A Feature Distribution Smoothing Network Based on Gaussian Distribution for QoS Prediction

Tongxin Lu[1,2], Xiaohong Zhang[1,2*], Ziliang Wang[1,2], Meng Yan[1,2]
[1]Key Laboratory of Dependable Service Computing in Cyber Physical Society (Chongqing University), Ministry of Education, China
[2]School of Big Data and Software Engineering, Chongqing University, Chongqing, China
Email: {lutongxin}@stu.cqu.edu.cn, {xhongz, wangziliang, mengy}@cqu.edu.cn

***Abstract*—With the increasing number of services and their homogenization, the use of Quality of Service (QoS) for recommendations has become necessary. However, existing QoS prediction solutions have limitations in solving the noise and label imbalance problems of dataset, which greatly limit the improvement of QoS prediction accuracy. In this paper, we propose FSNet that contains a feature distribution smoothing module and an improved W-Huber loss function. The feature distribution smoothing module mitigates the effect of noise problem by fitting potential Gaussian distribution of known features with a supervised feedforward neural network. W-Huber loss function mitigates the impact of label imbalance problem on QoS prediction by reweighting the two components of Huber loss function. We conduct extensive experiments on real large-scale QoS dataset, and the results demonstrate that the proposed FSNet method outperforms existing QoS prediction methods.**

***Index Terms*—Service recommendation, Noisy data, Label imbalance, Feature distribution smoothing, QoS prediction**

I. Introduction

In recent years, the popularity of Software Oriented Architecture (SOA), Internet of Services (IoS), and cloud computing technologies has resulted in a significant increase in the number of their core components, Web services. According to ProgrammableWeb.com, a Web services repository, there were more than 24,000 Web services maintained as of October 2022. With the exponential growth of Web services, a significant number of them become homogenized, i.e., multiple services share the same or similar functional characteristics. Therefore, it is an urgent problem for users to choose the most suitable one from these homogeneous Web services. Quality of service (QoS), which is judged by service availability, response time, and throughput [1] [2], is a non-functional characteristic of Web services that can effectively assist users in service selection [3]. The quality of service is largely influenced by the context of Web services, i.e., for the same service, the QoS values are likely to differ significantly when invoked by different users [4]. Therefore, it is necessary to obtain personalized QoS values to adapt different users in performing service recommendation. Considering the time cost and resource overhead, it is impractical for users to invoke all services one by one to obtain the corresponding QoS values [5]. Weighing the merits, predicting unknown QoS values based on existing QoS information is thus more feasible.

*Corresponding author.

However, predicting QoS faces many challenges. The first one is the problem of noisy data. Existing QoS prediction research is based on the infirmative assumption that the information collected about users and available services is true and reliable. Due to lack of effective monitoring and verification mechanisms in data collection process, some noise will inevitably appear, which can adversely affect prediction accuracy. To mitigate the impact of this problem, Chen it et al. propose a structured noise complementation algorithm (OSMCSN) based on operator splitting techniques to identify and remove noisy data [6]. Wu it et al. propose MSDDAE, a new multilayer noise reduction autoencoder model to obtain more accurate QoS prediction results [7]. However, these studies only consider the noise in QoS values and neglet noise in other information about users and services.

Secondly, label imbalance problem with QoS data is also worth to be reckoned with. QoS dataset obtained from real world generally have an unbalanced data distribution, resulting in a long-tail effect on the label distribution. For example, when response times are used as labels, since most service invocations complete properly, they usually have short response times, causing most labels to be distributed in intervals with small values. However, in a few cases, due to unstable network conditions or other reasons, the response time of some service invocations becomes longer, making a few labels distributed in intervals with larger values. In this case, some researchers have proposed various approaches in the field of QoS. Suomi it et al. detect the imbalanced parts of the dataset and address them using a QoS preprocessing matrix called OffDQ [8]. Li it et al. mitigate the effect of imbalanced data on the model by using a more robust Huber loss function [9]. In contrast, Mhedhbi it et al. use importance sampling to obtain data with lower imbalance [10]. However, it can be drawn from these researches that some are only generalized methods to cope with the label imbalance problem, which makes them not targeted enough for QoS prediction; and other methods can only be applied to scenarios that improve the correctness of the judgment of outliers in QoS data, while have do no work on overall accuracy of QoS prediction. In addition, methods like sampling not only destroy distribution of the original dataset, but also reduce data utilization and exacerbate QoS data sparsity.

In response to above challenges, in this paper we propose

a feature distribution smoothing QoS prediction method based on Gaussian distribution, namely FSNet. For the noise problem, FSNet invites a feature distribution smoothing module, which is capable of learning potential Gaussian distribution of features through a supervised feedforward neural network and resampling the features. This module can eliminate the interference data in dataset and obtain sampled features that better represent the real data. With respect to the label imbalance problem, the W-Huber loss function is proposed in this paper. The W-Huber loss function is an improved Huber loss function that reweights the linear and nonlinear parts of original Huber loss function, respectively, to mitigate the effect of label imbalance by strengthening the nonlinear part. Overall, FSNet firstly learns the potential Gaussian distribution of features and resamples from the distribution to obtain sampled features. Then, it obtains latent features through the latent state learning algorithm in Study [11]. Finally, FSNet combines sampled features and latent features to feed into a multilayer fully connected network for high accuracy QoS prediction. Throughout the process, W-Huber loss function is applied as the loss function of FSNet.

