Combining Personalized Federated Hypernetworks and Shared Residual Learning for Distributed QoS Prediction

GUOBING ZOU, SHIYI LIN, SHAOGANG WU, SHENGXIANG HU, and SONG YANG, School of Computer Engineering and Science, Shanghai University, Shanghai, China
YANGLAN GAN, School of Computer Science and Technology, Donghua University, Shanghai, China
BOFENG ZHANG, School of Computer and Information Engineering, Shanghai Polytechnic University, Shanghai, China
YIXIN CHEN, Washington University, St. Louis, MO, United States

Connected vehicles due to the high mobility and dynamic network topologies of connected vehicles require accurate QoS that includes high throughput and low latency to assess satisfactory QoE. Existing methods mainly focus on centralized QoS prediction while paying little attention to distributed mobile QoS prediction, making it challenging to protect user privacy information when invoking Web services. Moreover, even though some advanced centralized methods can be transformed into federated architectures, they often face difficulty in capturing latent feature representations of users and services and learning personalized prediction layers between them due to the heterogeneity of the QoS dataset. To address the above issues, we propose a novel framework for distributed QoS prediction, called Combining Personalized **Federated Hypernetworks and Shared Residual Learning for Distributed QoS Prediction (FHR-DQP)**. FHR-DQP adopts the **federated averaging (FedAvg)** to aggregate location-aware residual shared feature information across all clients. Additionally, a hypernetwork is leveraged to generate personalized networks for user-service QoS prediction in each client. These components are integrated as a hybrid framework that performs training using a federated approach and makes personalized QoS predictions within each client. Extensive experiments are conducted on a real-world benchmark QoS dataset called WS-DREAM, containing nearly 2,000,000 historical QoS invocation records. Compared with both centralized and federated competing baselines, the results demonstrate that FHR-DQP achieves the highest performance for distributed QoS prediction, when it provides privacy-preserving of users' QoS invocations.

CCS Concepts: • **Information systems** → **Service discovery and interfaces**; • **Networks** → **Cloud computing**;

Additional Key Words and Phrases: Web Service, Distributed QoS Prediction, Personalized Federated Learning, Hypernetworks, Residual Learning

This work was supported by the National Natural Science Foundation of China (No. 62272290, 62172088) and Shanghai Natural Science Foundation (No. 21ZR1400400).

Authors' Contact Information: Guobing Zou, School of Computer Engineering and Science, Shanghai University, Shanghai, China; e-mail: gbzou@shu.edu.cn; Shiyi Lin (corresponding author), School of Computer Engineering and Science, Shanghai University, Shanghai, China; e-mail: sylin@shu.edu.cn; Shaogang Wu, School of Computer Engineering and Science, Shanghai University, Shanghai, China; e-mail: sgwu@shu.edu.cn; Shengxiang Hu, School of Computer Engineering and Science, Shanghai University, Shanghai, China; e-mail: shengxianghu@shu.edu.cn; Song Yang, School of Computer Engineering and Science, Shanghai University, Shanghai, China; e-mail: yangsong@shu.edu.cn; Yanglan Gan, School of Computer Science and Technology, Donghua University, Shanghai, China; e-mail: ylgan@dhu.edu.cn; Bofeng Zhang, School of Computer and Information Engineering, Shanghai Polytechnic University, Shanghai, China; e-mail: bfzhang@sspu.edu.cn; Yixin Chen, Washington University, St. Louis, MO, United States; e-mail: chen@cse.wustl.edu.

ACM 1556-4703/2025/9-ART23
https://doi.org/10.1145/3709141

ACM Reference format:
Guobing Zou, Shiyi Lin, Shaogang Wu, Shengxiang Hu, Song Yang, Yanglan Gan, Bofeng Zhang, and Yixin Chen. 2025. Combining Personalized Federated Hypernetworks and Shared Residual Learning for Distributed QoS Prediction. *ACM Trans. Autonom. Adapt. Syst.* 20, 3, Article 23 (September 2025), 25 pages.
https://doi.org/10.1145/3709141

1 Introduction

With the rapid advancement of Web 3.0 and 5G edge artificial intelligence technologies, there has been a substantial increase in the number of **Internet of Things (IoT)** applications. These applications often have strict requirements for high **throughput (TP)** and low latency, where the resource management poses substantial challenges due to their high mobility and the dynamic nature of network topologies. Frequent changes in network topology and connection interruptions complicate the maintenance of the **Quality of Experience (QoE)** that users expect [7, 13, 38]. To address the needs of end users, various standardization bodies and organizations are actively involved in developing frameworks for immersive content consumption systems and metaverses, particularly in the context of **Augmented Reality (AR)** and **Virtual Reality (VR)**. In the realm of AR/VR immersive media consumption, enhancing the QoE of users' invoking services is of great importance [2, 29]. Web services, a fundamental component of **Service-Oriented Architecture (SOA)**, encompass functions such as service discovery, selection, composition, recommendation, and mashup creation for facilitating downstream tasks [5, 6, 8, 27, 49]. However, due to different service providers offering a multitude of similar or functionally equivalent Web services, it makes difficulty in selecting satisfactory Web services to meet user QoE in practical scenarios for service consumers in real-world scenarios, such as IoT [1, 18]. To adequately meet user QoE, it is critical to predict vacant QoS value and assess the user's QoS expectations when invoking Web services.

Quality of Service (QoS) is commonly used as a critical factor in describing the non-functional characteristics of network services and plays an important role in selecting Web services with similar or equivalent functionality. Specifically, QoS includes **response time (RT)**, TP, availability, cost, etc. Due to the dynamic network environment and heterogeneous geographical locations, users may observe diverse QoS values when invoking the same Web service. Moreover, it is also impractical and time-consuming for users to invoke all Web services in a constantly evolving network environment and record their corresponding QoS values. Therefore, predicting missing QoS values based on sparse user-service QoS historical invocation records and applying them to downstream SOA scenarios has become a research hotspot [4, 23]. Figure 1 shows an intuitive real-world example that simplifies the cloud-edge-client computing paradigm for understanding the core components involved. It illustrates two distinct ways of predicting QoS: centralized (represented by the red line) and distributed (represented by the yellow line). In scenarios where service users operate terminal devices such as smart cars, mobile phones, and watches to invoke Web services, these devices generate QoS invocation records.

In the **centralized QoS prediction (CQP)**, client-generated QoS data is typically uploaded directly to the central cloud center. Here, the cloud aggregates QoS invocation records from all service users and employs **collaborative filtering (CF)** techniques to learn a QoS prediction model from the collected dataset. Subsequently, it predicts QoS values for all unknown service invocations. However, this direct data transmission approach in CQP may lead to data leakage during the transfer process, compromising users' privacy. Conversely, in the **distributed QoS prediction (DQP)**, after terminal devices generate QoS data, each service user trains a QoS prediction model using the collected local QoS invocation records, and then uploads the trained gradient information to

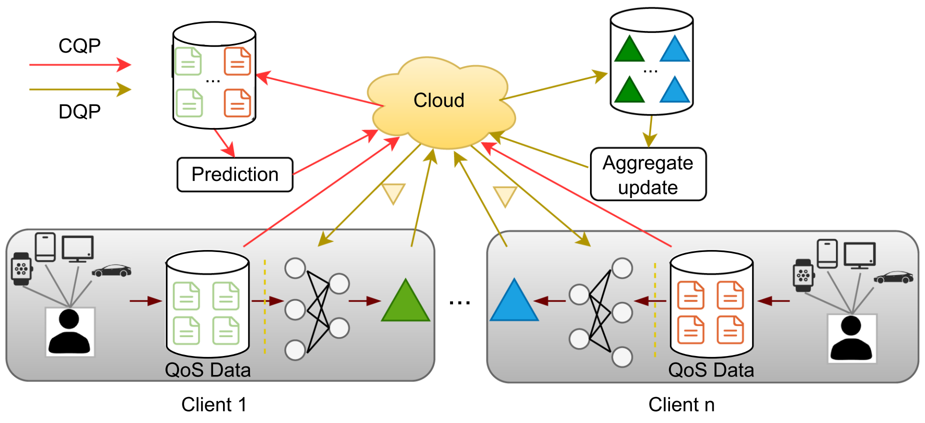


Fig. 1. An intuitive real-world example that illustrates the differences of centralized and distributed QoS prediction.

the cloud center. The cloud center aggregates these gradients, using methods such as averaging or weighted averaging [30], and propagates the aggregated gradient information back to each service user. By doing so, it helps build a shared bridge for CF, enabling continuous parameter updates for each model until global convergence. Thus, a service user can make local QoS predictions for unknown service invocations. In addition, techniques such as differential privacy or homomorphic encryption are often employed to further enhance the security of gradient transmission during the cloud parameter aggregation.

CF, as the most important technique, has received many research investigations to predict the vacant QoS values. CF-based QoS prediction can be categorized into memory-based and model-based approaches. Memory-based CF approaches first calculate similarity to generate similar neighborhoods of users or services, and then predict the unknown QoS based on the similar user/service historical QoS invocations [35]. However, these kinds of methods are highly dependent on data sparsity, and their performance is unsatisfactory when dealing with low-density QoS datasets in real-world application scenarios. To alleviate the issue of data sparsity, model-based methods represent the features of users and services in the latent space instead of using historical QoS invocations directly. Specifically, these kinds of approaches begin by projecting users and services independently into the latent space and then connecting their latent features using downstream operations such as dot product in **matrix factorization (MF)** [20, 46], **multilayer perceptrons (MLP)** in **neural collaborative filtering (NCF)** [12, 41, 53, 54], etc.

Although existing model-based CF methods improve QoS prediction performance, they still suffer from twofold deficiencies. First, most conventional methods primarily concentrate on CQP problems, where user-service history QoS invocations are aggregated in a centralized manner for model training, instead of distributing service records among users. As a result, DQP has not been considered, making it challenging to protect the QoS privacy information of users. Second, despite some model-based methods having adopted various advanced deep learning models for QoS prediction, they have difficulty adapting to federated architectures and accurately extracting latent features of users and services. Especially in the case of the non-IID dataset, the local optimization directions of various clients will deviate from the global optimum, resulting in client drift phenomena [26]. It is observed that the existing models in the federated environment lack deep shared feature extraction and personalized prediction networks for data disparities across clients facing the non-IID QoS dataset, reducing the accuracy of QoS prediction. Therefore, existing model-based

CF methods in terms of considering privacy-preserving cannot effectively learn latent features and personalized interactions of users and services in the non-IID QoS dataset for better DQP performance.

