Mobility-aware Fast QoS Forecasting in Mobile Edge Computing

Journal:	*Transactions on Mobile Computing*
Manuscript ID	TMC-2022-12-1236
Manuscript Type:	Regular
Keywords:	Mobile edge computing, Edge region recognition, edge fast QoS forecasting

Mobility-aware Fast QoS Forecasting in Mobile Edge Computing

Huiying Jin, Pengcheng Zhang, *Member, IEEE*, Hai Dong, *Senior Member, IEEE*, and Yuelong Zhu

Abstract—We propose a novel mobility-aware fast QoS forecasting approach – MEC-RDESN (MEC QoS forecasting based on Region recognition and Dynamic Echo State Network) for the mobile edge computing environment. MEC-RDESN can make fast and accurate QoS forecasting on the premise of ensuring forecasting performance. Mobile sensing technology is employed to real-timely recognize the edge region to which a user belongs during the user's movement. A dynamic echo state network is adopted to fulfil the need of real-time training while ensuring prediction accuracy. We conduct a series of experiments using public and self-collected data sets. The results show that the efficiency of our approach is superior to other forecasting approaches. MEC-RDESN is also validated to be more suitable for fast edge forecasting.

Index Terms—Mobile edge computing, Edge region recognition, ESN, edge fast QoS forecasting.

1 Introduction

With the rise of the Internet of Things, service-oriented computing (SOC) has attracted much attention from industry and academia [1]. SOC is a type of computing generics that uses services as the basic elements to develop applications [2]. The main implementation technology of SOC is Web services [3]. Web services are widely used in e-commerce, big data analysis and other application scenarios [4]. Quality of Service (QoS, also called non-functional attributes) becomes the focus to select among Web services with similar functions [5]. QoS mainly includes *response time*, *throughput*, and *security* [6].

Edge Computing is an emerging computing mode that performs at the edge of network [7]. Mobile Edge Computing (MEC) provides IT services and cloud computing capabilities at the edge. It is generally deployed in the Radio Access Network (RAN), which is geographically adjacent to mobile users, enabling computing power closer to data sources [8]. Once MEC nodes receive Web service requests from users nearby, these nodes can timely respond to those requests with significant reduction on network transmission delay. In this regard, MEC can provide users with a superior service experience. The emergence of edge computing provides key enabling technologies for the vigorous development of smart transportation, smart life, virtual reality and other delay-sensitive applications [9].

Edge devices (e.g., smartphones, smart wearable, etc.) generate massive streaming data, making real-time decisions almost impossible when analytics are performed on remote clouds. To complement the computation performed on the cloud, edge nodes closer to end users can be utilized to reduce network latency [7]. However, the low-latency feature of MEC puts forward higher requirements for edge services (i.e., services, such as micro-services, Web services, etc. delivered by edge nodes) quality forecasting [10]. There are two major challenges in this regard: 1) *the necessity of user mobility-awareness*, and 2) *the high demand for fast and accurate QoS forecasting*. The following scenario demonstrates these two challenges.

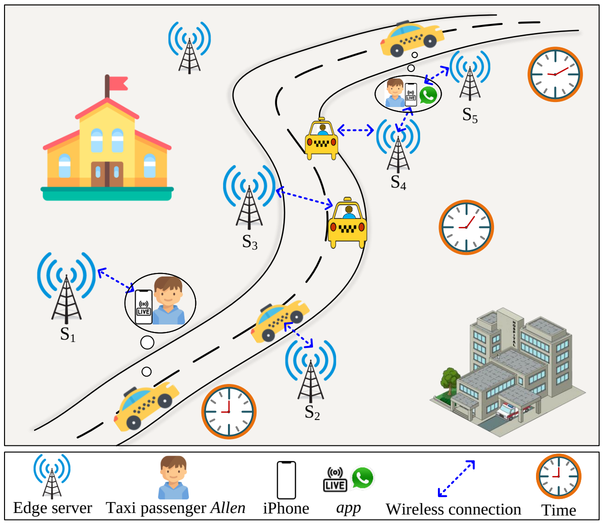


Fig. 1: A mobility-aware edge service invocation scenario

Fig. 1 shows the scenario of an urban road and its surrounding edge server distribution. A taxi was running at a relatively fixed speed from southwest to northeast at 9:00-9:10 am, while the taxi passenger *Allen* was watching a *Twitch* game live video during this journey. The video quality needs to be continuously forecasted. Since if the predicted video quality deteriorates, *Twitch* can compress video data while maintaining certain clarity to reduce bandwidth consumption and fix video stuttering. During this journey, the taxi rapidly moves and *Allen* quickly switches among different edge servers. Thus, it is particularly important to

• *H. Jin, P. Zhang and Y. Zhu are with the Key Laboratory of Water Big Data Technology of Ministry of Water Resources and the College of Computer and Information, Hohai University, Nanjing, China*
E-mail: 367046895@qq.com; pchzhang@hhu.edu.cn; ylzhu@hhu.edu.cn
• *H. Dong is with the School of Computing Technologies, RMIT University, Melbourne, VIC 3000, Australia*
E-mail: hai.dong@rmit.edu.au

Manuscript received XXX XX, XXXX; revised XXX XX, XXXX.

achieve fast and accurate video quality forecasting. From 9:00am to 9:10am, *Allen* continuously accesses the edge servers S_1, S_2, S_3, S_4 and S_5. During this process, the service quality prediction is usually based on the service data previously generated by him (from 9:00am). Since *Allen*'s position is constantly changing, the status (e.g., bandwidth, communication rate, etc.) of the network environment where he is located is dynamic. Therefore, not all the previous QoS data generated by *Allen* are valid for the subsequent forecasting. In addition, *Allen* was making a voice call using *WhatsApp* from 9:05am to 9:10am. Similarly, the call quality also needs to be continuously forecasted so that *WhatsApp* can compress audio to ensure the call quality if the predicted call quality drops. Thus, both of the quality of *Twitch* and *WhatsApp* needs to be forecasted in the second half of the journey.

We summarize the challenges faced by the edge QoS forecasting in the above scenario as follows:

i). Existing QoS forecasting approaches cannot adapt to MEC users' dynamic movement in real-time. Users have the characteristics of activity and mobility in the MEC environment. Each edge server has a certain coverage area and provides services to users within its coverage area [11]. Since users may continuously accesses different edge servers during the movement process, each edge server serves a different number of users with limited resources [12]. Therefore, the network condition where a user is can keep changing dynamically. However, the location-aware schemes in the existing QoS forecasting approaches only focus on geographic area awareness (i.e., users' movement between areas covered by different edge servers) and overlook users' real-time movement and moving speed [10], [13], [14]. This may lead to low forecasting accuracy and unsatisfied forecasting speed, i.e., forecasting cannot be delivered before a user arrives in an area covered by a different edge server. Therefore, it is great important to sense user movement and obtain effective time series data in real time for QoS forecasting.

ii). Realizing fast and accurate QoS forecasting is an urgent problem in MEC. The core concept of MEC is to use the wireless access network to provide computing power nearby, and create a high-performance and low-latency service environment. Similarly, realizing fast QoS forecasting is a necessary condition for improving user service experience. In this regard, most traditional time series QoS forecasting approaches [15], [16], [17] need to go through a complex model training process, which are not suitable for continuous online training. In addition, these approaches are only designed for performing predictions for existing services without considering new services. More specifically, most existing approaches can only predict QoS for a fixed set of services. However, they ignore the fact that an MEC environment is extremely dynamic and there would be many new services being invoked in the next moment. Therefore, fast and accurate QoS forecasting with the consideration of new services in the mobile edge environment has great research significance.

We propose a novel mobility-aware fast QoS forecasting approach in the mobile edge environment, abbreviated as MEC-RDESN (<u>MEC</u> QoS forecasting based on <u>R</u>egion recognition and <u>D</u>ynamic <u>E</u>cho <u>S</u>tate <u>N</u>etwork). A user-centered edge region is continuously identified during a user's movement to obtain valid historical time series data to ensure the prediction accuracy. We adopt an improved Echo State Network to meet the requirement of fast forecasting with significantly reduced training time. Overall, the contributions of this paper include the following three aspects:

- We design a mobility-aware user-centered edge region recognition scheme. Mobile sensing technology is a low-cost but efficient way to collect environmental data [18]. The radius of the current 5G urban base station signal coverage is around 300-500m [19]. Based on the sensing technology and the signal coverage statistics of edge servers, we introduce the concept of mobile recognition in user-centered edge regions. Let us assume that the current location of a user is covered by one or many edge severs with the signal coverage radii of 300-500m. Taking this location as the center, a circle with the radius of **500m** is used to portray the user-centered edge region. This region will contain the set of edge server(s) that this user may visit next. This region also covers the server(s) previously accessed by this user, by which we can capture the relevant historical QoS data of services (denoted as s_{old}) of this user. The user-centered edge region is constantly changing along with the user's dynamic movement to update the data captured from the latest environment.
- We devise a fast QoS forecasting approach in the mobile edge environment. This approach is employed to predict the QoS performance of services from the edge servers that a user may access next based on the QoS data of s_{old}. First, an initial model is pre-trained based on Echo State Network (ESN). Then, an improved ESN model is adopted for real-time QoS prediction during user movement. The neurons in the reserve pool of the hidden layer of ESN are randomly connected and do not require training. Only the connection weights in the output layer of ESN require training, thus making continuous online training possible [20]. Since the types and numbers of services invoked by users in the edge environment are diverse, the improved ESN model is multi-service adaptive and can store information about services invoked by users. This model only needs to dynamically allocate memory for new services (denoted as s_{new}). In comparison to ESN, the improved model is able to further optimize training costs, and stabilize the connection weights between neurons to improve prediction accuracy.
- We conduct a series of experiments based on public and self-collected data sets to evaluate the effectiveness of MEC-RDESN. The experiments verify the impact of mobility awareness on QoS forecasting and the performance of the improved ESN. The experimental results also demonstrate that MEC-RDESN achieves the goal of fast forecasting while ensuring the prediction accuracy in diverse application scenarios, such as cycling, driving, taking trains, etc.

The structure of the paper is organized as follows. Section 2 surveys state-of-the-art QoS forecasting in traditional environments and mobile edge environments. Section 3 introduces the background knowledge used by our approach. Section 4 presents the details of our approach. Section 5 elaborates the experimental design and result analysis. Section 6 concludes the paper and plans our future work.

2 RELATED WORK

2.1 Traditional QoS Forecasting

Traditional QoS forecasting can be mainly achieved by similarity-based, model-based, location-based, and time-based approaches.

Zheng et al. [21] proposed a web service QoS prediction method based on user similarity and service similarity. Wei et al. [22] integrated the geographical information to build up an extended matrix factorization (MF) approach for personalized QoS prediction. Chen et al. [23] employed the location information and QoS values to cluster users and services, and made personalized service recommendation based on the clustering results. Shen et al. [24] proposed a QoS prediction method based on the geographic location information of candidate services to improve the prediction accuracy. These traditional location-based QoS prediction methods make service recommendation based on location of services or users. They ignore user mobility issues.

To further analyze the dynamics of QoS data and pursue more accurate predictions, many scholars consider the time factor during the forecasting. Existing time series QoS forecasting approaches can be divided into numerical and model-based approaches.

Numerical forecasting mostly target predicting null values by mining the relationship between historical values. Wang et al. [25] considered the influence of network states and time changing on service performance. They presented a spatio-temporal QoS prediction method, which can improve the prediction accuracy by more than 10%. Nevertheless, the forecasting bases on other available QoS values in the current time slot and cannot be directly applied to forecast future temporal QoS values. Ding et al. [26] developed a similarity-enhanced collaborative filtering method to capture the temporal features of user similarity. It performs QoS prediction based on autoregressive integrated moving average model (ARIMA). However, this method ignores the impact of user locations on prediction performance. Qi et al. [27] added the time factor to the traditional locality-sensitive hashing (LSH) technique. They proposed a time-aware privacy protection service recommendation method based on LSH. This method achieves a good trade-off between recommendation accuracy and efficiency. Nevertheless, this method assumes that the recommendation decisions only depend on a single quality criterion. Thereby, it is not suitable for a multi-criteria recommendation scenario. Ye et al. [28] used QoS historical data and short-term advertisements to predict the long-term QoS behavior of service providers. They select the optimal service combination to meet users' long-term needs. However, QoS is inherently dynamic. The QoS history and short-term advertisement cannot capture this dynamism.

Model-based forecasting generally builds a prediction model and trains the model based on time series QoS data. Wang et al. [16] designed a time-aware QoS prediction model based on momentum-incorporated latent factorization tensor. It integrates a momentum method into a tensor latent factorization model based on stochastic gradient descent (SGD). This model effectively improves the convergence speed while ensuring the prediction accuracy. However, how to realize the momentum coefficient that can determines model performance self-adaptively is still an open issue. Zhang et al. [29] proposed an online long-term QoS prediction method based on radial basis function (RBF) to solve the problems of correlation of multiple attributes, inaccurate long-term prediction and lack of dynamic update mechanism. This method achieves higher efficiency and lower error rates than traditional methods. Nevertheless, the establishment of complex relationships among multiple QoS attributes needs to be further improved. White et al. [15] adopted an LSTM-based network to predict QoS values and identify whether a service may be about to fail. This method has high training cost. Zou et al. [30] devised a time-aware QoS prediction method based on deep learning and feature integration. It integrates binarization feature and similarity feature, and learns and mines temporal features between users and services based on gated recurrent units (GRU) to realize better service QoS prediction. This method does not consider the influence of geographic locations of users and services on the prediction result.