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

a) We propose a QoS prediction method FSNet that includes a feature distribution smoothing module to effectively mitigate the effect of noise in features and improve the QoS prediction accuracy.
b) We propose an improved W-Huber loss function that reweights the linear and nonlinear parts of original Huber loss function to effectively mitigate the effect of label imbalance problem on QoS prediction.
c) We conduct extensive experiments on real large-scale QoS dataset, and the results show that our approach significantly outperforms on both MAE and RMSE metrics compared to all baseline methods.

The remainder of this paper is organized as follows. The general framework of FSNet is given in Section 2, which details the feature distribution smoothing module and the improved W-Huber loss function. Section 3 describes the specific setup of the experiments, including the experimental environment, dataset, evaluation criteria and baseline method. Section 4 gives the experimental results and analysis. Section 5 reviews the related work. Finally, the full text is summarized in Section 6.

II. Approach

This section will describe our proposed QoS prediction method in detail. The overall framework of FSNet is illustrated in Fig. 1. It comprises three modules: supervised feature distribution smoothing, latent features generation and QoS prediction. We apply the improved W-Huber loss function as the loss function of FSNet.

A. Supervised Feature Distribution Smoothing

This module fits the potential Gaussian distribution in features by means of a supervised feedforward neural network to smooth out the noise in features. Specifically, it can eliminate the interference data in dataset and obtain more realistic and reasonable sampled features.

Before performing feature distribution smoothing, we first preprocess the features. Considering that the embedding operation can encode the objects with low-dimensional vectors and preserve their meanings, we input user known features and service known features into the Embedding layer of Keras. This step is responsible for converting the high-dimensional sparse feature vector into a dense low-dimensional feature vector, which alleviates the effect of deep learning not being good at handling sparse feature vectors.

Then, the obtained embedding vectors are used to construct known user feature map and known service feature map separately. The factors taken into known user feature map(K_u) are user ID vector(I_u^k), user AS information vector(As_u^k) and user country information vector(C_u^k). Similarly, known service feature map(K_s) includes factors of service ID vector(I_s^k), service AS information vector(As_s^k) and service country information vector(C_s^k). Therefore, the size of both known feature maps will be 1×3.

$$K_u = \left[I_u^k, As_u^k, C_u^k\right] \tag{1}$$

$$K_s = \left[I_s^k, As_s^k, C_s^k\right] \tag{2}$$

After that, we do feature distribution smoothing on K_u and K_s mentioned above respectively. Specifically, K_u and K_s are approximated as multivariate Gaussian distributions, which roughly consists of three steps. In the following, we use K_u as an example to illustrate.

First, K_u is put into feedforward neural network based on fully connected layers to obtain the nonlinear relationship between the features.

$$K_u' = f^{[2^v, 2^{v-1}]}(K_u; \mathbf{W}_{K_u}) \tag{3}$$

where $f^{[2^v, 2^{v-1}, 1]}$ denotes the fully connected layers and numbers, such as 2^v mean the number of neurons in this layer.

Then, K_u' is divided into two halves, and the mean and variance of it are calculated.

$$\mu_u = f_{\mu_u}(K_u'; \mathbf{W}_{\mu_u}) \tag{4}$$

$$\sigma_u = f_{\sigma_u}(K_u'; \mathbf{W}_{\sigma_u}) \tag{5}$$

Third, we sample the distribution constructed from the mean and variance.

$$S_u = f_S\left\{f_M(\mu_u, \sigma_u)\right\} \tag{6}$$

where f_M means the MultivariateNormalDiag layer in Tensorflow and f_S means sampling from the distribution.

Similarly, we have

$$S_s = f_S\left\{f_M(\mu_s, \sigma_s)\right\} \tag{7}$$

The result S of feature distribution smoothing can finally be obtained by concatenating and flattening S_u and S_s. The specific operations are as follows:

$$S = F(S_s) \oplus F(S_u) \tag{8}$$

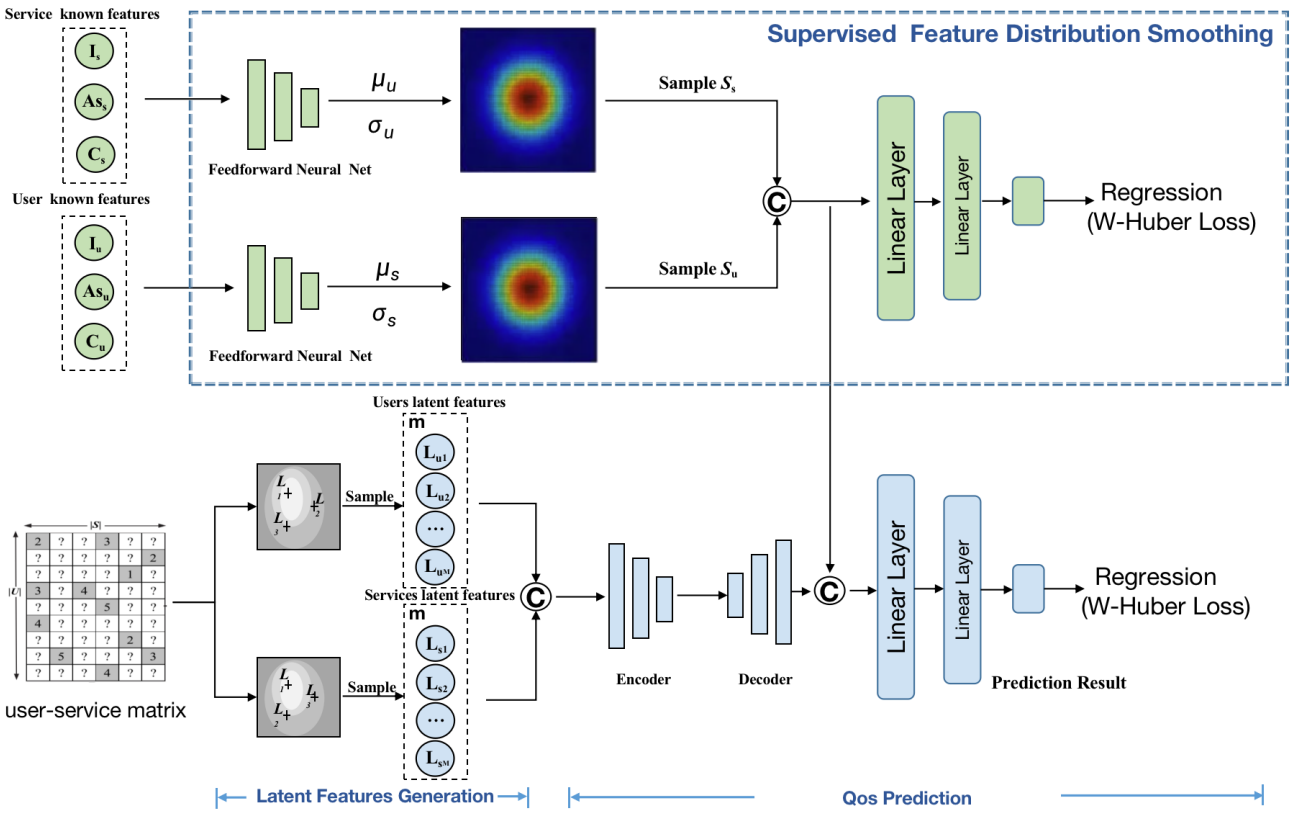


Fig. 1. The framework of FSNet. Supervised feature distribution smoothing: learns the variance and expectation of known features by a feedforward neural network and resamples features; Latent feature generation: generates user latent features and service latent features; QoS prediction: accomplish high accuracy QoS prediction through encoder, decoder and a multilayer fully connected network.

where F denotes the flatten operation and $\oplus$ denotes the concatenate operation by concatenate of Keras.

As shown in Fig. 1, we also connect several linear layers to S and apply the result as the second output of our network. Although this output is also a prediction of service quality, it is not the final prediction result of the entire network. Instead, we use the other output of the network to represent the final prediction of our method and calculate the evaluation value. The output of this sub-network can be used to adjust the parameters of the feedforward neural network for a more accurate predictions. The specific steps are shown as follows

$$P = f^{[2^v, 2^{v-1}, 1]}(S; \mathbf{W}_S) \tag{9}$$

where $f^{[2^v, 2^{v-1}, 1]}$ denotes the fully connected layers, numbers in these layers such as 2^v mean the number of neurons in this layer, and P means the prediction value.

B. Latent Features Generation

To alleviate the effect of data sparsity and make fuller use of known information, we use the method in the Study [11] to generate latent features. The study can extract latent features from known features in two steps, latent states generation and parameter estimation. We generate the user latent features L_u and the service latent features L_s, respectively.

Then, L_u and L_s are fed into the Embedding layer of Keras and finally the latent feature map L is constructed with obtained results.

C. Qos Prediction

First, the latent feature map is passed through the encoder and decoder modules. The encoder module compresses the feature map to obtain compressed information, which is then decoded by the decoder module. This process results in a better representation of latent features compared to the original L. The specific steps of this process are shown below:

$$E = f^{[2^v, 2^{v-1}]}(F(L); \mathbf{W}_L) \tag{10}$$

$$L' = f^{[2^{v-1}, 2^v]}(E; \mathbf{W}_E) \tag{11}$$

where F denotes the flatten operation , $f^{[2^v, 2^{v-1}]}$ denotes the fully connected layers and the numbers such as 2^v denote the number of neurons in this fully connected layer.

Then,the sampled known features is connected with the latent features as follows:

$$X = F(S) \oplus F(L') \tag{12}$$

In the end,we feed the obtained fused feature map X into a multilayer fully connected network for prediction:

$$R = f^{[2^v, 2^{v-1}, 10, 1]}(X; \mathbf{W}_X) \tag{13}$$

where the output R denotes the final QoS prediction value of FSNet.

