hidden states. By doing this, HSA-Net can use more information to provide accurate predictions.

2.2 Deep Learning-Based QoS Prediction Approaches

In recent years, deep neural networks have emerged as an effective approach to recommended services. This approach, which our approach belongs to, captures the nonlinear relationship by considering the hidden state information of the object. He *et al.* are the first researchers who applied deep learning techniques in the recommended field [12]. Moreover, Kim *et al.* proposed the ConvMF model for QoS prediction that integrates the convolutional neural network (CNN) into probabilistic matrix factorization (PMF) to captures known information [49]. Similarly, deep learning approaches that use the timeslices information are also widely used in QoS prediction tasks. For example, Xiong *et al.* proposed a novel personalized LSTM based matrix factorization approach that can seize the dynamic latent representations of users and services and can be seasonably updated to deal with the new data [50].

Deep learning improves the accuracy of prediction to some extent but requires a lot of additional effective information as prediction conditions [51], [52], [53], [54], [55]. To further mining hidden information, Xiong *et al.* proposed a deep hybrid collaborative filtering approach for service recommendation that can find the complex invocation relations between mashups and services by using a multilayer perceptron [56]. HSA-Net is inspired by other domains of the research base on deep learning methods using known information for QoS prediction [57]. The main difference between the HSA-Net and those methods is that HSA-Net uses more hidden information and further captures the latent relationship between different information rather than only the relationship between independent features and QoS. Consequently, the HSA-Net can mitigate the challenges of independent features for QoS prediction.

3 PROPOSED APPROACH

The architecture of HSA-Net is shown in Figure 1, which includes three steps: hidden state initialization, hidden state perception, and QoS prediction:

Hidden State Initialization: An LDA-based hidden state initialization algorithm to initialize M hidden states of users and services: (p_1...p_m) and (q_1...q_m).

Hidden State Perception: The hidden state perception step includes four different HSP modes to capture the impact of different information. We abstract and fuse (p_1...p_m), (q_1...q_m) and known information of users and services (K_{u_1}...K_{u_o}, K_{s_1}...K_{s_o}) through different convolution operations to obtain four fusion features. Furthermore the size of the convolution kernel of each HSP mode can be automatically adapted according to the number of the known information (O) from the dataset and the number of hidden information (M).

QoS Prediction: The QoS prediction step captures the relationship between the four fusion features of the second step and QoS through a fully connected neural network to achieve high-precision QoS prediction.

3.1 Hidden State Initialization

Since the original information from the QoS dataset contains little information, we use a hidden state initialization algorithm to initialize M hidden states of user and M hidden states of service closely related to response time. The response time is a complex variable that is affected by various factors of the user and the service itself. In most cases, we cannot access sufficient information such as user network speed and service load to predict response time; hence response time is modeled as a mixture of hidden state information and known information.

Figure 2 describes how the hidden state variables interact with the observation variables to learn the distribution of the hidden state. We provided the definition of notations in Table 1 and the hidden state initialization steps in Algorithm 1.

3.1.1 Parameter definition

In this section, four probability distributions are first defined and how to calculate them is then discussed.

Definition 1: Let $L = \langle U, S, T \rangle$ denote the QoS records of the form (u_i,s_i,t_i), $u_i \in U$, $s_i \in S$, $t_i \in T$, $1 \leq i \leq n$. U is the user identifier. S is the service identifier, and T is the response time for user U to access service S. Let parameter M denote the number of hidden state type. The set P_i=$\{p_k(1 \leq k \leq M, 1 \leq i \leq n_u)\}$ contains m hidden state that are contained by the user i. The set Q_i=$\{q_k(1 \leq k \leq M, 1 \leq i \leq n_s)\}$ contains M hidden state that are contained by the service i. n, n_u, n_s are the number of records, the number of users and the number of services, respectively.

The matrix $\beta_u = \left\{\beta_{u\{i,j\}}\right\}$ is used to denote the probability for assigning all M hidden states to all n_u users $(1 \leq i \leq M, 1 \leq j \leq n_u)$ and the matrix $\beta_s = \left\{\beta_{s\{i,j\}}\right\}$ denotes the probability for assigning all M hidden states to all n_s services. We initialize β_u and β_s with the standard normal distribution.

Definition 2: The Θ_s is the frequency of various hidden state assignments on services. $\Theta_{s\{i\}}$ is the fraction of distinct services contains hidden state i, satisfying $\Sigma_{i=1}^{m}\Theta_{s\{i\}} = 1$. Dirichlet Prior is the smoothing distribution function to guarantee that the element of the hidden state would not be too small or too big. The Θ_s was defined the prior distribution that its probability density function by Dirichlet prior:

$$\Theta_s \sim Dirichlet(), \Theta_{s_i} = 1/M, 1 \leq i \leq n_s \quad (1)$$

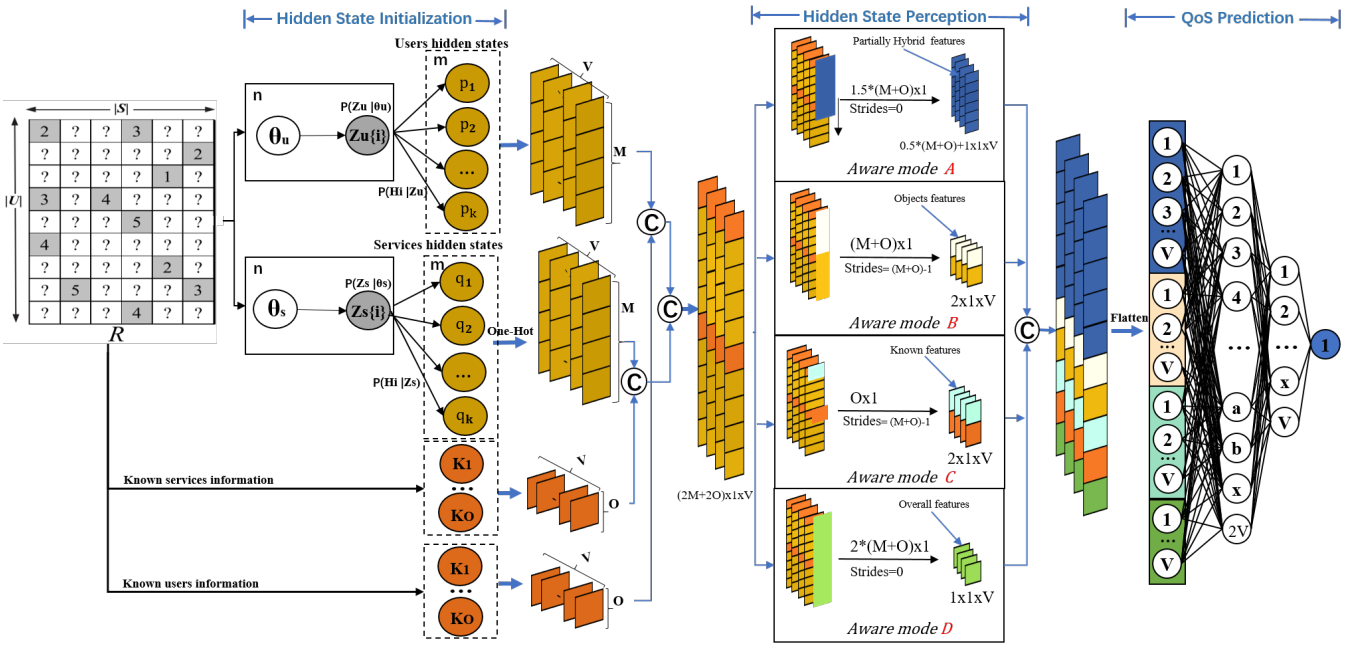


Fig. 1: The framework of HSA-Net. Step I: HSA-Net initializes M hidden states information of users and M hidden states information of services. Then $2 \times M$ hidden state information and $2 \times$O known information are encoded into V-dimensional vector by Ont-hot. Step II: Four HSP modes generate four interpretable fusion features through four different convolution operations. Step III: HSA-Net uses a three-layer fully connected network to achieve high-precision QoS prediction.

Similarly, we define Θ_u by using the same Dirichlet Prior with the same prior distribution:

$$\Theta_u \sim Dirichlet(), \Theta_{u_i} = 1/M, 1 \leq i \leq n_u \tag{2}$$

We consider the following sampling process: the hidden state is a sample from the state set of size m, according to the frequency Θ_u. The probability of the hidden states contained by u_i is exactly Z_u. Formally:

$$Z_u|\Theta_u \sim Discrete(\Theta_u) \tag{3}$$

Similarly, the probability of the hidden state contained by s_i is exactly Z_s. Formally:

$$Z_s|\Theta_s \sim Discrete(\Theta_s) \tag{4}$$

Definition 3: For each record, we consider the probability of assigning hidden state Z_u to the user U_i as well as the factors Z_s to the service S_j. The probability β_u is defined as the parameter for modeling the relation between the hidden state Z_u and u_i. We have:

$$u_i|Z_u, \beta_u \sim Discrete(\beta_{u\{Zu\}}) \tag{5}$$

In a similar fashion, we have:

$$s_i|Z_s, \beta_s \sim Discrete(\beta_{s\{Zs\}}) \tag{6}$$

as show in Figure 2, β_u and β_s have size $n_u \times M$ and $n_s \times M$, respectively.