2.2 Edge QoS Forecasting

In recent years, edge QoS forecasting studies have been carried out vigorously. The existing approaches can be grouped into environment-sensitive and model-based categories.

Wang et al. [13] proposed a collaborative filtering-based service recommendation method. It selects Top-k similar neighbors for prediction based on the similarity of users or edge servers. However, the prediction accuracy of this method is affected by the data density and distribution of edge servers. Li et al. [14] designed a trusted location-aware QoS prediction method. They integrate location clustering information and user reputation information into hybrid MF models to predict unknown QoS values. They only consider user information as an important factor to predict unknown values. Liu et al. [31] considered three important contextual factors (i.e., task type, task volume, and service workload). They proposed a context-aware multi-QoS prediction method, which uses improved case-based reasoning (CBR) to predict the values of multi-QoS attributes simultaneously. This method only focuses on services and ignores service users. Liu et al. [32] further proposed two context-aware MEC service QoS prediction schemes by combining user-related and service-related contextual factors and various MEC service scheduling scenarios. At the same time, they developed adaptive QoS prediction strategies to predict the suitable QoS data format for different MEC service scheduling scenarios. However, these schemes do not consider the impact of user mobility on MEC service QoS prediction.

Yin et al. [33] devised a novel MF model in mobile edge environments based on deep feature learning. This model learns deep latent features through convolutional neural network (CNN) to infer user or service features to build a comprehensive prediction framework. It effectively addresses overfitting under high data density. Nevertheless, the potential of context information has not been fully exploited. White et al. [17] introduced a stacked autoencoder approach on edge architectures. This method demonstrates effective reduction on training and request time while maintaining prediction accuracy. However, its accuracy heavily relies on the amount of data authorized by users. White et al. [34] also presented a forecasting approach based on noisy ESN. This approach considers that traditional time series forecasting methods have long training time and are not suitable for dynamic environments. It fulfills the need for accurate short-term forecasting in dynamic systems. Nonetheless, they did not consider dynamic and persistent prediction. Zhang et al. [35] considered the privacy and reliability issues in the MEC environment. They developed a trusted privacy-preserving QoS prediction model. This model protects the credibility of personal information and predicted results by using federated learning techniques and a reputation mechanism. Nonetheless, this method may be maliciously attacked during federated learning-based data transmission. Zhang et al. [36] developed a privacy-preserving distributed edge QoS prediction model. It employs Laplace vector mechanism to inject random noise into QoS data. It also considers user preferences and edge environment during the prediction process. However, the trade-off between prediction accuracy and privacy needs to be further explored. Our previous work [10] focuses on achieving real-time and accurate forecasting when users' geographic area changes. We did not consider real-time user movement and impact of moving speed on prediction effectiveness.

In MEC, QoS values are highly correlated with service invocation time. Mobility is a unique feature of users in the mobile edge environment, which may pose a major hurdle for timely QoS forecasting. Therefore, it is crucial to achieve fast forecasting. There is currently no fast QoS forecasting approach both considering the user mobility and time factor in mobile edge environments according to our literature survey.

3 PRELIMINARIES

3.1 Mobile Edge Computing

Large-scale resources and extensive services in cloud computing make it possible to generate new computing-intensive applications, e.g., virtual reality and intelligent environments [37], [38]. However, cloud computing relies heavily on the centralization of computing and data resources. Services provisioned in cloud data centers are usually far away from users, which cannot meet the needs of latency-sensitive applications, such as low latency, location awareness, and mobile support. In this context, researchers introduce the MEC technology to provide services to users by utilizing edge network resources.

The literature [39] defines MEC as "bringing the computing services of the wireless access network close to mobile users, thereby serving delay-sensitive and context-aware applications". The distinguishing features of MEC are dense geographical distribution of servers, close connection with end users and mobile support. Therefore, to ensure the service satisfaction of edge users, higher requirements are put forward for QoS prediction in the edge environment.

3.2 Mobile Sensing Technology

The concept of mobile sensing technology was first proposed by Professor Burke of University of California in 2006 [40], who described it as a new network architecture that can improve the trustworthiness, quality, privacy and sharing, and encourage the participation of individuals, societies and cities. In recent years, as smartphones and other types of mobile devices have become mainstream computing and communication methods, mobile sensing technology aims to use sensors on smart devices to collect and process data in real time to provide advanced application services.

In addition, most mobile sensing application scenarios usually involve collection and processing of multiple types of massive data when acquiring user information in a sensing area in real time. Therefore, techniques such as machine learning and data mining are widely used in data analysis, management and feedback. At present, mobility-aware systems are widely used in intelligent transportation, social networking, environmental monitoring and other fields [41].

3.3 Echo State Network

The Echo State Network (ESN) is a new type of neural network. It was proposed by H. Jaeger in 2001 [42]. The unique features of ESN are that it uses randomly sparsely connected neurons as the hidden layer, and the connection matrix is not adjusted after it is generated. At the same time, the generation process is independent of the training process, which greatly simplifies the training process [43]. ESNs are widely used in time series prediction [44], nonlinear system recognition [45] and image detection [46].

ESN consists of an input layer, a hidden layer and an output layer. The hidden layer is a dynamic reserve pool containing a large number of neurons. The reserve pool contains the operating state of the system, which has a short-term memory function. The core principle of ESN is to use a large-scale random sparse network (i.e., reserve pool) as an information processing medium. ESN maps the input signal from a low-dimensional input space to a high-dimensional state space, and uses a linear regression method to train the partial connection weights of the network in the high-dimensional state space [43]. Its structure is shown in Fig. 2. ESN solves the problems that the traditional recurrent neural network training more likely falls into local minimum values with low convergence speed, which achieves a key breakthrough.

4 THE MEC-RDESN APPROACH

The workflow of MEC-RDESN is outlined in Section 4.1. The four steps of MEC-RDESN are introduced in details in Section 4.2, Section 4.3, Section 4.4 and Section 4.5.

4.1 Overview of MEC-RDESN

We propose a mobility-aware fast QoS forecasting approach (MEC-RDESN) in the mobile edge environment. MEC-RDESN works towards the goals of mobility-aware, fast

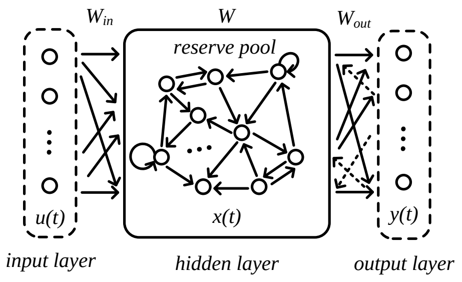


Fig. 2: The structure of ESN

and accurate edge QoS forecasting. The system workflow is shown in Fig. 3. It is mainly divided into four steps:

1). Data collection and processing. First, time-series QoS data, edge nodes and user movement information are collected from service providers, network infrastructure providers and mobile devices to form a spatio-temporal mobility-aware edge QoS data set. The edge node information includes geographic distribution of edge servers. Since edge servers are commonly deployed in base stations for mobile user access [47], here we assume that edge servers and base stations are in one-to-one correspondence. In addition, a user's mobile device records the user's moving distance and average speed. Here we adopt the scenario of *Allen* in Fig. 1 as an example. First, *Allen*'s iPhone records his movement information. Next, the network infrastructure provider provides the edge servers accessed by *Allen* along the way. Finally, we obtain the spatio-temporal mobility-aware edge QoS data set based on the time-series QoS data collected by the service provider, e.g., the time-series QoS data of *Allen* watching the *Twitch* game live video along the road.

2). Model pre-training. The data previously generated by the user after departure is used to activate the model. We employ ESN to train an initial model for the user to provide optimal hyper-parameters for subsequent predictions. In *Allen*'s scenario, the QoS data generated in the first few seconds after clicking *Twitch* is used for model pre-training. The pre-training terminates when the optimal hyper-parameters are obtained.

3). User-centered edge region recognition. The user's whole movement path is overlaid by the signal coverage of multiple edge servers. When the model pre-training is over, we take the user's current location as the center, and draw the circular edge region with the radius of 500m, where 500m is the maximum signal coverage radius of an urban 5G base station [19]. This user-centered edge region contains at least one edge server that stores the user's historical QoS data and at least one edge server that the user may visit next. Whenever the user's actual moving distance reaches 500m from the center of the current edge region, the user is switched to a new edge region. Fig. 4 shows all user-centered edge regions in *Allen*'s scenario.

4). Fast forecasting. The user employ the pre-trained model in step 2) in Section 4.1 as the initial QoS forecasting model. With the real-time tracking of the user-centered edge region, the input of the ESN is determined by whether the user switches to a new edge region. If the answer is no, the current QoS values are used as the input; otherwise the input is the average QoS values obtained from the previous 500m. We design a dynamic ESN model to perform QoS forecasting. The model stores information about services that the user previously invoked (i.e., previously trained connection weights among the neurons assigned to these services). When QoS data of new services is generated, the model assigns connection weights between neurons to these new services every 500m based on the real-time QoS data generated by the user. This ensures the accuracy of edge QoS forecasting results. The network training and forecasting actions are terminated as the user stops moving. In *Allen*'s example, the prediction is performed by iPhone based on the QoS data of the *Twitch* game live video instantly generated from its service provider or the average QoS data generated within the last 500m, which is provided by its service provider in the form of data logs. At the same time, the model parameters are periodically updated every 500m through training with the new data.

4.2 Data collection and processing

First, a user's mobile device records his/her mobile information. The information is recorded as $UM_{info} = \{(D_{t_1}, D_{t_2}, \ldots, D_{t_k}), \bar{V_u}\}$, where D_{t_k} is the user's real-time moving distance at time t_k (e.g., 0.5km at 9:01 am) and $\bar{V_u}$ is the user's average speed in the last 500m. Then, edge server information is collected from its network infrastructure provider. It is recorded as $EN_{info} = \{(L_{t_1}, L_{t_2}, \ldots, L_{t_k}), S\}$, where L_{t_k} is the location (i.e., longitude and latitude) of an edge server accessed by the user in real time, and S is the signal coverage radius of base stations (i.e., 300-500m). Next, the user's time series QoS data is collected from service providers. It is recorded as $QoS = \{Q_{t_1}, Q_{t_2}, \ldots, Q_{t_k}\}$, where Q_{t_k} is the QoS data generated by the user at time t_k. Finally, we aggregate the three data sets into the user's spatio-temporal mobility-aware edge QoS data set, which is expressed as $[UM\_EN\_QoS]_{info} = \{(D_{t_k}, \bar{V_u}), (L_{t_k}, S), Q_{t_k}\}$. The data set forming process is illustrated in Fig. 5. Among them, UM_{info}, EN_{info} and QoS are each one-dimensional data, which are finally formed a three-dimensional spatial data set.

Let us make use of the scenario of the taxi passenger *Allen* to explain this process. During *Allen*'s journey, his iPhone records the real-time moving distance, e.g., (10km, 9:10am), and the average speed in the last 500m of the taxi, e.g., 50km/h. Edge servers S_1, S_2, S_3, S_4 and S_5 respectively log the *Twitch* game live video quality data, e.g., frame rates and bit rate. These QoS data and accessed edge server information (i.e., longitude and latitude, signal coverage radius, etc.) along the way are recorded by service providers and network infrastructure providers, which are shared with *Allen*'s iPhone. We fuse these three groups of data to form a spatio-temporal mobility-aware edge QoS data set, e.g., $[UM\_EN\_QoS]_{info}$=[(10km,9:10am,50km/h),((118.79783, 31.92262),500m),(25fps,1800kbps)]. We then employ a map software [1] to map the user's movement and locate the accessed edge servers.