D. Improved W-Huber Loss Function

As the most frequently used loss function in the Qos prediction problem, MAE has both advantages and disadvantages. It makes the target close to the median and thus be robust to outliers, but at the same time its property of always constant update gradient is likely to make it miss the minima. In this case, Huber loss that combines the advantages of MAE and MSE is more applicable. It is defined as follows:

$$Huber_{loss}(y_i, \hat{y}_i) = \begin{cases} \frac{1}{2}(y_i - \hat{y}_i)^2 & |y - \hat{y}|_{abs} < \delta. \\ \delta|y_i - \hat{y}_i|_{abs} - \frac{1}{2}\delta^2 & else. \end{cases} \tag{14}$$

where y_i represents the real value and the $\hat{y}_i$ represents the predicted value. δ is the hyperparameter in Huber loss function, which plays a selection role. When the prediction deviation is less than δ, the squared error is used; when the prediction deviation is greater than δ, the linear error is used.

As stated in Section 1, the dataset for QoS prediction generally suffer from severe label imbalance and we choose to alleviate the impact of this problem by using the improved W-Huber loss function, which turns out quite effective To be more specific, a factor is added to both the linear and nonlinear components of Huber loss as a weight to form a new loss function, W-Huber.

The W-Huber loss function is defined as follows:

$$W - Huber_{loss}(y_i, \hat{y}_i) = \begin{cases} \gamma\frac{1}{2}(y_i - \hat{y}_i)^2 & |y - \hat{y}|_{abs} < \delta. \\ \beta(\delta|y_i - \hat{y}_i|_{abs} - \frac{1}{2}\delta^2) & else. \end{cases} \tag{15}$$

where γ is the hyperparameters of the linear component and β is the hyperparameters of the nonlinear component.

In determining the values of γ and β, as an attempt to reduce the impact of outliers, we set a smaller value for the parameter of the component whose predicted value differs more from the true value(i.e., β). Correspondingly, to increase the effect of normal values, we set a bigger value for the parameter of the component whose predicted value differs less from the true value (i.e., γ) .

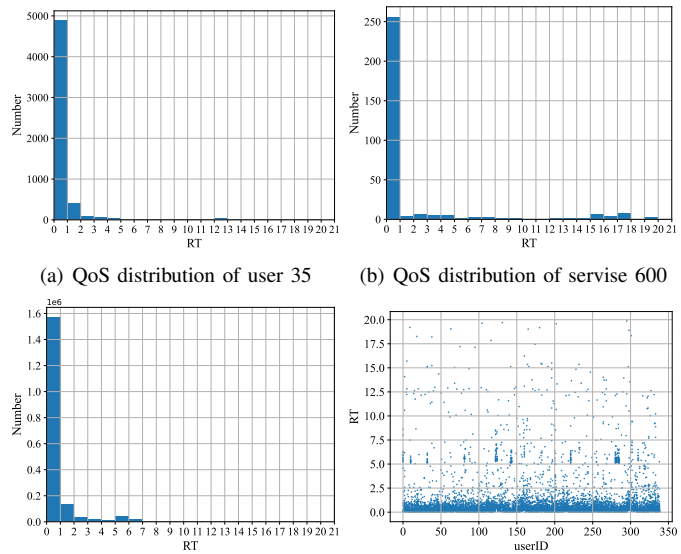
III. EXPERIMENTAL SETUP

In this section, we design experiments to answer the following research questions (RQs):

- Does our proposed method outperform state-of-the-art QoS prediction methods?
- What is the impact of feature distribution smoothing?
- What is the impact of W-Huber loss function?

A. Experimental Environment

All our experiments were performed on a server equipped with two 2.4GHz Intel Xeon cpus and 16-GB of RAM and running Ubuntu 16.04. Our methods and all baseline methods were implemented using Python version 3.6 and TensorFlow version 2.5.0.



(a) QoS distribution of user 35 (b) QoS distribution of servise 600

(c) Original data distribution (d) Original data distribution

Fig. 2. QoS distribution of datasets

B. Dataset

To verify the effectiveness of our method in real-world, we conduct extensive experiments on the QoS dataset known as WS-Dream [12], which has been widely used in the field of QoS prediction. The dataset consists of two subsets, one with response time (rtdata) and the other with throughput (tpdata) , both being quality of service metrics, and the former subset is chosen as experimental dataset. This dataset, collected and maintained by Zheng et al., contains 5,825 services, 339 users, and 1,974,675 QoS call records. Each of these records contains the corresponding user and service location information. Additionally, we perform outlier processing using the isolated forest approach, following the method described in Study [13]. Outliers are identified and assigned a score of 0.1.

The missing value count is performed on the rtdata subset, and the result shows that there are 100,837 missing values in it. In addition to the missing values, there are also some timeout response times that are captured as 20 due to collector limitations. Excluding above mentioned observable noise, there are also some unobservable noises in dataset. Fig. III-B shows the label distribution of rtdata subset. It can be clearly observed that the QoS dataset is significant label imbalance.

In real world, there is often little known data that can be used for network learning when making QoS predictions due to issues such as cost or privacy. In order to better simulate the data sparsity problem encountered in realistic scenarios, we randomly remove the interaction entries with different densities from the QoS matrix. For example, $density$ = 2% means 2% QoS interaction elements are selected as training set to predict the remaining 98% QoS elements.