Definition 4: The C_u is the complexity factor of the user representing user's sophistication factor on response time. It is a vector of size n_u. In this case, it represents the complexity factor of all users and initialized to $C_{u_i} = 10$, $1 \leq i \leq n_u$. Identically, C_s is the complexity factor of service represents the service's sophistication factor on response time. It represents the complexity factor of all services and initialized to $C_{s_j} = 10$, $1 \leq i \leq n_s$.

TABLE 1: Notations and Descriptions

Notations	Descriptions
O	The number of known information.
M	The number of hidden states.
N_s	The number of services.
N_u	The number of users.
N	Number of records.
Θ_s	$\Theta_{s\{i\}}$ denotes the probability of existence hidden state i for all services.
Θ_u	$\Theta_{u\{i\}}$ denotes the probability of existence hidden state i for all users.
β_s	$\beta_{s\{i\}}$ denotes the distribution of hidden states owned by service i, $\beta_{s\{i,j\}}$ denotes the service i's hidden state j.
β_u	$\beta_{u\{i\}}$ denotes the distribution of hidden states owned by user i, $\beta_{u\{i,j\}}$ denotes the user i's hidden state j.

3.1.2 Response time sampling

The most important step is sampling response time t_i given the i QoS record (u_i, s_i, t_i). t_i as the object of observation is affected by different factors of users and services. For each record, the conditional probability is used to represent its distribution:

$$t_i|Z_u, Z_s, C_u, C_s, w \sim \Phi(t_i; \lambda_{i,j,k}) \tag{7}$$

where function Φ is exponential distribution with parameter λ regarding response time:

$$\Phi(t_i, \lambda_{i,j,k}) = \lambda_{i,j,k} e^{(-t_i * \lambda_{i,j,k})} \tag{8}$$

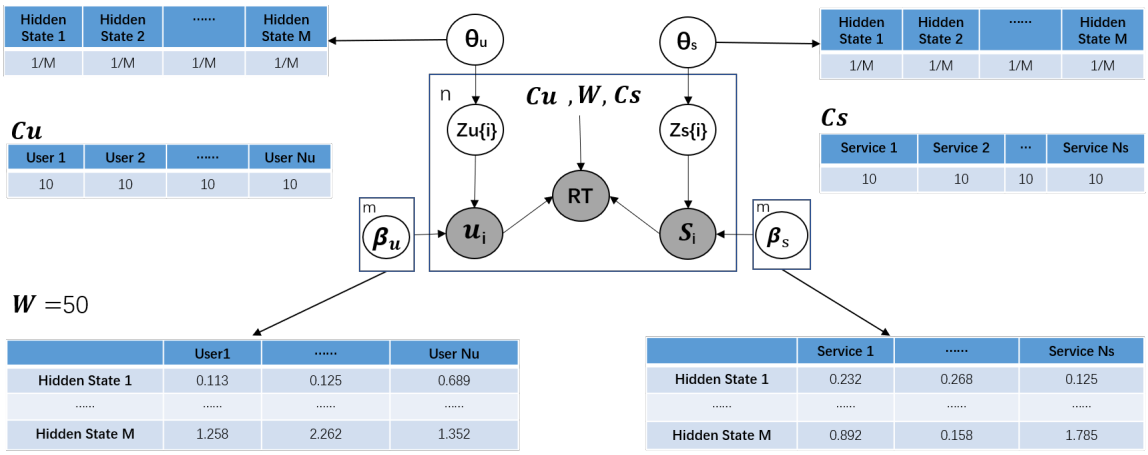


Fig. 2: The graphical model of hidden state initialization step. Θ_u is the hidden state probability distribution for all users. β_u is the hidden state probability distribution of each user. Θ_s is the hidden state probability distribution for all services. β_s is the hidden state probability distribution of each service.

where

$$\lambda_{i,j,k}^{-1} = \begin{cases} Cu_j Cs_k & \text{if } t_i < \eta \\ Cu_j Cs_k w & \text{else} \end{cases} \tag{9}$$

where j denotes the service and k denotes the user in the ith record. w is a penalty coefficient, the initial value is 50. The range of response time is 0-20 second. According to the exponential distribution in Eq. 8, the records with lower response times will get higher feedback. When the response time is close to 0 second, the feedback value is close to 1, and when the response time is close to 20 second, the feedback value is smaller. The significance of the penalty coefficient is to increase the score gap between the record with a response time of 5-20 second and the record with a response time of 0-5 second. According to the exponential distribution, a record of 5-20 will get a lower score. Because in the QoS prediction problem, we hope to successfully predict records with a too-long response time, which will reduce the recommendation of these services to users. The initial value of the complex coefficient does not have any meaning, and the purpose is to give an initial gradient to the gradient descent algorithm. $\eta$is a constant. When the response time is less than η seconds, the penalty does not be imposed.

3.1.3 Parameter estimation

The $\Gamma(\Theta_u, \Theta_s, \beta_u, \beta_s, w, C_u, C_s)$ is a set of parameters in our hidden states initialization algorithm, which must be learned. In this place, we choose maximum a posteriori (MAP) to complete parameter estimation. In this step, we use the initialization values of the parameters as the prior distribution and use the QoS record Q as the observation object to obtain the posterior distribution $P(\Gamma|Q)$. The specific calculation approach is as follows:

$$P(\Gamma|Q) = Z \prod_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{m} \tau_{i,j,k} \Phi(s_{ij}; \lambda_{i,j,k}) \tag{10}$$

where

$$\tau_{i,j,k} = \beta_{u\{j,u_i\}} \beta_{s\{j,s_i\}} \Theta_{u\{j\}} \Theta_{s\{k\}} \frac{1}{B(M)} \prod_{i=1}^{m} (\Theta_{s\{i\}} \Theta_{s\{i\}})^{M-1} \tag{11}$$

Z is a constant and keeps the sum of all probabilities equal to 1. In the above announcement, u_i and s_i represent the service number of the user recorded in the i-th. To solve this MAP problem, the EM (expectation-maximization) algorithm is used to update the set of parameters $\{\Theta_u, \Theta_s, \beta_u, \beta_s\}$, and the set $\{C_u, C_s, w\}$ are updated with the gradient descent (GD) algorithm at the same time. In the case that the objective function and the data are determined, the solution space is also determined. Different initial settings may have different initial gradients, but they will produce the same final value.

In Algorithm 1, E-step refers to the expectation step. We calculate the probability of hidden states by Bayes theorem:

$$\begin{aligned} F_{i,j,k}^{t} &= P(u_i, t_k | u_i, s_i, t_i, \Gamma) \\ &= \tau_{i,j,k} \Phi(s_{ij}; \lambda_{i,j,k}) \end{aligned} \tag{12}$$

M-step refers to the maximization step. We update parameters Γ by maximizing the condition expectation.

$$\begin{aligned} T(\Gamma|\Gamma^t) &= \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{m} F_{i,j,k}^{t} log P(\Gamma|Q) \\ &= \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{m} F_{i,j,k}^{t} log(\tau_{i,j,k} \Phi(s_{ij}; \lambda_{i,j,k})) \end{aligned} \tag{13}$$

Then we have:

$$\begin{aligned} \Theta_s^{t+1} &= arg_{\Theta_u} max T(\Gamma|\Gamma^t) \\ &= arg_{\Theta_u} max \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{m} T(log\Theta_{s\{j\}} + \\ &\quad (m-1) \sum_{i=1}^{m} log(\Theta_{s\{i\}})) \\ &= \frac{(m-1)n + \sum_{i'=1}^{n} \sum_{k=1}^{m} F_{i',i,k}^{t}}{(m(m-1)+1)n} \end{aligned} \tag{14}$$

and

$$\begin{aligned} \beta_{u_{j,p}}^{t+1} &= arg_{\beta_u} max T(\Gamma|\Gamma^t) \\ &= \sum_{i=1}^{n} \sum_{k=1}^{m} F_{i,j,k}^{t} I(u_i = p) \end{aligned} \tag{15}$$

Algorithm 1 Hidden state initialization algorithm

Input: Q: (u_i,s_i,t_i) variables: n_u is the number of users, n_s is the number of services. m is the number of hidden state and n is the number of records.

