Security-Aware QoS Forecasting in Mobile Edge Computing based on Federated Learning

Huiying Jin
College of Computer and Information
Hohai University
Nanjing, China
Email: 367046895@qq.com

Pengcheng Zhang
College of Computer and Information
Hohai University
Nanjing, China
Email: pchzhang@hhu.edu.cn

Hai Dong
School of Science
Royal Melbourne Institute of Technology
Melbourne, Australia
Email: hai.dong@rmit.edu.au

***Abstract*—This paper proposes a novel security-aware QoS (Quality of Service) forecasting approach – Edge QoS Per-PM (Edge QoS forecasting with Personalized training based on Public Models in mobile edge computing) by migrating the principle of integrating cooperative learning and independent learning from federated learning. Edge QoS Per-PM can make fast and accurate forecasting on the premise of ensuring enhanced security. We train private model based on public model for personalized forecasting. The private models are invisible to other users to ensure the absolute security. At regular intervals, a Long Short-Term Memory (LSTM) model is trained based on the latest private data to meet the real-time requirements of the dynamic edge environment and ensure the accuracy of prediction results. A series of experiments is conducted based on public network data sets. The results demonstrate that Edge QoS Per-PM can train appropriate models and achieve faster convergence and higher accuracy.**

***Keywords*-Mobile edge computing; Quality of Service; Public model; Private model; LSTM; Security Forecasting**

I. Introduction

Service Oriented Architecture (SOA) is an application architecture, in which all functions are defined as independent services. Web service is one of the technologies to implement SOA [1]. Industry experts often orchestrate these services to fulfil users with changing requirements. In recent years, with the development of Web services, its non-functional attributes (i.e., Quality of Services (QoS)) have attracted increasing attentions [2], [3]. Nowadays, there are a large number of Web services with redundant functions on the Web. It is therefore particularly important to employ QoS to select appropriate web services to meet users' requirements [4].

With the advent of 5G era, mobile edge computing technologies are being increasingly used [5]. Computing power is located on the edge of the network and very close to users or information sources. In this way, response delay can be greatly reduced. Providing mobile edge based services responses to users has become the current development trend [6]. However, it would also generate security problems.

At present, researchers have made tremendous efforts in the field of privacy-preserving QoS forecasting. Current solutions include local sensitive hash [7], multidimensional anonymity [8], differential privacy [9] and location aware protection [10]. It can be seen that the existing methods are mostly suitable for traditional environments and they mostly use data encryption to manipulate QoS data directly to realize security.

However, the existing security-aware QoS forecasting approaches mainly face the following problems in the mobile edge environment:

i). The security mechanisms of traditional approaches are inadequate. Because of the dynamic and regional characteristics of the edge environment, local users are more frequently and closely connected. They can access each other's historical data for forecasting [11]. With increasing interactions, the user behaviors are easier to be inferred, and the encryption rules of data are easier to be cracked [12]. Therefore, the traditional data encryption based QoS forecasting approaches are more vulnerable in the edge environment.

ii). The prediction accuracy is greatly reduced. Because mobile edge computing technology has the characteristics of proximity and low latency, it can effectively improve user experience. Meanwhile, the requirements on real-time data and data accuracy are more rigorous. However, the traditional approaches do not have mechanism to update the data dynamically, and consequently the prediction accuracy in mobile edge computing cannot be guaranteed.

To overcome the problems aforementioned, this paper proposes a novel security-aware QoS forecasting method in the mobile edge environment, abbreviated as Edge QoS Per-PM (Edge QoS forecasting with Personalized training based on Public Models in mobile edge computing). We combine public model and private model training for personalized forecasting. The private models are only visible to their own. The parameters of the private models are different from one to another, which enables the security. The main contributions of this paper are as follows:

- We propose a novel personalized security-aware QoS forecasting method inspired by the idea of cooperation and independence in federal learning [13], to

 H. Jin, P. Zhang and H. Dong, "Security-Aware QoS Forecasting in Mobile Edge Computing based on Federated Learning," 2020 IEEE International Conference on Web Services (ICWS), Beijing, China, 2020, pp. 302-309, doi: 10.1109/ICWS49710.2020.00046.

address the deficiency of traditional encryption methods and fundamentally eliminate the possibility of being cracked. First, we extract the general data from the whole data set to form a public data set. Next, a Long Short-Term Memory (LSTM) is trained via the public data set. The weight parameters obtained in the training process are transmitted to private users. Finally, the LSTM will be trained based on the users' private data to perform personalized forecasting.

