Swift and Accurate Mobility-Aware QoS Forecasting for Mobile Edge Environments

Huiying Jin, Pengcheng Zhang, *Member, IEEE*, Hai Dong, *Senior Member, IEEE*, Athman Bouguettaya, *Fellow, IEEE*, and Albert Y. Zomaya, *Fellow, IEEE*

Abstract—We propose an innovative approach named MEC-RDESN /mekˌrːdɪˈsaɪn/ (MEC QoS forecasting based on Region recognition and Dynamic Echo State Network) enabling mobility-aware and swift QoS forecasting in the mobile edge computing environment. MEC-RDESN offers efficient QoS forecasting while maintaining high accuracy. We can identify the edge region to which a user belongs in real time while moving by leveraging mobile sensing technology. We employ a *dynamic echo state network* characterized by multi-service adaptability to retain information about services invoked by users to ensure real-time training and forecasting accuracy. Our approach is validated through a series of experiments using both public and collected datasets. The experiments demonstrate that MEC-RDESN achieves the goal of fast forecasting while ensuring its forecasting accuracy in diverse application scenarios.

Index Terms—Mobility-aware, User-centered edge region recognition, dynamic ESN, Swift 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 computing paradigm that uses services as the basic elements to develop applications [2]. The main implementation technology of SOC is service [3]. Services are widely used in e-commerce, big data analysis and other application scenarios [4]. Quality of Service (QoS, also called non-functional attributes) is typically used as a discriminant between services with similar functions [5], [6]. QoS includes *response time*, *throughput*, and *security* [7].

Mobile Edge Computing (MEC) specifically focuses on bringing computing capabilities to the edge of mobile networks. It is generally deployed on the Radio Access Network (RAN), which is geographically adjacent to mobile users, to provide computing power right where it is needed [8]. Once MEC nodes receive service requests from nearby users, these nodes can respond with significant reductions in network transmission delay [8]. In this regard, MEC can provide users with a superior service experience [8]. 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 wearables, etc.) generate massive streaming data, making real-time decisions impractical when analytics are performed on remote clouds. Edge nodes that are closer to end users may be utilized to reduce network latency to complement the computation performed on the cloud [10]. However, the low-latency feature of MEC is highly constraining for edge services quality prediction [11]. In addition, the real-time movement of users and the variability of movement speed also pose higher requirements for edge QoS forecasting. For example, the *Washington Post* reported on Jun 13, 2023 that Tesla's Autopilot was involved in 736 crashes since 2019, including 17 fatalities. How to provide real-time, reliable autopilot services for users has always been a concern of National Highway Traffic Safety Administration [12]. Therefore, it is critical to achieve swift and accurate QoS forecasting while providing intelligent services. There are two major challenges in this regard: 1) *The continuous movement of users and rapid switching among different edge servers* pose challenges in predicting and adapting to the dynamic network environment; 2) *The need to forecast and maintain acceptable QoS in such dynamic network environments* requires rapid and accurate prediction capabilities. The following scenario demonstrates these two challenges.

Fig. 1 shows the scenario of an urban road and its surrounding edge server distribution. Assume a taxi is going at a relatively fixed speed from southwest to northeast between 9:00-9:10am. Also assume that the taxi passenger *Allen* is watching a *Twitch* game live video during this journey. The video quality needs therefore to be continuously predicted. 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 may go at higher speeds requiring *Allen* to quickly switch between different edge servers. Hence, it is paramount to quickly and accurately predict

- *H. Jin is with the College of Computer, Nanjing University of Posts and Telecommunications, Nanjing, China E-mail: hyjin@njupt.edu.cn*
- *P. Zhang (Corresponding author) is with the Key Laboratory of Water Big Data Technology of Ministry of Water Resources* and *the College of Computer and Software, Hohai University, Nanjing, China E-mail: pchzhang@hhu.edu.cn*
- *H. Dong is with the School of Computing Technologies* and *the Centre for Cyber Security Research and Innovation, RMIT University, Melbourne, VIC 3000, Australia E-mail: hai.dong@rmit.edu.au*
- *Athman Bougettaya and Albert Y. Zomaya are with the School of Computer Science, University of Sydney, Sydney, NSW 2006, Australia E-mail: {athman.bouguettaya, albert.zomaya}@sydney.edu.au*

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

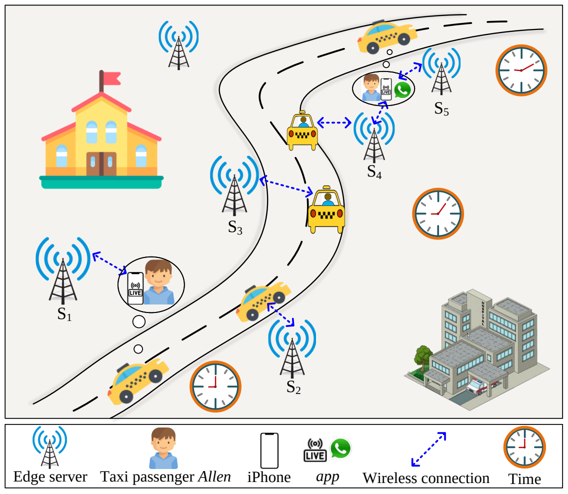


Fig. 1: A mobility-aware edge service invocation scenario

video quality. Assume that between 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 forecasting 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 prediction. In addition, *Allen* may decide to make a voice call using *WhatsApp* from 9:05am to 9:10am. Similarly, the call quality also needs to be continuously predicted so that *WhatsApp* can compress audio to ensure the call quality. Thus, both of the quality of *Twitch* and *WhatsApp* needs to be predicted in the second half of the journey.