Output: (β_u, β_s)

//E-STEP
1: **while** $L^{t+1}/L^t > \gamma$ **do**
2: **for** $i = 1$ to n **do**
3: **for** $j = 1$ to m **do**
4: **for** $k = 1$ to m **do**
5: $F^t_{i,j,k} = \tau_{i,j,k}\Phi(s_{ij}; \lambda_{i,j,k})$
6: **end for**
7: **end for**
8: **end for**
//M-STEP
9: **for** $i = 1$ to m **do**
10: $\Theta^{t+1}_u = \frac{(m-1)n+\sum_{i'=1}^{n}\sum_{k=1}^{m} F^t_{i',i,k}}{(m(m-1)+1)n}$
11: $\Theta^{t+1}_s = \frac{(m-1)n+\sum_{i'=1}^{n}\sum_{j=1}^{m} F^t_{i',j,i}}{(m(m-1)+1)n}$
12: **for** $q = 1$ to n_u **do**
13: $\beta^{t+1}_{u\{i,q\}} = \sum_{i'=1}^{n}\sum_{k=1}^{m} F^t_{i',i,k} I(u_i = q)$
14: **end for**
15: **for** $q = 1$ to n_s **do**
16: $\beta^{t+1}_{s\{i,q\}} = \sum_{i'=1}^{n}\sum_{j=1}^{m} F^t_{i',j,i} I(s_i = q)$
17: **end for**
18: **end for**
//GD-STEP
19: **for** $q = 1$ to n_u **do**
20: $C^{t+1}_{u_i} = C^t_{u_i} + \varrho * \frac{\partial P(\Gamma|Q)^t}{\partial C^t_{u_i}}$
21: **end for**
22: **for** $q = 1$ to n_s **do**
23: $C^{t+1}_{s_i} = C^t_{s_i} + \varrho * \frac{\partial P(\Gamma|Q)^t}{\partial C^t_{s_i}}$
24: **end for**
25: $w^{t+1} = w^t + \varrho * \frac{\partial P(\Gamma|Q)^t}{\partial w^t}$
26: $L^{t+1} = P(\Gamma|Q)$
27: **end while**
return (β_u, β_s)

$$\begin{aligned}\beta^{t+1}_{s_{k,q}} &= arg_{\beta_s} max T(\Gamma|\Gamma^t) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{m} F^t_{i,j,k} I(s_i = q)\end{aligned} \tag{16}$$

$I(s_i = q)$ means that when $s_i = q$, the return value is 1, otherwise the value is 0. We supplement the steps of the GD algorithm in detail to update the parameters (C_u, C_s, w):

$$\frac{\partial P(\Gamma|Q)^t}{\partial C^t_{s_i}} = \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{m} F^t_{i,j,k}(-\frac{1}{C_{u_q}} + \frac{t_i}{C_{u_{(u_i)}} C^2_{s_q}}(I(j=k) + \frac{1}{w} I(j \neq k))))I(u_i = q) \tag{17}$$

for C_s:

$$\frac{\partial P(\Gamma|Q)^t}{\partial C^t_{s_i}} = \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{m} F^t_{i,j,k}(-\frac{1}{C_{s_q}} + \frac{t_i}{C_{s_{(s_i)}} C^2_{u_q}}(I(j=k) + \frac{1}{w} I(j \neq k))))I(s_i = q) \tag{18}$$

and the gradient for w:

$$\frac{\partial P(\Gamma|Q)^t}{\partial w} = \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{m} F^t_{i,j,k}(-\frac{1}{w} + \frac{t_i}{C_{u_{(u_i)}} C_{s_{(s_i)}} w})I(j \neq q) \tag{19}$$

The γ is a constant less than 0.01, which is used as the convergence condition of the algorithm. In algotithm1, line 1 and line 26 provides a way to calculate the termination condition. Finally, we use the results returned by Algorithm 1 as the hidden state of users and services.

3.2 Hidden State Perception

3.2.1 Input layer

To train the neural network, we input the user known information K_{u_i}, user's hidden states P_i, service known information K_{s_j}, and service's hidden states Q_j into the embedding layer of Keras, which can be regarded as a fully connected layer without bias term. The latent state initialization algorithm calculates the latent state of users and services, which is a floating-point value. To implement the embedding layer operation in the neural network, we expand each state value by 100 times and de-integer. In the sense of solely, each information goes through one-hot encoding that makes the feature to 1xV dimension vector so that the ith position is 1 and the other is 0. The mathematical expression of this step is as follows:

$$K^v_{u_i} = f_1(T_1^T u_i + b_1) \tag{20}$$

$$P^v_i = f_1(T_1^T pi + b_2) \tag{21}$$

$$K^v_{s_j} = f_1(H_1^T s_j + b_3) \tag{22}$$

$$Q^v_j = f_1(H_1^T q_j + b_4) \tag{23}$$

where T_1 and H_1 stand for the users and service's embedding weight matrix, respectively. For better feature combination, we combine the user known information vector with the user hidden state information vector to obtain the user feature vector. Similarly, we can get the service feature vector. Then, we concatenate them to get the input vector. The simplified step is as follows:

$$Input = \Phi(K_{u_i}, p_i, K_{s_j}, q_j, U, S, axis = 1) \tag{24}$$

where Φ is a mergence operation by Keras, and axis=1 indicates that the operation is performed line by line.

3.2.2 HSP modes

We designed four HSP modes by considering the meaning of hidden information in different situations through real experience. In the section, we described the setting and meaning of HSP modes in detail. We divide all QoS information into four categories: user known information ($K_{u_1}...K_{u_o}$), service known information ($K_{s_1}...K_{s_o}$), user hidden states information ($p_1...p_m$), and service hidden states information ($q_1...q_m$). To select a more reasonable combination of features according to the degree of contribution to the predicted response time, different convolution kernels have been used to calculate the impact between different information to generate specific fused features.

The hidden state perception step in the network can be expressed as:

$$F_m = \Phi(f_1, f_2, f_3, f_4) \tag{25}$$

F_m is the output of HSP modes, f_1, f_2, f_3 and f_4 are the fused features obtained after convolution operation with different settings, which are described as follows:

$$f_i = w_i * F + b_i \tag{26}$$

Figure 2 shows the HSP mode proposed in this paper. The following details the meaning and output of each HSP mode:

The hybrid mode solves a defect of the hidden state initialization algorithm: the user and the service initialize the same number of hidden states. However, service providers tend to have more stable networks, and their hidden states may not be as large as users. It is necessary to consider the relationship between the different number of hidden states of the service and other states.

Hybrid Mode: The convolution kernel of hybrid mode is $1.5 * (M + N) \times 1$ and the $strides = 0$. Each convolution operation of hybrid mode is focused on different K user features and J service features. At the same time, K is decreasing and J is increasing during each convolution. $0.5 * (M + O) < K < M + O$, $0.5 * (M + O) < j < M + O$, $j + k = 1.5 * (M + O)$.

Hybrid Features: The hybrid mode produces $0.5 * (M + N) + 1$ hybrid features. Each hybrid feature focuses on different user features and service features. The reason for this is that the quality of service may be affected by different amounts of service information or user information rather than the same amount of users and service information. The effect of hybrid features is to explore the impact of fused features from different numbers of service features and user features on service quality, respectively. For example, the interaction between all user information and a small amount of service information is considered in the first hybrid feature.

The object mode considers the states of users or services alone by generating object features. We can focus on the state of the object itself to judge the quality of service. This is because there are services and users with poor states. In Figure 3, the response time of many users to call certain services is about 12s, which is caused by the poor states of the objects of these services.

Object Mode: The convolution kernel of object mode is $(M + O) \times 1$ and the $strides = (M + O) - 1$. The object mode performs twice convolution operations. The first convolution operation fuses all user features ($K_{u_{1:o}}$ and $p_{1:m}$) to obtain user object features. The second convolution operation fuse all service features ($K_{s_{1:o}}$ and $q_{1:m}$) to obtain the service object feature.

Objects Features: The object mode produces two different object features: user object features and service object features. The effect of object feature is to explore the impact of user and service information on service quality respectively. For example, if user x cannot access a certain service due to network states, the states of his state features will have a decisive effect on QoS prediction.

In QoS prediction, the known information is location information, and this module will play a role similar to the distance module in other studies.

Known Mode: The convolution kernel of known mode is $O \times 1$ and the $strides = (M + O) - 1$. The known mode performs twice convolution operations. The first convolution operation fuses all user known features (K_{u_o}) to obtain user known fused features. The second convolution operation fuse all the service known features (K_{s_o}) to obtain the service known fused feature.

Known Features: The known mode produces two different known features: users known features and services known features. Light blue indicates that the known features of the user. Orange indicates the known fused features of the service. The effect of known features is to explore the impact of known information on service quality respectively. For example, the known features in the dataset D2 may be easier to pay attention to the latent relationship (distance) because the known information of D2 is location information.

The overall model calculates the standard value through all the features. The standard value can also be directly regarded as the performance of the final feature—response time. We will use other fusion features that from the three HSP modes to calibrate this value further. **Overall Mode**: The convolution kernel of overall mode is $2 * (M + O)$ and the $strides = 0$. The overall mode performs once convolution operations. This convolution operation fuses all information to obtain the overall fused feature.

Overall Features: Green represents the overall fused feature of all information. The overall fused feature focuses on the impact of all information on service quality.

M is the number of hidden states. O is the number of known information of the user or service given by the source dataset. When $O = 1$, the known information includes identification(ID). The known information includes ID, location information, and autonomous system (AS) information when $O = 3$.

3.3 QoS Prediction

In this step, a fully connected neural network is used to achieve high-precision QoS prediction. The network is used to process the input vector from the hidden state perception step for learning the high-dimensional nonlinear relationship and complete the QoS prediction. First of all, we reshape the dimensions of the fused feature through the flatten layer of Keras to meet the input required of the fully connected layer. Secondly, we use the Batch Normalizing (Bn) layer to prevent overfitting. Bn layer can also speed up network training and prevent the gradient from disappearing. Finally, we chose ReLu as the activation function for the first two layers because it can speed up the convergence of the model. To get an accurate prediction value, we chose the "Linear" function as the activation function of the last layer. The prediction step is defined as follows:

$$FC^{2*V} = f_3(W_3^T x + b_3) \tag{27}$$

$$FC^{V} = f_2(W_2^T x + b_2) \tag{28}$$

$$\hat{Q}_{u,s} = FC^1 = f_1(W_1^T x + b_1) \tag{29}$$

where $\hat{Q}_{u,s}$ is the predictive QoS value that user i use service j. V is the number of neurons. The FC^v means that v neurons are set in this layer. W_d is the corresponding weight matrix and b_d is the bias term of the d-th layer.