TABLE I
PREDICTION RESULTS OF BASELINES AND FSNET ON WS-DREAM DATASET

Method	2%		4%		6%		8%		10%	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
UPCC	0.542	1.022	0.466	0.820	0.428	0.787	0.389	0.754	0.555	1.317
IPCC	0.520	1.121	0.473	0.900	0.437	0.838	0.427	0.829	0.596	1.342
UIPCC	0.506	1.075	0.462	0.878	0.427	0.820	0.415	0.808	0.584	1.329
HMF	0.349	0.674	0.277	0.579	0.261	0.551	0.256	0.532	0.256	0.526
LDCF	0.349	0.987	0.279	0.794	0.247	0.751	0.240	0.711	0.213	0.692
CMF	0.327	0.657	0.294	0.605	0.250	0.536	0.231	0.496	0.205	0.461
HSA-Net	0.182	0.558	0.159	0.495	0.128	0.470	0.128	0.448	0.126	0.442
PLRes	0.174	0.524	0.143	0.461	0.135	0.424	0.125	0.409	0.122	0.407
FSNet	**0.149**	**0.481**	**0.128**	**0.433**	**0.118**	**0.392**	**0.114**	**0.379**	**0.109**	**0.363**

C. Evaluation Metrics

AS QoS prediction is a regression problem, in our experiments, mean absolute error (MAE) and root mean square error (RMSE) are to measure the accuracy of QoS prediction of all methods.

MAE is defined as follows:

$$MAE = \frac{(\sum_{i,j} |r_{i,j} - \hat{r}_{i,j}|_{abs})}{N} \tag{16}$$

where $r_{i,j}$ is the real value , $\hat{r}_{i,j}$ is the value predicted by a trained model, and N is the total number of samples to be predicted in QoS records. From the definition of MAE, it is known that when the gap between $r_{i,j}$ and $\hat{r}_{i,j}$ becomes smaller, the value of MAE also becomes smaller, which means lower value of MAE stands for higher accuracy of QoS prediction.

RMSE is defined as follows:

$$RMSE = \sqrt{\frac{(\sum_{i,j}(r_{i,j} - \hat{r}_{i,j})^2)}{N}} \tag{17}$$

Similarly, the lower value of RMSE means the more accurate prediction we have.

D. Baseline Methods

To evaluate the effectiveness of our method, we compare it with the following eight methods:

IPCC [14]: It is a classical item-based collaborative filtering algorithm.

UPCC [15]: Unlike item-based IPCC, UPCC is a user-based collaborative filtering algorithm.

UIPCC [12]: It is a hybrid method that combines predicted results of UPCC and IPCC for higher accuracy prediction.

HMF [16]: It is a context-based collaborative filtering algorithm. HMF clusters users and services using their location information and then predicts missing QoS values by using a global QoS matrix and several local location-based QoS matrices generated by user-service clusters.

LDCF [17]: It is a deep learning algorithm based on collaborative filtering using location information. It also uses a similarity adaptive corrector to correct the prediction results.

CMF [13]: It is a QoS prediction method with outlier resilience. This algorithm uses the Corsi loss to measure difference between true QoS value and predicted value and considers temporal information.

HSA-Net [11]: It is a LDA-based QoS prediction method. First, it performs hidden state generation by LDA, and then performs hidden state perception and completes QoS prediction using neural networks.

PLRes [18]: This approach uses ResNet in QoS prediction to reuse the historical call probability distributions and location features of users and services.

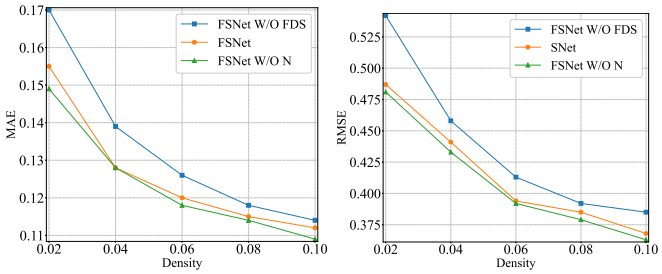
IV. RESULTS

A. RQ1: Does our proposed method outperform to state-of-the-art QoS prediction methods?

We experiment the proposed FSNet with above baseline methods in same experimental settings and compare all results. To ensure high quality and continuous stability of this experiment, we run each baseline method 10 times and calculate its average value to represent the effect of this method.

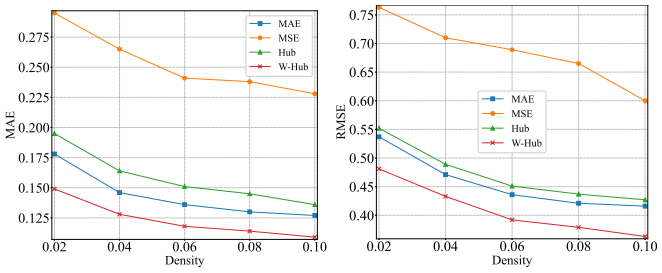
In terms of fixed settings and parameters, our method uses $Nadam$ as the model optimizer and $ReLU$ as the activation function; training batch size is set to 256 and learning rate of FSNet is set to 0.001. In addition, we set the parameter δ in W-Huber loss function to 0.3 and the number of generated latent features of users and services to 5. To find the optimal variable parameters mentioned above, we run experiments with different variable parameters on the dataset with $density = 2\%$ and $density = 4\%$ and carry out 40 iterations. The parameters of each baseline method are set as the default parameters in the references.