We summarize the challenges faced by the edge QoS forecasting are 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 [13]. Since users may continuously access different edge servers during the movement process, each edge server serves a different number of users with limited resources [14]. However, the location-aware schemes in existing QoS forecasting approaches mainly 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 [11], [15], [16]. This may lead to *low forecasting accuracy* and *unsatisfied forecasting lead time*, i.e., the forecasting result cannot be delivered before a user arrives in an area covered by a different edge server. Therefore, it is of great importance to monitor user movement in real-time and achieve a satisfactory QoS forecasting lead time.

ii). Realizing swift 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, thus creating a high-performance and low-latency service environment. Similarly, realizing quick QoS forecasting is a necessary condition for improving user service experience. In this regard, most traditional time series QoS forecasting approaches [17], [18], [19] 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, swift 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 swift QoS forecasting approach in the mobile edge environment, abbreviated as MEC-RDESN (MEC QoS forecasting based on Region recognition and Dynamic Echo State Network). A user-centered edge region is continuously identified during a user's movement to obtain valid historical time series data to ensure forecasting accuracy. We adopt an improved Echo State Network to meet the requirement of fast forecasting with significantly reduced training time. The contributions of this paper include the following two aspects:

- We propose a mobility-aware user-centered edge region recognition scheme. Mobile sensing technology is a low-cost but efficient way to collect environmental data [20]. The radius of the current 5G urban base station signal coverage is around 300-500m [21]. We introduce the concept of mobile recognition in user-centered edge regions based on the sensing technology and the signal coverage statistics of edge servers. Let us assume that the current location of a user is covered by one or many edge servers 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 describe an efficient 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 forecasting during user movement. 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. In comparison to ESN, the improved model is able to further optimize training costs, and stabilize the connection weights between neurons to improve forecasting accuracy.

We conduct a series of experiments based on public and 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 swift forecasting while ensuring forecasting accuracy in diverse application scenarios, such as cycling, driving, taking trains, etc.

The structure of the paper is organized as follows. Section 2 surveys relevant research about "mobility-aware" and state-of-the-art QoS forecasting methods. 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 Mobility-Awareness

A. T. Campbell is recognized as a pioneer in mobility-awareness. He introduced the "human-centered" awareness mode in 2006 [22]. In 2011, IBM Research [23] studied the "mobile crowd sensing" application. In 2012, Y. Liu [24] proposed the concept of "group-aware computing", marking a significant milestone in this area.

In recent years, many scholars have conducted research on mobility-awareness in edge environments. Peng et al. [25] devised a method to assess mobility-related fitness values between unallocated edge users and available edge servers. They also designed a mobility model to perform user allocation according to the fitness values. Hoang et al. [26] investigated a mobility-aware computational offloading method for in-vehicle wireless networks. This method uses an unbounded simulation area migration model to simulate the movement of intelligent connected cars, thereby constructing a mobility model. Ma et al. [27] used data mining and machine learning technology to estimate the probability of a user moving to a certain location based on the user's historical movement trajectory, thereby enhancing service delivery. Liu et al. [28] proposed a mobility-aware dynamic edge service migration scheme. This scheme calculates the probability of users crossing the boundaries of cellular networks based on network shapes, making the mobility behaviour of users predictable.

However, the above mobility-aware solutions predominantly focus on monitoring changes in users' locations rather than continuously tracking users' real-time movement status.

2.2 QoS Forecasting

Traditional QoS forecasting. Traditional QoS forecasting can mainly be achieved via similarity-based, model-based, location-based, and time-based approaches. Zheng et al. [29] proposed a web service QoS forecasting method based on user similarity and service similarity. Ding et al. [5] considered the hidden environmental preference information to build up a joint QoS forecasting method based on the deep fusion of features. Liu et al. [30] employed Graph Neural Networks (GNNs) for QoS forecasting. GNNs model the impact between users/services and learn feature vectors effectively. Zou et al. [31] proposed a novel framework for data-protected QoS prediction. It ensures user data protection and the efficacy of predicting missing QoS values. Chen et al. [32] employed the location information and QoS values to cluster users and services. They made personalized service recommendations based on the clustering results. Shen et al. [33] proposed a QoS forecasting method based on the geographic location information of candidate services to improve forecasting accuracy. These traditional location-based QoS forecasting methods make service recommendations based on the location of services or users. They ignore user mobility issues.

Many scholars consider the time factor during forecasting to further analyze the dynamics of QoS data and pursue more accurate predictions. Existing time series QoS forecasting approaches can be divided into numerical and model-based approaches. Numerical forecasting mostly targets predicting null values by mining the relationship between historical values. Wang et al. [34] considered the influence of network states and time changes on service performance. They presented a spatio-temporal QoS forecasting method, which can improve the forecasting accuracy by more than 10%. Nevertheless, the forecasting leverages other available QoS values in the current time slot and cannot be directly applied to forecast future temporal QoS values. Ye et al. [35] used QoS historical data and short-term advertisements to predict the long-term QoS behavior of service providers. However, QoS is inherently dynamic. The QoS history and short-term advertisement cannot capture this dynamism.

Model-based forecasting generally builds a forecasting model and trains the model based on time series QoS data. Zhang et al. [36] proposed an online long-term QoS forecasting method based on radial basis function (RBF) to solve the problems of correlation of multiple attributes, inaccurate long-term forecasting 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. Zou et al. [37] devised a time-aware QoS forecasting method based on deep learning and feature integration. It integrates the binarization feature and the similarity feature. It learns and mines temporal features between users and services based on gated recurrent units (GRU) to realize better service QoS forecasting. This method does not consider the influence of geographic locations of users and services on the forecasting result.

Edge QoS Forecasting. Existing edge QoS forecasting approaches can be grouped into environment-sensitive and model-based categories. Wang et al. [15] proposed a collaborative filtering-based service recommendation method. It selects Top-k similar neighbors for forecasting based on the similarity of users or edge servers. However, the forecasting accuracy of this method is affected by the data density and distribution of edge servers. Li et al. [16] designed a trusted location-aware QoS forecasting method. They integrate location clustering information and user reputation information into hybrid MF models to forecast unknown QoS values. They only consider user information as an important factor in forecasting unknown values. Liu et al. [38] proposed two context-aware MEC service QoS forecasting schemes by combining user-related and service-related contextual factors and various MEC service scheduling scenarios. At the same time, they developed adaptive QoS forecasting strategies to forecast 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 forecasting.

White et al. [39] 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 forecasting. Zhang et al. [40] considered the privacy and reliability issues in the MEC environment. They developed a trusted privacy-preserving QoS forecasting 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. Our previous work [11] focuses on achieving real-time and accurate forecasting when users' geographic area changes. We did not consider real-time user movement and the impact of moving speed on forecasting 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 swift forecasting. There is currently no swift 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. 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. They 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 [41] 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, higher requirements are put forward for QoS forecasting in the edge environment to ensure the service satisfaction of edge users.