3.3.1 Loss function

We can roughly classify the mainstream loss functions into two types: loss functions for regression problems and loss functions for classification problems. QoS prediction problem is a regression problem. The commonly used loss functions for the regression problem are mean square error function (MSE), mean absolute error function (MAE), and Huber function. In this paper, after a detailed comparison experiment, we chose the MAE function as the loss function, which is defined as follows:

$$Loss = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|_{abs} \tag{30}$$

In Section 4, we discuss in detail the reasons for choosing MAE function.

3.3.2 Complexity analysis

In this section, we analyze the time complexity of the proposed HSA-Net.

In algorithm 1, the time complexity of lines 2-8 is $O(n \times m \times m)$, where n represents the number of records and m is the number of hidden states($m \ll n$). The time complexity of lines 9-11 is $O(1)$ because m is a constant. The time complexity of lines 12-17 is $O(m \times n_u)$ +$O(m \times n_s)$, where n_u and n_s represent the number of users and services, respectively. The time complexity of lines 19-26 is $O(1)$. Therefore, the time complexity of the step (lines 2–26) is $O(n \times K)$. Then the complexity of algorithm 1 is given by

$$\begin{aligned} T &= O(n \times m \times m + 1 + m \times n_u + m \times n_s + 1) \\ &\approx \Theta(K \times n) \end{aligned} \tag{31}$$

In the HSP modes, the time complexity of Eq.(20) - Eq.(26) is O(n). Then we perform a total of four one-dimensional convolution operations. Therefore, the time complexity of the forward propagation process is $O(n \times k \times 4)$ in Eq.(28).

In the QoS prediction step, the time complexity of Eq.(32) is $O(n \times k)$. The time complexity of Eq.(29) - Eq.(31) is $O(n \times k)$ because the time complexity of the forward propagation is the same as the backpropagation.

Overall, the time complexity of HSA-Net is $O(n \times k)$.

4 STUDY SETUP

This section provides a detailed introduction to the experimental setup, including an introduction to the datasets and the evaluation metrics that we used.

TABLE 2: The example of data

The Dateset D2						
The Dateset D1			Other known information [14]			
User ID	Service ID	RT	IP Address	User-AS	Service-Country	Service-AS
0	0	5.982	29	115	70	118
1	0	2.13	16	94	70	118
2	0	0.854	16	94	70	118
...	...	...	...	...	...	...
239	3945	0.132	27	101	70	43
...	...	...	...	...	...	...
338	64	0.44	29	87	1	43

TABLE 3: Properties of all the designed test cases.

No.	Density	Train:Test	TrainingData	TestingData
D1.1	0.05	5%:95%	98,721	1,774,099
D1.2	0.10	10%:90%	197,440	1,676,166
D1.3	0.15	15%:85%	296,182	1,577,571
D1.4	0.20	20%:80%	394,926	1,478,867
D2.1	0.05	5%:95%	98,721	1,774,099
D2.2	0.10	10%:90%	197,440	1,676,166
D2.3	0.15	15%:85%	296,182	1,577,571
D2.4	0.20	20%:80%	394,926	1,478,867

4.1 Datasets

We conduct the experiments on two benchmark datasets, which are Web service QoS data collected by the WS-Dream system. This is a large-scale real-world Web services dataset collected and maintained by Zheng *et al.* [24], containing 1,974,675 QoS values of Web services collected from 339 users on 5,825 services with location information about users and services. In this paper, the QoS dataset acts in the form of a user-service matrix, where the row-index means the user identifier, the column-index expresses the service identifier, and each value in the matrix is the Corresponding response time (RT). The benchmark datasets are shown in Table 2:

1) D1: The known information includes user ID, service ID, and response time (RT).

2) D2: The known information includes user ID, service ID, response time, city information, and AS (autonomous system) information. Among them, the city information and AS information come from [14].

The city information refers to the city where the user and service are located, for example, Japan, India. The AS information refers to the network service machine of the user and the service network provider, for example, AS17 and AS173. Most of the time, a region has multiple AS providers. Different regions and different ASs often have different network environments, and different environments often will lead to different quality of service. HSA-Net uses location code and AS code to protect user privacy instead of detailed longitude and latitude information.

For both datasets, cross-validation techniques are utilized to obtain more objective results. Due to the massive amount of QoS data, it is often impossible to collect all the data in reality. We simulate the real environment by artificially setting sparse data. We design different test cases for validating the effectiveness under different data densities, as shown in Table 3. The Train: Test means the ratio of training data to testing data; Among them, D1.1 (D2.1) represents the first dataset that the first dataset is divided according to the ratio of $5\% : 95\%$. And D1.2 (D2.2), D1.3 (D2.3), D1.4 (D2.4) represent the other three density divisions. The $5\% : 95\%$ means that 5% of original data is chosen randomly as training data and the remaining data is used as the test data.

TABLE 4: DESCRIPTIONS OF ALL THE COMPARISON MODELS.

Approach	Description
	CF-Based Approach
IPCC [58]	IPCC(Item Pearson Correlation Coefficient) borrows the thinking from item-based collaborative filtering algorithm to compute the similarity between two services.
UPCC [30]	This approach used similar user behavior information for QoS prediction.
LACF [39]	This approach uses both locations information of users and services for service recommendation.
LFM [43]	This approach used the QoS matrix to learn latent features of users and services for predicts QoS.
PMF [59]	This approach used probabilistic factors into MF for service recommendation.
DCALF [60]	this approach proposes a data-characteristic-aware hidden state model to implement highly accurate QoS predictions, which is a state-of-art CF-based approach.
	Deep learning-Based Approach
JCF [57]	This approach use the CNN learns the neighbors' features, forms a feature matrix for prediction QoS.
NCF [12]	This is an advanced deep learning approach that combines MLP and MF for QoS prediction.
AutoRec [61]	This paper puts forward a DNNs-based QoS predictor which is an autoencoder framework for CF and consists of I- AutoRec and U-AutoRec.
Cluster* AE [62]	This paper proposes a deep learning-based approach that combining a matrix factorization model based on a deep auto-encoder (DAE) and a clustering technique, which is the state-of-art Deep learning-based approach.
LDCF [14]	This is an advanced deep learning approach with location-aware for QoS prediction.

4.2 Evaluation Metrics

In the QoS prediction issue, the important criterion for evaluation is prediction accuracy. In most studies, two metrics are used to measure accuracy. The first metric is the mean absolute error function (MAE):

$$MAE = \frac{\left(\sum_{i,j} |r_{i,j} - \hat{r}_{i,j}|_{abs}\right)}{N} \tag{32}$$

where $r_{i,j}$ is the real value and $\hat{r}_{i,j}$ is the predicted value of QoS property, and N is the number of QoS records. The second metric is the root mean squared error (RMSE):

$$RMSE = \sqrt{\frac{\left(\sum_{i,j}(r_{i,j} - \hat{r}_{i,j})^2\right)}{N}} \tag{33}$$

The lower values of these two metrics mean the more accurate prediction we have.

5 EXPERIMENTS

In this section, we perform several experiments to compare the performance of the proposed approach against state-of-the-art baselines and to examine the effect of different parameters on the performance of the proposed approach.

5.1 Performance Comparison

Approach: To evaluate the performance of the proposed approach, we selected ten baselines (see Table 4) for comparison, including traditional algorithms and the latest research.

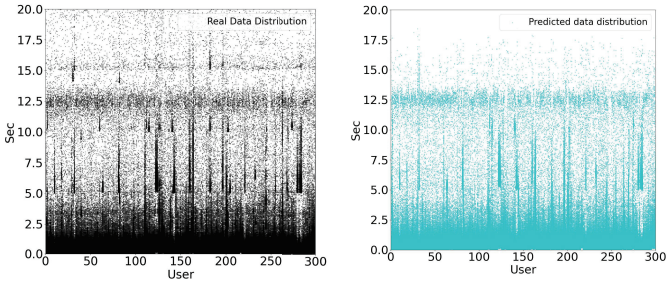


Fig. 3: Real and predicted QoS values for 300 users. Each dot represents a QoS value generated by a call to the service by that user. In this dataset, each user has 5000+ points on the Y-axis.

We conduct experiments with various data densities on two datasets D1 and D2. Since some baselines are not universal between different datasets, we choose six comparison baselines on two datasets, respectively.

From Tables 4, 5, we have the following observations:

(1) As shown in Table 4. The performance of LFM, DCALF, AutoRec, JCF, and PMF performance is similar. When the dataset D1 density was 0.05, the MAE of all methods exceeded 0.5. This indicates that the existing approach, which is based on CF and deep learning, has low predictive accuracy when the sparsity of features is high. Using the HSA-Net approach proposed in this paper, we have improved the MAE performance by 28.73% and the RMSE also dropped by 9.0% than the baseline model DCALF. In the case of other densities, the MAE performance of HSA-Net is improved by 28.28%, 28.93% and 29.42% respectively than the best performance of DCALF, and RMSE decreased by 6.34%, 6.1% and 7.14%, respectively.

(2) The experimental results on the dataset D2 is shown in Table 6. HSA-Net improves Cluster* AE by 4.1%, 2.5%, 5.4% and 2.7% in terms of MAE, by 2.5%, 3.1%, 2.8% and 2.7% in terms of RMSE among D2.1, D2.2, D2.3. D2.4 respectively.