To address the above issues, we propose a novel **personalized federated learning (PFL)** framework for DQP named Combining Personalized **Federated Hypernetworks and Shared Residual Learning for Distributed QoS Prediction (FHR-DQP)**. The framework mainly consists of an initialization phrase and three mutually cooperating modules. Initially, each client collaboratively performs model training based on shared and personalized parameters transmitted by the server. At the beginning, the client uploads the locally trained user-service shared feature extraction layer to the server, and the server performs **federated averaging (FedAvg)** on these shared parameters, where an advanced centralized residual QoS prediction model proposed by us [54] is utilized that considers the location information of users and services. Subsequently, the client selectively uploads the locally trained parameter difference of personalized prediction layer to the server, and the server employs an **hypernetwork (HN)** to generate updated personalized prediction layer parameters individually for each client. The above processes including shared feature extraction and personalized network generation are repeated until global convergence. Finally, we apply the co-trained client network to predict unknown QoS values in a distributed manner, where each client has shared feature extraction and personalized prediction layers for better non-IID QoS prediction performance.

To evaluate the effectiveness of FHR-DQP for DQP, extensive experiments have been conducted on a public large-scale dataset called WS-DREAM, which consists of a total number of 1,974,675 historical user-service QoS invocations [52]. By comparing FHR-DQP with ten federated and centralized competing baselines, the results validate that our proposed FHR-DQP receives the best prediction accuracy on both **mean absolute error (MAE)** and **root mean square error (RMSE)**.

The main contributions of this article are summarized as follows:

— We propose a novel federated personalized framework for DQP. It independently generates personalized network parameters on the server, which protects the privacy information of QoS invocations.
— To improve the accuracy of DQP, we extract shared features by federated aggregation with residual learning to more effectively reveal the QoS characteristics, and leverage an HN to yield personalized network parameters on the server to alleviate the heterogeneity of QoS invocations across multiple users.
— Extensive experiments are conducted on a large-scale real-world QoS dataset. The experimental results demonstrate that the proposed FHR-DQP remarkably outperforms state-of-the-art federated approaches for DQP, even surpassing centralized baselines in the low-density QoS dataset.

The remainder of this article is organized as follows. Section 2 defines and formulates the DQP problem. Section 3 illustrates the overall framework of FHR-DQP and elaborates on its components. Section 4 shows and analyses the experimental results. Section 5 reviews the related work. Section 6 discusses the computational complexity, limitations and future works. Finally, we conclude the article in Section 7.

2 Problem Formulation

In this section, we focus primarily on the understanding of DQP problem by giving a set of formal definitions, clarifying what the solution is to a DQP problem.

Definition 1 (Web Service). We focus on evaluating the non-functional properties of each Web service for QoS prediction. Let $S = \{s_1, s_2, ..., s_m\}$ be a set of Web services where s is described by a five-tuple $s =< ID, RG, AS, Lat, Lon >$. In this tuple, ID identifies the service, while the other attributes collectively represent the location information, including **region (RG)**, **autonomous system (AS)**, **latitude (Lat)**, and **longitude (Lon)**.

Definition 2 (Service User (Client)). A service user is someone who has used one or more Web services. Let $U = \{u_1, u_2, ..., u_n\}$ be a set of users where each u is characterized by a five-tuple $u =< ID, RG, AS, Lat, Lon >$. Here, ID identifies the user and the remaining attributes can be collectively regarded as representing the location information.

Definition 3 (User-Service Invocation Record). A user-service invocation record is defined as a three-tuple $< u, s, y_{u,s} >$, where $u \in U$ has invoked a Web service $s \in S$, with $y_{u,s}$ representing the QoS value. A user-service invocation set Y can be obtained by aggregating all the invocation records among users, where each row represents a user's QoS values for Web service invocations, and each column represents the QoS values of a service invoked by service users. A tuple $< u, s, y_{u,s} >$ is an element of Y if the user u has invoked the service s, otherwise $< u, s, y_{u,s} >\notin Y$.

Definition 4 (CQP Problem). Given a set of users U, a set of services S, and observed QoS invocation matrix Y, the CQP problem can be expressed as a five-tuple $CQP =< U, S, Y, u, s >$, where u is the target user, s is the target service, and $< u, s, y_{u,s} >\notin R$. The target of this problem is to predict the missing QoS value $\widehat{y}_{u,s}$ based on a CQP model. Therefore, a corresponding solution to CQP can be denoted as $< u, s, \widehat{y}_{u,s} >$.

Definition 5 (DQP Problem). Given a set of user-service invocation submatrices $Y' = \{Y_1, Y_2, ..., Y_n\}$, where n is the number of users, a DQP problem is defined as a five tuple $DQP =< U, S, Y', u, s >$, where u is a target user and s is a target service. The solution to a DQP problem is represented by $< u, s, \widehat{y}_{u,s} >$ of the target user u invoking s.

Here, the significant difference between a CQP and DQP problem is that the former can learn the complex non-linear invocation features from the aggregated invocation matrix Y, while DQP models is limited to collaboration by their independent submatrices $Y' = \{Y_1, Y_2, ..., Y_n\}$.

3 Approach

3.1 The Framework of FHR-DQP

Figure 2 is the overall framework of FHR-DQP for DQP. It consists of three crucial components: shared feature extraction, personalized network generation and federated QoS prediction. The process of each component in FHR-DQP is described as below.

— In the stage of shared feature extraction, a two-tower deep residual network in the **neighborhood-based collaborative residual learning (NCRL)** model [54] is used to extract shared features of users and services. The residual network parameters are transmitted to the server for FedAvg aggregation. Subsequently, the client receives global average residual network parameters from the server to facilitate local feature extraction layer updates.

— In the stage of personalized network generation, it aims to learn the personalized user-service prediction layer parameters by applying a federated HN. The personalized prediction network parameters for DQP are also uploaded to the server, while federated aggregation is not performed. Instead, they are employed to update the HN to generate personalized network parameters for individual clients, achieving a personalized update strategy in a targeted QoS manner.

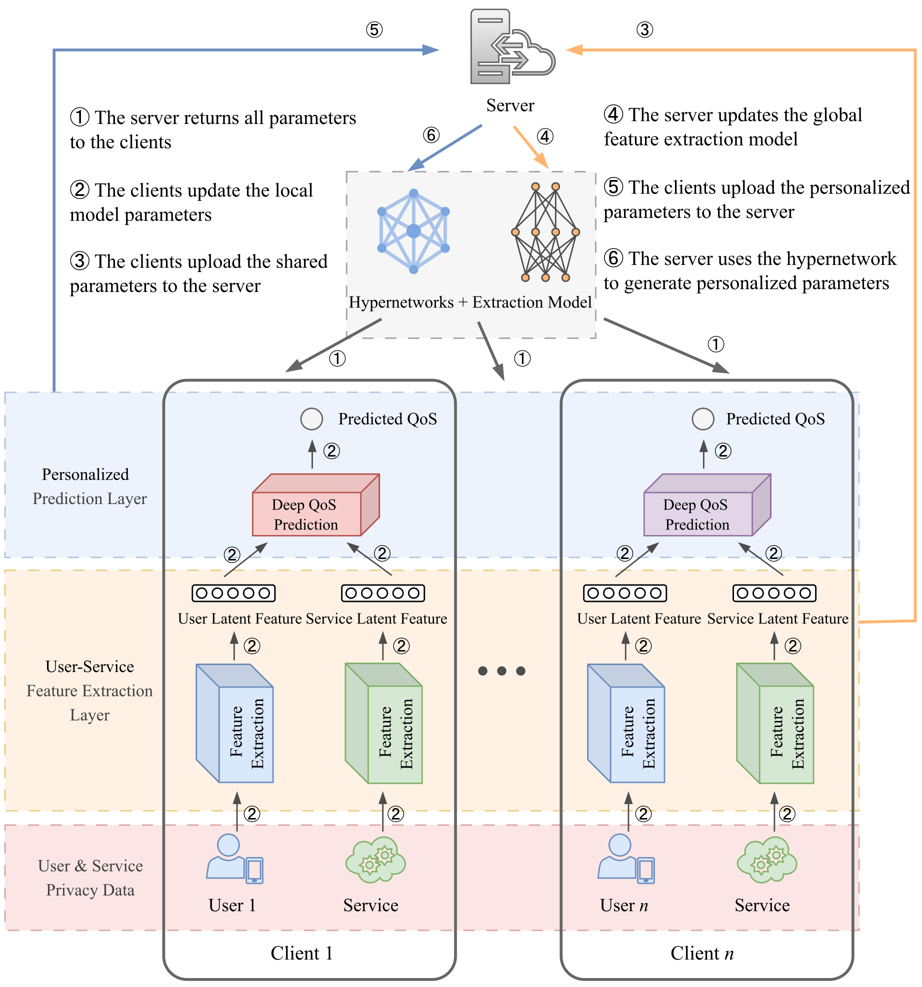

⑤
③
Server
① The server returns all parameters to the clients
② The clients update the local model parameters
③ The clients upload the shared parameters to the server
④ The server updates the global feature extraction model
⑤ The clients upload the personalized parameters to the server
⑥ The server uses the hypernetwork to generate personalized parameters
⑥
④
Hypernetworks + Extraction Model
①
Personalized Prediction Layer
User-Service Feature Extraction Layer
User & Service Privacy Data
Predicted QoS
Deep QoS Prediction
User Latent Feature
Service Latent Feature
Feature Extraction
User 1
Service
Client 1
User n
Client n

Fig. 2. The overall framework of FHR-DQP for DQP.

—In the stage of federated QoS prediction, the converged parameters of the user-service shared feature extraction and personalized prediction layers are integrated into each client model for personalized QoS prediction in distributed service-oriented systems.

3.2 Shared Feature Extraction

3.2.1 Local Feature Extraction with Residual Learning. In this section, we present the forward propagation process with residual learning in a centralized manner, whose parameters can be divided into the shared parameter ϕ and the personalized parameter θ for further DQP. First, we transform the initial features for the user's and service's ID, RG, and AS into their corresponding embedded feature vectors: $E_{u_{ID}}, E_{u_{RG}}, E_{u_{AS}}, E_{s_{ID}}, E_{s_{RG}}, E_{s_{AS}}$. After embedding the initial features, we combine the above embedded features and the Lat & Lon features to obtain a user's and a service's embedded feature vectors (denoted as x_u and x_s). The concatenation can be exressed as follows:

$$x_u = \Phi(E_{u_{ID}}, E_{u_{RG}}, E_{u_{AS}}, u_{Lat}, u_{Lon}) = \begin{bmatrix} E_{u_{ID}} \\ E_{u_{RG}} \\ E_{u_{AS}} \\ u_{Lat\&Lon} \end{bmatrix}, \tag{1}$$

$$x_s = \Phi(E_{s_{ID}}, E_{s_{RG}}, E_{s_{AS}}, s_{Lat}, s_{Lon}) = \begin{bmatrix} E_{s_{ID}} \\ E_{s_{RG}} \\ E_{s_{AS}} \\ s_{Lat\&Lon} \end{bmatrix}, \tag{2}$$

where Φ denotes the concatenation operation; x_u and x_s represent a user's and a service's embedded feature vectors, respectively. These embedded feature vectors are fed into the feature extraction layer, where residual networks are leveraged to extract latent features of the user and service, solving the problem of neural network performance degradation caused by increasing layers. Inspired by NCRL [54], we replace the convolution kernel with a modified MLP residual unit to learn the latent feature representation of the user and service, instead of using conventional convolutional layers [10] in the residual network. The residual feature extraction layer consists of a set of residual units for both the user and service. Each of these residual units is composed of two non-linear layers and an identity shortcut. The input feature vector of the residual unit is added back after being passed through the two non-linear layers. Formally, the feature propagation and aggregation of the residual unit for the user or service is given by:

$$Y^h = \mathbf{W}_0^h g_a(x^h) + \mathbf{b}_0^h, \tag{3}$$

$$Z^h = \mathbf{W}_1^h g_a(Y^h) + \mathbf{b}_1^h, \tag{4}$$

$$x^{h+1} = Z^h + x^h, \tag{5}$$

where x^h is the input of the hth residual unit and x^{h+1} is the output of the hth residual unit; g_a represents the GELU activation function [14], $\mathbf{W}^h_{\{0,1\}} \in \phi$ and $\mathbf{b}^h_{\{0,1\}} \in \phi$ are the parameters in the hth residual layer.

