Space-Time-Aware Proactive QoS Monitoring for Mobile Edge Computing

Shunhui Ji, Jiajia Li, Huiying Jin, Ting Wei, Hai Dong, *Senior Member, IEEE*, Pengcheng Zhang, *Member, IEEE*, and Athman Bouguettaya, *Fellow, IEEE*

***Abstract*—This paper presents a novel probabilistic Quality of Service (QoS) monitoring method named DLSTM-BRPM (Double Long Short Term Memory (DouLSTM-Den) based Bayesian Runtime Proactive Monitoring) to accurately and efficiently monitor QoS in a mobile edge environment. This method consists of a DouLSTM-Den model and a Gaussian Hidden Bayesian classifier. The DouLSTM-Den model aims to predict a user's future movement trajectory in real time and proactively monitor the spatio-temporal QoS performance of services based on the predicted trajectory. The Gaussian Hidden Bayesian classifier is employed to accurately *monitor* QoS by constructing parent attributes to reduce the interdependence between QoS attributes. Our experiments based on public synthetic datasets demonstrate the effectiveness of the proposed method over state-of-the-art solutions. We also conducted experiments in a real-world edge environment to validate the feasibility of the proposed method.**

***Index Terms*—Mobile/Multi-Access edge computing, Quality of Service, Monitoring, Bayesian classifier, LSTM model.**

I. Introduction

MOBILE (or Multi-Access) edge computing is a new distributed computing paradigm that transfers the computing power from cloud data centers to the edge of a network [1]. Mobile edge networks employ Software Defined Network (SDN) and Network Functions Virtualization (NFV) technologies to enable networking, computing, storage, and communication resources close to end users [1]. It can process and analyze data in real-time to achieve the goal of cost and delay reduction.

Mobile edge services refer to the services provisioned in mobile edge environments. Users' requirements on mobile edge services have gradually shifted from functional requirements to non-functional requirements, i.e. Quality of Service (QoS) [2], [3]. QoS is a key discriminant in service selection [4]. Extensive studies focused on selecting a service that meets a user's QoS requirements among many services with similar functions [5]. In mobile edge environments, monitoring QoS is crucial for ensuring system reliability, performance, and security. Through QoS monitoring, services can be more effectively selected to align with user QoS requirements. In instances where issues like high network latency or service interruptions hinder users from fulfilling their requirements, the reliance on QoS monitoring can aid in early problem identification. This allows users to promptly switch to alternative services, thereby ensuring the uninterrupted fulfillment of their needs.

Effective service monitoring techniques have been extensively studied [6]–[8]. Traditional methods adopt Eclipse plugin[1] and timed automata to monitor the service providers' compliance with Service Level Agreement (SLA) [9] or model-driven methods [10]. These *server-side methods* only consider enterprise-defined standards and *ignore users' personal preferences* [11].

There is a strong trend for QoS requirements to be represented by *probabilistic* quality attributes [7]. For example, a service reliability requirement can be described as "*the average downtime probability of a service does not exceed 10% per year*". This can more precisely reflect users' expectations toward QoS. A variety of monitoring methods have been devised for probabilistic quality attributes. These include QoS monitoring methods based on traditional probability statistics [2], hypothesis testing [3], [12] and Bayes' theorem [13], [14]. Those methods aim to perform continuous QoS monitoring based on user-defined standards in addition to computation overhead reduction. However, these methods encounter the following challenges in the mobile edge environment:

Challenge 1. Traditional QoS monitoring approaches lack a proactive mechanism [15]. Most of the existing QoS monitoring methods rely on passively monitoring the current state of a service. However, monitoring the current state of a service cannot fully prevent the service from failing in the next moment. In addition, the monitoring results received by a user can only reflect the service status in the past due to network transmission delay. Therefore, it is essential to develop *proactive service monitoring* solutions to detect service failure in advance. More specifically, proactive service monitoring in a mobile edge environment needs to determine if the QoS of a service provisioned by an edge server meets a user's demands in advance. This can be achieved by predicting the user's future access to the edge server *based on the user's trajectory*. However, most existing monitoring strategies [10], [11], [14], [16] do not consider proactive service monitoring.