(3) Experience shows that the Friedman Rank test is effective in validating the performance of multiple approaches on multiple datasets [63]. We perform the Friedman test [64] in MAE recorded respectively. From the Friedman Rank statistical results, we observe that HSA-Net has the lowest Friedman Rank values among all QoS predictors approaches, which means that HSA-Net has the best prediction accuracy among the tested approaches. In Figure 3, we visualized the experimental results to detect the prediction of the discrete value by HSA-Net. This shows the unbalanced distribution of the user's call records, and most of the call records are within 0-3 seconds. The predicted QoS distribution well demonstrates that the proposed method can effectively predict such data distribution.

Results: The experimental results are shown that the proposed approach has the smallest MAE and RMSE in all situations. This means that the proposed approach performs better than other baselines. The accuracy of the proposed approach has obvious advantages, especially when the information known in D1 is limited. Especially when known information is limited in D1, the accuracy of the proposed approach has obvious advantages.

TABLE 5: Experimental results in D1

Method	D1.1				D1.2				D1.3				D1.4				
	MAE	impro.%	RMSE	impro.%	MAE	impro.%	RMSE	impro.%	MAE	impro.%	RMSE	impro.%	MAE	impro.%	RMSE	impro.%	FriedmanRank
UPCC	0.769	52.27%	1.694	26.26%	0.720	54.58%	1.610	27.57%	0.652	52.91%	1.495	24.68%	0.592	50.16%	1.399	22.01%	0.68
IPCC	0.732	49.86%	1.634	23.56%	0.712	54.07%	1.543	24.43%	0.678	54.71%	1.410	20.14%	0.652	54.75%	1.310	16.71%	0.71
LFM	0.578	36.50%	1.501	16.78%	0.564	42.02%	1.320	11.66%	0.543	42.46%	1.271	11.40%	0.535	44.85%	1.226	11.01%	0.555
AutoRec	0.546	32.66%	1.373	9.03%	0.485	32.57%	1.264	7.75%	0.464	33.36%	1.210	6.94%	0.452	34.73%	1.189	8.24%	0.486
JCF	0.513	27.49%	1.332	6.23%	0.466	29.82%	1.250	6.72%	0.450	31.77%	1.185	4.97%	0.436	32.33%	1.180	7.5%	0.466
DCALF	0.512	28.32%	1.373	9.0%	0.454	27.97%	1.245	6.34%	0.434	29.26%	1.200	6.1%	0.424	30.42%	1.175	7.14%	0.455
HSA-Net	**0.367**	-	**1.249**	-	**0.327**	-	**1.166**	-	**0.307**	-	**1.126**	-	**0.295**	-	**1.091**	-	**0.324**

TABLE 6: Experimental results in D2

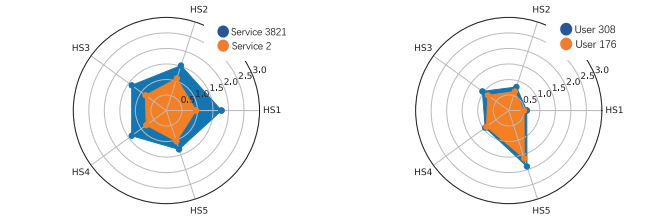
Method	D2.1				D2.2				D2.3				D2.4				
	MAE	impro.%	RMSE	impro.%	MAE	impro.%	RMSE	impro.%	MAE	impro.%	RMSE	impro.%	MAE	impro.%	RMSE	impro.%	FriedmanRank
UPCC	0.634	45.42%	1.377	12.27%	0.553	43.39%	1.311	12.97%	0.511	42.27%	1.258	12.71%	0.483	41.40%	1.220	12.29%	0.545
IPCC	0.635	45.51%	1.387	12.90%	0.584	46.40%	1.305	12.56%	0.507	41.81%	1.252	12.30%	0.454	37.66%	1.203	11.05%	0.545
PMF	0.568	40.04%	1.537	21.40%	0.487	35.72%	1.321	13.62%	0.451	34.58%	1.221	10.07%	0.433	34.64%	1.171	8.62%	0.484
LACF	0.682	49.26%	1.500	19.46%	0.650	51.84%	1.468	22.27%	0.610	51.63%	1.416	22.45%	0.582	51.37%	1.381	22.51%	0.631
NCF	0.440	21.36%	1.325	8.8%	0.385	18.70%	1.283	11.06%	0.372	20.69%	1.253	14.11%	0.362	21.82%	1.205	11.20%	0.389
LDCF	0.413	16.22%	1.310	7.78%	0.385	18.70%	1.255	9.08%	0.362	18.50%	1.204	10.35%	0.346	18.20%	1.185	10.59%	0.376
Cluster* AE	0.361	4.1%	1.240	2.5%	0.321	2.5%	1.178	3.1%	0.312	5.4%	1.130	2.8%	0.291	2.7%	1.114	3.9%	0.321
HSA-Net	**0.346**	-	**1.208**	-	**0.313**	-	**1.141**	-	**0.295**	-	**1.098**	-	**0.283**	-	**1.070**	-	**0.309**

5.2 The impact of the hidden state initialization

Approach: In this section, we evaluated the impact of the hidden state initialization step in the HSA-Net. We designed related comparative experiments on dataset D2. To evaluate the effectiveness of the hidden state initialization step, we define the following approaches:
a) LDCF: The baseline.
b) HSA-Net: The approach proposed in this paper, and M=5.
c) HSA-Net-HS: M=0.
d) HSA-Net-1: M=5, and hidden state values are set to 1 in HSA-Net.

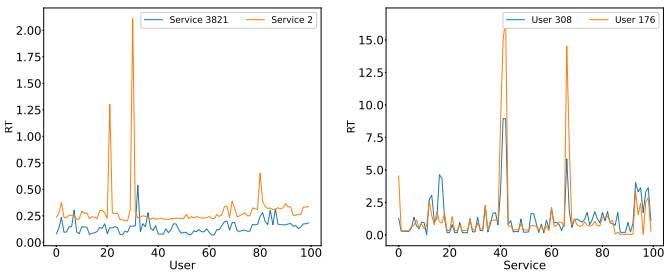
We evaluate the effectiveness of Algorithm 1 by sampling and analyzing the results. For example, we initialize the hidden states of two users and two services: service 3281 and service 2, user 308 and user 176. Figure 4a shows that service 3281 and user 308 have a higher value than service 2 and user 176 for all M hidden states (M=5), respectively. The blue solid polygon shows the state values of service 3281, which are around 1, much higher than the state values of service 2, whose ability values are around 0.5, as shown in the red-orange solid polygon. Next, we sample one hundred services and one hundred users (service 0 - service 99 and user 0 - user 99) from the QoS dataset [24]. Figure 5a illustrates the response time comparison of service 3821 and service 2, and Figure 5b illustrates the response time comparison of user 308 and user 176. We observed that the average response time of user 308 using service 0 - service 99 is 1.32s, and the average response time of user 176 is 1.42s. The average response time of service 3281 used by user 0 - user 99 is 0.143s and the average response time of service 2 is 0.298s.

Results:
(1) As shown in Figure 6, the prediction accuracy of HSA-Net is higher than HSA-Net-1 that without using the hidden state initialization step and its hidden state is initialized to 1.
(2) The prediction accuracy of HSA-Net is higher than that of HSA-Net-HS, without the hidden state. The hidden state initialization step can improve the accuracy of QoS prediction.
(3) Through case analysis, we find out that the user or



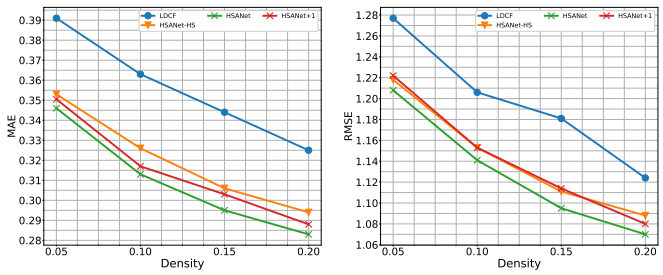
(a) hidden states of service 3821 and service 2 (b) hidden states of user 308 and user 176

Fig. 4: The users and services's hidden state and QoS.



(a) The QoS of service 3821 and service 2 (b) The QoS of user 308 and user 176

Fig. 5: The demo of hidden states and QoS.



(a) MAE of HSA-Net (b) RMSE of HSA-Net

Fig. 6: Results of approaches proposed in 5.2.

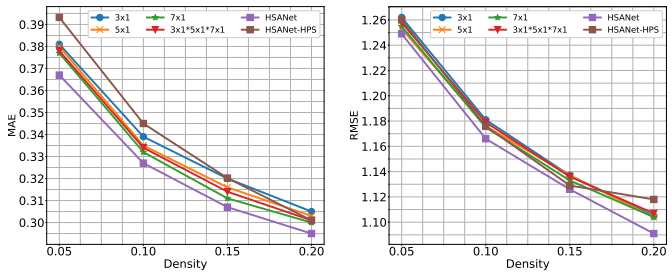


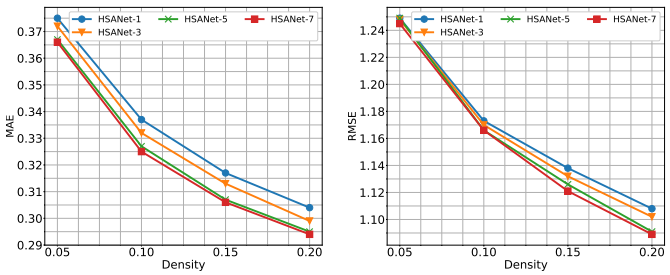
Fig. 7: Results of different convolution approaches proposed in 5.2.

service with a higher state value has a better average time and maximum state value than the lower user or service.