Consequently, given the embedding vector x_u and x_s, the latent features can be extracted and representeded as follows:

$$x'_u = RL_u^H(x_u), \tag{6}$$

$$x'_s = RL_s^H(x_s), \tag{7}$$

where RL_u^H and RL_s^H represent the functions of residual layers with H Residual Units in user-tower network and service-tower network, respectively. x'_u and x'_s are the extracted latent features of a user and a service at the last residual unit. In shared residual learning, the user tower network and service tower network are designed with the same multi-layer architecture to extract latent features of users and services, and they can be trained independently and in parallel through specific network hyperparameter optimization. These features are concatenated and fed into an MLP transformation, as shown below:

$$X_{u,s} = \Phi(x'_u, x'_s) = \begin{bmatrix} x'_u \\ x'_s \end{bmatrix}, \tag{8}$$

$$\hat{y}_{u,s} = \mathbf{W}^O X_{u,s} + \mathbf{b}^O, \tag{9}$$

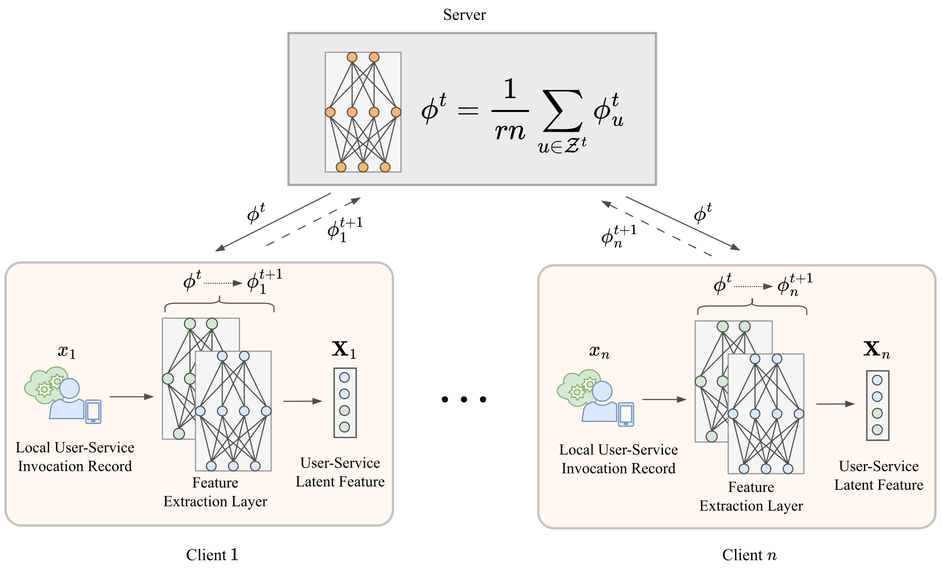

Server
$\phi^t = \frac{1}{rn}\sum_{u\in\mathcal{Z}^t}\phi_u^t$
ϕ^t
ϕ_1^{t+1}
ϕ^t
ϕ_n^{t+1}
$\phi^t \rightarrow \phi_1^{t+1}$
x_1
$\mathbf{X}_1$
Local User-Service Invocation Record
Feature Extraction Layer
User-Service Latent Feature
Client 1
$\phi^t \rightarrow \phi_n^{t+1}$
x_n
$\mathbf{X}_n$
Local User-Service Invocation Record
Feature Extraction Layer
User-Service Latent Feature
Client n

Fig. 3. Shared feature extraction based on federated aggregation.

where Φ represents the concatenation operation. $\mathbf{W}^O \in \theta$ and $\mathbf{b}^O \in \theta$ are weight matrix and bias vector of user-service prediction layer, and $\hat{y}_{u,s}$ is predicted QoS when a target user u invokes a target service s.

3.2.2 Federated Shared Residual Learning. In this section, we utilize federated learning to jointly train the shared feature extraction layer of users and services while preserving local data privacy. In the edge computing environment, services are deployed near mobile users on edge nodes through distributed computing mode. Furthermore, the privacy information of both users and service invocation records are securely stored on edge nodes or terminal devices [28]. Since users and services have similar location information such as RGs, ASs and Lat & Lon, all clients have the shared feature space. Therefore, we share the feature extraction layer across clients to represent user and service features, enabling more efficient extraction of shared user-service invocation features.

Figure 3 illustrates the process of shared feature extraction based on FedAvg aggregation. For client u, the shared feature extraction layer parameters are trained using local data x_u on the client and averaged on the server to learn the global parameter ϕ. The pseudocode involved in the training and FedAvg aggregation is detailed in Algorithm 1. At the start of each training round, the server sets the client participation rate r and then selects rn clients from all available clients to participate in the current round. First, the server sends the initialized parameters to each of the selected clients $u \in \mathcal{Z}^t$. Each client $u \in \mathcal{Z}^t$ initializes its local model using the global parameters and then performs local training on the user-service feature extraction layer. The local training process for each epoch is expressed as follows:

$$\phi_u^t \longleftarrow \mathcal{G}(\phi_u^t, \eta, x_u), \tag{10}$$

where $\mathcal{G}$ denotes the optimization algorithm used for gradient descent during model training and we use Adam optimizer [19] to update all the parameters that need to be optimized in FHR-DQP. ϕ_u^t refers to the model parameter of client u in the tth round, η is the learning rate for model training, and x_u represents the local data of client u. After completing the local training, the updated

Algorithm 1: Shared Feature Extraction based on Federated Aggregation

```
Input: $r$ is the fraction of clients, $n$ is the number of clients, $T$ is the federated training
       rounds; $x_u$ is the privacy data of client $u$, $E$ is the client training epochs, $\eta$ is the
       client learning rate
Output: Shared Feature Global Residual Network $\phi^T$
1  Initialize $\phi^0$
2  for $t = 1, ..., T$ do
3      $\mathcal{Z}^t \longleftarrow$ random set of $rn$ clients
4      foreach client $u(u \in \mathcal{Z}^t)$ do
5          $u$ receive $\phi^t$ from Server: $\phi^t_u \longleftarrow \phi^t$
6          for local epoch $e = 1, 2, ..., E$ do
7              ClientUpdate: $\phi^t_u \longleftarrow \mathcal{G}(\phi^t_u, \eta, x_u)$
8          end
9          $\Delta\phi^t_u \longleftarrow \phi^t_u - \phi^t$
10         $u$ upload $\Delta\phi^t_u$ to Server
11     end
12     $\phi^{t+1} \longleftarrow \phi^t - \frac{1}{rn} \sum_{u \in \mathcal{Z}^t} \Delta\phi^t_u$
13 end
14 return $\phi^T$
```

parameter difference denoted as $\Delta\phi^t_u$ is uploaded to the server for a single update of the global model.

After receiving model parameters from the selected clients, the FedAvg aggregation performs on these model parameters in the server is expressed as:

$$\phi^{t+1} \longleftarrow \phi^t - \frac{1}{rn} \sum_{u \in \mathcal{Z}^t} \Delta\phi^t_u, \tag{11}$$

where the updated global model parameter ϕ^{t+1} is returned to the clients who participated in the current round. These clients use the updated parameter for further model training in subsequent iterations.

3.3 Personalized Network Generation

Although the federated aggregation algorithm in shared feature extraction discussed in the last section combines client-side user and service features, enabling feature sharing among clients without exposing local data and improving model efficiency, the global update based on federated aggregation remains inadequate for the personalized QoS prediction layer between users and services. The primary reason is the large variance in service invocation records among clients, which undermines the convergence of the federated aggregation model. Specifically, in the WS-DREAM dataset, user 1 contains 573 service invocation records while user 228 only has 353. In addition, the average RT for service invocations from user 1 is 2.95 seconds, compared to only 0.47 seconds for user 316. The disparity in service invocation records among clients is considerable in terms of both data quantity and distribution. Therefore, adopting FedAvg aggregation on the non-IID QoS dataset is highly likely to result in client drift, which ultimately hampers the stable convergence of the global model [17].

In this section, HNs are employed to personalize the parameters of the user and service feature prediction layer. HNs are neural network models that generate parameters for another neural

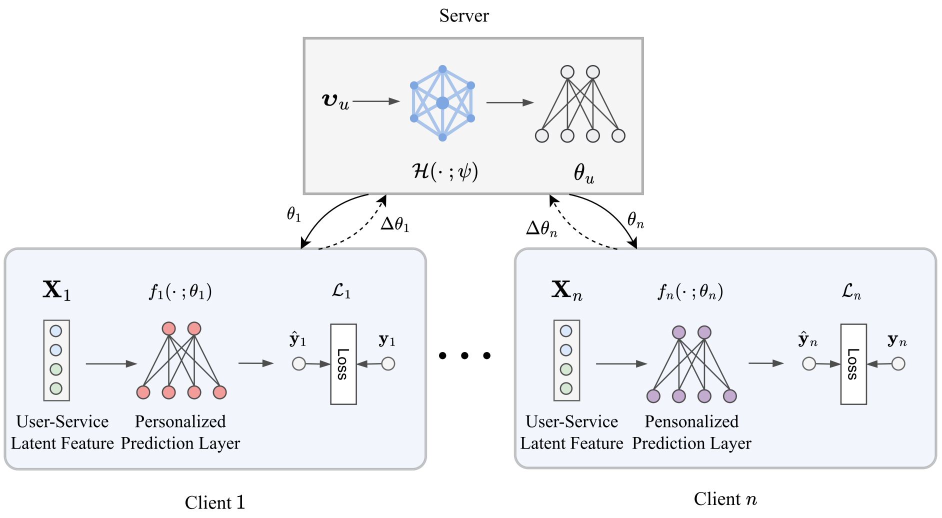


Fig. 4. Personalized network generation based on HNs.

network [9], whose weights can dynamically change based on input conditions, thus producing personalized model parameters for each client. HNs are well-suited to learning a set of personalized models with the same structure, generating desired target networks based on input conditions to optimize personalized federated aggregation in non-IID scenarios [34].