- The LSTM model can update the weight parameters dynamically to improve the accuracy of QoS prediction, considering the requirements of real-time and data accuracy in the mobile edge environment. Within every time interval, users perform LSTM training on the private data generated in that interval. The latest weight parameters are imported into the LSTM for attribute value forecasting in the next time interval. As private data is continuously imported over time, periodic training is constantly performed to dynamically update the weight parameters. This process repeats until no new private data is generated (i.e.,the user terminates the service(s)). The training process completes and the edge QoS forecasting results are obtained.
- We conduct a series of experiments to explore the proposed Edge QoS Per-PM approach based on public network data sets. The experiments verify the influence of different learning rates and training time on public model training, and the effectiveness of the optimal public weight parameters on personalized forecasting. The experimental results also demonstrate that Edge QoS Per-PM can achieve the goal of security, ensuring the accurate forecasting performance.

The structure of this paper is organized as follows: Section 2 states the related work of the existing research. Section 3 introduces the background knowledge and relevant theoretical basis of our approach. Section 4 presents the Edge QoS Per-PM approach proposed in this paper. Section 5 delivers the experimental design and result analysis. Section 6 summarizes the paper and plans future work.

II. Related work

A. QoS privacy preserving

With regards to QoS privacy preserving, Liu et al. [9] combined a differential privacy encryption algorithm with collaborative filtering to design a QoS privacy protection method. Qi et al. [7] developed a distributed recommendation system based on local sensitive hash in 2017. Shahriar et al. [10] devised a protection protocol for attribute value encryption and location hiding. It is obvious that the existing QoS privacy protection methods are mostly applicable to static environments, and the prediction accuracy is hindered after data encryption. In addition, they are easier to be cracked in the edge environment [11], [12].

B. QoS security services

With regard to security services, researchers have proposed a variety of security mechanisms and guidelines. Shen et al. [14] introduced a distributed dynamic management mechanism considering both security and QoS. However, the mechanism is only suitable for specific environments, e.g., when network traffic is not too busy. Alessandro et al. [15] adopted an integrated tool support approach, which can achieve the maximum trade-off between safety and quality. However, this approach does not consider the changes in the dynamic environment. Jalal et al. [16] presented a security-aware QoS optimization method in distributed real-time environments to reach an agreement between confidentiality, integrity and authentication security. Nevertheless, it is mostly used in IP routing protocols. Charuenporn et al. [17] proposed a new paradigm for developing the QoS security metrics (QoS-SM) using QoS ontologies. Nevertheless, this paradigm can only be applied with two defined information system standards (COBIT and ITIL).

To solve the security limitations in mobile edge computing, this paper proposes a security-aware QoS prediction approach in the mobile edge environment called Edge QoS Per-PM. Inspired by the ideas of sharing and independence from federated learning [13], this approach extracts and trains data with the same characteristics of the horizontal federated learning.

III. Preliminaries

A. Security and challenges of mobile edge computing

Mobile edge computing (MEC) is a new computing model and extends centralized cloud computing to network edges [18]. MEC provides users with services and computing functions at the edge of a mobile network to reduce latency. It also ensures efficient network operations and service delivery. However, due to its characteristics of interoperability, decentralization and mobile support, mobile edge computing faces new security challenges. Because of the technical characteristics of MEC, there are many differences between MEC security and traditional security. From the perspective of privacy security, users in a node are easy to obtain other users' sensitive data, which causes user privacy disclosure [19]. This results from the fact that users are close to nodes in MEC. In addition, MEC is an open ecosystem, and user mobility also brings some challenges to security mechanism.

B. Long Short-Term Memory

Long Short-Term Memory (LSTM) is a type of deep learning methods, which was first proposed by Sepp Hochreiter and Jürgen Schmiduber in 1997 [20]. It is a specific form of RNN. RNN is the general term of a series of neural networks that can process sequence data.

All RNNs contain a chain form of repetitive neural network modules, while LSTM is designed to solve long-term problems. The structure of a single node is shown in Fig. 1. The repeated modules comprise of four layers, which interact in a special way. Different from the single neural network layer, the subtlety of LSTM is to update the input threshold, the forgetting threshold and the output threshold, so that the weight of self cycle changes constantly. Therefore, when the model parameters are fixed, the integral scale can change dynamically at different time, so as to effectively avoid the problem of gradient disappearance or gradient expansion.

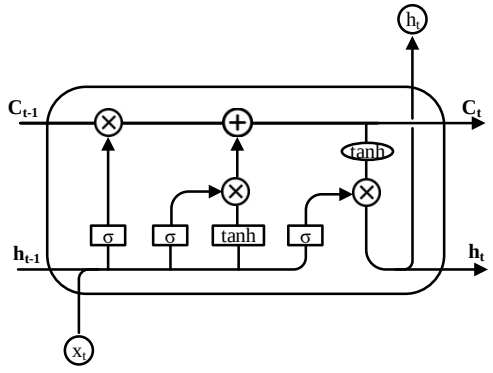


Fig. 1. LSTM structure diagram

C. Federated Learning

Federated learning is a new basic technology of artificial intelligence. Its design goal is to ensure data security when model training needs data exchange among multiple repositories. Under the premise that terminal data and personal privacy data are legal and compliant, efficient machine learning is carried out among multiple participants or computing nodes. As a distributed machine learning paradigm, federated learning can effectively solve the data island problem, i.e., data stored and maintained independently among different individuals obstacles data sharing. It allows participating models to collaborate on the basis of not sharing data [21], and technically break the data island to achieve AI cooperation.