1. https://www.ldmap.net/

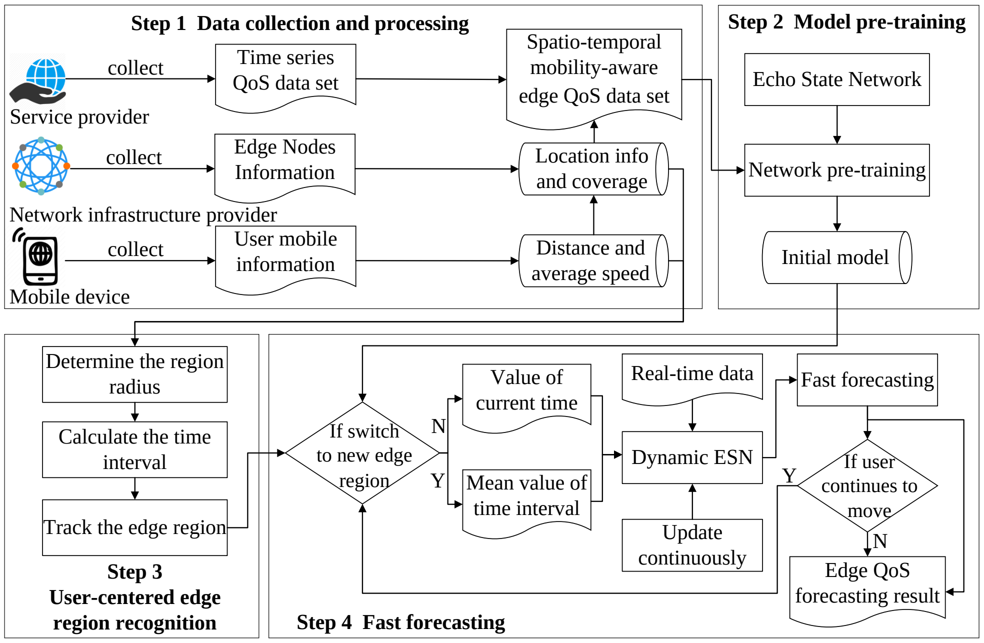

Step 1 Data collection and processing
collect
Service provider
Time series QoS data set
Spatio-temporal mobility-aware edge QoS data set
collect
Edge Nodes Information
Location info and coverage
Network infrastructure provider
collect
User mobile information
Distance and average speed
Mobile device
Step 2 Model pre-training
Echo State Network
Network pre-training
Initial model
Determine the region radius
Calculate the time interval
Track the edge region
Step 3 User-centered edge region recognition
If switch to new edge region
N
Y
Value of current time
Mean value of time interval
Real-time data
Dynamic ESN
Update continuously
Fast forecasting
If user continues to move
Y
N
Edge QoS forecasting result
Step 4 Fast forecasting

Fig. 3: MEC-RDESN overview

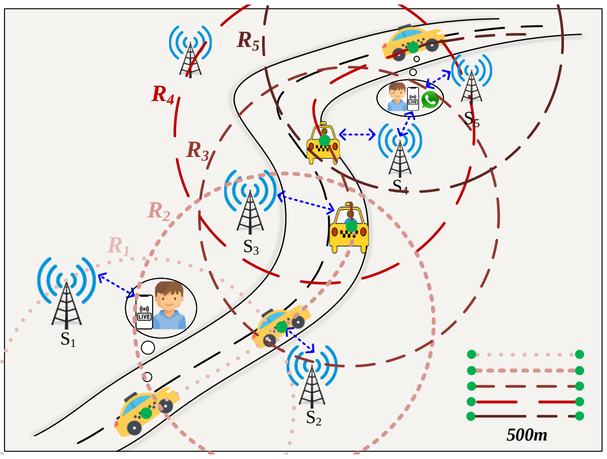


Fig. 4: User-centered edge regions in *Allen*'s scenario

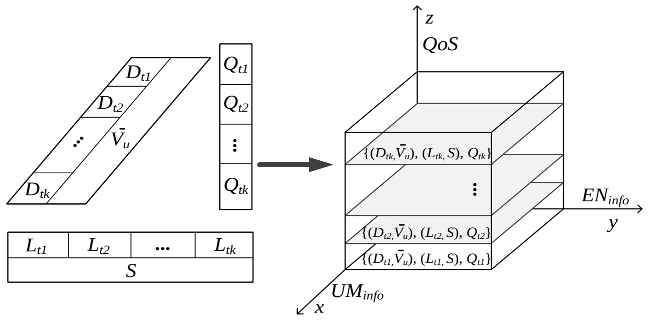

D_{t1}
D_{t2}
$\bar{V}_u$
D_{tk}
Q_{t1}
Q_{t2}
Q_{tk}
L_{t1}
L_{t2}
...
L_{tk}
S
z
QoS
$\{(D_{tk},\bar{V}_u), (L_{tk}, S), Q_{tk}\}$
$\{(D_{t2},\bar{V}_u), (L_{t2}, S), Q_{t2}\}$
$\{(D_{t1},\bar{V}_u), (L_{t1}, S), Q_{t1}\}$
EN_{info}
y
UM_{info}
x

Fig. 5: The edge QoS data set forming process

4.3 Model pre-training

We view the QoS data generated in a user's departure phase as the initial data, and obtain a pre-trained ESN model based on the initial data. Due to the structural characteristics of the ESN, its main training process is concentrated between the reserve pool and the output unit, and the connection weights between neurons in the reserve pool are randomly generated without training. Therefore, the ESN model training process is simple and efficient. For *Allen*, once he generates the data set in the departure phase, the ESN can quickly complete the pre-training process. The purpose of model pre-training is to provide him with an initial model for predicting QoS of the *Twitch* game live video.

When ESN is used for time-series forecasting, its input and output are both time-series data. At this point, the ESN can be regarded as a nonlinear filter to realize the conversion from input to output. The update process of the ESN is as follows:

$$\widetilde{x}(t) = f\left(W_{in}\left[1; u(t)\right] + Wx(t-1)\right) \quad (1)$$

$$x(t) = (1-\alpha)x(t-1) + \alpha\widetilde{x}(t) \quad (2)$$

Where $f(\cdot)$ is the activation function of the neurons in the reserve pool, and the common activation functions include *sigmoid*, *tanh* and *relu*. $u(t) \in \mathbb{R}^{N_u}$ is the input, $x(t) \in \mathbb{R}^{N_x}$ is the state of the reserve pool and $\widetilde{x}(t) \in \mathbb{R}^{N_x}$ is its update. $\alpha \in (0,1]$ is the leaking rate. At time step t, $f(\cdot)$ is applied element-wisely. $[\cdot;\cdot]$ stands for a vertical vector (or matrix) concatenation. $W_{in} \in \mathbb{R}^{N_x \times (1+N_u)}$ and $W \in \mathbb{R}^{N_x \times N_x}$ are the input and recurrent weight matrices respectively. They do not need to be trained and remain unchanged after initial generation.

We design an improved ESN that stores information about services invoked by users, without regenerating the weight connections between all neurons in each subsequent round of training and prediction. In other words, MEC-RDESN stores W_{in} for s_{old} (i.e., old services) and W, and only needs to allocate W_{in} for s_{new} (i.e., new services).

Generally, it is necessary to generate an appropriate reserve pool to ensure the echo state characteristics of the

ESN, i.e., the spectral radius $\rho(W)$ of the connection weight matrix W of the reserve pool, is less than 1. The calculation process of $\rho(W)$ is:

$$W = \alpha_w \frac{W^r}{|\lambda_{max}|} \tag{3}$$

Where α_w $(0 < \alpha_w < 1)$ is a scaling factor, W^r is a randomly generated sparse matrix, and λ_{max} is the largest eigenvalue of the matrix W^r.

The learning methods of the ESN can be divided into two types: unsupervised and supervised. The training process of the unsupervised method is only based on the input variables of the network, and has nothing to do with the target output. Its training process is to learn and explore the state variables of the reserve pool for optimization. The supervised method considers both the input and target output variables of the network, and the goal of training is to minimize the difference between the network output and the target output.

A typical supervised ESN training process can be expressed as: 1) the sparse connection weight matrix W between the processing units of the reserve pool is randomly generated in advance; 2) training data stimulates the processing units of the reserve pool to generate state variables through the randomly generated weight matrix W_{in}; 3) after collecting the state variables, linear regression is used to minimize the training error to obtain W_{out}. The linear readout layer is defined as:

$$y(t) = W_{out}\,[1; u(t); x(t)] \tag{4}$$

Where $y(t) \in \mathbb{R}^{N_y}$ is the output of the ESN, $W_{out} \in \mathbb{R}^{N_y \times (1+N_u+N_x)}$ is the output weight matrix, and $[\cdot;\cdot;\cdot]$ again stands for a vertical vector (or matrix) concatenation.

The ridge regression is one of the methods used in the weight learning process. The calculation formula is expressed as:

$$W_{out} = Y_{target} X^T (XX^T + \lambda_r I)^{-1} \tag{5}$$

Where $X = [x_{(1)} \ldots x_{(t)}]$ is the pool state, $Y_{target} = [y_{(1)} \ldots y_{(t)}]$ is the target value, λ_r $(\lambda_r > 0)$ is the ridge parameter, and I is the identity matrix.

For a given input signal $u(t) \in \mathbb{R}^{N_u}$, the desired target output signal $y_{target}(t) \in \mathbb{R}^{N_y}$ is known. Our goal is to learn a model whose output is $y(t) \in \mathbb{R}^{N_y}$ making the error E between y_{target} and $y(t)$ is as small as possible and can be applied to more data. We use the Root Mean Square Error (RMSE) to measure E:

$$E(y, y_{target}) = \frac{1}{N_y} \sum_{i=1}^{N_y} \sqrt{\frac{1}{T} \sum_{t=1}^{T} (y_i(t) - y_{itarget}(t))^2} \tag{6}$$

It is also the mean of the i dimension of the output N_y, where i is the total dimension of N_y. T is the total number of discrete time points in the training data set.

4.4 User-centered edge region recognition

In real scenarios, the signal coverage of a base station where an edge server co-locates can cover a certain circular region. Mobile users within the region can access the edge server. Correspondingly, if the whole area where the user is located is fully covered by the signal coverage of edge servers, for a user, all the edge servers that s/he can access are also within a circular area. We name this area as a user-centered edge region. We denote the maximum signal coverage radius (i.e., 500m [19]) of urban base stations as SC, and use SC as the radius R to draw a user-centered edge region (i.e., $R_{u\in ER} = SC$). We calculate the time T_u spent by the user when his/her moving distance reaches R (i.e., $T_u = R/\bar{V}_u$), where $\bar{V}_u$ is the user's average speed over R. Whenever the user's moving distance reaches $D' = D + R$, a new user-centered edge region is created.

We record the straight line as SL and the detour as DL. There are four cases for the moving path of the two time intervals, namely SL+SL, SL+DL, DL+SL and DL+DL. Fig. 6 abstracts the user moving path in two adjacent time intervals $T_u - 1$ and T_u in the real scene, and the meaning of the legend is explained on the right side. It covers all the possibilities. It can be seen that all the moving paths are within the drawn edge region (i.e., the user is moving in this edge region during the time intervals $T_u - 1$ and T_u). Thus, the historical data used for prediction are generated in the same edge region environment. The edge region recognition mode provides effective time series historical data for QoS prediction, and the edge region follows the user's movement in real time to further ensure the freshness of historical data.

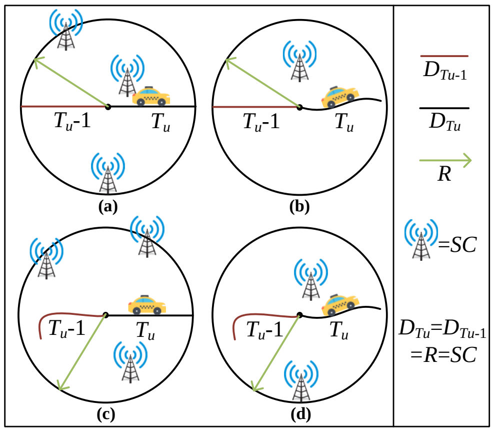


Fig. 6: Four types of user moving path

4.5 Fast forecasting

The fast QoS forecasting of an edge service deployed by a user can be accomplished by means of a dynamic ESN. The dynamic ESN is based on a pre-trained ESN and the historical QoS data obtained from the edge servers in the user-centered edge region. According to the user-centered edge region recognition solution depicted in Section 4.4, we first judge whether the user has entered a new user-centered edge region. If not, the prediction will be made by feeding the ESN with the current QoS value; otherwise, the average QoS value in the latest time interval T_u will be fed into the ESN. The forecasting process is calculated in term of equation (1)∼(4). During the process of user movement, when the user-centered edge region changes, the dynamic ESN is updated based on the latest historical data in the last T_u (i.e., the QoS data generated in the last 500m moving distance of the previous region) to ensure the prediction accuracy. Here the dynamic ESN can memorize the services have been invoked. In other words, the model stores the W_{in} of the invoked services and W, and only needs to generate random connection weights for new services. This improvement can assist saving training and prediction costs. In *Allen*'s case, the model stores the weights of *Twitch* game live video. If he uses new app services such as *WhatsApp*, the model will assign weights to these new app services.