The process of personalized network generation based on HNs is summarized in Figure 4. For each client u, the user-service invocation features $\mathbf{X}_u$ can be obtained by the process in Figure 3 (or Section 3.2.2). After training the personalized prediction layer locally, the difference in parameter update $\Delta\theta_u$ is uploaded to the HN, which can dynamically output personalized model parameters θ_u for different clients. The process consists of three steps and the pseudocode detailed in this process is given in Algorithm 2. Begins with the initialization of the HN model parameters, the personalized parameters for client u are generated as follows:

$$\theta_u = \mathcal{H}(\boldsymbol{v_u};\psi), \tag{12}$$

where $\boldsymbol{v_u}$ is the embedding vector of client u in the server, and θ_u represents the model parameters of user and service personalized prediction layer in client u. Subsequently, the personalized prediction layer of the client is initialized using the parameters θ_u to generate $f_u(\cdot\,;\theta_u)$, which represents the user-service personalized prediction neural network. The predicted QoS value is derived by the personalized prediction layer, by the following expression:

$$\hat{\mathbf{y}}_{u,s} = f_u(\mathbf{x}_{u,s};\theta_u), \tag{13}$$

where $\mathbf{x}_{u,s}$ denotes the invocation features when user u invokes service s, and $\hat{\mathbf{y}}_{u,s}$ is the corresponding QoS prediction value; f_u represents the identity function. During local training, the MAE is utilized to calculate the loss function.

$$\mathcal{L}_u = \frac{1}{K}\sum_{s=1}^{K}\left|\hat{\mathbf{y}}_{u,s} - \mathbf{y}_{u,s}\right|, \tag{14}$$

After completing the local training process, the parameter difference $\Delta\theta_u$ between these new parameters $\bar{\theta}_u$ and the previous ones θ_u are uploaded to the server. At the server-side, the parameters

Algorithm 2: Personalized Network Generation based on Hypernetworks

```
Input: $T$ is the federated training rounds, $\gamma$ is the hypernetworks learning rate; $\mathbf{X}_u$ is the
       personalized feature of client $u$, $E$ is the client training epochs, $\eta$ is the client
       learning rate
Output: The Parameters of Personalized Prediction Layer $\{\theta_1, \theta_2, ..., \theta_n\}$
1  Initialize model parameter $\psi$ of hypernetworks
2  for $t = 1, ..., T$ do
3      foreach client $u \in \mathcal{Z}^t$ do
4          $\theta_u \longleftarrow \mathcal{H}(\boldsymbol{v}_u; \psi)$
5          $\bar{\theta_u} \longleftarrow \theta_u$
6          for $e = 1, 2, ..., E$ do
7              foreach training batch $\mathcal{B} \subseteq \{\mathbf{X}_u, \mathbf{y}_u\}$ do
8                  $\bar{\theta_u} \longleftarrow \bar{\theta_u} - \eta \nabla_{\bar{\theta_u}} \mathcal{L}(\bar{\theta_u}; \mathcal{B})$
9              end
10         end
11         $\Delta\theta_u \longleftarrow \bar{\theta_u} - \theta_u$
12         $u$ upload $\Delta\theta_u$ to Server
13         Server update $\psi$ and $\boldsymbol{v}_u$:
14         $\psi \longleftarrow \psi - \gamma \nabla_\psi \theta_u^\top \Delta\theta_u$
15         $\boldsymbol{v}_u \longleftarrow \boldsymbol{v}_u - \gamma \nabla_{\boldsymbol{v}_u} \psi^\top \nabla_\psi \theta_u^\top \Delta\theta_u$
16     end
17 end
```

of the HN ψ and the embedding vectors $\boldsymbol{v}_u$ are updated based on Equations (16) and (17), respectively:

$$\Delta\theta_u \longleftarrow \bar{\theta_u} - \theta_u, \tag{15}$$

$$\psi \longleftarrow \psi - \gamma \nabla_\psi \theta_u^\top \Delta\theta_u, \tag{16}$$

$$\boldsymbol{v}_u \longleftarrow \boldsymbol{v}_u - \gamma \nabla_{\boldsymbol{v}_u} \psi^\top \nabla_\psi \theta_u^\top \Delta\theta_u, \tag{17}$$

where γ represents the learning rate of the HN, and $\gamma \nabla_\psi \theta_u^\top \Delta\theta_u$ and $\gamma \nabla_{\boldsymbol{v}_u} \psi^\top \nabla_\psi \theta_u^\top \Delta\theta_u$ indicate the gradient calculated by the chain rule in back propagation. Therefore, the HN parameters including parameters $\psi, \boldsymbol{v}_1, \cdots, \boldsymbol{v}_n$ are updated according to Algorithm 2.

3.4 Federated QoS Prediction

FHR-DQP employs FedAvg aggregation to optimize residual layer parameters of user and service feature extraction globally and utilizes HNs to generate personalized parameters of the user and service prediction layer, enabling personalized QoS prediction in distributed scenarios. As mentioned in Sections 3.2.2 and 3.3, FHR-DQP requires learning two sets of model parameters for federated QoS prediction: the shared parameter ϕ of the user and service feature extraction layer and the personalized parameter θ_u of the user and service prediction layer. The above federated QoS prediction process for a given client u can be represented as follows:

$$\hat{\mathbf{y}}_u = \mathcal{F}_u(\mathbf{x}_u; \phi; \theta_u), \tag{18}$$

where $\mathcal{F}_u$ represents the federated QoS prediction model of client u. Each client has its IID data distribution in the federated QoS prediction scenario, thus the overall optimization objective of

Table 1. Statistics of WS-DREAM Dataset

Item name	Value
Users	339
Services	5,825
Service Invocations	1,974,675
Users' RGs	31
Users' ASs	137
Services' RGs	74
Services' ASs	992
Services' Providers	2,699

FHR-DQP is:

$$\min_{\phi,\theta_1,\ldots,\theta_n\in\mathbb{R}^d} F(\phi;\theta) := \frac{1}{n}\sum_{u=1}^{n}\mathbb{E}_{(\mathrm{x},\mathrm{y})\sim\mathcal{D}_u}[\mathcal{L}_u(\phi;\theta_u;\mathbf{x},\mathbf{y})], \tag{19}$$

where $\mathcal{D}_u$ denotes the data distribution of client u and $\mathcal{L}_u$ denotes the loss function on client u. Here, MAE is used as the loss function during training in the article, as shown below:

$$\mathcal{L}_u(\phi;\theta_u;\mathbf{x},\mathbf{y}) = \frac{1}{K}\sum_{s=1}^{K}|\mathcal{F}_u(\mathbf{x}_{u,s};\phi;\theta_u) - \mathbf{y}_{u,s}|, \tag{20}$$

where K denotes the number of QoS samples on client u, $\mathbf{x}_{u,s}$ and $\mathbf{y}_{u,s}$ denote the sth training sample data on client u.

4 Experiments

4.1 Experimental Setup and Dataset

All the experiments are carried out on our workstation equipped with two NVIDIA RTX 4090 GPUs, two Intel(R) Xeon(R) Silver 4210R @2.40 GHz CPUs and 1.0TB RAM. The components of FHR-DQP in the experiments are implemented by Python 3.7.15 with Pytorch 1.13.1. To validate the effectiveness of the DQP performance of FHR-DQP, we conduct extensive experiments on a real-world Web service QoS dataset called WS-DREAM[1] [52], which has been widely used for service QoS prediction. Here, it consists of two types of service invocation QoS criteria, including RT and TP, which collected from 339 users and 5,825 Web services with 1,974,675 historical QoS invocation records. In addition, the contextual location information is provided in RT and TP, such as RG, Lat, and Lon. Comprehensive statistics of the QoS dataset is shown in Table 1.

The QoS dataset of RT or TP can be formalized as a user-service QoS matrix. In this matrix, each row represents a collection of QoS values that a user invokes from all services, and each column represents a set of QoS values that a service has been invoked by all users. Due to the sparsity of user-service interactions in real-world scenarios, QoS dataset is trained with four different low densities: 2.5%, 5%, 7.5%, and 10% on RT and TP, respectively. For the comparisons of QoS prediction accuracy, remaining QoS samples under each density are treated as testing data in the experiments.

[1]https://wsdream.github.io/

4.2 Evaluation Metrics

MAE and RMSE are used as the two evaluation metrics to measure the accuracy of QoS prediction among the competing approaches in the experiments. MAE and RMSE are defined as follows:

$$MAE = \frac{\sum_{u,s} |\mathbf{y}_{u,s} - \hat{\mathbf{y}}_{u,s}|}{N}, \tag{21}$$

$$RMSE = \sqrt{\frac{\sum_{u,s} (\mathbf{y}_{u,s} - \hat{\mathbf{y}}_{u,s})^2}{N}}, \tag{22}$$

where $\mathbf{y}_{u,s}$ is the original QoS value of a target user u invoking a service s and $\hat{\mathbf{y}}_{u,s}$ is the predicted QoS; N is the number of test samples of predicted QoS values.

MAE reflects the overall accuracy of QoS prediction by averaging absolute deviations from the original QoS values. Compared with MAE, RMSE is more sensitive to outliers as it assigns relatively higher weights to large errors in predicted QoS values. Smaller deviations on MAE and RMSE indicate better performance of QoS model prediction.

4.3 Competing Approaches

To evaluate the effectiveness of FHR-DQP, we compare it with ten competing approaches, including three Memory-based, five Model-based and two FL-based methods. They are described as below.

— *Memory-Based Methods*:
  (i) *UPCC* [35]: It is a user-based CQP method that calculates a set of similar users as the neighborhood of a target user by **Pearson Correlation Coefficient (PCC)**, and combines the average QoS values of the target user with the deviation values of similar users to achieve QoS prediction.
  (ii) *IPCC* [51]: It is an item-based CQP method that calculates a set of similar items as the neighborhood of a target item by PCC, and combines the average QoS values of the target item with the deviation values of similar items to achieve QoS prediction.
  (iii) *UIPCC* [51]: It is a hybrid CF method by the combination of UPCC and IPCC, which applies a weighting coefficient to adjust their relative importance. It is a memory-based representative approach for CQP.

— *Model-Based Methods*:
  (i) *PMF* [31]: It is a variant MF method for CQP, which leverages prior Gaussian distribution to optimize hyperparameters in probability model.
  (ii) *FM* [44]: It is a reinforced MF method for CQP, which integrates the linear regression model and the MF model to model multiple variable interactions with linear complexity.
  (iii) *NCF* [12]: It is an advanced NCF method that combines multi-layer perceptron and generalized MF, which learns complex non-linear interactions between users and services to achieve CQP.
  (iv) *NDMF* [53]: It is an advanced neighborhood-aware NCF method, which integrates user-selected neighborhoods into collaborative loss function via a **deep neural network (DNN)** to achieve neighborhood-integrated CQP.
  (v) *DNM* [41]: It is an advanced contextual-aware NCF method, which maps contextual features into a shared latent space and integrates their high-order interactions through DNN to achieve multi-attribute CQP.