According to data set differences, federated learning can be divided into three types: horizontal federated learning, vertical federated learning and federated transfer learning [13]. Among them, horizontal federated learning is mainly used for data sets with more overlapped user features and less overlapped user numbers, e.g., for two banks in different regions, the number of same users is very small but the business features of users are similar, while vertical federated learning is the opposite, which is suitable for data sets with more overlapped user numbers and less overlapped user features. Federated transfer learning is used in the case of fewer user numbers and user features. We use transfer learning to overcome the situation of small data scale or insufficient labelled samples, so as to improve the learning performance of learning models.

IV. The Edge QoS Per-PM approach

A. Overview of Edge QoS Per-PM

We propose a security-aware QoS forecasting approach (Edge QoS Per-PM) in the mobile edge environment. The main goals of Edge QoS Per-PM are security-aware, accurate and efficient forecasting. The main workflow is shown in Fig. 2. It includes two primary phases:

1) Public model training: First, the edge station location information and a QoS data set are fused to form a spatio-temporal edge user QoS data set. The QoS data set is then employed to determine the edge server locations based on the geographical distribution of the users. The whole edge network area is divided into several edge regions according to the geographical distribution of edge servers. Next, the public edge QoS data set in each edge region is generated based on the previous edge user QoS data set. Finally, a public LSTM is trained via the public data set of each region to obtain a regional model. The weight parameters of the regional models are transmitted to private users in the corresponding regions for personalized forecasting.

2) Personalized forecasting: A user uses the weight parameters of the public LSTM in his/her belonged region as the initial parameters of his/her private LSTM. The private LSTM will be further trained based on the user's private data to make the personalized forecasting. A user's private data is the temporal QoS data generated in the process that the user interacts with services. The private data is used to train the private LSTM in each time interval to update its weight parameters continuously, so as to ensure the real-time performance of the weight parameters in the dynamic edge environment and improve the prediction accuracy. With the increase of training iterations, the private model is continuously optimized. QoS forecasting results will be generated for future time slots.

B. Public model training

The purpose of the public model training is to provide an initial forecasting model for all users in an edge region. The training bases on privacy-insensitive public data. The trained public model parameters are passed to each individual user to train the user's private model using his/her private data. It thus can improve the private model training efficiency, while achieving the purpose of security without disturbing the personalized forecasting. The public model training consists of the following three steps.

1) Region division : First, the QoS data set and the edge location data are integrated to form a spatio-temporal edge user QoS data set. Next, according to the geographical distribution of the edge users, the actual locations of edge servers on the map are obtained through a map service[1]. We

[1]https://www.dituwuyou.com/

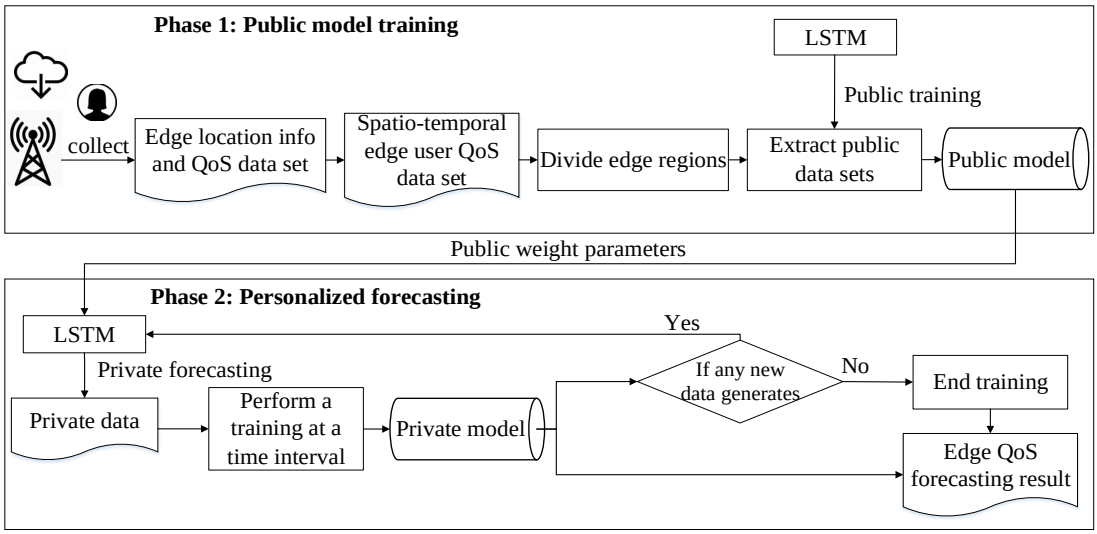


Fig. 2. Edge QoS Per-PM overview

assume that the servers are evenly distributed in each edge region. The geographically close edge servers are assumed to share the similar edge environment [22]. Therefore, in order to improve the forecasting accuracy as much as possible, we divide the whole edge network area into *k* edge regions, where *k* ($k \geq 2$) is determined by the geographical distribution of edge servers. Each edge region contains a group of edge servers that share the same edge environment.