The experimental results are shown in Table 1. According to the Table, the following observations can be drawn: a) In general, simple memory-based methods(e.g., IPCC, UPCC) are significantly less effective than other methods. HMF achieves higher accuracy than them because it is a context-based method that takes into account the inclusion of location information for modeling while using collaborative filtering algorithm. b) DL-based methods perform better than other methods in terms of RMSE and MAE. Among



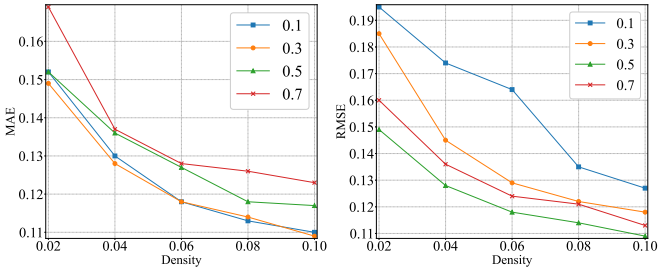
(a) MAE (b) RMSE

Fig. 3. The results of FSNet with and without feature distribution smoothing



(a) MAE (b) RMSE

Fig. 4. The results of FSNet with different loss function



(a) MAE (b) RMSE

Fig. 5. The results of FSNet with different δ

all baseline methods, HSA-Net and PLRes, which have a clear advantage in prediction accuracy, are both DL-based methods. The result confirms that neural network models provide powerful modeling capabilities for QoS prediction. c) Among all methods, FSNet performs best results on both MAE and RMSE metrics. Compared with PLRes, which is the most effective neural network-based baseline methods, FSNet improves MAE performance by 14.37%, 10.49%, 12.59%, 8.80%, and 10.66%, RMSE by 8.21%, 6.07%, 7.55%, 7.33%, and 10.81%, respectively, on dataset at density of 2%, 4%, 6%, 8%, and 10%.

In summary, it is safe to say that our method outperforms the state-of-the-art QoS prediction methods.

B. RQ2: What is the impact of feature distribution smoothing?

To better validate the effect of feature distribution smoothing to improve prediction accuracy by reducing the impact of noise, we generate a new dataset with 10% noise added to training set in the original dataset. Specifically, we randomly select 10% country code and AS code from the training set and reassign them to random values in the range of values taken.

Then, we design three comparison experiments. The first named FSNet uses the complete network structure with feature distribution smoothing module. The second named FSNet W/O FDS uses the network structure without feature distribution smoothing module. The last, named FSNet W/O N, uses the same complete network structure as FSNet with feature distribution smoothing module. Among them, FSNet and FSNet W/O FDS run on the new dataset, while FSNet W/O N runs on the original dataset.

According to Fig. 3, the following conclusions can be drawn: a)There is no significant difference between FSNet and FSNet W/O N in terms of MAE and RMSE in dataset of all density. This result indicates that the complete FSNet model is virtually immune to noisy data. b)FSNet W/O FDS underperformed significantly compared with FSNet in both MAE and RMSE metrics in dataset of all density. This result indicates that the model with feature distribution smoothing module removed is susceptible to noisy data and thus degrades QoS prediction accuracy.

In conclusion, we can claim that the feature distribution smoothing module can improve QoS prediction accuracy by mitigating the effect of noise.

C. RQ3: What is the impact of W-Huber loss function?

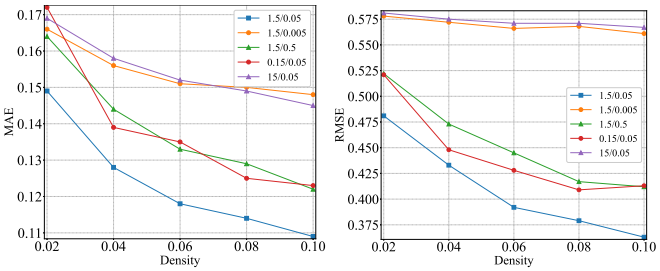
In verification of the effectiveness of W-Huber loss function, we design four comparison experiments. These experiments set the loss functions of our method as MAE, MSE, Huber loss function and W-Huber loss function, respectively. Among them, W-Huber loss function has two hyperparameters γ and β that are added to reweight the two components of Huber loss function. According to experimental results in the later section , we set γ to 1.5 and set β to 0.05. Default values are used and kept consistent for all other parameters involved in the experiments.

Fig. 5 reveals that MSE performs the worst among all loss functions, and Huber loss function combining MSE and MAE produces poorer results than MAE. However delight to see that effect of the W-Huber loss function is significantly improved compared with the Hubers loss function. Conclusively speaking, W-Huber loss function performs the best among all loss functions. The experimental results show that W-Huber loss function can effectively alleviate the impact of label imbalance and thus improve the QoS prediction accuracy.