Challenge 2. The current QoS monitoring approaches ignore the temporal and spatial characteristics of QoS [15]. Most of the existing QoS monitoring methods [2], [3], [17],

S. Ji, J. Li, T. Wei, and P. Zhang are with the College of Computer and Software, Hohai University, Nanjing, China (e-mail: shunhuiji@hhu.edu.cn; jiajiali@hhu.edu.cn; 458279713@qq.com; pchzhang@hhu.edu.cn)

H. Jin is with the School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing, China (e-mail: hyjin@njupt.edu.cn)

H. Dong is with the School of Computing Technologies, RMIT University, Melbourne, VIC 3000, Australia (e-mail: hai.dong@rmit.edu.au)

A. Bouguettaya is with the School of Computer Science, The University of Sydney, NSW, Australia (e-mail: athman.bouguettaya@sydney.edu.au)

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

[1]https://axis.apache.org/axis2/java/core/tools/eclipse/plugin-installation.html

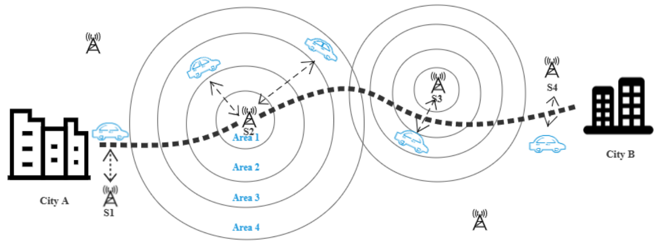


Fig. 1: Motivation scenario of service monitoring

[13], [18] , [16] overlook the impact of user mobility on QoS. In reality, a user may request services in different locations at different moments. The QoS of a service is highly correlated to *when* and *where* the service is requested. This is due to the fact that the QoS (observed from the client) depends on the status of the service (on the server) and the network environment [19]. The service status is affected by factors such as server capacity, workload, and allocated computing resources. The network environment is affected by a user's and a server's locations, network bandwidth and traffic, number of clients, etc. Both are highly dynamic in time and space [20]. For instance, different QoS values may be observed when a user invokes the same service from the same server in different locations, as a result of signal strength differences between these locations. Therefore, considering the space-time awareness QoS monitoring may enhance the monitoring accuracy.

Our previous work [15] proposed a preliminary mobile-aware QoS active monitoring method. It contains 1) an LSTM model for predicting a user's future movement trajectory and 2) a Bayesian classifier for actively monitoring service QoS. The LSTM model extracts user mobility features by analyzing historical user movement trajectories. Subsequently, perceived QoS data for the next moment is fed into a Bayesian classifier to achieve proactive service monitoring. However, this method has the following major limitations:

- *Lack of automated edge region division.* Users in the same edge region often have similar geographical locations and network communication environments. Therefore, the QoS perceived by users in the same edge region is similar. In [15], edge regions were manually divided, resulting in low accuracy. As a result, this approach is also impractical for servers with a large number of service calls.
- *Insufficient experimental validation.* The work in [15] included only a single experiment to validate the method. Such limited experimental scale and limited processing constrained its ability to demonstrate the comprehensiveness and reliability of the proposed method. Additionally, there was a lack of usability assessment in real scenarios.

This paper presents an approach named DLSTM-BRPM (DouLSTM (Double Long Short Term Memory)-Den-based Bayesian Runtime Proactive Monitoring) to address the above QoS monitoring problems in mobile edge environments. The ultimate goal of this proactive monitoring approach is to provide users with a better service experience while saving computing resources for service monitoring. We use the following mobile edge service scenario to illustrate our motivation.

As shown in Fig. 1, a user drives from city A to city B. The user requests a service from an edge server (S1) during this journey. When S1 receives this service call, our proposed solution is able to predict that the user will likely next access the same service in the edge server S2 according to the user's historical trajectory. Therefore, S2 proactively monitors the spatio-temporal QoS of the same service. We divide the signal range of S2 into several circular areas in terms of their distance to S2. These circular areas indicate different signal strengths. The signal strength has a strong correlation with the QoS performance of the edge service. Assuming that Our approach predicts that the user will next be in Area 4. We extract the historical QoS data from the same circular area during the same time period (e.g. the same day) for QoS monitoring. The extracted QoS data is then classified by a Gaussian hidden Bayesian classifier to determine whether it meets the user's QoS requirements. The classification result is viewed as the final monitoring result.