2) Public data extraction: The public data extraction process is divided into three steps. **Step 1**: the QoS data set of each edge region is resorted according to the order of user ID, time period ID, service ID and attribute value. **Step 2**: The values of a QoS attribute shared by all services invoked by all users in an edge region in time period T_1 are extracted from the regional data set and expressed in the form of a service-user matrix. We take the median value of each row of the matrix to get a service-T_1 column vector, and make the column vector as the public QoS data of the services invoked in the region in time period T_1. **Step 3**: We repeat Step 2 to get service-T_2, ... , service-T_n column vectors. Finally, the *n* column vectors are synthesized to generate a two-dimensional (service-time interval) matrix as the public QoS data set in an edge region for the public LSTM training.

3) Public data training: Each regional LSTM performs training based on the regional public data set. Before the model training, we have reached a consensus between the learning rate and training iterations. That is, the larger the learning rate, the faster the error adjustment speed. Hence we can use less training iterations to achieve the same training goal. A combination of parameters in the model training is learning rate and training iterations. Because there is a negative correlation between learning rate and training iterations, the training iterations decrease with the increase of the learning rate. For example, if the learning rate of the LSTM is increased in [0.001, 0.01] with the step size of 0.001, the corresponding training iterations are decreased in [100, 1000] with the step size of 100 to achieve a trade-off between the learning rate and the training iterations.

After the parameter setting, we use the regional public QoS data set to train the regional LSTMs. The following are the calculation formulas of the forgetting gate, input gate, current time unit state and output gate.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{1}$$

where W_f is the weight matrix of the forgetting gate, h_{t-1} and x_t respectively represent the output of the previous time and the input of the current time, b_f is the bias term of the forgetting gate. LSTM selectively keeps the cell state of the previous time to the current time by the forgetting gate.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2}$$

where W_i is the weight matrix of the input gate, b_i is the bias term of the input gate. The input gate selectively saves the current input to the unit state. Next, we calculate the cell state at the current time c_t. It bases on the aggregation of the cell state of the last moment c_{t-1} multiplied by the forgetting gate f_t, and the cell state of the current input $\tilde{c}_t$ multiplied by the input gate i_t, as shown in formula (3) (4):

$$\tilde{c}_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{3}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \tag{4}$$

In this way, we combine LSTM's current memory $\tilde{c}_t$ and long-term memory c_{t-1} to form a new unit state c_t. The operations on the forgetting and input gates can save the

information of a long time ago and avoid the current unimportant content from entering the memory. Equation (5) is the calculation process of the output gate, which controls the effect of long-term memory on the current output.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{5}$$

The final output of LSTM is determined by the output gate and unit state.

$$h_t = o_t \circ tanh(c_t) \tag{6}$$

The loss function is estimated by calculating the RMSE (Root Mean Square Error):

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(h_{t_i} - h'_{t_i})^2} \tag{7}$$

Where h_{t_i} is the output value of LSTM at time t_i, h'_{t_i} is the real value at time t_i, and *N* is the total number of times.

Through the derivation of loss function, the weight parameters are adjusted continuously. Hence, with the increase of the number of training iterations, the training error decreases continuously until the obtainment of the optimal initial weight parameter values for personalized forecasting.

C. Personalized forecasting

In the personalized forecasting, a user makes QoS forecasting based on his/her private LSTM with the initial weight parameters passed from the public LSTM. The calculation process is shown in formula (1)∼(6). After a QoS forecasting is made for time period T_1, if the actual QoS is obtained from T_1, we adjust the model weight parameters with the deviation between the predicted and actual QoS data. The weight parameters are adjusted with the following formula (8):

$$weight_i = weight_i - l * error \tag{8}$$

Among them, $weight_i$ is the initial weight parameter, *l* is the learning rate and $error = 2 * (h_{t_i} - h'_{t_i})'$ is the derivation of prediction error.

After completing the forecasting and training in time period T_1, the forecasting will be performed for the next period. If there is actual QoS data obtained from the next time period after the forecasting, the training will be executed again to further adjust the weight parameters. This forecasting followed by training process will repeat until no new QoS data is obtained from the next time period (i.e., the user terminates the service(s)).