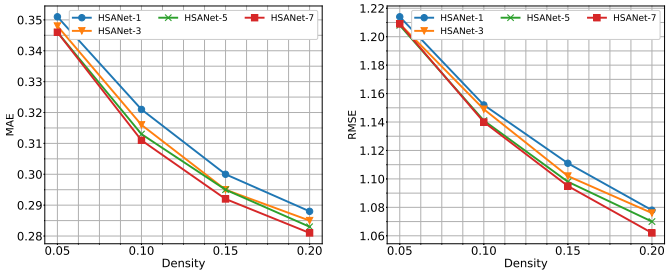
5.3 The impact of HSP modes

Approach: To evaluate the effectiveness of the HSP modes, we define the following four generic convolution approaches to replace the HSP modes.
a) 3x1: Only use 3x1 one-dimensional convolution kernel.
b) 5x1: Only use 5x1 one-dimensional convolution kernel.
c) 7x1: Only use 7x1 one-dimensional convolution kernel.
d) 3x1*5x1*7x1: Use three different one-dimensional convolutions: 3x1, 5x1, and 7x1, and connect the results through the concatenate layer.
e) HSANet: The four convolution kernels of HSP modes that we proposed above.
f) HSANet-HPS: The HSA-Net approach without HSP modes.

As shown in Figure 7, the HSA-Net(e) performance is superior to the others. Specifically, the MAE of HSA-Net(e) is 0.367 and its RMSE is 1.249 when density=0.05. As a comparison, the MAE of HSA-Net(f) is 0.393 and its RMSE is 1.26. The HSA-Net(f) without HSP modes have the worst prediction accuracy when the data density is low. HSA-Net(a), HSA-Net(b) and HSA-Net(c) has similar prediction accuracy. In the same way of using multiple convolution kernels, it can be seen that the prediction performance of HSA-Net(e) is much better than that of HSA-Net(d). **Results:** Compared with the general convolution approaches, HSP modes achieve higher prediction accuracy. These results show that HSP modes are helpful for improving the performance of HSA-Net.

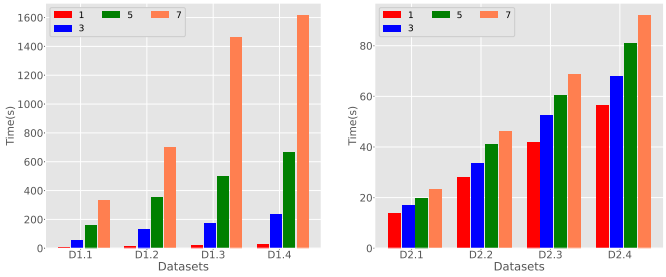


(a) MAE on the dataset D1 (b) RMSE on the dataset D1



(c) MAE on the dataset D2 (d) RMSE on the dataset D2

Fig. 8: The impact of M on the HSA-Net



(a) The each epoch's running time of algorithm 1 (b) The each epoch's running time of HSP modes

Fig. 9: The impact of M on the HSA-Net

5.4 Hyper-parameters Sensitivity Tests

1) The impact of the number of hidden state (M)

Approach: To evaluate the effectiveness of the number of hidden states (M), we set the M values for experiments to 1, 3, 5, and 7.
a) HSANet-1: In Algorithm 1, each user and service initializes only one type of latent information. In this setting, M=1.
b) HSANet-3: In Algorithm 1, each user and service initializes only three types of latent information. In this setting, M=3.
c) HSANet-5: In Algorithm 1, each user and service initializes only five types of latent information. In this setting, M=5.
d) HSANet-7: In Algorithm 1, each user and service initializes only seven types of latent information. In this setting, M=7.

Figure 8 shows that as the number of hidden states (M) increases, the model becomes more accurate. However, as the number of hidden state information increases, the gains brought by M increase have decreased. For example,

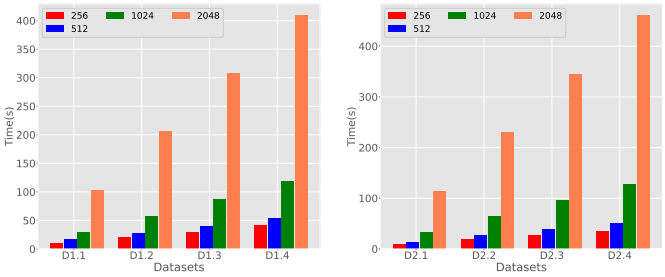


Fig. 10: The impact of V on the HSA-Net.

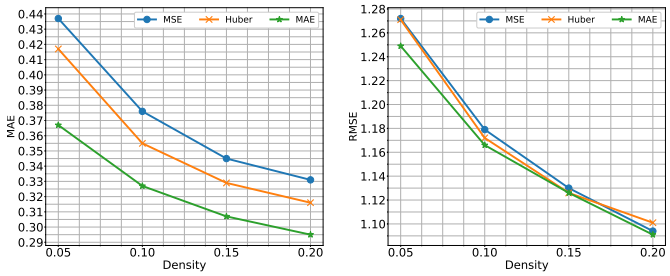


Fig. 11: The impact of loss fuction on the HSA-Net.

the accuracy rate of M=5 is 2% higher than M=1, but the improvement is not obvious when M=7 is compared with M=5.

Results: The prediction accuracy increases as M increases, but the degree of improvement will gradually decrease. At the same time, the cost of learning will increase significantly as M increases. Considering the calculation cost brought by initializing hidden states as shown in Figure 9(a), the default setting of M is 5.

2) The impact of the number of neurons (V)

In HSA-Net, we map each information into a 1xV dimensional vector. At the same time, we set the number of channels of the convolution operation to V. The number of neurons in the QoSprediction step is 2V, V, and 1. In this section, we adjusted the value of V to confirm the influence of different parameters on the model.

Approach: We evaluate the running time of HSA-Net when V takes a different value. It can be seen from Figure 12 that as V increases, the learning cost becomes high. In the experiment, the MAE indicator fluctuates within 0.1, but the cost of learning increases significantly with the increase of V. As illustrated in Figure 10, when V=1024, the training time is twice that of V=512, and when V=2048, the training time is nine times that of V=512.

Results: The prediction accuracy does not increase significantly with the increase of V, but the learning cost increases rapidly. Considering the cost of learning, the default setting of V is 512.

3) The impact of the loss function

Approach: This section compares the performance of the Huber loss function against the MAE (absolute error loss) and MSE (squared error loss) functions.

As shown in Figure 10, the loss function has a greater impact on the MAE indicator than the RMSE indicator. Among the compared indicators, the MSE loss function has the worst prediction performance, and the MAE function achieves the highest precision prediction.

Results: The MAE loss function has the best prediction performance because the MAE function can ignore the influence of outliers as much as possible.

The experimental results are summarized in Figure 11. The results show that the MAE function has the best performance between MAE and RMSE for the QoS prediction. Through the analysis of the dataset, we discover a phenomenon that there are a large number of outliers in the QoS datasets. The response time of most service records is less than 1 second, and the response time of the overtime service is 20.

4) The impact of learning rates

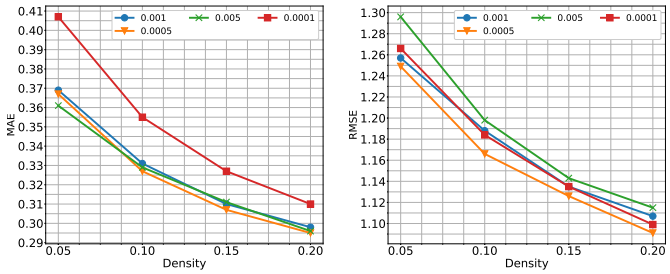


Fig. 12: The impact of learning rates on the HSA-Net

Approach: We choose the Nadam algorithm as our optimizer. This algorithm adds first-order momentum accumulation to the mini-batch adaptive moment estimation (Adam), which has the benefits of high computational efficiency and smaller memory requirement [65]. It has a better learning rate as well as a stronger influence on the update of the gradient. To evaluate the impact of the learning rate on the model, we use the dataset D1 to conduct experiments. The number of hidden states M is set to 5. We set up the maximum number of iterations for 100. Figure 12 shows the change of MAE and RMSE when the learning rate is 0.0001, 0.005, 0.0005, and 0.001.

Results: The results show that the learning rate of 0.0005 is significantly better than other results. MAE and RMSE increase relatively when the learning rate is 0.0001, 0.005, and 0.001.

6 DISCUSSION

In this section, we discuss in detail the influence of the parameters in Algorithm 1 through two comparative experiments. And we discussed the performance of HSA-Net comparison with the point-by-point prediction-based method.

1) The impact of Complexity Factor

Approach: To evaluate the effectiveness of the complexity factor in hidden state initialization algorithm, we set the penalty coefficient (w) value to 50, and set the complexity factor value to 10, 30 and 50.

TABLE 7: The results of Algorithm 1 use different complex factors on the dataset D1.1.

CF	User0-HS1	User0-HS2	User0-HS3	Service1-HS1	Service1-HS2	Service1-HS3
10	0.680	1.542	2.277	0.118	1.996	0.948
30	0.0671	1.520	2.246	0.118	1.998	0.950
50	0.0668	1.513	2.234	0.118	1.987	0.950

Results: During the experiment, different complexity factors lead to a different number of iterations, and the final results are almost the same.