Algorithm 1 Mobility-aware fast edge QoS Forecasting

```
Require: The moving distance of mobile user $u$ at time $t_k$ is $D_{t_k}$, the average speed of $u$ is $\bar{V_u}$. $SC$ is the base station maximum signal coverage radius, $\{Q_{t_1}, Q_{t_2}, \ldots, Q_{t_k}\}$ is the real-time data generated by $u$ accesses edge server $S = \{S_1 \ldots S_k\}$ along the way. $A$ is the user-centred edge region. $s$ are services invoked by $u$, $s_{old}$ are all services have been invoked by $u$, and $s_{new}$ are new services invoking by $u$.
Ensure: Edge QoS forecasting results for $u$
 1: Record $D_{t_k}$, $\bar{V_u}$ of $u$;
 2: Collect $Q_{t_k}$;
 3: Pre-train the network based on $\{Q_{t_1}, Q_{t_2}, \ldots\}$;
 4: Draw the edge region with the $u$'s location as the center, and the $SC$ value as the radius $R$;
 5: Calculate the time interval $T_u = R/\bar{V_u}$;
 6: for $D_{t_k}$++ do
 7:   if $D_{t_k} < D_{T_u - n} + R$ $(n = 1, 2 \ldots)$, i.e., $u.location \in$ Edge region $A_n$ then
 8:     Perform forecasting based on the current moment value $Q_{t_k}$ as the network input;
 9:     Output edge QoS forecasting results;
10:   else
11:     $T_{u-n}++$, $A_n++$;
12:     if $s \cap s_{old} \neq \emptyset$ then
13:       Assign new connection weights to $s_{new}$;
14:     else
15:       Use stored connection weights;
16:     end if
17:     Continuously train the network based on the latest time interval value $Q_{T_u - n}$;
18:     Take the mean value of the latest time interval $Q_{T_u - n}$ as the network input for prediction;
19:     Output edge QoS forecasting results;
20:   end if
21: end for
```

Based on Algorithm 1 and equation (5)~(6), every time when a user enters a new edge region during the prediction process, the dynamic ESN is updated based on the historical QoS data generated in the last time interval and then produces the QoS forecasting results. The training and prediction are iterated until the user stops moving or no new QoS data is generated (e.g., *Allen* reaches his destination or stops watching the video).

5 EVALUATION

We conduct all the experiments in a computer system with Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz, 16.0 GB RAM, Windows 10, and MATLAB R2018b. We collect real data in Tomcat 9.0.59, Android studio 4.1.2, JDK 1.8 and Gradle 6.5 environment. We perform model pre-training, edge region recognition, and fast forecasting in this environment.

We conduct both simulation experiments based on several existing data sets and real world experiments on a university campus to verify the feasibility and effectiveness of our model.

5.1 Research Questions

A set of dedicated experiments are performed to explore the following research questions:

- RQ1: How much data can effectively activate the model?
- RQ2: What are the optimal hyper-parameters for user mobility-aware model pre-training?
- RQ3: What is the performance of the proposed method in comparison to existing forecasting methods?
- RQ4: What are the impact of the base station signal coverage radius and movement speed on the time efficiency of edge forecasting?

5.2 Simulation Experiment

5.2.1 Data Set Description

Our simulation experiments base on two data sets – a GPS trajectory data set and a time series QoS data set. These data sets can be downloaded from the data sources used in [48], [49]. The first data set [2] is provided by Geolife project. It contains 17,621 trajectories with a total distance of 1,292,951 kilometers and a total duration of 50,176 hours. The data set is in the form of a sequence of time-stamped points, each of which contains the information of latitude, longitude and altitude. We use the longitude and latitude information to describe user movement paths. The second data set [3] describes real-world QoS evaluation results of 142 users (IDs: 0-141) on 4,500 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). In addition, we employ a base station distribution service [4] to obtain the geographical locations of base stations.

5.2.2 Experimental Data Preprocessing

In the trajectory data set, there are 73 users who labeled their trajectories with transportation mode. We choose four trajectories, each of which is with a distinct mode (namely *bike*, *taxi*, *subway* and *train*). Then, we locate the base stations around the four paths by referencing the base station distribution data, and label them with IDs from 0 to 141. Next, we use the IDs to identify the corresponding QoS data set. Thereby, we can obtain the spatio-temporal mobility-aware edge QoS data set of each path. Fig. 7 shows the distribution of the four paths and their surrounding edge servers. The trajectory and base station information is shown in Table 1.

We set the radius of the user-centered edge regions to *R* (*R*=500m), and determine the time interval T_u of each user

2. https://www.microsoft.com/en-us/download/details.aspx?id=52367
3. https://github.com/wsdream/wsdream-dataset
4. https://www.opengps.cn/Data/Cell/Region.aspx

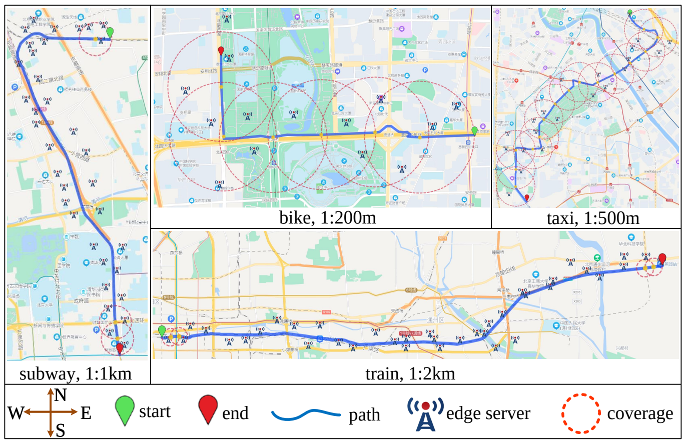


Fig. 7: Paths with different transportation modes and edge server distribution

TABLE 1: Trajectory and base station information

Trajectory	Time	Length(km)	BSs number
bike	2008/7/22	3.55	17
taxi	2008/6/28	8.59	26
subway	2008/5/14	14.37	43
train	2008/8/7	35.63	56

according to their average speed (e.g., if $T_{u_1} = R/\bar{V}_{u_1} = T$, thus when $V_{u_2} = 2\bar{V}_{u_1}, T_{u_2} = R/V_{u_2} = T/2$). In this experiment, we set the average speed of four transportation modes as: $\bar{V}_{bike} = 20km/h, \bar{V}_{taxi} = 50km/h, \bar{V}_{subway} = 100km/h, \bar{V}_{train} = 200km/h$. Therefore, the corresponding time intervals are: $T_{bike} = R/20, T_{taxi} = R/50, T_{subway} = R/100, T_{train} = R/200$. Since T_{train} is the smallest time interval, we fuse T_{train} with one time slice of the QoS data set depicted in Section 5.2.1, and thus obtain the number of time slices contained in each time interval of the four paths. The corresponding process is shown in Fig. 8.

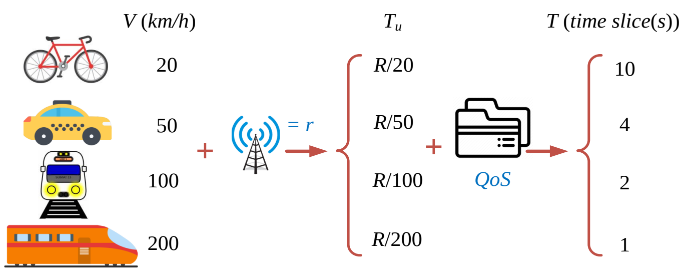


Fig. 8: Correspondence between time interval and time slices of the four paths

5.2.3 Experimental Procedure

It is expected that the experiments can prove that the proposed mobility-aware fast edge QoS forecasting approach can achieve accurate and fast prediction.

To address *RQ1*, the ESN is activated to generate the first W_{out}. Since theoretically the data of the first two time slices can activate the model (i.e., obtaining the W_{out}), we calculate the prediction error values of different time slices (i.e., 3T-6T, where the last time slice of each is the prediction result).

To address *RQ2*, we adjust the hyper-parameters of the ESN through pre-training and try to find the optimal values for subsequent user mobility-aware model training.

To address *RQ3*, we compare MEC-RDESN with several mainstream time series methods, including a baseline method, a time series model, two classical neural networks, a vanilla ESN model, and a vanilla ESN model with region recognition. The six comparative approaches are as follows:

- *Average.* A simple time series forecasting model that uses the average of a time series as the forecasting value for the next period without training.
- *SARIMA.* An extension of AutoRegressive Integrated Moving Average (ARIMA), which is used to model periodic time series data and predict future values [50].
- *RNN.* A recurrent neural network that takes sequence data as input for forecasting, where all nodes are connected in a chain [51].
- *LSTM.* An RNN that is able to learn long-term dependence within time-series data. It contains three control gates and a cell structure to make the network have memory capabilities [52].
- *ESN.* A forecasting model based on the original echo-state network [42].
- *RESN.* A forecasting model based on region recognition and ESN that does not store connection weights for invoked services during movement.

To address *RQ4*, we calculate the leading time of edge forecasting at different signal values and speeds.

5.2.4 Experimental Results

(1) **The amount of data to activate the model**

Theoretically the data of two time slices can activate the model to obtain the W_{out} value for prediction. Table 2 shows the prediction errors for *RT* and *TP* data sets under different time slices. The time slice with the lowest error is used to activate the model.

TABLE 2: RMSE of different time slices

	Path/T	3	4	5	6
RT	bike	0.8025	**0.7153**	1.0499	1.0296
	taxi	1.0534	0.9723	**0.85**	1.1605
	subway	2.4601	**1.5353**	1.5768	1.6826
	train	**0.6114**	1.3205	2.0614	0.7908
TP	bike	35.8133	25.7665	**17.2443**	49.5525
	taxi	30.4186	24.1308	**1.7344**	12.6351
	subway	**5.0482**	6.508	36.2323	44.5756
	train	20.6927	58.0275	14.6619	**1.635**

(2) **The optimal hyper-parameters for model pre-training**

Leaking Rate. The leaking rate α of the reserve pool in equation (2) can be regarded as the rate of dynamic update of the reservoir. For time-varying data, α is an important parameters that determines the duration of short-term memory in ESN [20]. Fig. 9 and Fig. 10 show the prediction error values of the four different paths depicted in Section 5.2.2 on *RT* and *TP* data sets at different leaking rates. It can be seen that, with the increase of the leaking rate, all the prediction errors show a U-shaped curve. We take the leaking rate value of each path when the error is the lowest, e.g., 0.5 for *RT*, and 0.1 for *TP* in the *bike* path.

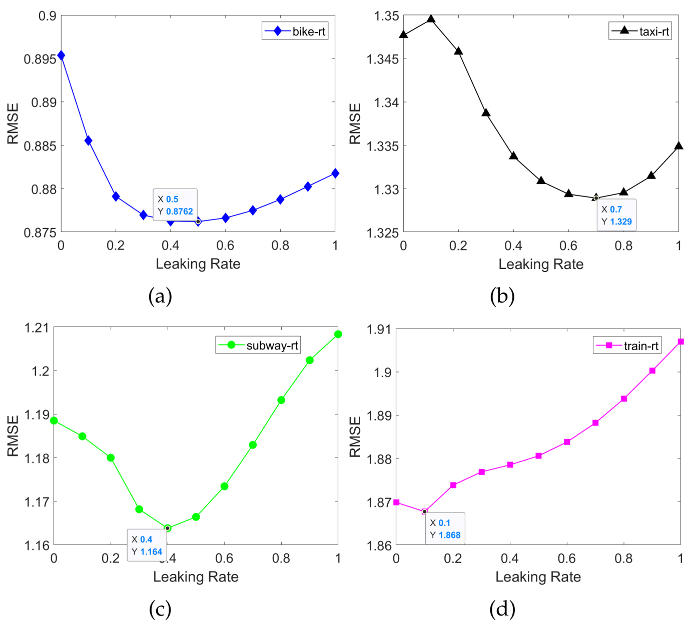


Fig. 9: Forecasting errors of the *RT* data set of four paths with increasing leaking rate: (a) bike, (b) taxi, (c) subway, (d) train.

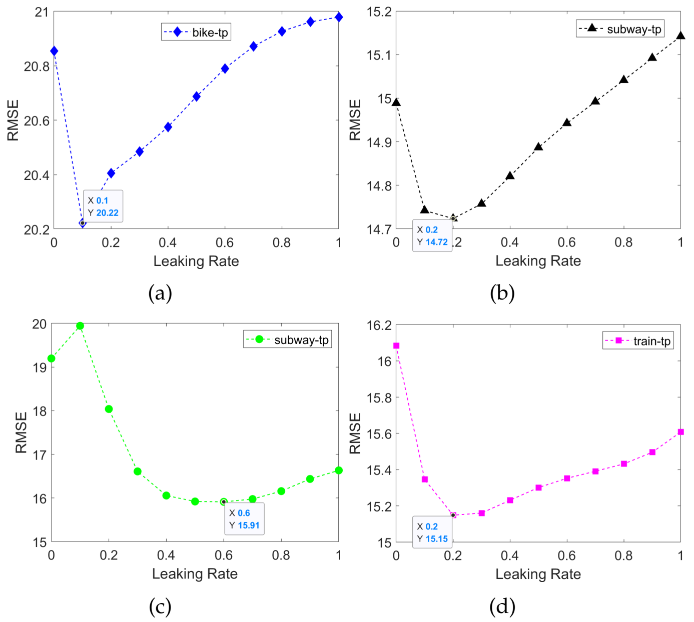


Fig. 10: Forecasting errors of the *TP* data set of the four paths with increasing leaking rate: (a) bike, (b) taxi, (c) subway, (d) train.