The main contributions are summarized as follows:

- This paper proposes a proactive QoS monitoring mechanism taking into account users' mobility in mobile edge environments. We construct a DouLSTM-Den model to predict a user's future movement trajectories. The model is trained upon the user's historical movement path. The QoS can be proactively monitored along with the user's predicted trajectory. This would reduce the burden on real-time monitoring and subsequent decision making. Therefore, it can reduce time and computing resources for monitoring client-side QoS.
- Our service monitoring implements spatio-temporal awareness when capturing the QoS of edge services. This is achieved by considering the geographical location and relative distance of edge servers and users, service invocation time, etc. All of these pose significant impacts on user-perceived QoS. In addition, we use the DBSCAN clustering algorithm to realize automatic partitioning of server coverage, reducing the error caused by manual partitioning for space-time-aware QoS monitoring. The inclusion of the contextual dependency and automatic partitioning greatly enhances the QoS monitoring performance.
- We use a Gaussian hidden Bayes classifier to monitor each probabilistic QoS attribute for a given service. We train a Gaussian hidden Bayesian classifier using spatio-temporal QoS data for accurate QoS monitoring. At the same time, this classifier allows us to construct a parent attribute for each QoS attribute to eliminate the inter-attribute dependency between QoS attributes and effectively improve monitoring accuracy.
- A set of dedicated experiments is performed based on a public data set and a self-created real-world data set. The experimental results validate that DLSTM-BRPM outperforms the state-of-the-art approaches in terms of feasibility and effectiveness.

The rest of the paper is summarized as follows. Section II analyzes related QoS monitoring methods and their limitations. Section III introduces the preliminary knowledge used in this approach. Section IV gives a detailed description of the DLSTM-BRPM approach. Section V discusses the evaluation of this approach on both public and real-world data sets. Section VI summarizes and prospects the work.

II. Related Work

Many monitoring methods were proposed for probabilistic QoS attributes. [2] compared the proportion of successful samples with the pre-defined probability standard to judge whether the service requirements are met. This method needs to collect a large number of service call records, which consumes a lot of resources and performance. A new toolchain, WS-PSC, was provided in [16]. It monitors time attributes in composite services based on graphical specification attribute sequence diagrams and time attribute sequence diagrams. However, other QoS attribute values cannot be monitored. A monitoring method $ProMo$ based on sampling and continuous hypothesis testing was introduced in [17]. The probabilistic attribute CSL^{MON} was defined for monitoring. It employs the sequential probability ratio test (SPRT) to perform verification. This method does not support continuous monitoring. An improved SPRT (iSPRT) method was proposed in [12]. iSPRT realizes dynamic monitoring by reusing previous monitoring information. However, it requires additional memory space to store historical monitoring results. All the aforementioned methods were built on the prerequisite of fixed probabilistic QoS requirements. In reality, users' service requirements vary along with changes in their surrounding environments. In this regard, fixed probabilistic QoS requirements cannot reflect users' dynamic service requirements.

Many probabilistic QoS monitoring techniques based on Bayesian classifiers were proposed to address the limitation of the aforementioned methods on variable user requirements. A Bayesian probability monitor ($BaProMon$) was introduced in [13]. This method checks whether the runtime QoS information supports the null or alternative hypothesis by computing the Bayes factor. In this way, it realizes effective QoS monitoring. A Web Service QoS monitoring method named weighted naive Bayes running monitoring (wBSRM) was demonstrated in [14]. This method uses the Term Frequency-Inverse Document Requency (TF-IDF) algorithm to calculate the influence of environmental factors. It cannot accurately monitor the future state of the service. A weighted naive Bayes runtime monitoring method called IgS-wBSRM was delivered in [21]. It is based on the information gain theory and the sliding window mechanism. IgS-wBSRM addresses the limitation of real-time monitoring that does not consider historically redundant data. Therefore, It can maintain monitoring accuracy in dynamic environments. The defect of IgS-wBSRM is that it ignores the impact of user preferences on QoS monitoring results. A new mobility and dependency-based QoS monitoring method named ghBSRM-MEC was presented in [22]. This method constructs a parent attribute for each QoS attribute, thereby reducing the dependencies between attributes to improve the accuracy of monitoring. It relies on historical data to monitor the current service status. The common drawback of the above methods is that they cannot judge whether the future status of services can continuously meet user demand. In addition, they cannot tolerate invalid or inaccurate monitoring results caused by network delays. In this regard, proactive QoS monitoring is required. Proactive QoS monitoring can determine if a service will continue to meet users' needs in the future and minimize the impact of network delay.