V. Evaluation

A series of experiments are performed to validate Edge QoS Per-PM based on several public data sets. We conduct the experiments in a computer system with Intel(R) Core(TM) i5-8250U CPU @1.60GHz, 8.00GB RAM, Windows 10, and MatLab R2018b to train models and carry out personalized forecasting.

A. Data Set Description

We use two data sets: a time series QoS data set and an edge station location data set, which can be downloaded from the data sources used in [23], [11]. The first data set [2] describes real-world QoS evaluation results from 142 users (IDs: 0-141) on 4500 Web services over 64 consecutive time slices (with a 15-minute interval between each two slices). The QoS attributes mainly include *response time (RT)* and *throughput (TP)*. The general information of the first data set is shown in Table I.

Table I
QoS data set information

User ID	Time ID	Service ID	Attribute value
24	18	100	0.3060
24	18	2485	0.1100
96	63	3895	1.3310
140	9	289	0.5760

The second data set [3] is provided by *Shanghai Telecom.* It contains more than 7.2 million Internet access records of 9481 mobile phones collected by 3233 base stations [24]. We use the latitude and longitude information of the base stations in the data set to locate the edge server positions. The information *Shanghai Telecom* data set is shown in Table II.

Table II
Shanghai Telecom data set information

Longitude	Latitude	District	Server ID
121.407680	31.137509	Minhang	17
121.390889	31.002479	Minhang	13
121.480699	31.240686	Huangpu	47
121.454410	31.128227	Xuhui	31

B. Data fusion and public data extraction

We randomly select 142 sets of base station positions from *Shanghai Telecom* data set and fuse them with the QoS data set in the following steps: First, the 142 sets of positions are numbered from 0 to 141 and matched with the 142 users in the QoS data set in terms of identical IDs. Thereby, the spatio-temporal QoS data set of edge users is obtained. Next, the edge users in the same positions are mapped to the same edge servers. 67 edge servers are located through the map service[1]. Considering the geographical distribution of the 67 edge servers and the municipal districts in Shanghai, we divide the whole edge network area into three edge regions, in which each edge region maps with 1-4 municipal districts. The division results are shown in Fig. 3. The numbers of edge servers in the edge regions are respectively 22, 24 and 21. The spatio-temporal edge user QoS data set is also

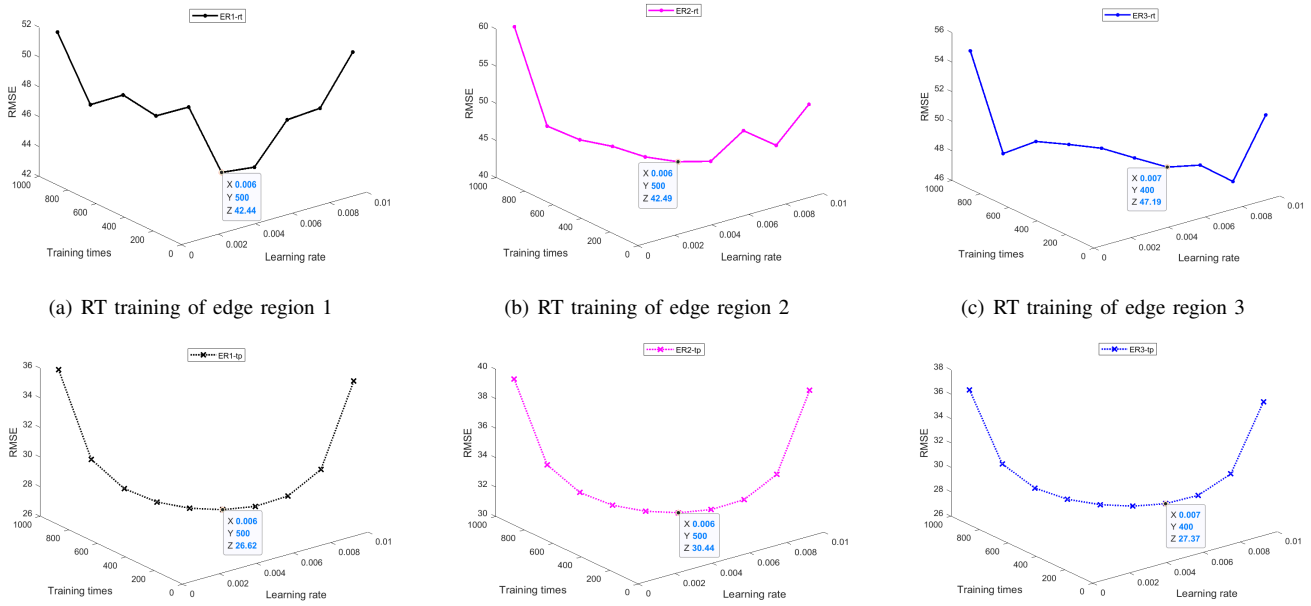
[2]https://github.com/wsdream/wsdream-data set
[3]http://sguangwang.com/Telecomdata set.html

(a) Edge region 1 distribution : Minhang District

(b) Edge region 2 distribution : Huangpu District, Hongkou District and Pudong New District

(c) Edge region 3 distribution : Xuhui District, Songjiang District, Qingpu District and Jingan District

Fig. 3. Chart of edge region distribution



(a) RT training of edge region 1

(b) RT training of edge region 2

(c) RT training of edge region 3

(d) TP training of edge region 1

(e) TP training of edge region 2

(f) TP training of edge region 3

Fig. 4. Training performance of different learning rate and training iterations combination

divided into three subsets in correspond to the geographical locations of the edge regions.