Reservoir Size. Another important parameter in equation (2) is the size of the reserve pool N_x, which refers to the number of neurons contained in the reserve pool. The choice of the size of the reserve pool is related to the complexity of the model and the number of training samples. In general, the larger the value of N, the stronger the description ability of the network and the higher the prediction accuracy. However, if N is too large, overfitting will easily occur. Fig. 11 and Fig. 12 show the changes in the prediction error values of the four different paths on the *RT* and *TP* data sets with the reservoir size increased from 50 to 500 in the step of 50. It can be seen from Fig. 11(a) and Fig. 11(c) that, with the increase of the reservoir size, the prediction error is decreasing. Thus, we take 500 as the reservoir size of the *bike* and *subway* on the *RT* model. For *taxi* and *train*, we take the reservoir size of 350 and 100 respectively as the parameter of the model corresponding to the lowest error values in their cases.

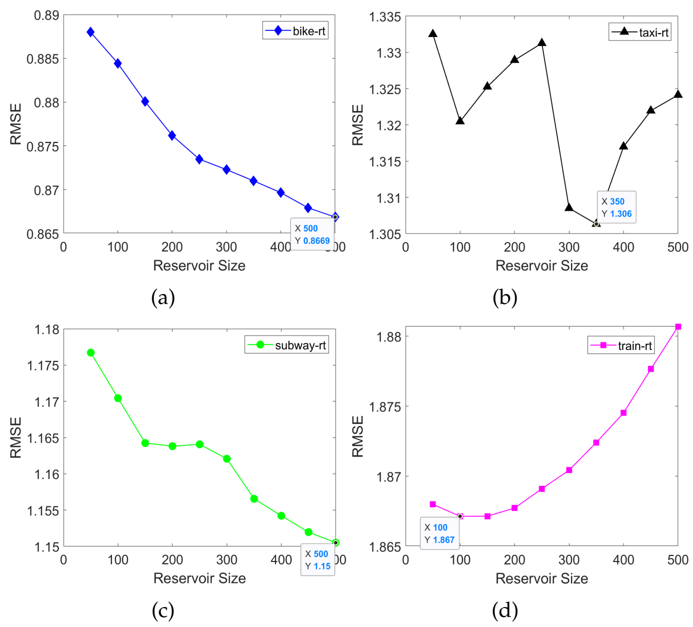


Fig. 11: Forecasting errors of the *RT* data set of the four paths with increasing reservoir size: (a) bike, (b) taxi, (c) subway, (d) train.

Activation Function. Table 3 shows the error values of the four paths on the *RT* and *TP* data sets under different activation functions. It can be seen that *tanh* achieves the best prediction performance in most cases, with little difference in the *bike* and *train* paths of the *RT* data set. In the *TP* data set, *tanh* has obvious advantages. Based on the above analysis, we choose *tanh* as the uniform model activation function. The average time for model pre-training on the *RT* and *TP* data sets is respectively 1.59s and 1.46s, which indicates that a user only needs less than 2s after departure to obtain a pre-trained model.

(3) **MEC-RDESN Performance**

The user performs forecasting based on the pre-trained model. Whenever s/he enters a new edge region, model training is performed based on the historical data in the latest time interval to improve the real-time performance. The training frequency exhibits periodic characteristics as the user moves. Based on the number of time slices in

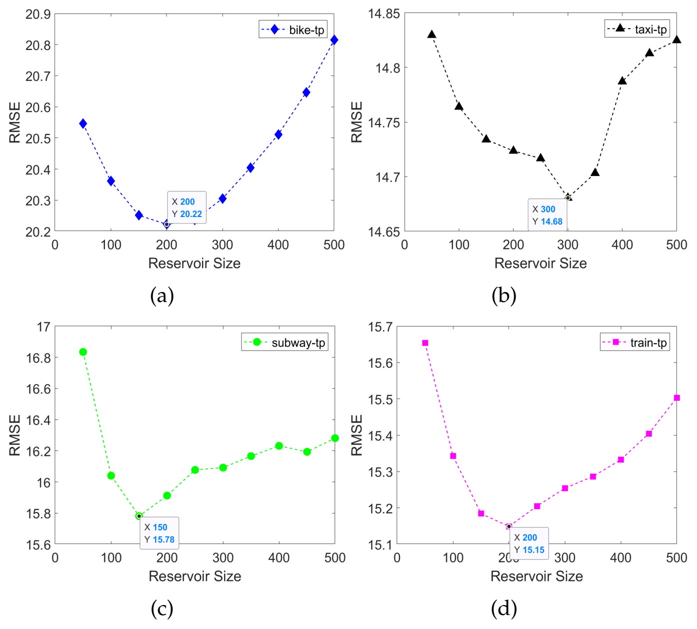


Fig. 12: Forecasting errors of the *TP* data set of the four paths with increasing reservoir size: (a) bike, (b) taxi, (c) subway, (d) train.

TABLE 3: RMSE of Activation Functions on: (a) *RT*, (b) *TP*.

(a)

Fuction/Path	bike	taxi	subway	train
sigmoid	0.8738	1.3093	1.1777	**1.867**
tanh	0.8669	**1.3063**	**1.1505**	1.8671
relu	**0.8628**	1.3098	1.1565	1.9931

(b)

Fuction/Path	bike	taxi	subway	train
sigmoid	20.2983	14.8	17.6772	15.1565
tanh	**20.2225**	**14.6805**	**15.7806**	**15.149**
relu	30.7396	16.3455	17.619	20.6051

the time interval of each path introduced in Section 5.2.2, we perform periodic training and predict the QoS value in the next time interval. The forecasting performance in the movement process is measured in term of training time, forecasting time and forecasting accuracy.

Training Time. Fig. 13 shows the training time of the *RT* data set for the four paths in each edge region during the movement process, and Fig. 14 shows the training time of the *TP* data set. The *Average* method does not require training. Since the number of services invoked by a user changes dynamically, the training time fluctuates with the number of services. From the results of the two data sets, it can be seen that there are significant differences among the training time of these models. Among all the models, the training time of *SARIMA* has the least fluctuations, mostly around 6s. The training time of *RNN* and *LSTM* fluctuates greatly, especially *LSTM*. In contrast, the training time of the *ESN* series of methods is very short, and the longest training time is about 1s. The training time of *MEC-RDESN* is the shortest in the *ESN* series, because it further saves the time to generate connection weights.

Table 4 shows the total training time of the six methods on *RT* and *TP* data sets on each path. It can be seen that our proposed MEC-RDESN method has the lowest total training time. Therefore, it greatly reduces the training cost.

TABLE 4: Total training time of four paths: (a) *RT*, (b) *TP*.

(a)

RT (s)	bike	taxi	subway	train
SARIMA	33.155	60.986	121.047	205.344
RNN	7.094	8.187	38.618	70.448
LSTM	30.443	39.172	393.608	645.1
ESN	1.287	1.669	14.971	16.478
RESN	1.285	1.768	14.508	14.646
MEC-RDESN	**1.039**	**1.084**	**12.055**	**13.724**

(b)

TP (s)	bike	taxi	subway	train
SARIMA	34.475	64.846	128.964	226.441
RNN	6.681	7.82	32.679	67.015
LSTM	26.558	46.048	321.82	548.967
ESN	0.787	1.328	9.242	17.529
RESN	0.755	1.535	8.518	14.379
MEC-RDESN	**0.61**	**1.07**	**7.581**	**12.764**

Forecasting Time. The forecasting time is very short compared to the training time. Table 5 shows the average forecasting time of each method on the *RT* and *TP* data sets. It can be seen that the *Average* has the shortest forecasting time because it only requires some simple calculations. The forecasting time of *MEC-RDESN* is about 0.005s, which completely meets the requirement of fast forecasting.

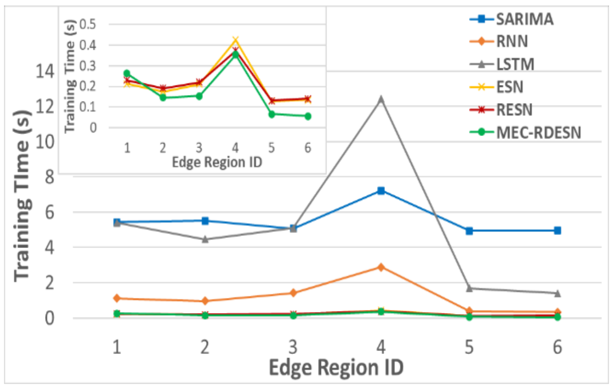
TABLE 5: Average forecasting time of *RT* and *TP*

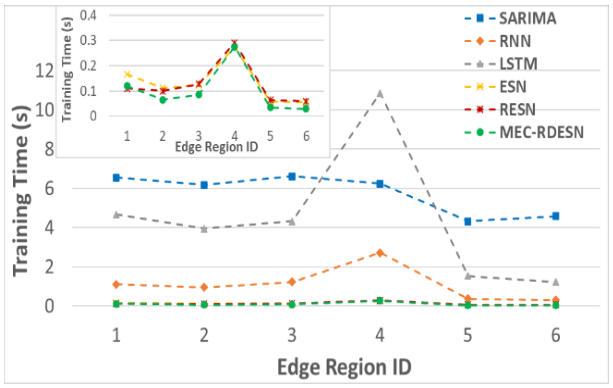
Data set	Average	SARIMA	RNN	LSTM	ESN	RESN	MEC-RDESN
RT (ms)	0.15	229.94	2.51	4.58	5.89	5.64	5.63
TP (ms)	0.15	233.67	2.67	4.93	4.04	4.26	4.43

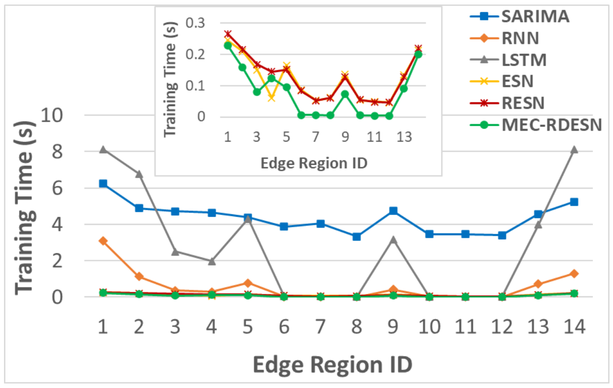
Forecasting Accuracy. Each time when a user enters a new edge region, an updated forecasting is performed. Table 6 and Table 7, Fig. 15 and Fig. 16 show the prediction errors of the four transportation modes in each edge region over the *RT* data set [5]. We record error values for the seven approaches. The most accurate forecasting results in Table 6 and Table 7 are marked in bold, and the lowest point of each region in Fig. 15 and Fig. 16 represents the most accurate prediction.

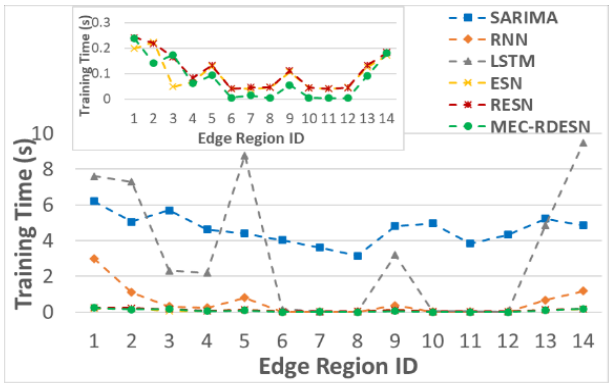
Similarly, Table 8 and Table 9, Fig. 17 and Fig. 18 show the prediction errors of the *TP* data set [5]. Further analysis of the experimental results shows that MEC-RDESN achieves the most accurate prediction results on 80% of the *RT* data set and 78% of the *TP* data set. When the environmental changes in the adjacent edge regions are not obvious and

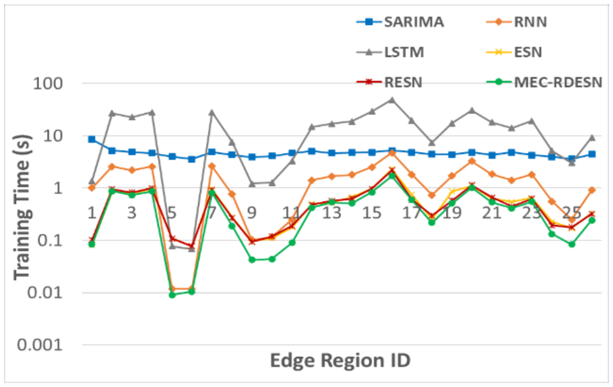
5. Note: Since the paths of *subway* and *train* modes contain too many time intervals, their results are shown in figures instead of tables.