QoS monitoring has also been implemented based on neural networks. An approach for cloud-edge-based dynamic service workflow reconfiguration was introduced in [23]. The long-short-term memory neural network is used to predict the stability of the service. The stability and cost of the service are comprehensively evaluated for candidate service selection. This method does not consider the impact of user mobility characteristics on candidate service selection. A proactive approach for web service composition (WSC) was described in [24]. It relies on the Markov Decision Process to model the WSC process and a Reinforcement Learning technique to adapt to dynamic changes in the WSC environments proactively. Since this method relies on Web service execution log data to determine active adaptation, it may fail for new scenarios or application scenarios with high real-time requirements.

Proactive monitoring techniques have also been applied to other fields. A QoS monitoring algorithm that can quickly detect broken or congested links was depicted in [25]. This algorithm takes advantage of a multi-threaded design based on lock-free data structures. It improves the performance by avoiding synchronization among threads. Their work specifically focuses on real-time streaming. It does not realize proactive QoS monitoring. A proactive solution was introduced in [26]. It migrates the virtual machines before violating the actual delay threshold. The authors proposed a delay-aware resource allocation method that considers an adaptive delay warning threshold for various users. Their work focuses on dynamic resource allocation for hosting delay-sensitive vehicular services in a federated cloud. It cannot realize proactive QoS monitoring.

All the aforementioned monitoring methods do not consider proactive QoS monitoring through capturing user trajectories in mobile edge environments. They also ignore the importance of spatio-temporal QoS for monitoring accuracy. These deficiencies lead to their incapability to solve the problem of monitoring lag. This inspires us to devise a space and location-aware proactive QoS monitoring method to fully cater to mobile edge environments.

III. Preliminaries

A. LSTM neural network

The DouLSTM-Den model is based on an LSTM (Long Short Term Memory Networks) model. LSTM is a special RNN model that can predict what will happen next based on time sequences [27]. It is characterized by a time loop structure. The structure diagram of the LSTM neural network is shown in Fig. 2. An LSTM model consists of a set of interconnected recursive sub-networks. Each module contains one or more core cells for auto-correlation and three cells used to control the flow of information into the storage unit,

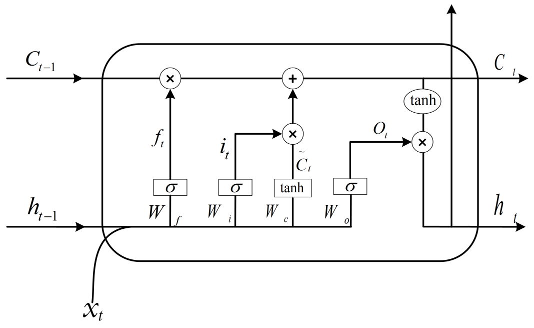


Fig. 2: Structure of LSTM network

the current unit, and the new units of the network, namely the forget gate, the input gate, and the output gate.

In LSTM, the gate is an optional way to propagate information [28]. The forget gate performs selective filtering based on the output at the previous moment. This decision is controlled by the $sigmoid$ function of the *forget gate* layer. This function is based on the output h_{t-1} and the current input x_t to generate an f_t value of 0 or 1. The forget gate is to decide whether to pass or partially pass the information C_{t-1} learned at the last moment. The f_t value is calculated as follows:

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

The input gate determines the information that needs to be updated. It adds new information to the cell state. This step consists of two parts. The first part is the *input gate* layer that uses the $sigmoid$ function to determine which values to update. The second part is the $tanh$ layer to generate new candidate values $\widetilde{C_t}$. It serves as the layer to process the data at the current moment. The calculation method is as follows:

$$\begin{aligned} i_t &= \sigma(w_i \cdot [h_{t-1}, x_t] + b_i) \\ \widetilde{C_t} &= tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \end{aligned} \tag{2}$$

Then, the output gate updates the old cell state to get the new candidate value C_t. It is expressed as follows:

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C_t} \tag{3}$$

Finally, the output gate is used to determine what value the model needs to output. First, it gets an initial output o_t through the $sigmoid$ layer. It then uses $tanh$ to scale the value of C_t to the interval [-1,1] and multiplies the output o_t obtained by the $sigmoid$ layer pair by pair to get the output of the model. The calculation method is:

$$\begin{aligned} o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ h_t &= o_t * tanh(C_t) \end{aligned} \tag{4}$$

The DouLSTM-Den model designed for our problem context will be introduced in Section 4.2.