We independently train a public model for each edge region using the public data subset in that region. We take the edge region 1 as an example. According to the step of public data extraction stated in Section IV, the attribute values of 4500 services invoked by all the users in edge region 1 in T_1 time period are extracted and expressed in a two-dimensional matrix. We take the median value of each row of the matrix to generate a service-T_1 column vector. We use the same method to generate the column vectors for T_2, ... , T_{64}. Finally, the column vectors are aggregated to from a service-time matrix.

C. Experimental Results

(1) The trade-off between learning and training

The training results on combinations of various learning rates and training iterations in the three edge regions are shown in Fig. 4. The horizontal axis represents the learning rate, and the two vertical axes respectively represent the training iterations and the average root mean square error. In the training process, when the learning rate increases, the adjustment range of the weight parameter also increases, to make the error rate decrease faster. Therefore, when the same training effect is achieved, the number of required training iterations can be reduced accordingly. Thus, the training iterations decrease with the increase of learning rate to achieve a trade-off between the two parameters.

Fig. 4(a) and 4(d) show the training performance of

edge region 1 on *RT* and *TP*. It can be seen that, as the learning rate increases (or the training iterations decrease simultaneously), the error rate curve shows a concave shape. At the point (0.006, 500), both *RT* and *TP* reach their lowest error rates. Therefore, we find the best combination of the learning rate and training iterations for edge region 1, and use them for the public model training. Similarly, from Fig. 4(b) and 4(e), it can be seen that the point (0.006, 500) is the best combination of the learning rate and training iterations for edge region 2. *RT* and *TP* in edge region 3 reach the lowest error rates at different points, i.e., (0.009, 200) and (0.005, 600), according to Fig. 4(c) and 4(f). To achieve the relatively lower error rates for both *RT* and *TP*, we choose the average values (0.007, 400) as the relatively better combination for edge region 3.

(2) Weight parameter of public model training

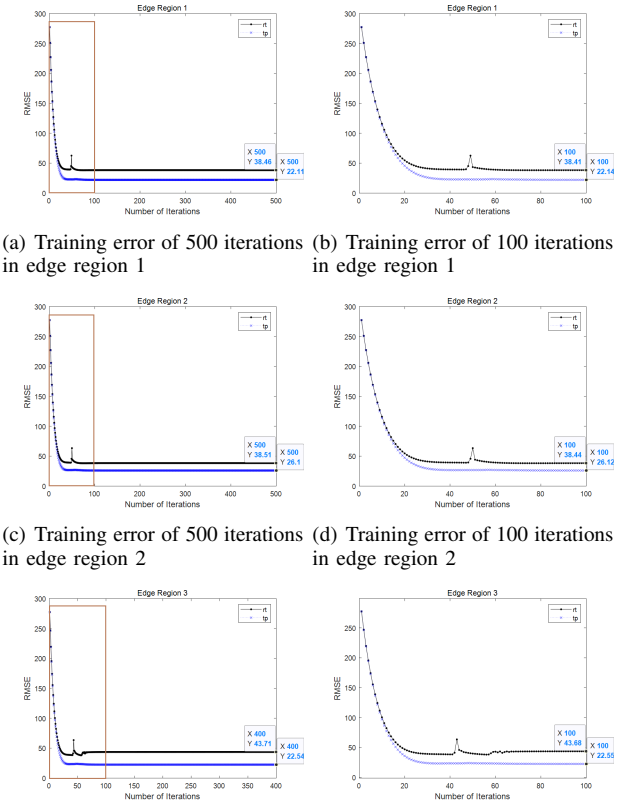
After determining the best parameter combination for each region's public model training, our next experiment is to find the optimal initial weight parameter values that each region's public model can provide for personalized forecasting. We perform training on both the *RT* and *TP* data sets of each region. Fig. 5 shows the variation of the data training error rate of the three public models with the increasing number of iterations. Fig. 5(b), 5(d) and 5(f) are enlarged views of the first 100 iterations of Fig. 5(a), 5(c) and 5(e).