2) The impact of Penalty Coefficient (w)

Approach: To evaluate the effectiveness of the penalty coefficient in the hidden state initialization algorithm, we set the complexity factors value to 30 and set the penalty coefficient value to 10, 30, and 50.

TABLE 8: The results of Algorithm 1 on the dataset D1.1 with different penalty coefficient (w)

w	User0-HS1	User0-HS2	User0-HS3	Service1-HS1	Service1-HS2	Service1-HS3
10	0.673	1.536	2.272	0.119	2.013	0.942
30	0.0671	1.508	2.227	0.119	1.998	0.947
50	0.671	1.520	2.246	0.118	1.988	0.950

Results: During the experiment, different penalty coefficient (w) only lead to a different number of iterations, and the final results are almost the same.

3) Comparision with point-by-point prediction based mehod

Approach: Chowdhury *et al.* [66] proposed a layered hybrid filtering prediction scheme (called CAHPHF) considering service and user context information. CAHPHF completes QoS prediction by predicting one record at a time due to the need to perform multiple coordination filtering and matrix decomposition for target users and services. The dataset and test method used in CAHPHF are different from other methods because it requires detailed context information and can only complete sample tests. To conduct a fair comparison, we follow the test method provided by CAHPHF on our dataset. In detail, we are randomly sampling 200 instances from the testing dataset for the comparison. We perform five experiments to calculate the average result. The results are as follows:

TABLE 9: The comparison of HSA-Net and CAHPHF in dataset D1

Method	D1.1		D1.2		D1.3		D1.4	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
CAHPHF[63]	2.04	2.886	1.989	3.024	1.760	3.008	1.656	2.887
HSA-Net	1.25	2.41	1.44	2.57	1.37	2.54	1.22	2.19

Results: We implemented CAHPHF and HSA-Net in the same environment that includes Python 3.8.3 and TensorFlow 1.15.0. To keep a fair comparison on the dataset with limited information, we remove the distance of collaborative filtering in CAHPHF. Because the dataset used in this paper does not contain detailed location information. All other parameters remain consistent with the original description of CAHPHF. The result of experiments in dataset D1 is shown in Table 9. The performance of HSA-Net is better than CAHPHF in the sampling test. Relatively speaking, CAHPHF is difficult to calculate similarity when the information is limited because it needs to go through multiple collaborative filtering base on rich context information.

7 CONCLUSION

This paper first analyzed challenges for the high accuracy QoS prediction, which include the data-sparse problem raised in many studies and the impact of missing information on the prediction task proposed in this paper. Based on the intuitions we got from the problem analysis, we proposed a hidden-state aware network named HSA-Net. In addition, we showed how to initialize the hidden state information of users and services, and to abstract and fuse various information to achieve QoS prediction tasks with high accuracy. In the first step of HSA-Net, a hidden state initialization algorithm was used to initialize five state information of users and services. This step can effectively alleviate the problem of insufficient information in the QoS prediction task, which reduces the accuracy of the algorithm. In the second step, the hidden state perception approach was utilized to acquire four interpretable fused features. Consequently, interrelationships between independent features can be obtained. Finally, a nonlinear relationship between fusion features and QoS is taught to achieve QoS prediction. The experimental results on two datasets demonstrated that MAE index of the proposed approach is reduced by 3.67%˜ 28.84%, RMSE index is reduced by 3.075%˜ 9.45% compared with ten baselines on average.

In the time-aware QoS prediction task, the states of many users and services will change over time, which may affect the accuracy of the prediction model. In future work, we plan to extend our approach to realize the prediction approach based on variable state perception. We plan to use temporal datasets to learn about the impact of user and service state changes on QoS.

REFERENCES

[1] D. A. Menascé, "Qos issues in web services," *IEEE internet computing*, vol. 6, no. 6, pp. 72–75, 2002.

[2] C. Jatoth, G. Gangadharan, and R. Buyya, "Computational intelligence based qos-aware web service composition: a systematic literature review," *IEEE Transactions on Services Computing*, vol. 10, no. 3, pp. 475–492, 2015.

[3] Z. Zheng, Y. Zhang, and M. R. Lyu, "Distributed qos evaluation for real-world web services," in *2010 IEEE International Conference on Web Services*. IEEE, 2010, pp. 83–90.

[4] Y. Ma, S. Wang, P. C. Hung, C.-H. Hsu, Q. Sun, and F. Yang, "A highly accurate prediction algorithm for unknown web service qos values," *IEEE Transactions on Services Computing*, vol. 9, no. 4, pp. 511–523, 2015.

[5] D. Ryu, K. Lee, and J. Baik, "Location-based web service qos prediction via preference propagation to address cold start problem," *IEEE Transactions on Services Computing*, 2018.

[6] Z. Chen, L. Shen, F. Li, and D. You, "Your neighbors alleviate cold-start: On geographical neighborhood influence to collaborative web service qos prediction," *Knowledge-Based Systems*, vol. 138, pp. 188–201, 2017.

[7] M. Tang, Z. Zheng, G. Kang, J. Liu, Y. Yang, and T. Zhang, "Collaborative web service quality prediction via exploiting matrix factorization and network map," *IEEE Transactions on Network and Service Management*, vol. 13, no. 1, pp. 126–137, 2016.

[8] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Wsrec: A collaborative filtering based web service recommender system," in *2009 IEEE International Conference on Web Services*. IEEE, 2009, pp. 437–444.

[9] ——, "Qos-aware web service recommendation by collaborative filtering," *IEEE Transactions on services computing*, vol. 4, no. 2, pp. 140–152, 2010.

[10] J. Cao, Z. Wu, Y. Wang, and Y. Zhuang, "Hybrid collaborative filtering algorithm for bidirectional web service recommendation," *Knowledge and information systems*, vol. 36, no. 3, pp. 607–627, 2013.

[11] J. Li and J. Lin, "A probability distribution detection based hybrid ensemble qos prediction approach," *Information Sciences*, vol. 519, pp. 289–305, 2020.

[12] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th international conference on world wide web*, 2017, pp. 173–182.

[13] H.-J. Xue, X. Dai, J. Zhang, S. Huang, and J. Chen, "Deep matrix factorization models for recommender systems." in *IJCAI*, vol. 17. Melbourne, Australia, 2017, pp. 3203–3209.

[14] Y. Zhang, C. Yin, Q. Wu, Q. He, and H. Zhu, "Location-aware deep collaborative filtering for service recommendation," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2019.

[15] W. Ahmed, Y. Wu, and W. Zheng, "Response time based optimal web service selection," *IEEE Transactions on Parallel and distributed systems*, vol. 26, no. 2, pp. 551–561, 2013.

[16] J. Li, J. Wang, Q. Sun, and A. Zhou, "Temporal influences-aware collaborative filtering for qos-based service recommendation," in *2017 IEEE International Conference on Services Computing (SCC)*. IEEE, 2017, pp. 471–474.

[17] O. G. Uyan and V. C. Gungor, "Qos-aware lte-a downlink scheduling algorithm: A case study on edge users," *International Journal of Communication Systems*, vol. 32, no. 15, p. e4066, 2019.

[18] L. Chen, F. Xie, Z. Zheng, and Y. Wu, "Predicting quality of service via leveraging location information," *Complexity*, vol. 2019, 2019.

[19] Y. Wu, F. Xie, L. Chen, C. Chen, and Z. Zheng, "An embedding based factorization machine approach for web service qos prediction," in *International Conference on Service-Oriented Computing*. Springer, 2017, pp. 272–286.

[20] S. H. Ghafouri, S. M. Hashemi, and P. C. Hung, "A survey on web service qos prediction methods," *IEEE Transactions on Services Computing*, 2020.

[21] Z. Zheng, L. Xiaoli, M. Tang, F. Xie, and M. R. Lyu, "Web service qos prediction via collaborative filtering: A survey," *IEEE Transactions on Services Computing*, 2020.

[22] K. Lee, J. Park, and J. Baik, "Location-based web service qos prediction via preference propagation for improving cold start problem," in *2015 IEEE International Conference on Web Services*. IEEE, 2015, pp. 177–184.

[23] X. Chen, X. Liu, Z. Huang, and H. Sun, "Regionknn: A scalable hybrid collaborative filtering algorithm for personalized web service recommendation," in *2010 IEEE international conference on web services*. IEEE, 2010, pp. 9–16.

[24] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Qos-aware web service recommendation by collaborative filtering," *IEEE Transactions on services computing*, vol. 4, no. 2, pp. 140–152, 2010.

[25] Z. Chen, L. Shen, F. Li, and D. You, "Your neighbors alleviate cold-start: On geographical neighborhood influence to collaborative web service qos prediction," *Knowledge-Based Systems*, vol. 138, pp. 188–201, 2017.

[26] H. Wu, K. Yue, B. Li, B. Zhang, and C.-H. Hsu, "Collaborative qos prediction with context-sensitive matrix factorization," *Future Generation Computer Systems*, vol. 82, pp. 669–678, 2018.

[27] 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.

[28] C. Wu, W. Qiu, Z. Zheng, X. Wang, and X. Yang, "Qos prediction of web services based on two-phase k-means clustering," in *2015 ieee international conference on web services*. IEEE, 2015, pp. 161–168.