B. Naive Bayesian classifier

The Naive Bayes classifier is a probabilistic classifier based on Bayes theorem [29]. It derives from the assumption that the attributes are independent of each other. The basic premise is that it considers the possibilities of an event to be classified by all the categories. Its theoretical basis is that the probability

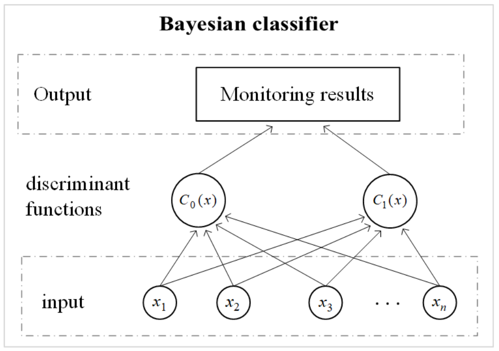


Fig. 3: Structure of Bayesian classifier

of each category is calculated when the event to be classified occurs. The category with the highest probability is considered as the category of the event [30] [31].

Bayes theorem is defined as follows: Given that the probability of the event B occurring under the condition that the event A occurs is known as $P(B|A)$ when the event B occurs, the probability of the event A occurring $P(A|B)$ can be expressed as [32]:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{5}$$

Given the predefined category set $C = c_0, c_1, ..., c_j$, the sample vector $X = x_1, x_2, ..., x_n$ is:

$$P(c_j|X) = \frac{P(c_j)P(X|c_j)}{P(X)} \tag{6}$$

When X belongs to the class c_j, the values of the elements in X are independent of each other, and $P(X)$ is the same for all the classification results. Hence the Bayesian classifier formula can be simplified to:

$$C(X) = \underset{c_j \in C}{\arg\max} \{P(c_j) \prod_{i=1}^{n} P(x_i|c_j)\} \tag{7}$$

C. QoS monitoring in mobile edge computing

QoS monitoring is one such effective method to determine whether the current service status meets user needs. Generally speaking, QoS attribute requirements can be expressed by probabilistic quality attributes. For example, the response time can be described as "The probability that the service response time to a customer request is less than 3.6 seconds is greater than 80%." Therefore, the QoS monitoring problem can be transformed into the probability calculation and analysis of whether the collected runtime information meets the probabilistic QoS requirement, proactively capturing service abnormalities.

The framework of QoS monitoring is shown in Fig. 3. QoS monitoring can be viewed as a binary classification task. The monitoring result is either *satisfactory* (i.e. the service meets the probabilistic QoS requirement) or *unsatisfactory* (i.e. the service does not meet the probabilistic QoS requirement). The probabilistic QoS monitoring process can be mathematically expressed as follows:

We define a set of QoS sample vectors of a certain service in an edge server as $X = \{x_1, x_2, x_3, ..., x_n\}$, where $x_k(k \in$

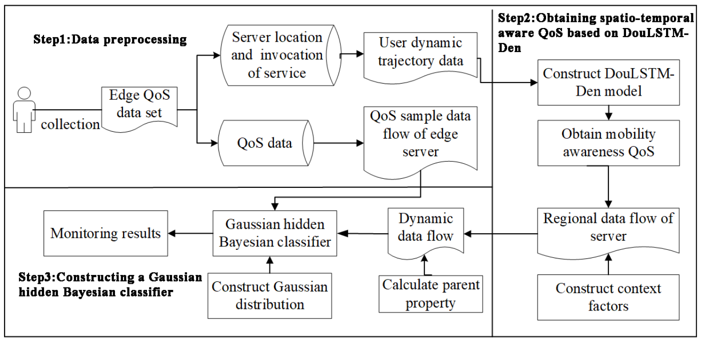

Step1:Data preprocessing
collection
Edge QoS data set
Server location and invocation of service
User dynamic trajectory data
QoS data
QoS sample data flow of edge server
Step2:Obtaining spatio-temporal aware QoS based on DouLSTM-Den
Construct DouLSTM-Den model
Obtain mobility awareness QoS
Regional data flow of server
Construct context factors
Monitoring results
Gaussian hidden Bayesian classifier
Dynamic data flow
Step3:Constructing a Gaussian hidden Bayesian classifier
Construct Gaussian distribution
Calculate parent property