3.2 Mobile Sensing Technology

The concept of mobile sensing technology was first proposed by Professor Burke of University of California in 2006 [42]. He 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, mobile sensing technology aims to use sensors on smart devices to collect and process data in real time to provide advanced application services, as smartphones and other types of mobile devices have become mainstream computing and communication methods.

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 [43].

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 [44]. The core principle of ESN is employing a large-scale random sparse network as an information processing medium. ESN maps the input signal from a low-dimensional input space to a high-dimensional state space. It then utilises a linear regression method to train the partial connection weights of the network in the high-dimensional state space [45].

ESN comprises an input layer, a hidden layer and an output layer. Its structure is shown in Fig. 2. The unique features of ESN lie in its utilization of randomly sparsely connected neurons within the hidden layer, coupled with the feature that the connection matrix remains unaltered once generated. At the same time, the generation process is independent of the training process, which greatly simplifies the training process [45]. ESN achieves a key breakthrough in the problem that the traditional recurrent neural network training more likely falls into local minimum values with low convergence speed.

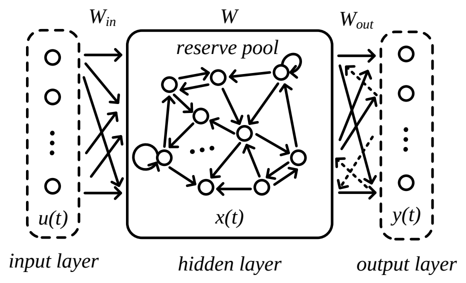


Fig. 2: The structure of ESN

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 swift and mobility-aware QoS forecasting approach (MEC-RDESN) in the mobile edge environment.

MEC-RDESN works towards the goals of mobility-aware, fast and accurate edge QoS forecasting. The system workflow is shown in Fig. 3. It is mainly divided into four steps:

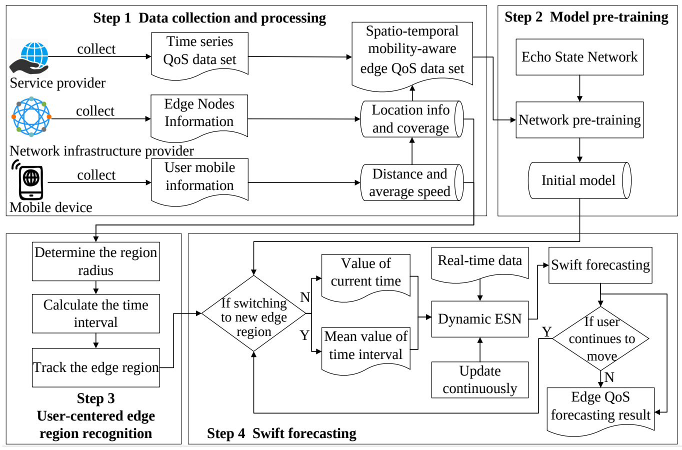

Step 1 Data collection and processing
Service provider
collect
Time series QoS data set
Spatio-temporal mobility-aware edge QoS data set
Network infrastructure provider
collect
Edge Nodes Information
Location info and coverage
Mobile device
collect
User mobile information
Distance and average speed
Step 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 switching to new edge region
N
Y
Value of current time
Mean value of time interval
Real-time data
Dynamic ESN
Update continuously
Swift forecasting
If user continues to move
Y
N
Edge QoS forecasting result
Step 4 Swift forecasting

Fig. 3: MEC-RDESN overview

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 [46], 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. 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 draw the circular edge region with a radius of 500m by taking the user's current location as the center. 500m is the maximum signal coverage radius of an urban 5G base station [21]. 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.

The user is switched to a new edge region whenever the user's actual moving distance reaches 500m from the center of the current edge region. Fig. 4 shows all user-centered edge regions in *Allen*'s scenario.

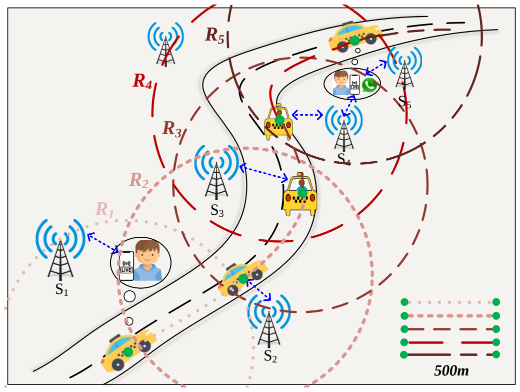


Fig. 4: User-centered edge regions in *Allen*'s scenario

4). Swift forecasting. The user employs the pre-trained model in step 2) in Section 4.1 as the initial QoS forecasting model. The input of the ESN is determined by whether the user switches to a new edge region with the real-time tracking of the user-centered 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). Once 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 forecasting 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. The QoS data 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 $[UM\_EN\_QoS]_{info} = \{(D_{t_k}, \bar{V_u}), (L_{t_k}, S), Q_{t_k}\}$.

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, 9:10 am), 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,9:10 am,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.

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. Its main training process is concentrated between the reserve pool and the output unit due to the structural characteristics of the ESN. In addition, 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 forecasting QoS of the *Twitch* game live video.

The input and output of ESN are both time-series data when it is used for time-series forecasting. 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:

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

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

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, which will remain unchanged after the 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 forecasting. In other words, MEC-RDESN stores W_{in} for s_{old} (i.e., old services) and W. It 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 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.

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) linear regression is used to minimize the training error to obtain W_{out} after collecting the state variables. The linear readout layer is defined as:

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

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 $X = \left[x_{(1)} \dots x_{(t)}\right]$ is the pool state, $Y_{target} = \left[y_{(1)} \dots y_{(t)}\right]$ 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, suppose the whole area where the user is located is fully covered by the signal coverage of edge servers. In that case, 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 [21]) of urban base stations as SC. We 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. The edge region recognition mode provides effective time series historical

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

data for QoS forecasting. The edge region follows the user's movement in real time to further ensure the freshness of historical data.