D. The impact of hyperparameters in W-Huber Loss

1) The impact of δ in W-Huber Loss

Because W-Huber loss is obtained by adding two hyperparameters to Huber loss for re-weighting, it has a hyperparameter δ to determine whether to use linear loss or mean square loss, just like Huber loss. The hyperparameter δ determines



(a) MAE (b) RMSE

Fig. 6. The results of FSNet with different γ and β

how sensitive the model is to data outliers. Specifically, the MAE loss is used when the predicted value is near the true value δ interval, otherwise the MSE loss is used. To discuss the impact of δ on QoS prediction performance, we set up four comparison experiments with δ set to 0.1, 0.3, 0.5 and 0.7, respectively. Except the δ, all parameters involved in the experiments were used with default values and kept consistent.

Fig. 5 shows that when δ is 0.3, MAE and RMSE achieve the minimum value on each density dataset, implying the best prediction performance. Therefore, 0.3 is set as the default value of δ for all experiments in this paper.

2) The impact of γ and β in W-Huber Loss

In order to obtain better values of the parameter settings, we determine optimal values of individual parameters before combining these parameters. We firstly add γ and β separately to obtain the optimal values of them. Then we combine the two optimal values and adjust the parameter values according to experimental results.

To show the impact of γ and β on the prediction accuracy, we design five comparison experiments with γ and β set to 1.5/0.05, 1.5/0.005, 1.5/0.5, 0.15/0.05 and 15/0.05, respectively.

Fig. 6 shows that when γ is set to 1.5 and β is set to 0.05, MAE and RMSE achieve the minimum value on each density dataset. Therefore, the default values of γ and β are set to 1.5 and 0.05, respectively, for all experiments in this paper.

V. RELATED WORKS

Existing work about QoS prediction can be generally grouped into two categories: collaborative filtering (CF)-based QoS prediction and deep learning (DL)-based QoS prediction. In this section, we overview the works that are closely related to these categories.

CF-Based QoS Prediction. The collaborative filtering algorithm is one of the most successful and widely used methods in the field of QoS prediction. It can be broadly classified into three types: memory-based CF methods, model-based CF methods and context-based CF methods. Traditional memory-based CF methods use only the user-service QoS matrix and make predictions by the similarity of users or services [19]. For example, Chen *et al.* used the similarity of users to predict QoS values [20] . The model-based CF algorithm uses all QoS values in the user service matrix to construct a global model for making QoS value predictions. Commonly used models include clustering, matrix factorization and time series [21]. For instance, He *et al.* used K-means clustering algorithm to cluster services and users, and then used matrix factorization method to predict QoS values [22]. Amin *et al.* combined ARIMA and GARCH models to propose a time series-based approach which can make a more accurate prediction of QoS values [23]. Tang *et al.* proposed a collaborative filtering approach to predict the QoS of mobile services based on factorization machines [24]. The context-based CF approach takes into account contextual information, such as location or time, in addition to users and services, to effectively improve the accuracy of QoS prediction. For example, Liu *et al.* significantly enhanced the data sparsity and cold start problems by adding the user's national geographic information and the service's web geographic information to the original user and service information [9]. Yang *et al.* proposed a novel method to predict QoS values based on factorization machine, which leveraged not only QoS information of users and services but also the user and service neighbor's information [25].

DL-Based QoS Prediction. Compared with CF-based methods, DL-based methods have been increasingly used in QoS prediction in recent years because of their ability to extract nonlinear features. White *et al.* added memory to the recurrent neural network to form an LSTM network and made predictions about service by using historical data of the service [26]. Li *et al.* proposed a topology-aware neural (TAN) model for collaborative QoS prediction. It projected the features of users, services, and intermediate nodes on the communication path as input features to a shared potential space [27]. In addition, some researchers made attempts to improve the accuracy of QoS prediction by mining more information from known features. For example, Wang *et al.* proposed HSA-Net, which could mine latent features of users and services through an LDA-based pretraining process [11]. There are also some researchers who focus on better predicting abnormal QoS values. Mhedhbi *et al.* improved the percentage of abnormal data in the training set through a sampling framework and then used the proposed (M,K)-NN algorithm for QoS prediction to detect outliers [10].

VI. CONCLUSION

In this paper, we start with pointing out shortcomings of existing QoS prediction methods in coping with the noise problem and label imbalance problem. Thus, we propose a QoS prediction method FSNet to mitigate the effect of above problems. FSNet reduces the effect of noise problem through a feature distribution smoothing module, which can eliminate the interference data in dataset and obtain more realistic and reasonable sampled features. As for alleviating the impact of label imbalance, we propose W-Huber loss function, which strengthens the linear part of the Huber loss function and weakens the non-linear part. To evaluate the effectiveness of our method, we conduct extensive experiments on real large-scale QoS dataset. The results show that compared to state-of-the-art baselines, FSNet improves MAE performance by

14.37%, 10.49%, 12.59%, 8.80%, and 10.66%, RMSE by 8.21%, 6.07%, 7.55%, 7.33%, and 10.81%, respectively, on dataset at density of 2%, 4%, 6%, 8%, and 10%.