Fig. 4: Structure of proactive QoS monitoring

$[1, n]$) is the value of the k th QoS sample of the service. $C = \{c_0, c_1\}$ is defined as a set of predefined categories, where c_0 represents that the sample meets the probabilistic QoS requirement, and c_1 represents that the sample does not meet the requirement. Next, a Bayesian classifier is built for each server based on these probabilistic criteria to calculate the ratio of the two types of posterior probability, where the ratio of posterior probabilities is defined as $P_c = \frac{afterPro(c_0)}{afterPro(c_1)}$. When $P_c > 1$, the monitoring result belongs to the category c_0; otherwise it belongs to c_1. In the process of service monitoring, the Bayesian classifier is trained based on historical data. Whenever a sample is collected, the state of the current sample must be tested for probability.

IV. The DLSTM-BRPM Approach

This section introduces the proposed DLSTM-BRPM approach in the mobile edge environment. Section 4.1 presents an overview of how DLSTM-BRPM facilitates proactive QoS monitoring for mobile users. Sections 4.2, 4.3, and 4.4 introduce the approach in detail.

A. Proactive QoS monitoring in mobile edge computing

The proactive QoS monitoring method is detailed in Algorithm 1. Its main framework is shown in Fig. 4. It mainly includes three steps:

1) *Data preprocessing.* First, we collect edge server locations, which are represented by latitude and longitude coordinates. The complete monitoring procedure must take into account the user's previous trajectory data and call service information to adapt to the user's mobile scene. As a result, we built a dataset of mobile edge servers and user movement trajectories. The second data preprocessing task is to filter out incorrect data, such as the sample data with a response time of -1 and 0. We also interpolate missing data in the samples to ensure the integrity of the sample data. This can make the experimental data closer to the real scenario.
2) *Obtaining spatio-temporal aware QoS based on DouLSTM-Den.* We designed a model named DouLSTM-Den to predict the user's location in the next moment. A user's moving path is tracked according to the sequence of base stations accessed by the user. The user's historical trajectory data is the input of the DouLSTM-Den model for training and prediction. The output of the model is the user's location in the next moment (see lines[1-4] in Algorithm 1). We can thus obtain the edge server that is possibly accessed by the user according to the predicted user location and the server's signal coverage. In addition, according to the findings of [33], users within the same signal coverage region of the server have similar edge environments, and user requests in the same area at the same time can obtain similar quality of service responses. Hence, we divide the signal coverage of the edge server into several circular areas. We can then determine in which circular area the user will be. This is achieved by calculating the distance between the predicted location and the server.(see lines[5-6] in Algorithm 1). In this way, the possible QoS values can be predicted based on the historical data in the same circular area with the same duration of the server (see line[7] in Algorithm 1).
3) *Constructing a Gaussian hidden Bayesian classifier.* First, a parent attribute is constructed for each attribute to enable the independence of the attribute. The value of the parent attribute is obtained based on the value of each corresponding QoS attribute (see line[8] in Algorithm 1). Next, a Gaussian hidden Bayes classifier is built upon the value of the corresponding parent attribute (see line[9] in Algorithm 1). The obtained user spatio-temporal awareness QoS value is entered into the classifier to determine whether the QoS value satisfies the probability requirement of the QoS attribute of the area to which it belongs. The posterior probability ratio of the QoS value is calculated to obtain the monitoring result (see lines[10-16] in Algorithm 1). In this way, the QoS of service possibly invoked by a user is proactively monitored based on the user's probabilistic requirement.

B. Data preprocessing

The purpose of the data collection phase is to collect the location information of an edge server and the QoS history data stored in the edge server. When there is no historical QoS data in the server to be accessed by the user, we try to find the adjacent edge servers. The historical data in the edge server is used to construct a Gaussian hidden Bayesian classifier to make monitoring decisions for the services to be called by users. The primary goal of data preprocessing is to filter out incorrect data, such as samples with a response time of -1, and to use the mean value to fill in the missing data for the samples in order to guarantee the samples' integrity and validity.

Our goal is to enable accurate proactive QoS monitoring based on the spatio-temporal dependency of QoS values. We model each QoS value as a three-dimensional tensor of $user-service-time$. Our proposed method monitors the QoS tensor and selects the historical data of similar users from the same edge server for proactive QoS monitoring. An example of the QoS tensors is shown in Fig. 5.