Table 2. Parameter Settings

Parameter	Value	Description
r	30%	the fraction of clients on each round
T	3,000	the federated training rounds
E	5	the number of client epochs
$\mathcal{B}$	-1	the batch size of client network
η	0.005	the learning rate of client network
γ	0.005	the learning rate of HNs
$\mathcal{E}$	16	the embedding size of HNs
$\mathcal{N}$	$< 200, 200, 200 >$	the structure of HNs
$\mathcal{R}$	$< 128, 256, 256, 64 >$	the structure of residual layer

—*FL-Based Methods*:

(i) *ENMF* [47]: It is a federated MF method for DQP that preserves user data privacy without aggregating raw QoS data for each client. It is a recent approach for DQP and is applied as an FL-based baseline.

(ii) *FedNCF*: It is a federated QoS prediction method based on NCF [12] for DQP. We apply FedAvg to federate NCF, where all parameters of the client model are included in the parameter aggregation phase.

4.4 Experimental Results and Analyses

To guarantee the fairness of comparison, we tune the parameters of all CQP approaches directly as they are suggested with the best performance in reference benchmark experiments, and the parameter settings of all federated QoS prediction approaches are shown in Table 2. The structure of HNs $\mathcal{N}$ is $< 200, 200, 200 >$ and residual layer $\mathcal{R}$ is $< 128, 256, 256, 64 >$. The learning rate of client network η is 0.005 and the learning rate of HNs γ is 0.005. We tune the hyperparameter fraction of clients on each round r in {10%, 30%, 50%, 100%}, the number of client epochs E in {1, 5, 10, 20}, the embedding size of HN $\mathcal{E}$ in {2,4,8,16,32} and the batch size of client network $\mathcal{B}$ in {32,64,128,-1}, where $\mathcal{B} = -1$ indicates that the client's local data is treated as a single batch. All approaches are trained on RT and TP training datasets, and QoS prediction performance is evaluated by comparing MAE and RMSE on the test samples. To prevent deviations, we run FHR-DQP and the competing methods three times to calculate the average results of predicted QoS for comparative analysis to reflect the fairness of the experiments.

Tables 3 and 4 show the QoS prediction experimental results on RT and TP among both centralized and federated competing baselines. The best results of distributed and centralized methods each column are marked in the form of dark and underline, respectively. It is observed that all competing methods demonstrate a reduction on MAE and RMSE as the QoS density increases from 2.5% to 10% on RT and TP, indicating the accuracy improvement in the QoS prediction. The improvement can be attributed to the increase in the number of available historical QoS, facilitating the calculation of similar neighborhoods in memory-based CF methods and enabling better learning of the invocation relationships between users and services during model training in model-based CF methods.

UPCC, IPCC, and UIPCC, as traditional memory-based CF methods, rely heavily on calculating similar users and services based on historical QoS invocations. As a result, they perform poorly in QoS prediction performance because they are easily affected by the density of the user-service QoS invocation matrix. As the traditional model-based MF method, PMF introduces a probabilistic model to achieve the decomposition of user and item features, which partially alleviates the sparsity of

Table 3. Performance Comparisons of QoS Prediction on RT

Methods	Density = 2.5%			Density = 5%			Density = 7.5%			Density = 10%		
	MAE	RMSE	NMAE	MAE	RMSE	NMAE	MAE	RMSE	NMAE	MAE	RMSE	NMAE
UPCC	0.7679	1.7888	0.8465	0.6166	1.5287	0.6795	0.5724	1.4197	0.6322	0.5550	1.3800	0.6120
IPCC	0.7380	1.7749	0.8105	0.6727	1.6981	0.7407	0.6471	1.6728	0.7147	0.6261	1.6367	0.6892
UIPCC	0.7515	1.7549	0.8261	0.6079	1.5023	0.6714	0.5670	1.4064	0.626	0.5502	1.3684	0.6058
PMF	0.6492	1.6149	0.7150	0.5753	1.4422	0.6332	0.5252	1.3370	0.5786	0.4954	1.2778	0.5455
FM	0.6876	1.5321	0.7566	0.6203	1.4406	0.6837	0.5592	1.3281	0.6165	0.5392	1.3052	0.5947
NCF	0.5444	1.5472	0.6007	0.4652	1.3904	0.5115	0.4159	1.3583	0.4567	0.3783	1.3040	0.4170
NDMF	0.5393	1.4036	0.5935	0.4880	1.3495	0.5365	0.4416	**1.2793**	0.4862	0.4304	**1.2349**	0.4739
DNM	0.4777	1.4829	0.5253	0.4147	1.4274	0.4553	**0.3843**	1.3745	**0.4228**	**0.3628**	1.3567	**0.3997**
ENMF	0.7031	1.7358	0.7752	0.6104	1.5048	0.6717	0.5254	1.3446	0.5792	0.4965	1.2821	0.5472
FedNCF	0.5926	1.6108	0.6539	0.4923	1.4780	0.5414	0.4606	1.3815	0.5059	0.4315	1.3747	0.4754
FHR-DQP	**0.4389**	**1.3855**	0.4835	**0.3988**	**1.2990**	**0.4392**	0.3879	1.2897	0.4279	0.3650	1.2532	0.4029
Gains	8.12%	1.29%	7.96%	3.83%	3.74%	3.54%	−0.94%	−0.81%	−1.21%	−0.61%	−1.48%	−0.80%

Table 4. Performance Comparisons of QoS Prediction on TP

Methods	Density = 2.5%			Density = 5%			Density = 7.5%			Density = 10%		
	MAE	RMSE	NMAE	MAE	RMSE	NMAE	MAE	RMSE	NMAE	MAE	RMSE	NMAE
UPCC	38.84	93.33	0.8151	25.42	65.68	0.5362	22.96	58.83	0.4827	21.25	57.24	0.4465
IPCC	37.25	97.50	0.7827	32.96	89.85	0.6898	31.02	87.56	0.6521	29.76	84.90	0.6277
UIPCC	36.87	91.85	0.7777	25.18	65.37	0.5293	22.93	59.17	0.4820	22.43	57.56	0.4720
PMF	30.18	74.67	0.6349	24.20	56.03	0.5090	22.52	55.97	0.4730	19.83	51.75	0.4175
FM	28.57	72.30	0.6020	21.59	57.60	0.4521	19.47	50.51	0.4086	17.69	48.62	0.3731
NCF	24.21	64.15	0.5066	18.68	54.65	0.3921	15.88	48.38	0.3327	14.40	46.22	0.3013
NDMF	20.31	58.17	0.4275	16.38	50.96	0.3451	15.28	47.60	0.3215	13.93	**43.91**	0.2934
DNM	18.29	65.65	0.3851	14.85	59.33	0.3125	13.82	56.55	0.2913	**12.92**	54.50	**0.2713**
ENMF	31.24	76.93	0.6540	26.92	68.86	0.5661	24.64	56.04	0.5174	19.62	53.37	0.4135
FedNCF	24.38	70.01	0.5125	18.52	57.27	0.3896	17.42	52.83	0.3673	16.02	51.40	0.3354
FHR-DQP	**17.26**	**55.52**	**0.3631**	**14.63**	**48.81**	**0.3082**	**13.72**	**46.27**	**0.2881**	13.16	44.65	0.2758
Gains	5.63%	4.56%	5.71%	1.48%	4.22%	1.38%	0.72%	2.79%	1.10%	−1.86%	−1.69%	−1.66%

user-service QoS invocation relationships. Compared with traditional memory-based CF methods, it demonstrates better QoS prediction performance. Additionally, FM focuses more on learning linear feature interactions and achieves better QoS prediction performance than MF. To further improve QoS prediction accuracy, advanced model-based neural CF methods are designed to model the deep non-linear interaction relationships between users and services. NCF employs an MLP to learn the non-linear interaction relationships from the embedding feature vectors of users and services. Although NCF outperforms traditional model-based CF methods, it remains inferior to NDMF and DNM because NCF neglects location information when extracting the latent features of users and services. NDMF considers the location information of users and calculates similar users as neighbors, which is integrated with the loss function to train the QoS prediction model, leading to significant prediction performance improvement on RMSE across all densities. DNM makes full use of QoS location information during the initial feature embedding and then integrates it into an MLP network to better capture implicit non-linear interaction relationships. Compared with NCF and NDMF, DNM achieves the best performance on MAE even though it performs relatively poorly on RMSE.

Table 5. Performance Comparisons of Our CQP and DQP Approaches on RT and TP

QoS	Methods	Density = 2.5%		Density = 5%		Density = 7.5%		Density = 10%	
		MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
RT	CQP	0.4220	1.3508	0.3655	1.2904	0.3458	1.2657	0.3459	1.2400
	DQP	0.4389	1.3855	0.3988	1.2990	0.3879	1.2897	0.3650	1.2532
TP	CQP	16.44	53.56	13.84	47.57	13.06	45.09	12.27	42.51
	DQP	17.26	55.52	14.63	48.81	13.72	46.27	13.16	44.65

Table 6. Results of Ablation Experiments on RT

Methods	Density = 2.5%			Density = 5%			Density = 7.5%			Density = 10%		
	MAE	RMSE	NMAE	MAE	RMSE	NMAE	MAE	RMSE	NMAE	MAE	RMSE	NMAE
FR-DQP	0.4745	1.4589	0.4277	0.5234	1.3455	0.4709	0.4089	1.3279	0.4508	0.3855	1.2941	0.4248
FH-DQP	0.5076	1.4244	0.5576	0.4557	1.4042	0.5028	0.4286	1.3856	0.4736	0.4056	1.3409	0.4467
FPR-DQP	0.4586	1.3970	0.5055	0.4089	1.3425	0.4498	0.3817	1.2995	0.4212	0.3793	1.2761	0.4190
FHR-DQP	0.4389	1.3855	0.4835	0.3988	1.2990	0.4392	0.3879	1.2897	0.4279	0.3650	1.2532	0.4029

Table 7. Results of Ablation Experiments on TP

Methods	Density = 2.5%			Density = 5%			Density = 7.5%			Density = 10%		
	MAE	RMSE	NMAE	MAE	RMSE	NMAE	MAE	RMSE	NMAE	MAE	RMSE	NMAE
FR-DQP	17.79	56.03	0.3747	15.06	49.84	0.3194	14.06	47.19	0.2966	13.72	46.42	0.2921
FH-DQP	20.29	59.22	0.4245	18.20	55.51	0.3831	17.03	53.26	0.3575	16.20	50.15	0.3398
FPR-DQP	17.73	56.90	0.3733	14.99	49.31	0.3346	14.01	47.03	0.3158	13.89	46.29	0.3093
FHR-DQP	17.26	55.52	0.3631	14.81	48.81	0.3082	13.72	46.27	0.2881	13.16	44.65	0.2758

As the QoS density increases from 2.5% to 10% on RT and TP, it can be observed that all federated learning methods achieve lower MAE and RMSE. This is because with increased QoS density, there are more QoS invocation records available for each client to train their local model, leading to better trained models that improve the federated QoS prediction accuracy. FHR-DQP improves the QoS prediction accuracy at all densities in comparison to EFMF and FedNCF, demonstrating the advantage of its PFL framework for DQP. Although FHR-DQP exhibits slightly worse performance in certain high QoS density scenarios, such as when TP density is equal to 7.5% or 10% on MAE compared to DNM, and 10% on RMSE compared to NDMF, it generally outperforms centralized baselines on both RT and TP. The possibility [16, 32] is that centralized competing baselines are better able to capture strong collaborative relationships in high-density datasets, whereas federated methods are more likely to fall into unpredicted suboptimal points when dealing with larger amounts of non-IID QoS datasets. As shown in Table 5, although there is a decrease in the accuracy of our federated approach DQP compared to our centralized approach CQP, the relatively slight decrease highlights the advancement of the proposed distributed federated framework in sustaining the effectiveness of missing QoS prediction.