Fig. 5(a) shows the training of the edge region 1 public model, in which the learning rate is set to 0.006 and the training iterations are 500 according to our above experiment *(1)* results. It can be seen that, with the increase of the iterations, the error rate first shows a sharp downward trend, followed by a relatively plain and stable trend after the iterations increase from 100 to 500. There is no gradient explosion or gradient disappearance being discovered in the model training process. Therefore, it can be seen from the experimental results that the public model training has basically reached the optimal state around 100 iterations. Considering the training cost-effectiveness, we provide the weight values of the forgetting gate, input gate, cell state and output gate of the LSTM after 100 training iterations as the initial weight parameters for users in edge region 1 to perform personalized forecasting. From Fig. 5(b), we can see that the training error rates for *RT* and *TP* are reduced from nearly 300 at the beginning to 38.41 and 22.14 respectively.

Similarly, in Fig. 5(c), 5(d), 5(e) and 5(f), compared with the error rates after 100 iterations, the error rates of the public models in edge regions 2 and 3 have reached their ideal state around 100 iterations. Therefore, we use the model weight values obtained at 100 iterations as the initial parameters for the users' private models in edge regions 2 and 3.

(3) Personalized forecasting performance

When users access the edge servers in different edge regions, personalized forecasting is carried out after obtaining the public model weight parameters of the corresponding



(a) Training error of 500 iterations in edge region 1

(b) Training error of 100 iterations in edge region 1

(c) Training error of 500 iterations in edge region 2

(d) Training error of 100 iterations in edge region 2

(e) Training error of 400 iterations in edge region 3

(f) Training error of 100 iterations in edge region 3

Fig. 5. Training performance of public model data (*RT* and *TP*) on RMSE in three edge regions

edge regions (e.g., when users access edge region 1, users get the weight parameters of the public model in edge region 1). In the process of user personalized forecasting, in order to ensure the real-time and accuracy of the weight parameters, we regularly train a user's private model. Each user's QoS data is recorded in 64 time slices (15 minutes each). Theoretically, the smaller the training interval and the higher the training frequency, the higher the accuracy of the model. However, we need to balance between training cost and training performance. Therefore, our training frequency is set to 4 time slices (one hour) for a cycle. 16 times of training and forecasting have been performed in total.

A group of edge servers with unbalanced user records (i.e., maximum (6, 3 and 6 respectively) and null accessing records) in three edge regions are accessed. Each user accesses each edge server lasting for 64 time slices. Taking *TP* data as an example, Table III gives the average RMSE of all the services over 16 training iterations. Among them, Edge QoS Per-PM is the proposed forecasting method, while Non Per-PM QoS which are purely private LSTMs and does not contain public models. Thereby it cannot receive initial weight parameters during the process of training and

forecasting. Each Non Per-PM QoS needs to train from the private QoS data of an individual user. It can be seen that the average RMSE of Edge QoS Per-PM is far less than Non Per-PM QoS. Hence, Edge QoS Per-PM not only ensures security, but also has better forecasting accuracy.

Table III
THE FORECASTING ACCURACY

	Edge Region 1		Edge Region 2		Edge Region 3	
Server ID	F-10	F-57	S-04	S-41	T-04	T-25
Edge QoS Per-PM	28.03	24.84	32.50	26.61	25.95	36.65
Non Per-PM QoS	133.58	113.78	138.46	113.08	105.05	100.94

Table IV shows the forecasting time required for the two approaches to achieve the same forecasting performance (e.g., the total forecasting error rate is less than 50). It can be seen from the table that the time required for Edge QoS Per-PM is far less than Non Per-PM QoS.

Table IV
THE FORECASTING TIME COST

	Edge Region 1		Edge Region 2		Edge Region 3	
Server ID	F-10	F-57	S-04	S-41	T-04	T-25
Edge QoS Per-PM	2.85s	2.03s	3.11s	1.91s	2.08s	1.92s
Non Per-PM QoS	18.04s	13.24s	18.25s	12.23s	12.86s	11.79s

In conclusion, our proposed Edge QoS Per-PM forecasting is more accurate and faster than Non Per-PM QoS via the experiments.

VI. CONCLUSIONS AND FUTURE WORK

Existing security-aware QoS forecasting approaches cannot meet the demand of the edge environment on security and high forecasting accuracy. We propose a novel security aware QoS forecasting approach for mobile edge environments named Edge QoS Per-PM. It combines public model and private model training for personalized forecasting to achieve the goal of security-aware, accurate and efficient forecasting.

For future work, first, we will consider the automatic clustering of edge servers to divide the sub edge regions. Second, the current approach only achieves a trade-off between learning rate and training iterations, we will further optimize the public model to improve forecasting performance. Finally, we will study multivariate QoS forecasting in the edge environment.

VII. ACKNOWLEDGEMENTS

The work is supported by the National Natural Science Foundation of China under Grant No. 61572171, the Natural Science Foundation of Jiangsu Province under Grant No. BK20191297, and the Fundamental Research Funds for the Central Universities under Grant No. 2019B15414.