4.5 Swift forecasting

The swift 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. We judge whether the user has entered a new user-centered edge region according to the user-centered edge region recognition solution depicted in Section 4.4. If not, the forecasting 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 prediction process is calculated in terms of equation (1)$\sim$(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 forecasting 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. It only needs to generate random connection weights for new services. This improvement can assist saving training and forecasting 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.

The dynamic ESN is updated based on the historical QoS data generated in the last time interval whenever a user enters a new edge region during the forecasting process (see Algorithm 1 and equation (5)$\sim$(6)). It then produces the QoS forecasting results. The training and forecasting 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).

Algorithm 1 Mobility-aware swift 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, $\{Q_{t_1}, Q_{t_2}, \dots, Q_{t_k}\}$ is the real-time data generated by $u$ accesses edge servers 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 $\{Q_{t_1}, Q_{t_2}, \dots\}$;
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 $D_{t_k} < D_{T_u - n} + R$ $(n = 1, 2 \dots)$, i.e., $u.location \in$ Edge region $A_n$ then
8:          Perform forecasting based on the current moment value $Q_{t_k}$ to obtain the forecasting result $q'$;
9:          $q' \rightarrow B$;
10:     else
11:         $T_{u-n} + +$, $A_n + +$;
12:         if $s \cap s_{old} \neq \emptyset$ then
13:             Assign new connection weights to $s_{new}$ and use all current connection weights;
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 forecasting to obtain the result $q'$;
19:         $q' \rightarrow B$;
20:     end if
21:     return $B$
22: end for
```

5 EVALUATION

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 [47], [48]. 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

2. https://www.microsoft.com/en-us/download/details.aspx?id=52367
3. http://wsdream.github.io/dataset/wsdream_dataset2.html

distribution service [4] to obtain the geographical locations of base stations.

5.2.2 Experimental Data Preprocessing

In the trajectory data set, there are some 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. We 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. Finally, we set the radius of the user-centered edge regions to R (R=500m). Fig. 5 shows the distribution of the four paths and their surrounding edge servers.

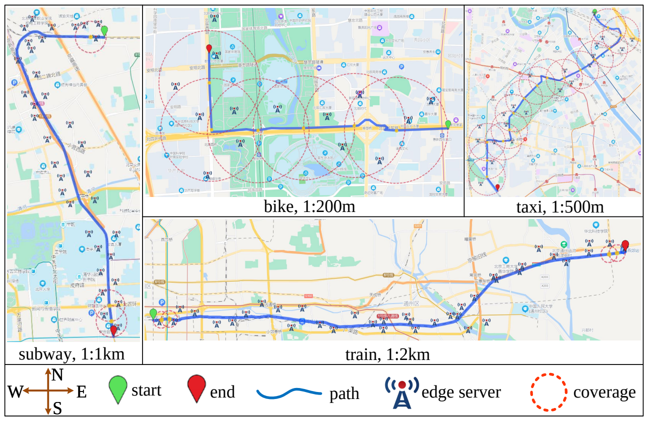


Fig. 5: Paths with different transportation modes and edge server distribution

We determine the time interval T_u of each user according to their average speed. 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 (T) of the QoS data set depicted in Section 5.2.1. Therefore, we obtain the number of time slices contained in each time interval of the four paths (i.e., $T_{bike} = 10T, T_{taxi} = 4T, T_{subway} = 2T, T_{train} = T$). We conduct experiments based on the RT data set.

5.2.3 Experimental Procedure

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

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 RMSE of different time slices (i.e., 3T-6T, where the last time slice of each is the forecasting 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.

4. https://www.opengps.cn/Data/Cell/Region.aspx

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: (1) *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. (2) *SARIMA*: An extension of AutoRegressive Integrated Moving Average (ARIMA), which is used to model periodic time series data and predict future values [49]. (3) *RNN*: A recurrent neural network that takes sequence data as input for forecasting, where all nodes are connected in a chain [50]. (4) *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 [51]. (5) *ESN*: A forecasting model based on the original echo-state network [44]. (6) *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 lead 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 forecasting. Table 1 shows the forecasting errors under different time slices. The time slice with the lowest error is used to activate the model.

TABLE 1: RMSE of different time slices

Path/T	3	4	5	6
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

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

For each user, the training of the initial model utilizes data generated specifically by that user after their departure. The pre-training terminates when optimal hyper-parameters are obtained. Hence, users have distinct initial models. By observing experimental results, we identify a relatively uniform hyper-parameter (i.e., the activation function *tanh*). This decision is depicted as follows:

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 parameter that determines the duration of short-term memory in ESN [52]. Fig. 6 shows the forecasting error values of the four different paths depicted in Section 5.2.2 at different leaking rates. It can be seen that all the forecasting errors show a U-shaped curve with the increase of the leaking rate. We take the leaking rate value of each path when the error is the lowest, e.g., 0.5 in the *bike* path.

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

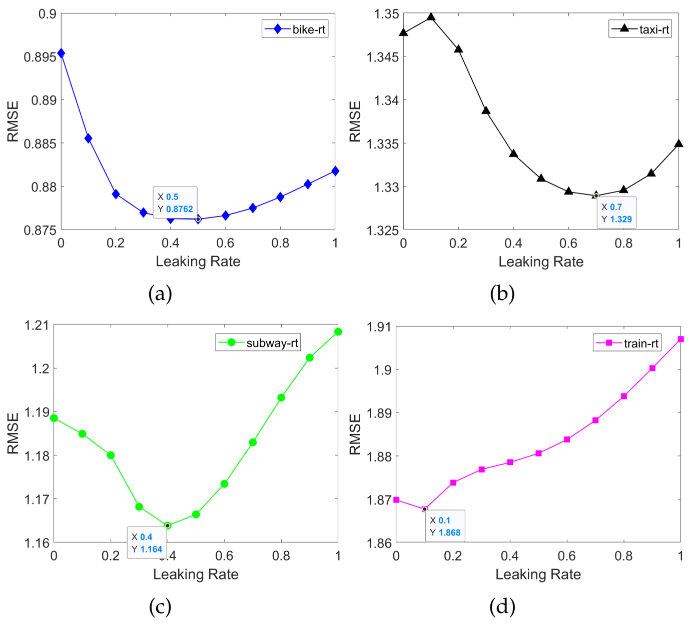


Fig. 6: Forecasting errors of four paths with increasing leaking rate: (a) bike, (b) taxi, (c) subway, (d) train.

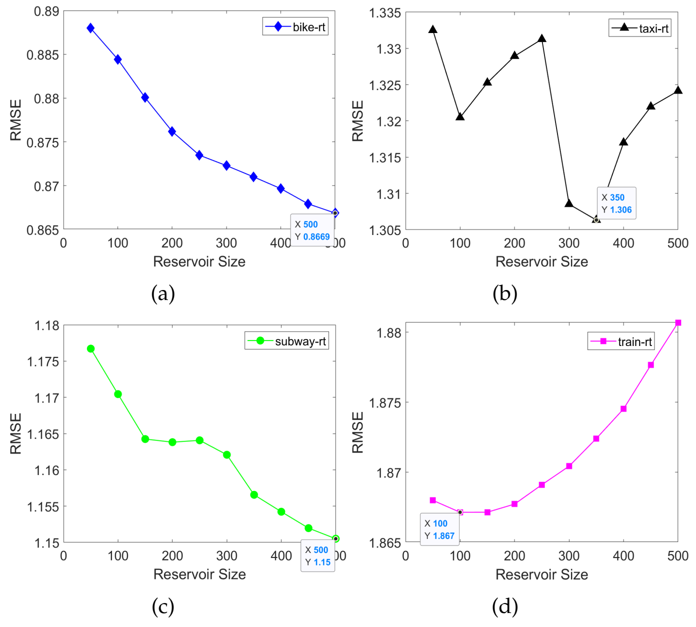


Fig. 7: Forecasting errors of the four paths with increasing reservoir size: (a) bike, (b) taxi, (c) subway, (d) train.

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 forecasting accuracy. However, if N is too large, overfitting will easily occur. Fig. 7 shows the changes in the forecasting error values of the four different paths with the reservoir size increased from 50 to 500 in the step of 50. It can be seen from Fig. 7(a) and Fig. 7(c) that the forecasting error is decreasing with the increase of the reservoir size. Thus, we take 500 as the reservoir size of the *bike* and *subway*. For *taxi* and *train*, overfitting occurred. Therefore, we take 350 and 100 respectively as the parameter of the model in their cases.

Activation Function. Table 2 shows the error values of the four paths under different activation functions. It can be seen that *tanh* achieves the best forecasting performance in the *taxi* and *subway* paths, with little difference in the *bike* and *train* paths. We choose *tanh* as the uniform model activation function based on the above analysis. The average time for model pre-training is 1.59s, indicating that a user only needs less than 2s after departure to obtain a pre-trained model.

TABLE 2: RMSE of Activation Functions

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

(3) MEC-RDESN Performance

The user performs a prediction 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. We perform periodic training and predict the QoS value in the next time interval based on the number of time slices in the time interval of each path introduced in Section 5.2.2. The forecasting performance in the movement process is measured in terms of training time, forecasting time and forecasting accuracy.

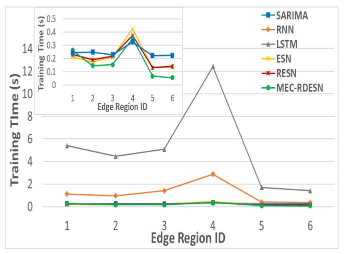
Training Time. Fig. 8 shows the training time of the four paths in each edge region during the movement process. The *Average* method does not require training. Since the number of services invoked by a user changes dynamically, the training time fluctuates with the number of services. Among all the models, the training time of *SARIMA* has the least fluctuations. 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, where 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 3 shows the total training time of the six methods 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.

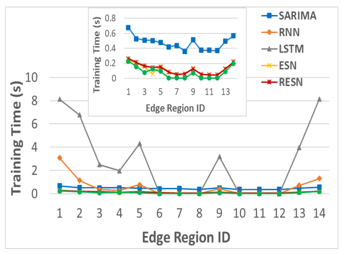
TABLE 3: Total training time of four paths/(s)

Method	bike	taxi	subway	train
SARIMA	1.493	6.622	17.986	39.95
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**

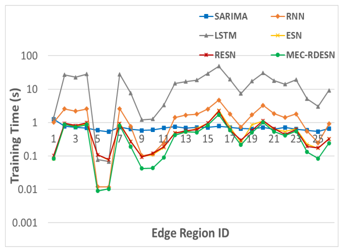
Forecasting Time. The forecasting time is very short compared to the training time. Table 4 shows the average forecasting time of each method on the RT data set. It



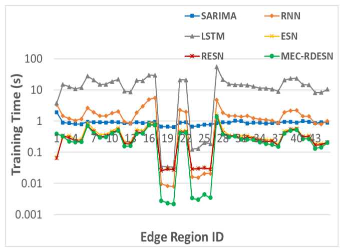
(a)



(b)



(c)



(d)

Fig. 8: Training Time of the four paths: (a) bike, (b) taxi, (c) subway, (d) train.

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 swift forecasting.

TABLE 4: Average forecasting time/(ms)

Average	SARIMA	RNN	LSTM	ESN	RESN	MEC-RDESN
0.15	24.22	2.51	4.58	5.89	5.64	5.63

Forecasting Accuracy. An updated forecasting is performed each time when a user enters a new edge region. Table 5 and Table 6, Fig. 9(a) and Fig. 9(b) show the forecasting errors of the four transportation modes in each edge region[5]. The most accurate forecasting results in Table 5 and Table 6 are marked in bold, and the lowest point of each region in Fig. 9(a) and Fig. 9(b) represents the most accurate forecasting.

TABLE 5: RMSE of *bike* path

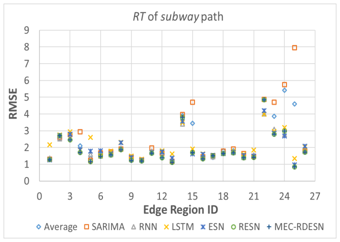
ER_ID	Average	SARIMA	RNN	LSTM	ESN	RESN	MEC-RDESN
a-1	0.8753	1.2351	0.9537	1.1451	0.8669	0.8699	**0.8669**
a-2	0.9343	1.265	1.0162	1.3066	0.9719	0.9305	**0.9276**
a-3	0.8209	0.9063	0.8483	1.1927	0.7853	0.7768	**0.7758**
a-4	0.6796	0.918	0.7206	1.1788	0.693	0.6624	**0.6617**
a-5	0.899	1.1344	0.9921	1.4238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