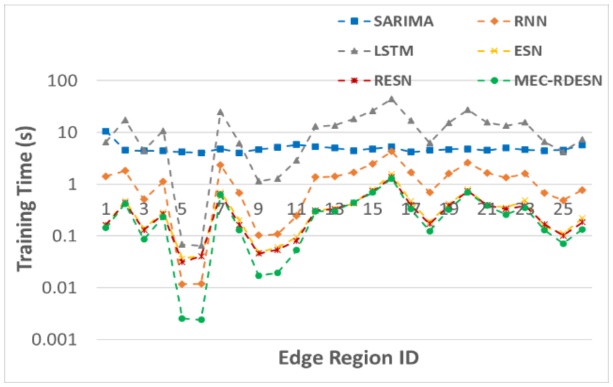

(a)

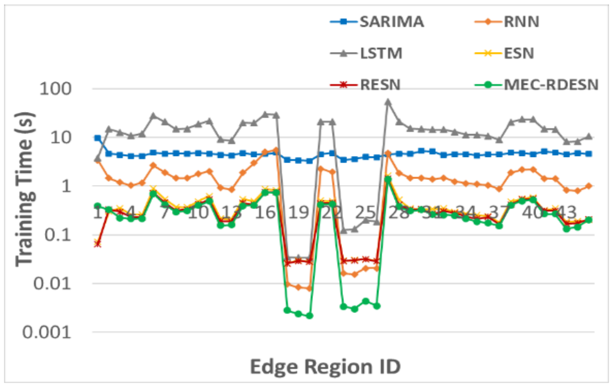

(a)


(b)


(b)


(c)


(c)


(d)

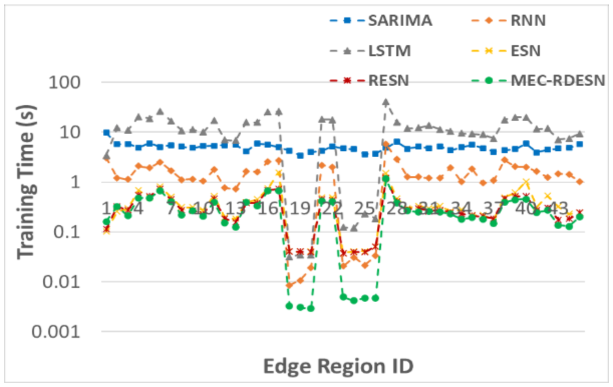

(d)

Fig. 13: Training Time of the *RT* data set of the four paths: (a) bike, (b) taxi, (c) subway, (d) train.

Fig. 14: Training Time of the *TP* data set of the four paths: (a) bike, (b) taxi, (c) subway, (d) train.

TABLE 6: RMSE of the *RT* data set of *bike* path.

ER_ID	Average	SARIMA	RNN	LSTM	ESN	RESN	MEC-RDESN
a-1	0.8753	1.2351	0.9537	1.1451	0.8669	0.8699	**0.8669**
a-2	0.9343	1.265	1.0162	1.3066	0.9719	0.9305	**0.9276**
a-3	0.8209	0.9063	0.8483	1.1927	0.7853	0.7768	**0.7758**
a-4	0.6796	0.918	0.7206	1.1788	0.693	0.6624	**0.6617**
a-5	0.899	1.1344	0.9921	1.4238	0.9301	0.8989	**0.8949**
a-6	0.8864	1.2583	1.0757	1.5897	0.9978	0.8818	**0.8754**

TABLE 7: RMSE of the *RT* data set of *taxi* path.

ER_ID	Average	SARIMA	RNN	LSTM	ESN	RESN	MEC-RDESN
b-1	1.4445	2.3673	1.3645	1.5123	1.3561	1.3572	**1.3561**
b-2	1.3361	1.6255	1.3903	1.7067	1.3864	1.3042	**1.3039**
b-3	2.7298	2.8079	**2.4027**	2.4925	2.7948	2.7576	2.7671
b-4	2.5257	2.7072	2.0887	2.5877	2.1117	2.0681	**2.0671**
b-5	1.5683	1.6111	1.3242	2.0292	1.5919	1.2508	**1.2504**
b-6	1.901	2.7531	1.8273	2.2586	1.8567	1.8247	**1.8025**
b-7	2.1888	3.5497	2.1922	2.6654	2.211	2.1918	**2.1428**
b-8	1.9869	1.9495	1.8079	2.8102	1.87	1.6871	**1.6781**
b-9	1.149	1.3741	1.271	1.6219	1.6638	1.1482	**1.1458**
b-10	1.0871	1.4476	**0.9557**	1.2301	1.3306	1.1351	1.1398
b-11	**1.6653**	1.7052	1.6885	2.3537	1.892	1.834	1.8192
b-12	1.5048	3.2046	1.416	1.718	1.4094	1.4216	**1.4032**
b-13	1.5021	1.8191	1.4983	2.0317	1.6146	1.5985	**1.4698**
b-14	1.5875	1.6213	1.498	1.612	1.4847	1.4791	**1.4781**

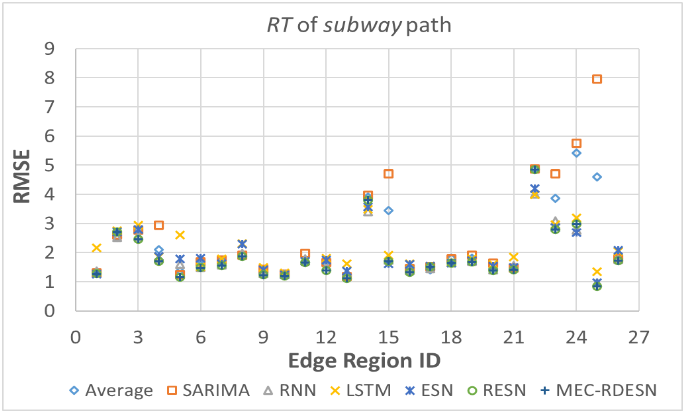


Fig. 15: RMSE of the *RT* data set of the *subway* path

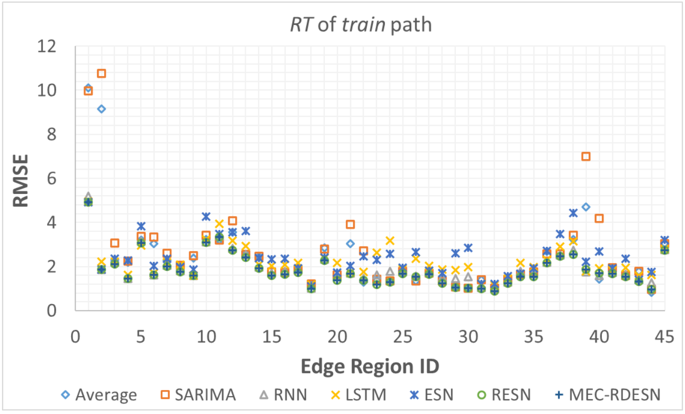


Fig. 16: RMSE of the *RT* data set of the *train* path

the data fluctuations are relatively stable (e.g., edge regions 9 and 10 in Fig. 17, and edge regions 15 and 16 in Fig. 18), the *Average* achieves lower prediction error values. The *RNN*-related approaches perform well in only a few edge regions (e.g., edge regions 3 and 10 in Table 7), because the training data in these regions has faster convergence rates. By comparing the results between *RESN* and *ESN*, it can be seen that obtaining valid regional historical data based on edge region recognition plays an important role in improving prediction accuracy. Since *MEC-RDESN* has a more stable weight connection than *RESN*, it has better prediction performance.

TABLE 8: RMSE of the *TP* data set of *bike* path.

ER_ID	Average	SARIMA	RNN	LSTM	ESN	RESN	MEC-RDESN
a-1	28.1072	21.4568	35.6324	54.2665	20.2225	20.2227	**20.2225**
a-2	53.0186	54.0161	59.4281	84.519	53.4868	52.3896	**52.3787**
a-3	32.8379	34.8863	34.6544	45.2192	36.1301	32.1866	**31.7577**
a-4	18.1733	25.6648	19.7381	31.7256	19.0686	17.1992	**17.1311**
a-5	30.0486	**30.033**	34.8626	48.455	43.2665	35.8936	35.8569
a-6	15.9805	17.5442	16.907	27.0956	27.1197	14.4448	**14.3796**

In addition, we employ Harvey, Leybourne and Newbold (HLN) to determine whether the difference between the improvement achieved by MEC-RDESN is significant [53]. The null hypothesis *H0* indicates that two time series forecasting models have the same prediction accuracy, and the alternative hypothesis *H1* indicates that the models have different prediction accuracy. We mark "significant at the 0.05 level" ($P < 0.05$) as ▲, and "significant at the 0.01 level" ($P < 0.01$) as ★. The HLN results of all the baseline methods compared to MEC-RDESN are shown in Table 10,

TABLE 9: RMSE of the *TP* data set of *taxi* path.

ER_ID	Average	SARIMA	RNN	LSTM	ESN	RESN	MEC-RDESN
b-1	43.3161	115.0875	22.4273	32.2243	21.7716	21.7719	**21.7716**
b-2	27.0205	57.7211	13.8041	17.7182	14.9182	9.0062	**8.9965**
b-3	13.6783	12.2412	9.7621	12.7523	9.6483	9.0987	**9.0957**
b-4	5.7767	9.9796	2.0535	2.7278	2.6493	1.6503	**1.6503**
b-5	1.8807	1.2065	1.0169	1.255	1.1175	0.9844	**0.9842**
b-6	18.2498	16.8396	14.8408	24.179	14.558	13.352	**13.3062**
b-7	4.393	4.3943	9.0965	13.9049	6.6988	3.9004	**3.8462**
b-8	8.4865	**7.9317**	9.7769	15.8058	11.9541	10.0207	10.0116
b-9	14.5217	14.5234	15.896	23.7599	15.3303	12.2369	**12.2142**
b-10	9.902	8.6138	10.4811	17.16	8.1645	6.4006	**6.3863**
b-11	15.3692	21.0232	15.4715	23.7833	17.8021	12.0885	**11.9828**
b-12	33.753	38.7282	39.319	51.8756	37.6753	25.9626	**25.8444**
b-13	19.9138	22.8103	22.6271	31.103	20.844	19.9219	**19.9132**
b-14	22.7999	23.8852	27.4249	38.8704	25.3176	**22.1647**	22.2746

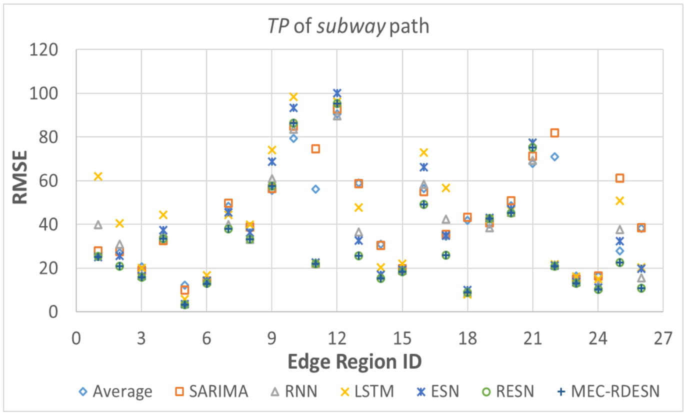


Fig. 17: RMSE of the *TP* data set of the *subway* path

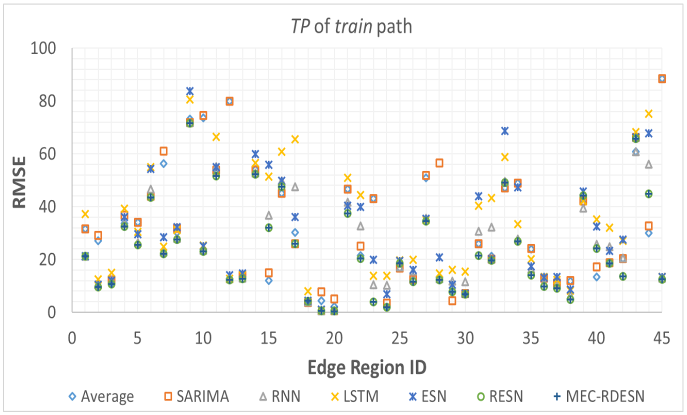


Fig. 18: RMSE of the *TP* data set of the *train* path

which shows that there is significant difference between the forecasts made by the baselines and our method.

TABLE 10: HLN test on *RT* and *TP*

Data set	Average	SARIMA	RNN	LSTM	ESN	RESN
RT	★	★	★	★	★	▲
TP	★	★	★	★	★	★

(4) **Leading time of edge forecasting**

Leading time refers to the amount of time that a user can utilize the forecasted QoS in the current time interval. It is an important indicator in fast moving scenes, as it measures the effective time length of the forecasting. Its formula is as follows:

$$t_L = (R - (t_T + t_F) \times \bar{V}_u)/\bar{V}_u \tag{7}$$

where R is the distance to be reached in the current time interval, t_T is the model training time, t_F is the service forecasting time, and $\bar{V}_u$ is the user's average moving speed. An explanation of the calculation process is shown in Fig. 19.

We explore the forecasting leading time of the seven methods at different base station signal coverage radii and speeds. Here we employ three radii values, i.e., 300m, 400m, and 500m, according to [19]. Taking the *RT* data set as an example, Fig. 20 shows the leading time of the four transportation modes under different signal coverage radii.