4.5 Ablation Study and Hyper-Parameters Impact

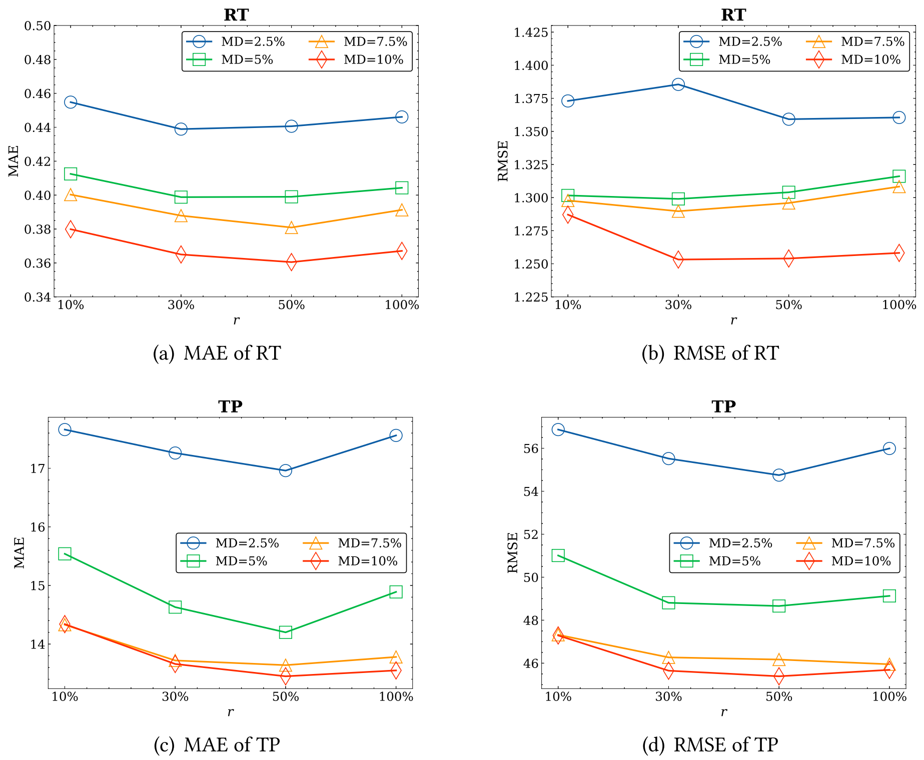
4.5.1 Ablation Study. Ablation experiments are conducted to validate the effectiveness of FHR-DQP proposed in this article. Tables 6 and 7 report the results of ablation experiments among FHR-DQP and its three variants, respectively. In the experiments, FR-DQP uses the FedAvg to perform global updates on all parameters of the model, which is the most commonly used aggregation algorithm in federated learning. Although FR-DQP is facilitate to deploy, it requires consistent data distribution across different clients to achieve satisfactory QoS prediction performance. In real-world scenarios, data distributions on different clients often vary considerably, resulting in the client drift problem and a significant QoS performance drop when applying the FedAvg. FH-DQP employs an HN to perform personalized updates on all parameters of the model without shared feature extraction, which severely undermines collaborative learning of shared features. It can be observed that FR-DQP is vulnerable to data heterogeneity, while FH-DQP has the worst prediction performance mainly because it ignores the common information of user and service features. Specifically, due to the fact that QoS network geographic information characteristics do not vary with different users, FH-DQP treats all parameters in the user network as personalized parameters, neglecting the common characteristics brought about by QoS network geographic information. For instance, when different users are located in the same AS, the inherent characteristics of that AS should be the same. Consequently, FH-DQP performance is inferior to the FR-DQP. FPR-DQP is a CQP approach based on the FedProx algorithm, which improves and optimizes the FedAvg for federated parameter aggregation. By comparing with FR-DQP and FPR-DQP, the effectiveness of the HN is demonstrated. In addition, as the amount of data increases, the HN can be applied in resource-constrained environments by deploying a large-scale model on the server side. With the consideration of both residual FedAvg update and HNs generation, FHR-DQP exhibits the best performance under different QoS densities.

4.5.2 Impact of the Fraction of Clients. In real-world scenarios, there are numerous clients connected to the server, making it difficult to aggregate model parameters from all clients during federated learning. Typically, a random subset of clients participate in federated training and aggregation based on the participation rate in each federated round. To test the performance impact of client participation rate on the model, we set its value as 10%, 30%, 50%, and 100% under different QoS densities, respectively, and the experimental results are shown in Figure 5.

It can be seen that as the client participation rate increases, both MAE and RMSE generally show a decreasing trend, because when the client participation rate is too low, the small number of training samples may lead to underfitting of the federated model. However, when all clients participate in every round of federated training, it does not significantly improve the prediction accuracy of the federated model. In some cases, it may even result in decreased performance, such as an increased MAE value on TP at 5% density. Although increasing the client participation rate requires longer federated training time, it does not yield significant QoS performance improvement. As a result, it is generally recommended to set the client participation rate between 30% and 50% to achieve better QoS prediction performance.

4.5.3 Impact of the Depth of Hypernetworks. An HN is a neural network model that generates parameters for another neural network and is used to generate network parameters for the personalized user-service prediction layer in FHR-DQP. The generated network parameters vary with the depth of the HNs, which in turn affects QoS prediction performance. To test the performance impact of HNs with different depths on QoS prediction, we conduct experiments with depths ranging from 1 to 5 with a fixed number of 200 neurons per layer under different QoS densities, and the experimental results are shown in Figure 6.

It is observed that as the QoS density increases, the network generated by the HN can better match the actual data distribution, leading to improved QoS prediction accuracy. As the number of HN layers increases, MAE and RMSE generally show a decreasing trend, followed by an increasing trend. Specifically, the decreasing trend reaches its peak when the depth increases from 1 to 3, and then the accuracy decreases when it increases from 3 to 5. The main reason for this phenomenon is that when using an HN with a small number of MLP layers, fewer parameters are insufficient to



(a) MAE of RT (b) RMSE of RT

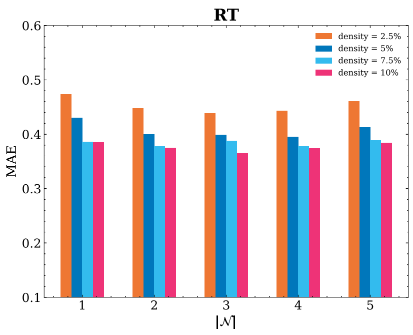
(c) MAE of TP (d) RMSE of TP

Fig. 5. Performance impact of *r* on FHR-DQP under different QoS densities.

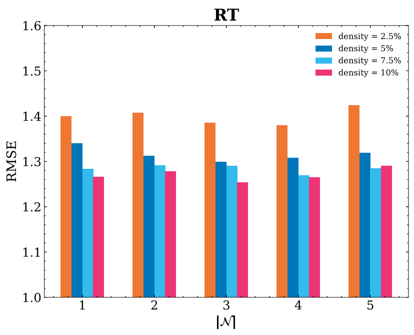
learn complex interaction relationships. Conversely, when the number of MLP layers is large, it may lead to an overfitting problem of network parameters. Both HN situations in HN are not beneficial to generating interaction parameters between users and services for QoS prediction. Therefore, considering the QoS prediction performance under different QoS densities and the scale of the HN, the HN depth is recommended to be 2 or 3 to achieve the optimal QoS prediction performance of FHR-DQP.

4.5.4 Impact of the Number of Client Epochs. The number of local training epochs determines the convergence efficiency of the client model. In theory, the more epochs local training has, the better the convergence efficiency of the client model will be. However, the risk of overfitting in the global model increases as the client model reaches its convergence parameters in federated learning [40]. To test the influence of the number of local training epochs on prediction performance, we vary its value by 1, 5, 10, and 20 with 2.5% QoS density, respectively, and the results are shown in Figure 7.

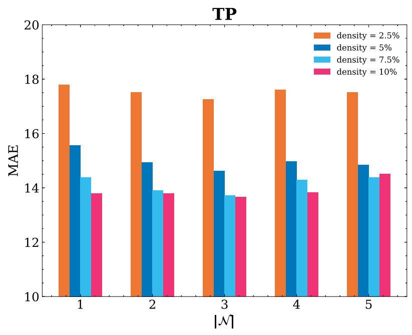
When the epoch is set to 1 and then the federated aggregation is applied, both MAE and RMSE exhibit an oscillating trend, indicating that it is challenging for the model to converge. The primary reason for this phenomenon is that the number of local training epochs is limited, hampering the convergence of the client model. When the epoch is set to 5, 10, or 20, MAE and RMSE tend to stabilize, indicating that the global model has converged. Nevertheless, the final convergence performance varies across the three different client epoch settings. Among them, the fitting ability of the global model is best when the number of epochs is set to 5 or 10 on RT, and the model shows the supreme fitting ability when running repeatedly for 5 epochs on TP. In addition, it can be seen