Further analysis of the experimental results shows that MEC-RDESN achieves the most accurate forecasting results on 80% of the *RT* data set. The *RNN*-related approaches perform well in only a few edge regions (e.g., edge regions 3 and 10 in Table 6), because the training data in these regions has faster convergence rates. Tt can be seen that

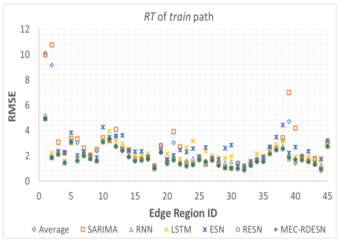
5. Note: Since the paths of *subway* and *train* modes contain too many time intervals, their results are shown in figures instead of tables.

TABLE 6: RMSE 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**



(a)



(b)

Fig. 9: RMSE of two paths: (a) subway, (b) train.

obtaining valid regional historical data based on edge region recognition plays an important role in improving forecasting accuracy, by comparing the results between *RESN* and *ESN*. Since *MEC-RDESN* has a more stable weight connection than *RESN*, it has better forecasting performance.

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 the two time series forecasting models have the same forecasting accuracy. The alternative hypothesis *H1* indicates that the models have different forecasting 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 7. It shows that there is a significant difference between the forecasts made by the baselines and our method.

TABLE 7: HLN results compared to MEC-RDESN

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

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

Lead time refers to the amount of time that a user can utilize the predicted QoS in the current time interval. It is an important indicator in swift 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. 10.

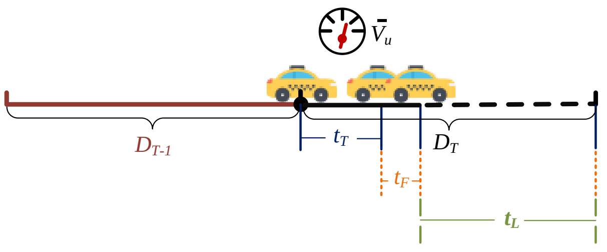


Fig. 10: Calculation process of lead time

We explore the forecasting lead 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 [21]. Fig. 11 shows the lead time of the four transportation modes under different signal coverage radii. 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 lead time of forecasting also decreases. Our method is second only to the trainingless *Average* method. In addition, it can be seen from Fig. 11(a)-11(d) that the faster the speed under the same R value, the shorter the lead time of forecasting. Negative numbers appear in Fig. 11(c) and Fig. 11(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 lead 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 swift moving scenes.

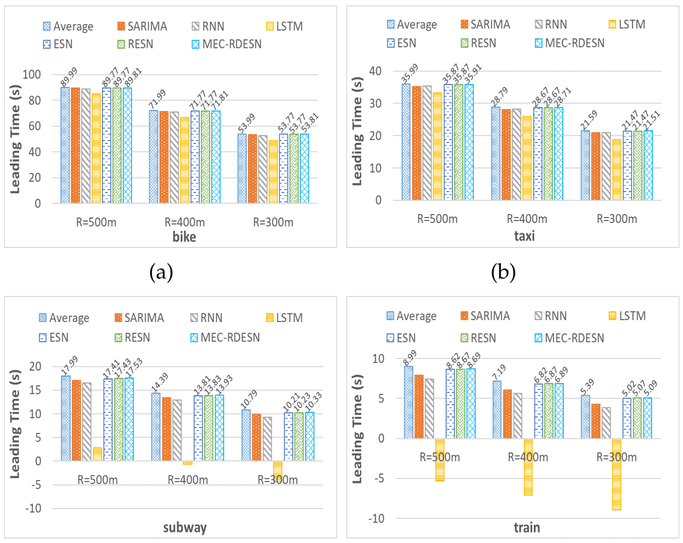


Fig. 11: Lead time of different signal coverage radii on the paths of (a) bike, (b) taxi, (c) subway, and (d) train.

In summary, the simulation experimental results preliminarily approve that our proposed MEC-RDESN forecasting approach can achieve the purpose of swift forecasting 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 university campus. The two students invoked a total of 273 different services. The coverage radius of the micro edge server in the campus is 150m. The user moving scenario is shown in Fig. 12. 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 (i.e., hhu rt) can be accessed from [6].

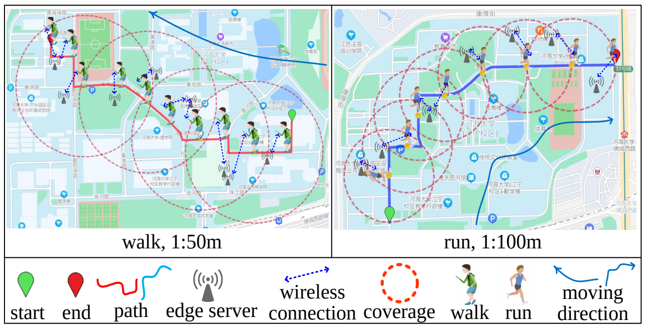


Fig. 12: The real-world experimental scenario on the campus

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 swift forecasting. The experimental results are discussed as follows.

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

Fig. 13 and Fig. 14 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, (0.6, 40) is used as the optimal hyper-parameter for *walk*, and (0.5, 90) is used for *run*.

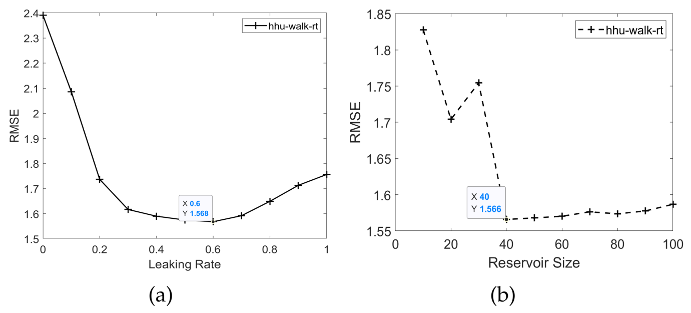


Fig. 13: hhu *walk* rt: (a) leaking rate, (b) reservoir size.

Table 8 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.

6. https://github.com/hyjin1996/mobility-aware-QoS-dataset

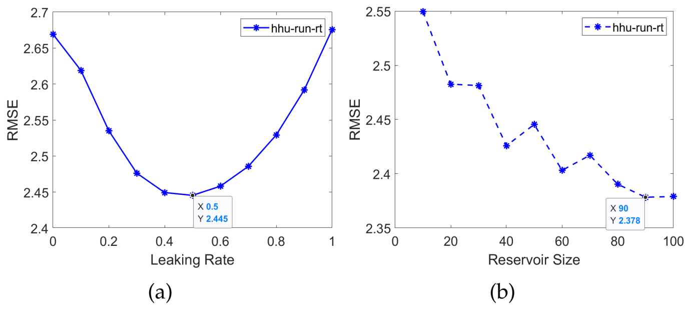


Fig. 14: hhu *run* rt: (a) leaking rate, (b) reservoir size.

TABLE 8: RMSE of Activation Functions on *walk* and *run*

Paths/Fuctions	sigmoid	tanh	relu
walk	1.6595	**1.5658**	1.5924
run	2.4316	**2.3783**	2.3822

(2) **Performance**

Table 9 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 low. Table 10 shows the forecasting errors for the two models, where our method shows higher forecasting accuracy. Table 11 shows the results of the HLN test. It shows that *MEC-RDESN* is significant compared to the other methods. In addition, as students' movement time and distance increase, more forecasting error values will be generated, which will further improve the HLN test performance.

TABLE 9: Total training time and average forecasting time

Mode	Approach	Total training time (s)	Average forecasting time (ms)
walk	Average	\	0.265
	SARIMA	0.619	41.971
	RNN	0.187	2.002
	LSTM	4.158	2.991
	ESN	0.026	0.723
	RESN	0.024	0.583
	MEC-RDESN	**0.013**	**0.689**
run	Average	\	0.13