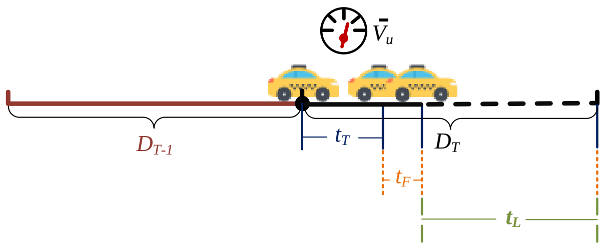


Fig. 19: Calculation process of leading time

Among them, *RESN* and *MEC-RDESN* belong to the *ESN* series of methods, and their results are close. In the view of single subfigures, when the *R* value decreases under the same transportation mode, the leading time of forecasting also decreases, and our method is second only to the trainingless *Average* method. In addition, it can be seen from Fig. 20(a)-20(d) that the faster the speed under the same *R* value, the shorter the leading time of forecasting. Negative numbers appear in Fig. 20(c) and Fig. 20(d), i.e., the results of the *LSTM* method, indicating the incompetence of the method to predict in time. In contrast, *MEC-RDESN* can achieve at least 10s and 5s leading time, with a significant gap ahead of *SARIMA* and *RNN*. Thus, *MEC-RDESN* is suitable for different transportation modes, the advantages of which is more obvious in fast moving scenes.

In summary, the simulation experimental results preliminarily approve that our proposed MEC-RDESN forecasting approach can achieve the purpose of fast prediction while ensuring the accuracy.

5.3 Real-world Experiment

5.3.1 Scene and Data set Description

We conduct experiments on two modes on the campus of Hohai University, Nanjing, China. The experimental environment overview is shown in Table 11. The two students invoked a total of 273 different services. The coverage radius of the micro edge server in the campus is 150m, and the user moving scenario is shown in Fig. 21. We record the students' real-time locations, service invocation time, locations of accessed edge servers, network conditions, service transmission bytes, and response time. The data set can be accessed from [6].

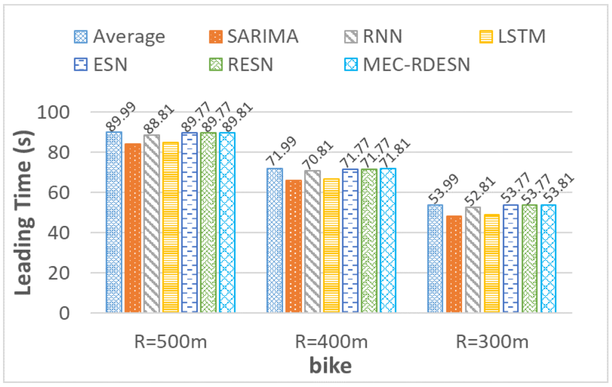
TABLE 11: Experimental environment overview

Student	Mobile device	Mode	$\bar{V}_u$(km/h)	Duration(min)	BSs number
Lee	iPhone	walk	5	10	9
Wang	Miphone	run	10	8	8

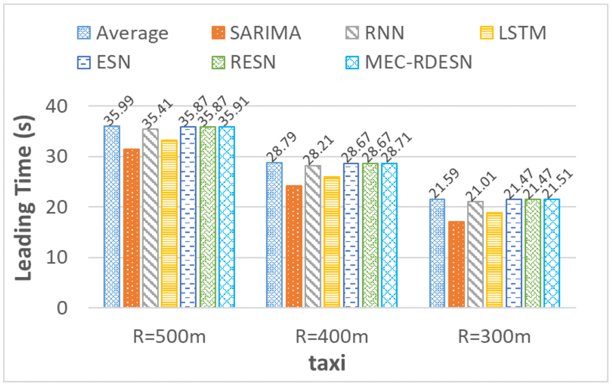
5.3.2 Experimental Process and Results

We use two time slices of data generated by students after departure to activate the model. We pre-train the model, followed by real-time region recognition and fast forecasting. The experimental results are discussed as follows.

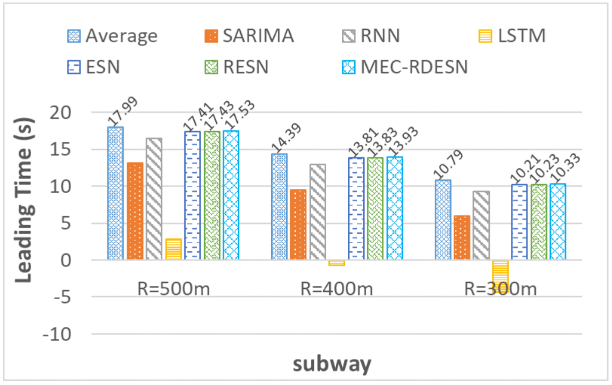
6. https://github.com/hyjin1996/HHU-Dataset



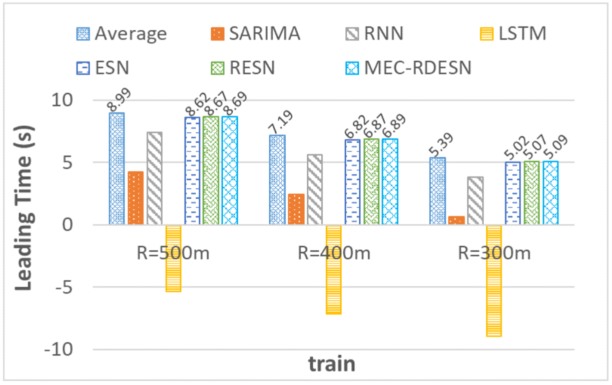
(a)



(b)



(c)



(d)

Fig. 20: Leading time of different signal coverage radii on the paths of (a) bike, (b) taxi, (c) subway, and (d) train.

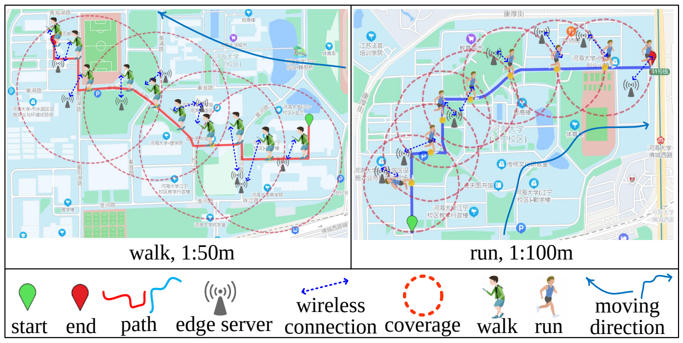


Fig. 21: The real-world experimental scenario on the campus of Hohai University

(1) **The optimal hyper-parameters for model pre-training** Fig. 22 and Fig. 23 show the error values of the two modes with various leaking rates and reserve sizes. We select the hyper-parameters corresponding to the lowest error values. As a result, (0.3, 40) is used as the optimal hyper-parameter for *walk*, and (0.7, 100) is used for *run*.

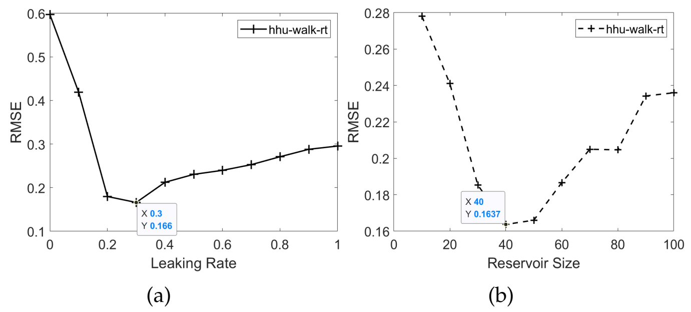


Fig. 22: hhu *walk* rt: (a) leaking rate, (b) reservoir size.

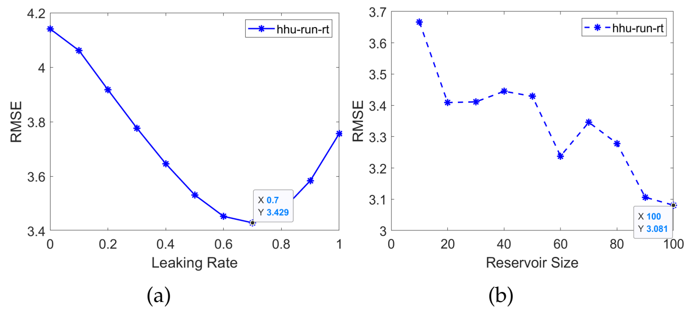


Fig. 23: hhu *run* rt: (a) leaking rate, (b) reservoir size.

Table 12 shows the error values under different activation functions, and we choose *tanh* as the activation function. Here the average time for model pre-training is 0.07s.

TABLE 12: RMSE of Activation Functions on *walk* and *run*

Paths/Fuctions	sigmoid	tanh	relu
walk	0.6445	**0.1637**	0.1689
run	3.7914	**3.0809**	3.1161

(2) **Performance**

Table 13 shows the total training time and average forecasting time of the two modes based on our collected response time (i.e. hht rt) dataset. It can be seen that the total training time of *MEC-RDESN* is the shortest, and its forecasting time is extremely small. Table 14 shows the forecasting errors for the two modes, where our method shows higher forecasting accuracy. Table 15 shows the results of the HLN test. *MEC-RDESN* is significant at the 0.01 level compared to *Average*, and at the 0.05 level compared to the other methods.

TABLE 13: Total training time and average forecasting time of hhu rt data set

Mode	Approach	Total training time (ms)	Average forecasting time (ms)
	Average	\	0.249
	SARIMA	14658.952	179.771
	RNN	118.049	1.7355
walk	LSTM	2430.639	2.283
	ESN	26.086	0.526
	RESN	24.151	0.25475
	MEC-RDESN	**10.069**	**0.3605**
	Average	\	0.331
	SARIMA	17977.021	176.9682
	RNN	145.092	1.4378
run	LSTM	3108.329	0.8016
	ESN	63.824	0.4708
	RESN	60.365	0.2954
	MEC-RDESN	**16.128**	**0.4322**

TABLE 14: RMSE of hhu rt data set: (a) *walk*, (b) *run*.

(a)

ER_ID	1	2	3	4
Average	4.1899	6.1274	6.2469	2.6033
SARIMA	10.4881	6.9776	5.6022	6.6685
RNN	0.9771	3.6612	2.2026	2.393
LSTM	0.884	3.6993	2.4704	2.367
ESN	0.1637	4.1956	1.5875	2.431
RESN	0.187	3.375	0.7517	1.7923
MEC-RDESN	**0.1637**	**3.2157**	**0.7332**	**1.7815**

(b)

ER_ID	1	2	3	4	5
Average	10.7841	3.5237	2.9011	3.9528	2.4833
SARIMA	4.5093	3.3166	1.747	3.8079	2.4495
RNN	3.5984	1.0974	1.695	**2.9819**	2.0492
LSTM	3.1212	0.6885	1.9463	3.1763	1.5752
ESN	3.0809	0.4488	1.7359	3.0383	2.6219
RESN	3.1355	0.3119	1.6925	3.3644	1.5022
MEC-RDESN	**3.0809**	**0.2912**	**1.6778**	3.2867	**1.4977**

(3) **Leading time of edge forecasting**

We apply MEC-RDESN to the trajectories of the *walk* and *run* modes. The forecasting leading time calculated based on equation (7) is shown in Fig. 24. It can be seen that *MEC-RDESN* achieves sufficient leading time in the both modes, which is equal to the *Average* method.

TABLE 15: HLN test on hhu rt

Data set	Average	SARIMA	RNN	LSTM	ESN	RESN
hhu rt	★	▲	▲	▲	▲	▲

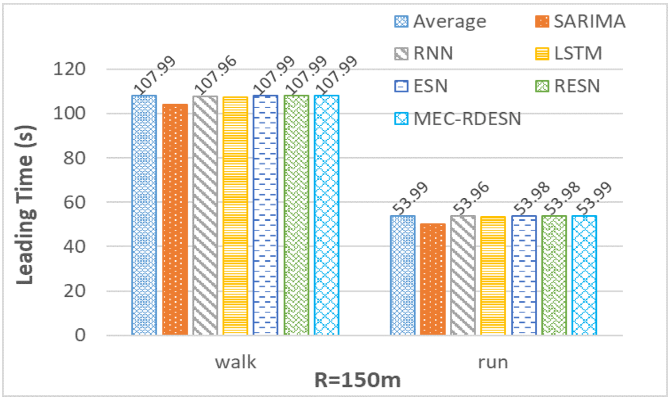


Fig. 24: Leading time of *walk* and *run* in hhu rt data set

In summary, the real-world experiment can preliminarily validate the usability and effectiveness of our proposed MEC-RDESN approach in the real environment.

6 CONCLUSIONS AND FUTURE WORK

Existing QoS forecasting approaches cannot meet the demand of the MEC environment on user-mobility-aware, fast and accurate QoS forecasting. We propose a novel edge QoS forecasting approach named MEC-RDESN, with user-mobility-awareness and high forecasting efficiency and accuracy. MEC-RDESN is based on a dynamic echo state network. In addition, we introduce the techniques of user-centered edge region recognition and model pre-training to achieve the goals of user-mobility-aware, fast and accurate QoS forecasting. In the future, we will focus on the following issues. First, the current user-centered edge region recognition bases on a uniform signal coverage radius. In reality, it needs to adapt to the different signal coverage radii of the surrounding base stations to achieve more accurate regional recognition. Second, we will further optimize other hyper-parameters in ESN to improve its forecasting performance.