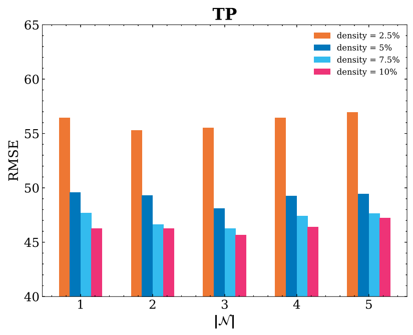
(a) MAE of RT



(b) RMSE of RT



(c) MAE of TP



(d) RMSE of TP

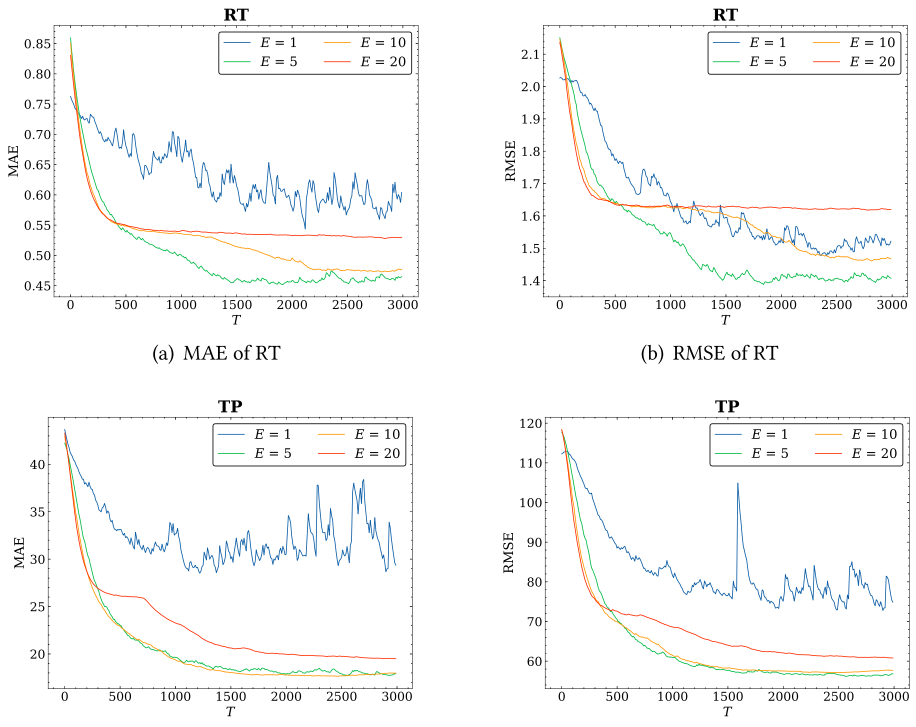
Fig. 6. Performance impact of $|\mathcal{N}|$ on FHR-DQP under different QoS densities.

that the prediction performance of the global model deteriorates gradually as the number of client epochs increases. The main reason is that the global model overfits when the client model receives too many training iterations, reducing its prediction performance. Based on the above analysis, when the number of epochs is set to 5 in our experiments, FHR-DQP achieves the best prediction accuracy.

5 Related Work

5.1 CQP

5.1.1 Memory-Based Methods. This kind of approaches firstly computes similarities between users or services, and then predicts unknown QoS by calculating average QoS and deviation migration based on historical QoS invocations. Shao et al. [35] introduced a user-based CF approach that predicts QoS values by finding similar users through PCC. Zheng et al. [51] proposed a hybrid CF approach called WSRec, which combines user-based and service-based CF by computing predicted QoS values with an adjusted weighting coefficient. Sun et al. [37] and Wu et al. [43] proposed enhanced similarity calculation algorithms for QoS prediction. Sun et al. introduced a normalization technique called normal recovery, whereby the QoS values of users are scaled to the same range to unify similarity across different multi-dimensional latent spaces. Wu et al. proposed a ratio-based approach to calculating user or service similarity. Chen et al. considered a wide range of QoS data and integrates it into a CF model by using a Top-K strategy to identify similar neighbors and combining bias information to generate QoS predictions. However, memory-based approaches



(a) MAE of RT (b) RMSE of RT

(c) MAE of TP (d) RMSE of TP

Fig. 7. Performance impact of T on FHR-DQP under different client epochs.

face a critical challenge due to the sparsity of historical QoS invocations, which substantially undermines their QoS prediction performance in real-world applications.

5.1.2 *Model-Based Methods.* MF and its variants are widely used as traditional model-based methods for QoS prediction, which directly embed user/service ID as a vector and model their linear interactions with inner product. Zhang et al. [46] designed a variant of MF named **non-negative matrix factorization (NMF)** by enforcing a non-negativity constraint in the linear model. Mnih et al. [31] proposed **probabilistic matrix factorization (PMF)**, which is another variant of MF that introduces a probability model to optimize the MF model. NMF and PMF enhance QoS prediction performance remarkably compared to MF. To further improve QoS prediction accuracy, researchers have proposed hybrid models that integrate neighborhood-based approaches with MF. Li et al. [25] proposed a location-aware reputation-based MF model for QoS prediction. which identifies the user neighborhood based on user's reputation and geographical information to reinforce the feature representation of users and services. Compared to memory-based approaches, MF and its variants can better predict vacant QoS by learning linear interaction relationships between latent features of users and services. Nevertheless, they cannot effectively capture the implicitly complex non-linear interaction relationships from user-service historical QoS invocations, resulting in unsatisfactory accuracy of QoS prediction.

Deep learning approaches have been widely used to solve QoS prediction problems. NCF [11] leverages an MLP to learn the interactive function of non-linear relationships, which has been applied for effective QoS prediction. Recent studies have proposed various deep learning models based on NCF to further improve QoS prediction accuracy. Wu et al. [41] proposed a **deep neural**

model (DNM) that considers multiple attributes of users and services, where contextual features are mapped into a shared latent space and their high-order interactions are captured through an MLP network. Xia et al. [45] proposed a QoS prediction approach that introduces implicit and explicit features into the initial dense vector representation and utilizes a convolutional neural network to compress and optimize the procedure of feature extraction. Li et al. [22] proposed a **topology-aware neural (TAN)** model that introduces network topology structure to solve the QoS prediction problem. By incorporating the network topology information, the TAN model can effectively capture the complex relationships among users and services. Zou et al. [54] proposed an NCRL model that utilizes a location-aware two-tower deep residual network for collaborative prediction, which is applied as the client model of our FHR-DQP.

These models have shown significant performance improvements for QoS prediction by effectively learning the complex non-linear interactive relationships among users and services. Although extensive deep learning models have been studied to enhance the accuracy of QoS prediction, most of them concentrate on developing a CQP model, neglecting the privacy-preserving significance of user-service QoS invocations.

5.2 PFL

Heterogeneous datasets are widespread in real-world applications, and many efforts in PFL [36] have been devoted to addressing the theoretical and practical challenges when applying federated learning to the non-IID dataset. PFL can be broadly classified into two categories: global model personalization and local model personalization.

There are two main global model personalization strategies: data-based and model-based. Data-based methods employ data augmentation to expand heterogeneous datasets. Zhao et al. [50] proposed data sharing strategies for federated learning by creating a small subset of data that is globally shared across all edge devices, which reduces the risk of overfitting to local data and improves prediction performance. Wu et al. [42] proposed a generative convolutional autoencoder for personalized health monitoring, which refines the model with a generated class-balanced dataset from the user's personal data. Besides, client selection mechanisms are also designed to sample from more IID distributed data to improve the global model generalization. Wang et al. [39] proposed a mechanism based on a reinforcement learning framework for device selection in federated learning, which selects a subset of devices in each communication round to maximize a reward that encourages the increase of validation accuracy and penalizes the use of more communication rounds. Model-based methods typically implement regularization between global and local models or apply **contrastive learning (CL)** to close the distance between local and global models. Li et al. [24] performed CL at the model level, which utilize the similarity between model representations to correct the local training of individuals. In addition, some researchers introduced meta-learning and transfer learning to accelerate the convergence speed of the global model and improve the effectiveness of local model personalization [16, 21].

Local model personalization strategies can be divided into architecture-based and similarity-based approaches. Architecture-based methods utilize parameter decoupling to train model networks locally to perform personalized tasks for specific scenarios [3], whose parameters are not shared with the server. Li et al. [21] enabled a personalized model architecture by using knowledge distillation [15] to select different network models based on multiple training objectives. Similarity-based methods produce personalization by establishing a relationship model between clients. Sattler et al. [33] clustered clients into different client groups and trained corresponding federated models on each homogeneous client group.

Considering the distributed and privacy data characteristics of user-service QoS invocation records and the heterogeneity of data across clients, we adopt a personalized federated HN framework to train personalized prediction layers collaboratively, significantly improving the performance of DQP.

6 Discussion and Future Work

6.1 Computational Cost and Communication Expenses

(i) *Computational Cost*. HN updates impose additional computational consumption on the server, increasing the overall computational complexity and resource requirements. Specifically, it mainly depends on the number of participating service users and the network size of the HN. Since the HN is deployed in the cloud for personalized user needs, it is not impacted by resource-constrained environments.

(ii) *Communication Expenses*. Communication is limited to local model parameters, not the larger HN itself, allowing for complex server-side models without additional overhead compared to FedAvg for federated parameter aggregation.

6.2 Limitation

(i) *Malicious User Influence*. Malicious users can modify data features or inject incorrect data subsets into the original dataset to embed backdoors into the model, thereby manipulating the training objectives of the local client [48]. Our proposed approach FHR-DQP introduces an HN so that destroying a service user's QoS prediction model does not leak information about other service users, because the server-side HN generates a single embedding for each user that remains on the server and cannot be interpreted. However, it still retains the noisy gradient introduced by the service user during the stage of federated parameter aggregation, which possibly reduces the model's QoS prediction accuracy.

(ii) *Computational Cost on the Server*. HN updates introduce additional computational consumptions on the server, increasing the overall computational complexity and resource requirements.

6.3 Future Work

To address these limitations and enhance FHR-DQP, several promising research directions are outlined.

(i) *Reputation Mechanism*. Reputation mechanism identifying untrusted users in mobile edge computing is based on the Byzantine Fault Tolerance mechanism, which can be recognized by the training gradient of surrounding users in a similar network environment.

(ii) *Reduce the Size of the Embedding Layer*. To mitigate privacy leakage, the fewer number of neurons in the residual network are shared between the server and each service user, which has a smaller impact on the accuracy of the centralized model, but it inevitably leads to a decrease in the accuracy of a single QoS prediction model.

(iii) *Improving Server Efficiency*. Server load balancing strategies, distributed processing techniques, or more efficient federated learning algorithms can be investigated and designed to optimize the computational burden on servers.

7 Conclusion

In this article, we propose a novel DQP framework called FHR-DQP, which integrates shared feature extraction with residual learning and personalized network generation via an HN. First, the server performs one round of parameter aggregation of shared feature extraction residual layer, which is uploaded by the client that has been trained using private data locally. Then, the client uploads locally trained personalized feature representations and gradients of personalized prediction networks to the server, and the server leverages the HN to generate personalized network parameters for different clients. Finally, predicting an unknown QoS for a distributed target user is attainable through the trained personalized QoS prediction model until global convergence.

References

[1] Amani Abusafia, Athman Bouguettaya, Abdallah Lakhdari, and Sami Yangui. 2023. Context-aware trustworthy IoT energy services provisioning. In *Proceedings of the International Conference on Service-Oriented Computing (ICSOC)*, 167–185.

[2] Muhammad Shahid Anwar, Ahyoung Choi, Sadique Ahmad, Khursheed Aurangzeb, Asif Ali Laghari, Thippa Reddy Gadekallu, and Andrew Hines. 2024. A moving metaverse: QoE challenges and standards requirements for immersive Media consumption in autonomous vehicles. *Applied Soft Computing* 159 (2024), 111577.