	SARIMA	0.704	27.039
	RNN	0.233	1.062
	LSTM	4.054	2.196
	ESN	0.064	0.653
	RESN	0.06	0.554
	MEC-RDESN	**0.014**	**0.544**

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

We apply MEC-RDESN to the trajectories of the *walk* and *run* modes. The forecasting lead time calculated based on equation (7) is shown in Fig. 15. It can be seen that *MEC-RDESN* achieves sufficient lead time in the both modes, which is equal to the *Average* method.

In summary, the real-world experiment can preliminarily validate the usability and effectiveness of our proposed

TABLE 10: RMSE of hhu rt data set: (a) *walk*, (b) *run*.

(a)

ER_ID	1	2	3	4
Average	3.4987	6.7966	4.0672	2.1114
SARIMA	8.0519	10.8191	3.9347	5.0079
RNN	2.1748	4.2754	3.2079	2.6666
LSTM	2.037	4.394	2.7525	2.7408
ESN	1.9265	5.4471	2.217	2.5438
RESN	1.9302	4.2361	2.0626	1.9485
MEC-RDESN	**1.8807**	**4.1278**	**1.9045**	**1.8926**

(b)

ER_ID	1	2	3	4	5
Average	3.4379	3.067	2.8614	2.6322	3.3958
SARIMA	3.6228	2.9591	2.958	3.2096	3.5882
RNN	2.5661	**1.9169**	2.0028	2.7286	3.0918
LSTM	2.7325	1.9507	1.9407	2.5953	3.0752
ESN	2.4564	2.3022	2.2657	2.798	2.9777
RESN	2.3934	1.9404	1.953	2.3918	2.737
MEC-RDESN	**2.3783**	1.9276	**1.8787**	**2.3889**	**2.6709**

TABLE 11: HLN results on hhu rt

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

MEC-RDESN approach in the real environment. In future, we will deploy our approach on terminals (e.g., mobile phones, car tablets, etc.) for evaluation in large-scale real-world scenarios characterized by densely distributed edge nodes, high numbers of users and significant mobility.

6 CONCLUSIONS AND FUTURE WORK

Existing QoS forecasting approaches cannot meet the demand of the MEC environment on user-mobility-aware, swift 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, swift 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 investigate the impact of client device resources on forecasting accuracy to optimize the proposed approach.

ACKNOWLEDGMENTS

This work is funded by the Natural Science Research Start-up Foundation of Recruiting Talents of Nanjing University of Posts and Telecommunications (Grant No. NY223166), the National Natural Science Foundation of China under Grant No.62272145 and No.U21B2016, and the Australian

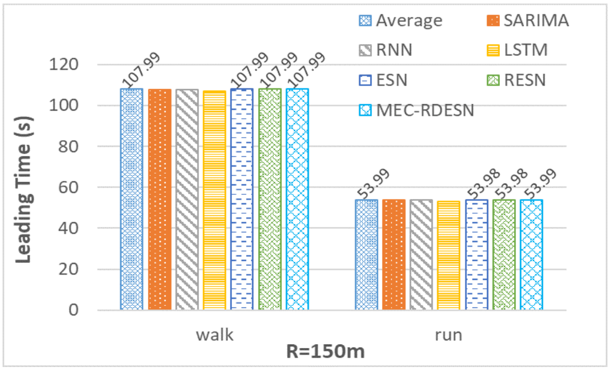


Fig. 15: Lead time of *walk* and *run* in hhu rt data set

Research Council's Discovery Projects funding scheme (DP220101823).

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] Q. Yu, X. Liu, A. Bouguettaya, and B. Medjahed, "Deploying and managing web services: issues, solutions, and directions," *The VLDB Journal*, vol. 17, pp. 537–572, 2008.

[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. Ding, J. Liu, G. Kang, Y. Xiao, and B. Cao, "Joint QoS prediction for web services based on deep fusion of features," *IEEE Transactions on Network and Service Management, DOI: 10.1109/TNSM.2023.3255253*, 2023.

[6] A. G. Neiat, A. Bouguettaya, T. Sellis, and S. Mistry, "Crowd-sourced coverage as a service: two-level composition of sensor cloud services," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 7, pp. 1384–1397, 2017.

[7] 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*, vol. 23, no. 2, pp. 1169–1185, 2024.

[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 and Tutorials*, vol. 20, no. 4, pp. 2961–2991, 2018.

[10] 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.

[11] 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*, vol. 16, no. 1, pp. 478–492, 2023.

[12] S. Blanco, "Report: Tesla autopilot involved in 736 crashes since 2019." https://www.caranddriver.com/news/a44185487/report-tesla-autopilot-crashes-since-2019/, 2023.

[13] E. Ahmed and M. H. Rehmani, "Mobile edge computing: opportunities, solutions, and challenges," *Future Generation Computer Systems*, vol. 70, pp. 59–63, 2017.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] E. Macias, A. Suarez, and J. Lloret, "Mobile sensing systems," *Sensors*, vol. 13, no. 12, pp. 17292–17321, 2013.

[21] 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.

[22] A. T. Campbell, S. B. Eisenman, N. D. Lane, E. Miluzzo, and R. A. Peterson, "People-centric urban sensing," in *Proceedings of the 2nd annual international workshop on Wireless Internet*, pp. 18–es, 2006.

[23] R. K. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: current state and future challenges," *IEEE Communications Magazine*, vol. 49, no. 11, pp. 32–39, 2011.

[24] Y. Liu, "Group-aware computing," *Communications of the China Computer Federation*, vol. 8, no. 10, pp. 38–41, 2012.

[25] Q. Peng, Y. Xia, Z. Feng, J. Lee, C. Wu, X. Luo, W. Zheng, S. Pang, H. Liu, Y. Qin, *et al.*, "Mobility-aware and migration-enabled online edge user allocation in mobile edge computing," in *2019 IEEE International Conference on Web Services (ICWS)*, pp. 91–98, IEEE, 2019.