REFERENCES

[1] S. P. Lee, L. P. Chan, and E. W. Lee, "Web services implementation methodology for soa application," in *2006 4th IEEE International Conference on Industrial Informatics*, pp. 335–340, IEEE, 2006.

[2] Y. Syu, J.-Y. Kuo, and Y.-Y. Fanjiang, "Time series forecasting for dynamic quality of web services: an empirical study," *Journal of Systems and Software*, vol. 134, pp. 279–303, 2017.

[3] P. Zhang, H. Jin, H. Dong, W. Song, and L. Wang, "LA-LMRBF: Online and long-term web service QoS forecasting," *IEEE Transactions on Services Computing, 2019, DOI: 10.1109/TSC.2019.2901848*.

[4] B. M. A. Madi, Q. Z. Sheng, L. Yao, Y. Qin, and X. Wang, "Plmwsp: Probabilistic latent model for web service qos prediction," in *2016 IEEE International Conference on Web Services (ICWS)*, pp. 623–630, IEEE, 2016.

[5] I. Ahmad, T. Kumar, M. Liyanage, J. Okwuibe, M. Ylianttila, and A. Gurtov, "Overview of 5g security challenges and solutions," *IEEE Communications Standards Magazine*, vol. 2, no. 1, pp. 36–43, 2018.

[6] M. La Polla, F. Martinelli, and D. Sgandurra, "A survey on security for mobile devices," *IEEE communications surveys & tutorials*, vol. 15, no. 1, pp. 446–471, 2012.

[7] L. Qi, H. Xiang, W. Dou, C. Yang, Y. Qin, and X. Zhang, "Privacy-preserving distributed service recommendation based on locality-sensitive hashing," in *2017 IEEE International conference on web services (ICWS)*, pp. 49–56, IEEE, 2017.

[8] X. Zhang, L. Qi, W. Dou, Q. He, C. Leckie, K. Ramamohanarao, and Z. Salcic, "Mrmondrian: Scalable multidimensional anonymisation for big data privacy preservation," *IEEE Transactions on Big Data*, 2017.

[9] S. Liu, A. Liu, Z. Li, G. Liu, J. Xu, L. Zhao, and K. Zheng, "Privacy-preserving collaborative web services qos prediction via differential privacy," in *Asia-Pacific Web (APWeb) and web-age information management (WAIM) joint conference on web and big data*, pp. 200–214, Springer, 2017.

[10] S. Badsha, X. Yi, I. Khalil, D. Liu, S. Nepal, E. Bertino, and K.-Y. Lam, "Privacy preserving location-aware personalized web service recommendations," *IEEE Transactions on Services Computing*, 2018.

[11] S. Wang, Y. Zhao, L. Huang, J. Xu, and C.-H. Hsu, "Qos prediction for service recommendations in mobile edge computing," *Journal of Parallel and Distributed Computing*, vol. 127, pp. 134–144, 2019.

[12] Y.-L. Huang, C.-R. Dai, F.-Y. Leu, and I. You, "A secure data encryption method employing a sequential-logic style mechanism for a cloud system," *International Journal of Web and Grid Services*, vol. 11, no. 1, pp. 102–124, 2015.

[13] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, p. 12, 2019.

[14] Z. Shen and J. P. Thomas, "Security and qos self-optimization in mobile ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 7, no. 9, pp. 1138–1151, 2008.

[15] A. Aldini and M. Bernardo, "A formal approach to the integrated analysis of security and qos," *Reliability Engineering & System Safety*, vol. 92, no. 11, pp. 1503–1520, 2007.

[16] A. Jalal and M. A. Zeb, "Security and qos optimization for distributed real time environment," in *7th IEEE International Conference on Computer and Information Technology (CIT 2007)*, pp. 369–374, IEEE, 2007.

[17] P. Charuenporn and S. Intakosum, "Qos-security metrics based on itil and cobit standard for measurement web services.," *J. UCS*, vol. 18, no. 6, pp. 775–797, 2012.

[18] E. Ahmed and M. H. Rehmani, "Mobile edge computing: opportunities, solutions, and challenges," 2017.

[19] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges," *Future Generation Computer Systems*, vol. 78, pp. 680–698, 2018.

[20] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[21] Y. Liu, T. Chen, and Q. Yang, "Secure federated transfer learning," *arXiv preprint arXiv:1812.03337*, 2018.

[22] Z. Li, R. Xie, L. Sun, and T. Huang, "A survey of mobile edge computing," *Telecommun. Sci*, vol. 34, no. 1, pp. 87–101, 2018.

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

[24] Y. Li and S. Wang, "An energy-aware edge server placement algorithm in mobile edge computing," in *2018 IEEE International Conference on Edge Computing (EDGE)*, pp. 66–73, IEEE, 2018.