[3] Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. 2019. Federated learning with personalization layers. arXiv:1912.00818. Retrieved from https://arxiv.org/abs/1912.00818

[4] Aishwariya Chakraborty and Sudip Misra. 2023. QoS-aware resource bargaining for federated learning over edge networks in industrial IoT. *IEEE Transactions on Network Science and Engineering* 10, 5 (2023), 2769–2778.

[5] Jifu Chen, Chengying Mao, and William Wei Song. 2023. QoS prediction for web services in cloud environments based on swarm intelligence search. *Knowledge-Based Systems* 259 (2023), 110081.

[6] Leticia Duboc, Rami Bahsoon, Faisal Alrebeish, Carlos Mera-Gómez, Vivek Nallur, Rick Kazman, Philip Bianco, Ali Babar, and Rajkumar Buyya. 2021. Systematic scalability modeling of QoS-aware dynamic service composition. *ACM Transactions on Autonomous and Adaptive Systems* 16, 3–4 (2021), 1–39.

[7] Mahmoud Hashem Eiza, Thomas Owens, Qiang Ni, and Qi Shi. 2015. Situation-aware QoS routing algorithm for vehicular ad hoc networks. *IEEE Transactions on Vehicular Technology* 64, 12 (2015), 5520–5535.

[8] Carlos Flores, Paul Grace, and Gordon S Blair. 2011. SeDiM: A middleware framework for interoperable service discovery in heterogeneous networks. *ACM Transactions on Autonomous and Adaptive Systems* 6, 1 (2011), 1–8.

[9] David Ha, Andrew Dai, and Quoc V. Le. 2016. Hypernetworks. arXiv:1609.09106. Retrieved from https://arxiv.org/abs/1609.09106

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity mappings in deep residual networks. In *Proceedings of the European Conference on Computer Vision (ECCV),* 630–645.

[11] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 355–364.

[12] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the International World Wide Web Conference (WWW)*, 173–182.

[13] Xiaoming He, Haodong Lu, Miao Du, Yingchi Mao, and Kun Wang. 2021. QoE-based task offloading with deep reinforcement learning in edge-enabled Internet of vehicles. *IEEE Transactions on Intelligent Transportation Systems* 22, 4 (2021), 2252–2261.

[14] Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (Gelus). arXiv:1606.08415. Retrieved from https://arxiv.org/abs/1606.08415

[15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. arXiv:1503.02531. Retrieved from https://arxiv.org/abs/1503.02531

[16] Yihan Jiang, Jakub Konečný, Keith Rush, and Sreeram Kannan. 2019. Improving federated learning personalization via model agnostic Meta learning. arXiv:1909.12488. Retrieved from https://arxiv.org/abs/1909.12488

[17] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. 2020. SCAFFOLD: Stochastic controlled averaging for federated learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 5132–5143.

[18] Taewoon Kim, Jenn-Wei Lin, and Chi-Ting Hsieh. 2023. Delay and QoS aware Low complex optimal service provisioning for edge computing. *IEEE Transactions on Vehicular Technology* 72, 1 (2023), 1169–1183.

[19] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv:1412.6980. Retrieved from https://arxiv.org/abs/1412.6980

[20] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[21] Daliang Li and Junpu Wang. 2019. FedMD: Heterogenous federated learning via model distillation. arXiv:1910.03581. Retrieved from https://arxiv.org/abs/1910.03581

[22] Jiahui Li, Hao Wu, Jiapei Chen, Qiang He, and Ching-Hsien Hsu. 2021. Topology-aware neural model for highly accurate QoS prediction. *IEEE Transactions on Parallel and Distributed Systems* 33, 7 (2021), 1538–1552.

[23] Jianxin Li, Tianchen Zhu, Haoyi Zhou, Qingyun Sun, Chunyang Jiang, Shuai Zhang, and Chunming Hu. 2022. AIQoSer: Building the efficient inference-QoS for AI services. In *Proceedings of the IEEE/ACM International Symposium on Quality of Service (IWQoS)*, 1–10.

[24] Qinbin Li, Bingsheng He, and Dawn Song. 2021. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 10713–10722.

[25] Shun Li, Junhao Wen, Fengji Luo, Tian Cheng, and Qingyu Xiong. 2017. A location and reputation aware Matrix factorization approach for personalized quality of service prediction. In *Proceedings of the IEEE International Conference on Web Services (ICWS)*, 652–659.

[26] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated optimization in heterogeneous networks. In *Proceedings of the Conference on Machine Learning and Systems (MLSys)*, 429–450.

[27] Wenjuan Li, Jian Cao, Shiyou Qian, and Rajkumar Buyya. 2019. TSLAM: A trust-enabled self-learning agent model for service matching in the Cloud market. *ACM Transactions on Autonomous and Adaptive Systems* 13, 4 (2019), 1–41.

[28] Pavel Mach and Zdenek Becvar. 2017. Mobile Edge computing: A survey on architecture and computation offloading. *IEEE Communications Surveys and Tutorials* 19, 3 (2017), 1628–1656.

[29] Bomin Mao, Yangbo Liu, Jiajia Liu, and Nei Kato. 2023. AI-assisted edge caching for metaverse of connected and automated vehicles: Proposal, challenges, and future perspectives. *IEEE Vehicular Technology Magazine* 18, 4 (2023), 66–74.

[30] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 1273–1282.

[31] Andriy Mnih and Russ R. Salakhutdinov. 2008. Probabilistic Matrix factorization. In *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*, 1257–1264.

[32] Khalil Muhammad, Qinqin Wang, Diarmuid O'Reilly-Morgan, Elias Tragos, Barry Smyth, Neil Hurley, James Geraci, and Aonghus Lawlor. 2020. FedFast: Going beyond average for faster training of federated recommender systems. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 1234–1242.

[33] Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. 2021. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE Transactions on Neural Networks and Learning Systems* 32, 8 (2021), 3710–3722.

[34] Aviv Shamsian, Aviv Navon, Ethan Fetaya, and Gal Chechik. 2021. Personalized federated learning using hypernetworks. In *Proceedings of the International Conference on Machine Learning (ICML)*, 9489–9502.

[35] Lingshuang Shao, Jing Zhang, Yong Wei, Junfeng Zhao, Bing Xie, and Hong Mei. 2007. Personalized QoS prediction for Web services via collaborative filtering. In *Proceedings of the IEEE International Conference on Web Services (ICWS)*, 439–446.

[36] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S. Talwalkar. 2017. Federated multi-Task learning. In *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*, 4424–4434.

[37] Huifeng Sun, Zibin Zheng, Junliang Chen, and Michael R. Lyu. 2012. Personalized Web service recommendation via normal recovery collaborative filtering. *IEEE Transactions on Services Computing* 6, 4 (2012), 573–579.

[38] Guoming Tang, Deke Guo, Kui Wu, Fang Liu, and Yudong Qin. 2020. QoS guaranteed edge cloud resource provisioning for vehicle fleets. *IEEE Transactions on Vehicular Technology* 69, 6 (2020), 5889–5900.

[39] Hao Wang, Zakhary Kaplan, Di Niu, and Baochun Li. 2020. Optimizing federated learning on non-IID data with reinforcement learning. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, 1698–1707.

[40] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. 2020. Tackling the objective inconsistency problem in heterogeneous federated optimization. In *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*, 7611–7623.

[41] Hao Wu, Zhengxin Zhang, Jiacheng Luo, Kun Yue, and Ching-Hsien Hsu. 2021. Multiple attributes QoS prediction via deep neural model with contexts. *IEEE Transactions on Services Computing* 14, 4 (2021), 1084–1096.

[42] Qiong Wu, Xu Chen, Zhi Zhou, and Junshan Zhang. 2022. FedHome: Cloud-edge based personalized federated learning for in-home health monitoring. *IEEE Transactions on Mobile Computing* 21, 8 (2022), 2818–2832.

[43] Xiaokun Wu, Bo Cheng, and Junliang Chen. 2017a. Collaborative filtering service recommendation based on a novel similarity computation method. *IEEE Transactions on Services Computing* 10, 3 (2017), 352–365.

[44] Yaoming Wu, Fenfang Xie, Liang Chen, Chuan Chen, and Zibin Zheng. 2017. An embedding based factorization machine approach for web service QoS prediction. In *Proceedings of the International Conference on Service-Oriented Computing (ICSOC)*, 272–286.

[45] Youhao Xia, Ding Ding, Zhenhua Chang, and Fan Li. 2022. Joint deep networks based multi-source feature learning for QoS prediction. *IEEE Transactions on Services Computing* 15, 4 (2022), 2314–2327.

[46] Sheng Zhang, Weihong Wang, James Ford, and Fillia Makedon. 2006. Learning from incomplete ratings using non-negative matrix factorization. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*, 549–553.

[47] Yilei Zhang, Peiyun Zhang, Yonglong Luo, and Jun Luo. 2020. Efficient and privacy-preserving federated QoS prediction for Cloud services. In *Proceedings of the IEEE International Conference on Web Services (ICWS)*, 549–553.

[48] Zhuosheng Zhang, Jiarui Li, Shucheng Yu, and Christian Makaya. 2023. Safelearning: Secure aggregation in federated learning with backdoor detectability. *IEEE Transactions on Information Forensics and Security* 18 (2023), 3289–3304.

[49] Yuqi Zhao, Bing Li, Jian Wang, Delun Jiang, and Duantengchuan Li. 2022. Integrating deep reinforcement learning with pointer networks for service request scheduling in edge computing. *Knowledge-Based Systems* 258 (2022), 109983.

[50] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. 2018. Federated learning with non-IID data. arXiv:1806.00582. Retrieved from https://arxiv.org/abs/1806.00582

[51] Zibin Zheng, Hao Ma, Michael R. Lyu, and Irwin King. 2011. QoS-aware web service recommendation by collaborative filtering. *IEEE Transactions on Services Computing* 4, 2 (2011), 140–152.

[52] Zibin Zheng, Yilei Zhang, and Michael R. Lyu. 2010. Distributed QoS evaluation for real-world web services. In *Proceedings of the IEEE International Conference on Web Services (ICWS)*, 83–90.

[53] Guobing Zou, Jin Chen, Qiang He, Kuan-Ching Li, Bofeng Zhang, and Yanglan Gan. 2020. NDMF: Neighborhood-integrated deep matrix factorization for service QoS prediction. *IEEE Transactions on Network and Service Management* 17, 4 (2020), 2717–2730.

[54] Guobing Zou, Shaogang Wu, Shengxiang Hu, Chenhong Cao, Yanglan Gan, Bofeng Zhang, and Yixin Chen. 2023. NCRL: Neighborhood-based collaborative residual learning for adaptive QoS prediction. *IEEE Transactions on Services Computing* 16, 3 (2023), 2030–2043.

Received 23 March 2024; revised 4 October 2024; accepted 13 December 2024