[26] V. H. Hoang, T. M. Ho, and L. B. Le, "Mobility-aware computation offloading in mec-based vehicular wireless networks," *IEEE Communications Letters*, vol. 24, no. 2, pp. 466–469, 2019.

[27] Y. Ma, W. Liang, J. Li, X. Jia, and S. Guo, "Mobility-aware and delay-sensitive service provisioning in mobile edge-cloud networks," *IEEE Transactions on Mobile Computing*, vol. 21, no. 1, pp. 196–210, 2020.

[28] F. Liu, B. Lv, J. Huang, and S. Ali, "Towards mobility-aware dynamic service migration in mobile edge computing," in *Collaborative Computing: Networking, Applications and Worksharing: 16th EAI International Conference, CollaborateCom 2020, Shanghai, China, October 16–18, 2020, Proceedings, Part I 16*, pp. 115–131, Springer, 2021.

[29] 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.

[30] M. Liu, H. Xu, Q. Z. Sheng, and Z. Wang, "QoSGNN: Boosting QoS prediction performance with graph neural networks," *IEEE Transactions on Services Computing*, vol. 17, no. 2, pp. 645–658, 2024.

[31] G. Zou, W. Yu, S. Hu, Y. Gan, B. Zhang, and Y. Chen, "FRLN: Federated residual ladder network for data-protected QoS prediction," *IEEE Transactions on Services Computing, DOI: 10.1109/TSC.2024.3377100*, 2024.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] J. A. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava, "Participatory sensing," 2006.

[43] 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.

[44] 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.

[45] H. Jaeger, "Adaptive nonlinear system identification with echo state networks," *Advances in neural information processing systems*, vol. 15, 2002.

[46] 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.

[47] 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.

[48] 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.

[49] 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.

[50] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

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

[52] M. Lukoševičius, "A practical guide to applying echo state networks," in *Neural networks: Tricks of the trade*, pp. 659–686, Springer, 2012.

[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 received the PhD degree in Software Engineering form the College of Computer and Software, Hohai University, Nanjing, China in 2023. She is now a lecturer with the College of Computer, Nanjing University of Posts and Telecommunications, Nanjing, China. Her current research interests include services computing and edge computing. 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 (Member, IEEE) 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 Software, 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 (Senior Member, IEEE) received the Ph.D. degree from Curtin University, Perth, Australia. He is currently a Senior Lecturer with the School of Computing Technologies, RMIT University, Melbourne, Australia. His primary research interests include services computing, edge computing, blockchain, cyber security, machine learning, and data science. His publications appeared in ACM Computing Surveys, IEEE Transactions on Industrial Electronics, IEEE Transactions on Industrial Informatics, IEEE Transactions on Mobile Computing, IEEE Transactions on Services Computing, and IEEE Transactions on Software Engineering, etc.

Athman Bouguettaya (Fellow, IEEE) is a Professor in the School of Computer Science at the University of Sydney, Australia. He received his PhD in Computer Science from the University of Colorado at Boulder (USA) in 1992. He was previously Science Leader in Service Computing at CSIRO ICT Centre, Canberra. Australia. He is a Fellow of the IEEE and a Distinguished Scientist of the ACM. including, the IEEE Transactions on Services Computing, IEEE Transactions on Knowledge and Data Engineering, ACM Transactions on Internet Technology, ACM Computing Surveys, and VLDB Journal. He has published more than 250 books, book chapters, and articles in journals and conferences in the area of databases and service computing (e.g., the IEEE TKDE, the ACM TWEB, WWW Journal, VLDB Journal, SIGMOD, ICDE, VLDB, and EDBT).

Albert Y. Zomaya (Fellow, IEEE) is Peter Nicol Russell Chair Professor of Computer Science and Director of the Centre for Distributed and High-Performance Computing at the University of Sydney. To date, he has published over 700 scientific papers and articles and is (co-)author/editor of more than 30 books. Also, he is a Fellow of the Australian Academy of Science, Royal Society of New South Wales, Foreign Member of Academia Europaea, and Member of the European Academy of Sciences and Arts. He is a Clarivate 2022 Highly Cited Researcher, and his research interests lie in parallel and distributed computing, networking, and complex systems.