[29] Z. Tan and L. He, "An efficient similarity measure for user-based collaborative filtering recommender systems inspired by the physical resonance principle," *IEEE Access*, vol. 5, pp. 27 211–27 228, 2017.

[30] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*, 2001, pp. 285–295.

[31] H. Ma, I. King, and M. R. Lyu, "Effective missing data prediction for collaborative filtering," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, 2007, pp. 39–46.

[32] H. Sun, Z. Zheng, J. Chen, and M. R. Lyu, "Personalized web service recommendation via normal recovery collaborative filtering," *IEEE Transactions on Services Computing*, vol. 6, no. 4, pp. 573–579, 2012.

[33] X. Luo, M. Zhou, S. Li, Z. You, Y. Xia, and Q. Zhu, "A nonnegative latent factor model for large-scale sparse matrices in recommender systems via alternating direction method," *IEEE transactions on neural networks and learning systems*, vol. 27, no. 3, pp. 579–592, 2015.

[34] Y. Shi, M. Larson, and A. Hanjalic, "Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges," *ACM Computing Surveys (CSUR)*, vol. 47, no. 1, pp. 1–45, 2014.

[35] Y. Koren and R. Bell, "Advances in collaborative filtering," in *Recommender systems handbook*. Springer, 2015, pp. 77–118.

[36] L. Wu, P. Sun, R. Hong, Y. Ge, and M. Wang, "Collaborative neural social recommendation," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2018.

[37] L. Ren and W. Wang, "An svm-based collaborative filtering approach for top-n web services recommendation," *Future Generation Computer Systems*, vol. 78, pp. 531–543, 2018.

[38] K. Chen, H. Mao, X. Shi, Y. Xu, and A. Liu, "Trust-aware and location-based collaborative filtering for web service qos prediction," in *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, vol. 2. IEEE, 2017, pp. 143–148.

[39] M. Tang, Y. Jiang, J. Liu, and X. Liu, "Location-aware collaborative filtering for qos-based service recommendation," in *2012 IEEE 19th international conference on web services*. IEEE, 2012, pp. 202–209.

[40] 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. 9, no. 5, pp. 686–699, 2015.

[41] D. Ryu, K. Lee, and J. Baik, "Location-based web service qos prediction via preference propagation to address cold start problem," *IEEE Transactions on Services Computing*, 2018.

[42] Y. Feng and B. Huang, "Cloud manufacturing service qos prediction based on neighbourhood enhanced matrix factorization," *Journal of Intelligent Manufacturing*, pp. 1–12, 2018.

[43] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[44] X. Luo, H. Wu, H. Yuan, and M. Zhou, "Temporal pattern-aware qos prediction via biased non-negative latent factorization of tensors," *IEEE transactions on cybernetics*, vol. 50, no. 5, pp. 1798–1809, 2019.

[45] X. Luo, M. Zhou, S. Li, Y. Xia, Z.-H. You, Q. Zhu, and H. Leung, "Incorporation of efficient second-order solvers into latent factor models for accurate prediction of missing qos data," *IEEE transactions on cybernetics*, vol. 48, no. 4, pp. 1216–1228, 2017.

[46] X. Luo, M. Zhou, Y. Xia, Q. Zhu, A. C. Ammari, and A. Alabdulwahab, "Generating highly accurate predictions for missing qos data via aggregating nonnegative latent factor models," *IEEE transactions on neural networks and learning systems*, vol. 27, no. 3, pp. 524–537, 2016.

[47] D. WU, Q. He, X. Luo, M. Shang, Y. He, and G. Wang, "A posterior-neighborhood-regularized latent factor model for highly accurate web service qos prediction," *IEEE Transactions on Services Computing*, 2019.

[48] Z. Luo, L. Liu, J. Yin, Y. Li, and Z. Wu, "Latent ability model: A generative probabilistic learning framework for workforce analytics," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 5, pp. 923–937, 2018.

[49] D. Kim, C. Park, J. Oh, S. Lee, and H. Yu, "Convolutional matrix factorization for document context-aware recommendation," in *Proceedings of the 10th ACM conference on recommender systems*, 2016, pp. 233–240.

[50] R. Xiong, J. Wang, Z. Li, B. Li, and P. C. Hung, "Personalized lstm based matrix factorization for online qos prediction," in *2018 IEEE International Conference on Web Services (ICWS)*. IEEE, 2018, pp. 34–41.

[51] X. Zhao, Y. Chen, J. Guo, and D. Zhao, "A spatial-temporal attention model for human trajectory prediction," *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 4, pp. 965–974, 2020.

[52] X. Hou, K. Wang, C. Zhong, and Z. Wei, "St-trader: A spatial-temporal deep neural network for modeling stock market movement," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 5, pp. 1015–1024, 2021.

[53] H. Wu, Z. Zhang, J. Luo, K. Yue, and C.-H. Hsu, "Multiple attributes qos prediction via deep neural model with contexts," *IEEE Transactions on Services Computing*, 2018.

[54] W. Li, Y. Xia, M. Zhou, X. Sun, and Q. Zhu, "Fluctuation-aware and predictive workflow scheduling in cost-effective infrastructure-as-a-service clouds," *IEEE Access*, vol. 6, pp. 61 488–61 502, 2018.

[55] J. Bi, H. Yuan, and M. Zhou, "Temporal prediction of multi-application consolidated workloads in distributed clouds," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 4, pp. 1763–1773, 2019.

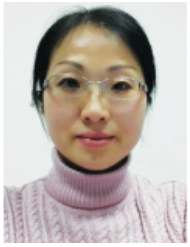
[56] R. Xiong, J. Wang, N. Zhang, and Y. Ma, "Deep hybrid collaborative filtering for web service recommendation," *Expert systems with Applications*, vol. 110, pp. 191–205, 2018.

[57] Y. Yin, L. Chen, Y. Xu, J. Wan, H. Zhang, and Z. Mai, "Qos prediction for service recommendation with deep feature learning in edge computing environment," *Mobile Networks and Applications*, pp. 1–11, 2019.
[58] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," *arXiv preprint arXiv:1301.7363*, 2013.
[59] J. M. Hernández-Lobato, N. Houlsby, and Z. Ghahramani, "Probabilistic matrix factorization with non-random missing data," in *International Conference on Machine Learning*, 2014, pp. 1512–1520.
[60] D. Wu, X. Luo, M. Shang, Y. He, G. Wang, and X. Wu, "A data-characteristic-aware latent factor model for web services qos prediction," *IEEE Transactions on Knowledge and Data Engineering*, 2020.
[61] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "Autorec: Autoencoders meet collaborative filtering," in *Proceedings of the 24th international conference on World Wide Web*, 2015, pp. 111–112.
[62] M. I. Smahi, F. Hadjila, C. Tibermacine, and A. Benamar, "A deep learning approach for collaborative prediction of web service qos," *Service Oriented Computing and Applications*, vol. 15, no. 1, pp. 5–20, 2021.
[63] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine learning research*, vol. 7, no. Jan, pp. 1–30, 2006.
[64] A. Benavoli, G. Corani, and F. Mangili, "Should we really use post-hoc tests based on mean-ranks?" *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 152–161, 2016.
[65] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[66] R. R. Chowdhury, S. Chattopadhyay, and C. Adak, "Cahphf: Context-aware hierarchical qos prediction with hybrid filtering," *IEEE Transactions on Services Computing*, 2020.

ZiLiang Wang He was received the B.S. degree from Nanchang Hangkong University, Jiangxi, China, in 2017. He is currently pursuing the Ph.D degree in software engineering in Chongqing University, Chongqing, China. His current research interests include service computing, smart city and system structure.

Meng Yan He is now an Assistant Professor at the School of Big Data and Software Engineering, Chongqing University, China. Prior to joining Chongqing University, he was a Postdoc at Zhejiang University. He got his Ph.D degree in June 2017 from Chongqing University, China. His current research focuses on how to improve developers' productivity, how to improve software quality and how to reduce the effort during software development by analyzing rich software repository data.

XiaoHong Zhang He received the M.S. degree in applied mathematics and the Ph.D. degree in computer software and theory from Chongqing University, China, in 2006. He is currently a Professor and the Vice Dean of the School of Big Data and Software Engineering, Chongqing University. His current research interests include data mining of software engineering, topic modeling, image semantic analysis, and video analysis. He has authored over 50 scientific articles and some of them are published in some authoritative journals and conferences, such as the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, the IEEE TRANSACTIONS ON IMAGE PROCESSING, PR, PRL, JSS, and IST.



Ling Xu Ling Xu was born in 1975 and received her B.S. in Hefei University of Technology in 1998, and her M.S. degree in software engineering in 2004. She received her Ph.D. degree in computer science technology from Chongqing University, PR China in 2009. She is currently an Associate Professor of the School of Big Data and Software Engineering, Chongqing University. Her research interests include data mining of software engineering, topic modeling, and image processing.

Dan Yang He was received the B.Eng. degree in automation, the M.S. degree in applied mathematics, and the Ph.D. degree in machinery manufacturing and automation from Chongqing University, Chongqing. From 1997 to 1999, he held a postdoctoral position at the University of ElectroCommunications, Tokyo, Japan. He is currently the President of Southwest Jiaotong University. He is also a Professor with the School of Big Data and Software Engineering, Chongqing University. He has authored over 100 scientific articles and some of them are published in some authoritative journals and conferences, such as the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, CVPR, and BMVC. His research interests include computer vision, image processing, pattern recognition, software engineering, and scientific computing.