ACKNOWLEDGMENTS

This work is funded by the National Natural Science Foundation of China under Grant No.62272145 and No.U21B2016, the 2021 Postgraduate Research and Practice Innovation Program of Jiangsu Province under grant number KYCX21_0549, and the Fundamental Research Funds for the Central Universities under grant number B210203070 and B210202075.

This research was also supported partially by the Australian Government through the Australian Research Council's Discovery Projects funding scheme (project DP220101823). The views expressed herein are those of the authors and are not necessarily those of the Australian Government or Australian Research Council.

References

[1] D. Karastoyanova and F. Leymann, "Service Oriented Architecture–overview of technologies and Standards," *it-Information Technology*, vol. 50, no. 2, pp. 83–85, 2008.
[2] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, "Service-oriented computing: State of the art and research challenges," *Computer*, vol. 40, no. 11, pp. 38–45, 2007.
[3] J. Wu, L. Chen, Z. Zheng, M. R. Lyu, and Z. Wu, "Clustering web services to facilitate service discovery," *Knowledge and information systems*, vol. 38, no. 1, pp. 207–229, 2014.
[4] A. Bouguettaya, M. Singh, M. Huhns, Q. Z. Sheng, H. Dong, Q. Yu, A. G. Neiat, S. Mistry, B. Benatallah, B. Medjahed, *et al.*, "A service computing manifesto: the next 10 years," *Communications of the ACM*, vol. 60, no. 4, pp. 64–72, 2017.
[5] L. Grunske, "Specification patterns for probabilistic quality properties," in *2008 ACM/IEEE 30th International Conference on Software Engineering*, pp. 31–40, IEEE, 2008.
[6] H. Jin, P. Zhang, H. Dong, X. Wei, Y. Zhu, and T. Gu, "Mobility-aware and Privacy-protecting Qos optimization in mobile edge networks," *IEEE Transactions on Mobile Computing, DOI: 10.1109/TMC.2022.3230856*, 2022.
[7] B. Varghese, N. Wang, S. Barbhuiya, P. Kilpatrick, and D. S. Nikolopoulos, "Challenges and opportunities in edge computing," in *2016 IEEE International Conference on Smart Cloud (SmartCloud)*, pp. 20–26, IEEE, 2016.
[8] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—a key technology towards 5g," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.
[9] P. Porambage, J. Okwuibe, M. Liyanage, M. Ylianttila, and T. Taleb, "Survey on multi-access edge computing for internet of things realization," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 2961–2991, 2018.
[10] H. Jin, P. Zhang, H. Dong, Y. Zhu, and A. Bouguettaya, "Privacy-aware forecasting of quality of service in mobile edge computing," *IEEE Transactions on Services Computing, DOI:10.1109/TSC.2021.3137452*, 2021.
[11] E. Ahmed and M. H. Rehmani, "Mobile edge computing: opportunities, solutions, and challenges," *Future Generation Computer Systems*, vol. 70, pp. 59–63, 2017.
[12] S. Deng, Z. Xiang, J. Taheri, M. A. Khoshkholghi, J. Yin, A. Y. Zomaya, and S. Dustdar, "Optimal application deployment in resource constrained distributed edges," *IEEE Transactions on Mobile Computing*, vol. 20, no. 5, pp. 1907–1923, 2020.
[13] 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.
[14] S. Li, J. Wen, and X. Wang, "From reputation perspective: a hybrid matrix factorization for QoS prediction in location-aware mobile service recommendation system," *Mobile Information Systems*, vol. 2019, 2019.
[15] G. White, A. Palade, and S. Clarke, "Forecasting QoS attributes using LSTM networks," in *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2018.
[16] Q. Wang, M. Chen, M. Shang, and X. Luo, "A momentum-incorporated latent factorization of tensors model for temporal-aware QoS missing data prediction," *Neurocomputing*, vol. 367, pp. 299–307, 2019.
[17] G. White, A. Palade, C. Cabrera, and S. Clarke, "Autoencoders for QoS prediction at the edge," in *2019 IEEE International Conference on Pervasive Computing and Communications (PerCom*, pp. 1–9, IEEE, 2019.
[18] E. Macias, A. Suarez, and J. Lloret, "Mobile sensing systems," *Sensors*, vol. 13, no. 12, pp. 17292–17321, 2013.
[19] J. Li, Y. Feng, and Y. Hu, "Load forecasting of 5g base station in urban distribution network," in *2021 IEEE 5th Conference on Energy Internet and Energy System Integration (EI2)*, pp. 1308–1313, IEEE, 2021.
[20] M. Lukoševičius, "A practical guide to applying echo state networks," in *Neural networks: Tricks of the trade*, pp. 659–686, Springer, 2012.
[21] 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.
[22] W. Lo, J. Yin, Y. Li, and Z. Wu, "Efficient web service QoS prediction using local neighborhood matrix factorization," *Engineering Applications of Artificial Intelligence*, vol. 38, pp. 14–23, 2015.
[23] X. Chen, Z. Zheng, Q. Yu, and M. R. Lyu, "Web service recommendation via exploiting location and QoS information," *IEEE Transactions on Parallel and distributed systems*, vol. 25, no. 7, pp. 1913–1924, 2013.
[24] Y. Shen, J. Zhu, X. Wang, L. Cai, X. Yang, and B. Zhou, "Geographic location-based network-aware qos prediction for service composition," in *2013 IEEE 20th International Conference on Web Services*, pp. 66–74, IEEE, 2013.
[25] X. Wang, J. Zhu, Z. Zheng, W. Song, Y. Shen, and M. R. Lyu, "A spatial-temporal QoS prediction approach for time-aware web service recommendation," *ACM Transactions on the Web (TWEB)*, vol. 10, no. 1, pp. 1–25, 2016.
[26] S. Ding, Y. Li, D. Wu, Y. Zhang, and S. Yang, "Time-aware cloud service recommendation using similarity-enhanced collaborative filtering and arima model," *Decision Support Systems*, vol. 107, pp. 103–115, 2018.
[27] L. Qi, R. Wang, C. Hu, S. Li, Q. He, and X. Xu, "Time-aware distributed service recommendation with privacy-preservation," *Information Sciences*, vol. 480, pp. 354–364, 2019.
[28] Z. Ye, S. Mistry, A. Bouguettaya, and H. Dong, "Long-term QoS-aware cloud service composition using multivariate time series analysis," *IEEE Transactions on Services Computing*, vol. 9, no. 3, pp. 382–393, 2014.
[29] 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*, vol. 14, no. 6, pp. 1809–1823, 2019.
[30] G. Zou, T. Li, M. Jiang, S. Hu, C. Cao, B. Zhang, Y. Gan, and Y. Chen, "Deeptsqp: Temporal-aware service QoS prediction via deep neural network and feature integration," *Knowledge-Based Systems*, vol. 241, p. 108062, 2022.
[31] Z. Liu, Q. Z. Sheng, W. E. Zhang, D. Chu, and X. Xu, "Context-aware multi-QoS prediction for services in mobile edge computing," in *2019 IEEE international conference on services computing (SCC)*, pp. 72–79, IEEE, 2019.
[32] Z.-Z. Liu, Q. Z. Sheng, X. Xu, D. Chu, and W. E. Zhang, "Context-aware and adaptive QoS prediction for mobile edge computing services," *IEEE Transactions on Services Computing*, vol. 15, no. 1, pp. 400–413, 2022.
[33] Y. Yin, L. Chen, Y. Xu, J. Wan, H. Zhang, and Z. Mai, "QoS prediction for service recommendation with deep feature learning in edge computing environment," *Mobile networks and applications*, vol. 25, no. 2, pp. 391–401, 2020.
[34] G. White and S. Clarke, "Short-term Qos forecasting at the edge for reliable service applications," *IEEE Transactions on Services Computing*, vol. 15, no. 2, pp. 1089–1102, 2022.
[35] Y. Zhang, P. Zhang, Y. Luo, and L. Ji, "Towards efficient, credible and privacy-preserving service QoS prediction in unreliable mobile edge environments," in *2020 International Symposium on Reliable Distributed Systems (SRDS)*, pp. 309–318, IEEE, 2020.
[36] Y. Zhang, J. Pan, L. Qi, and Q. He, "Privacy-preserving quality prediction for edge-based iot services," *Future Generation Computer Systems*, vol. 114, pp. 336–348, 2021.
[37] P. Abichandani, W. Fligor, and E. Fromm, "A cloud enabled virtual reality based pedagogical ecosystem for wind energy education," in *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*, pp. 1–7, IEEE, 2014.
[38] E. Ahmed, I. Yaqoob, A. Gani, M. Imran, and M. Guizani, "Internet-of-things-based smart environments: state of the art, taxonomy, and open research challenges," *IEEE Wireless Communications*, vol. 23, no. 5, pp. 10–16, 2016.
[39] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE internet of things journal*, vol. 3, no. 5, pp. 637–646, 2016.
[40] J. A. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava, "Participatory sensing," 2006.
[41] S. A. Hoseini-Tabatabaei, A. Gluhak, and R. Tafazolli, "A survey on smartphone-based systems for opportunistic user context recognition," *ACM Computing Surveys (CSUR)*, vol. 45, no. 3, pp. 1–51, 2013.
[42] H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks-with an erratum note," *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, vol. 148, no. 34, p. 13, 2001.

[43] H. Jaeger, "Adaptive nonlinear system identification with echo state networks," *Advances in neural information processing systems*, vol. 15, 2002.
[44] H. Jaeger and H. Haas, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication," *science*, vol. 304, no. 5667, pp. 78–80, 2004.
[45] S. I. Han and J. M. Lee, "Fuzzy echo state neural networks and funnel dynamic surface control for prescribed performance of a nonlinear dynamic system," *IEEE Transactions on Industrial Electronics*, vol. 61, no. 2, pp. 1099–1112, 2013.
[46] X. Sun, T. Li, Q. Li, Y. Huang, and Y. Li, "Deep belief echo-state network and its application to time series prediction," *Knowledge-Based Systems*, vol. 130, pp. 17–29, 2017.
[47] S. Wang, Y. Zhao, J. Xu, J. Yuan, and C.-H. Hsu, "Edge server placement in mobile edge computing," *Journal of Parallel and Distributed Computing*, vol. 127, pp. 160–168, 2019.
[48] Y. Zheng, Q. Li, Y. Chen, X. Xie, and W.-Y. Ma, "Understanding mobility based on gps data," in *Proceedings of the 10th international conference on Ubiquitous computing*, pp. 312–321, 2008.
[49] Z. Zheng, Y. Zhang, and M. R. Lyu, "Investigating Qos of real-world web services," *IEEE transactions on Services Computing*, vol. 7, no. 1, pp. 32–39, 2012.
[50] L.-M. Liu, G. B. Hudak, G. E. Box, M. E. Muller, and G. C. Tiao, *Forecasting and time series analysis using the SCA statistical system*, vol. 1. Scientific Computing Associates DeKalb, IL, 1992.
[51] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.
[52] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[53] D. Harvey, S. Leybourne, and P. Newbold, "Testing the equality of prediction mean squared errors," *International Journal of forecasting*, vol. 13, no. 2, pp. 281–291, 1997.

Huiying Jin is a PHD candidate with the College of Computer and Information, Hohai University, Nanjing, China. She received her bachelor degree in Software Engineering from Yangzhou University, Yangzhou, China in 2017. Her current research interests include services computing and data mining. She has won National scholarship for doctoral students. She has published in international journals such as *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Services Computing* and *Information and Software Technology*.

Pengcheng Zhang received the Ph.D. degree in computer science from Southeast University in 2010. He is currently a full professor in the College of Computer and Information, Hohai University, Nanjing, China. His research interests include software engineering, service computing, and data science. He co-authored more than 70 peer-reviewed conference and journal papers, and has served as technical program committee member on various international conferences. He is a member of the IEEE.

Hai Dong is a senior lecturer at School of Computing Technologies in RMIT University, Melbourne, Australia. He received a PhD from Curtin University, Australia and a Bachelor degree from Northeastern University, China. His research interests include Service-Oriented Computing, Distributed Systems, Cyber Security, Software Testing, Machine Learning and Data Science. He has published a monograph and over 110 research articles in international journals and conferences. He is a senior member of the IEEE.

Yuelong Zhu is currently a Professor and a Ph.D. Supervisor with the College of Computer and Information, Hohai University, Nanjing, China. He is the Deputy Chairman of the Water Resources Informatization Special Committee of the China Hydraulic Engineering Society, and the Vice Chairman of the Jiangsu Water Resources Protection Association. His main research interests include intelligent information processing and data mining, and water conservancy informatization. He was awarded the National Second Prize for Scientific and Technological Progress, the First Prize for Excellent Achievements of the Ministry of Education, and the Dayu Water Science and Technology First Prize.