ACKNOWLEDGMENT

This work was supported in part by the National Key Research and Development Project (No. 2021YFB1714200), the Fundamental Research Funds for the CentralUniversities (No.2022CDJDX-005), the Chongqing Technology Innovation and Application Development Project (No. CSTB2022TIAD-STX0007 and No. CSTB2022TIAD-KPX0067).

REFERENCES

[1] 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*, 2015, pp. 161–168.

[2] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, "Service-oriented computing: State of the art and research challenges," *Computer*, vol. 40, no. 11, pp. 38–45, 2007.

[3] J. El Hadad, M. Manouvrier, and M. Rukoz, "Tqos: Transactional and qos-aware selection algorithm for automatic web service composition," *IEEE Transactions on Services Computing*, vol. 3, no. 1, pp. 73–85, 2010.

[4] L. Shao, J. Zhang, Y. Wei, J. Zhao, B. Xie, and H. Mei, "Personalized qos prediction forweb services via collaborative filtering," in *IEEE International Conference on Web Services (ICWS 2007)*, 2007, pp. 439–446.

[5] Y. Zhang, K. Wang, Q. He, F. Chen, S. Deng, Z. Zheng, and Y. Yang, "Covering-based web service quality prediction via neighborhood-aware matrix factorization," *IEEE Transactions on Services Computing*, vol. 14, no. 5, pp. 1333–1344, 2021.

[6] C. Lei, Y. Geng, C. Zheng-yu, X. Fu, and X. Jian, "Web services qos prediction via matrix completion with structural noise," vol. 36, pp. 49–59, 2015.

[7] W. Mengwei, L. Qin, and W. Yingxue, "A multi-stack denoising autoencoder for qos prediction," vol. 13530, pp. 757–768, 2022.

[8] S. Chattopadhyay, R. Chanda, S. Kumar, and C. Adak, "Offdq: An offline deep learning framework for qos prediction." New York, NY, USA: Association for Computing Machinery, 2022.

[9] 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, 2016.

[10] M. Mhedhbi, S. Elayoubi, and G. Leconte, "Ai-based prediction for ultra reliable low latency service performance in industrial environments," in *2022 18th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2022, pp. 130–135.

[11] Z. Wang, X. Zhang, M. Yan, L. Xu, and D. Yang, "Hsa-net: Hidden-state-aware networks for high-precision qos prediction," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 6, pp. 1421–1435, 2022.

[12] 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, 2011.

[13] F. Ye, Z. Lin, C. Chen, Z. Zheng, and H. Huang, "Outlier-resilient web service qos prediction." New York, NY, USA: Association for Computing Machinery, 2021.

[14] J. B. D. CarlKadie, "Empirical analysis of predictive algorithms for collaborative filtering," *Microsoft Research Microsoft Corporation One Microsoft Way Redmond, WA*, vol. 98052, 1998.

[15] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms." New York, NY, USA: Association for Computing Machinery, 2001.

[16] P. He, J. Zhu, Z. Zheng, J. Xu, and M. R. Lyu, "Location-based hierarchical matrix factorization for web service recommendation," in *2014 IEEE International Conference on Web Services*, 2014, pp. 297–304.

[17] 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*, vol. 51, no. 6, pp. 3796–3807, 2021.

[18] W. Zhang, L. Xu, M. Yan, Z. Wang, and C. Fu, "A probability distribution and location-aware resnet approach for qos prediction," vol. 20, pp. 1189–1227, 2021.

[19] Z. Zheng, L. Xiaoli, M. Tang, F. Xie, and M. R. Lyu, "Web service qos prediction via collaborative filtering: A survey," vol. 15, pp. 2455–2472, 2022.

[20] L. Chen, Y. Feng, J. Wu, and Z. Zheng, "An enhanced qos prediction approach for service selection," in *2011 IEEE International Conference on Services Computing*, 2011, pp. 727–728.

[21] "Search based approach to forecasting qos attributes of web services using genetic programming," *Information and Software Technology*, vol. 80, pp. 158–174, 2016.

[22] P. He, J. Zhu, J. Xu, and M. R. Lyu, "A hierarchical matrix factorization approach for location-based web service qos prediction," in *2014 IEEE 8th International Symposium on Service Oriented System Engineering*, 2014, pp. 290–295.

[23] A. Amin, A. Colman, and L. Grunske, "An approach to forecasting qos attributes of web services based on arima and garch models," in *2012 IEEE 19th International Conference on Web Services*, 2012, pp. 74–81.

[24] M. Tang, W. Liang, Y. Yang, and J. Xie, "A factorization machine-based qos prediction approach for mobile service selection," *IEEE Access*, vol. 7, pp. 32 961–32 970, 2019.

[25] Y. Yang, Z. Zheng, X. Niu, M. Tang, Y. Lu, and X. Liao, "A location-based factorization machine model for web service qos prediction," *IEEE Transactions on Services Computing*, vol. 14, no. 5, pp. 1264–1277, 2021.

[26] G. White, A. Palade, and S. Clarke, "Forecasting qos attributes using lstm networks," in *2018 International Joint Conference on Neural Networks (IJCNN)*, 2018, pp. 1–8.

[27] J. Li, H. Wu, J. Chen, Q. He, and C.-H. Hsu, "Topology-aware neural model for highly accurate qos prediction," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 7, pp. 1538–1552, 2022.