C. Spatio-temporal QoS acquisition based on DouLSTM-Den

The primary purpose of this step is to construct and train the DouLSTM-Den model to obtain the user's space-time-aware

Algorithm 1 Proactive QoS Monitoring

```
Require: $T_{QoS} = [X_1, X_2, ...X_n]$: training sample
        $U$:User set
        $QoS\_Value$: QoS threshold
        $\beta$: QoS probabilistic standard
        $(x_1, y_1), (x_2, y_2), ..., (x_{t-1}, y_{t-1})$: user's trajectory coordinates
        $(X_1, Y_1), (X_2, Y_2), ...$: the locations of the servers
Ensure: Proactive monitoring result $c_j$
 1: for $u_i \in U$ do
 2:     Extracting the historical locations $(x_1, y_1), (x_2, y_2), ...$ $(x_{t-1}, y_{t-1})$ and the instant location $(x_t, y_t)$ of $u_i$
 3:     Predicting $(x_{t+1}, y_{t+1})$ of $u_i$ based on DouLSTM-Den
 4:     Calculating the distance between $(x_{t+1}, y_{t+1})$ and the servers to determine the $(X_{t+1}, Y_{t+1})$ that $u_i$ will access
 5:     Determining the $area_{t+1}$ that $u_i$ will be in
 6:     Obtaining historical QoS data $T_{QoS}$
 7:     Calculating mobility-aware QoS
        $QoS_{t+1} = T_{QoS1} + T_{QoS2} + ...T_{QoSn}$
 8:     Calculating the value of parent attribute
        $\pi(x_i) = (x_1 + x_2 + ...x_{k-1})/(k-1)$
 9:     Applying Gaussian distribution to probability distribution in (19)
10:     for $i = 1->n$ do
11:         Calculating posterior probability $P$ of $c_0$ and $c_1$
12:         if $P_{C_0} > P_{C_1}$ then
13:             return $c_0$
14:         else
15:             return $c_1$
16:         end if
17:     end for
18: end for
```

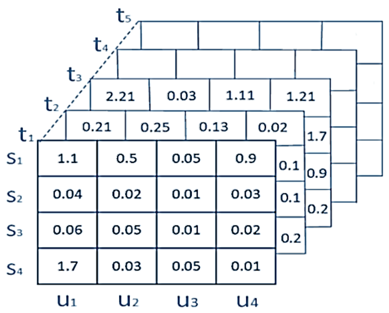

t₅
t₄
t₃
t₂
t₁
2.21 0.03 1.11 1.21
0.21 0.25 0.13 0.02
1.7
S₁ 1.1 0.5 0.05 0.9 0.1 0.9
S₂ 0.04 0.02 0.01 0.03 0.1 0.2
S₃ 0.06 0.05 0.01 0.02 0.2
S₄ 1.7 0.03 0.05 0.01
u₁ u₂ u₃ u₄

Fig. 5: A third-order tensor of QoS

QoS. Due to the continuous movement of users, relying solely on the current service status for monitoring leads to delayed results. To address this, we developed a trajectory prediction model to achieve spatiotemporal awareness of service quality. This model is then integrated into a Gaussian hidden Bayesian classifier to enable proactive monitoring of services, adapting to the user's movement. A user's historical trajectory is collected from the locations of the servers previously accessed by the user. The user's future location is predicted based on this historical trajectory. The edge server that is possibly accessed by the user in the next moment is thus determined by the user's future location and the signal coverage of servers. If

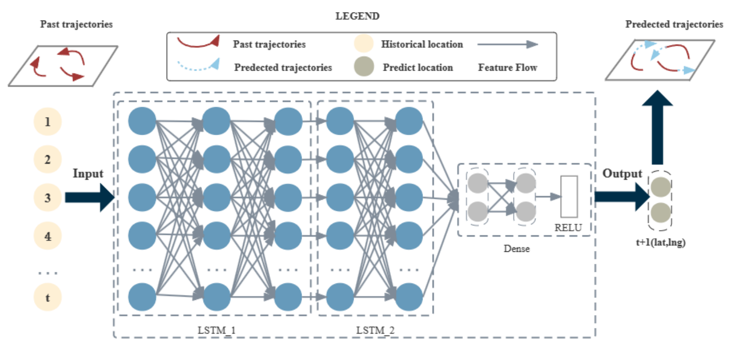


Fig. 6: Architecture of DouLSTM-Den model

the predicted user location is within two servers' overlapped areas, the server closer to the user will be selected. The LSTM is able to efficiently capture the long-term dependencies and temporal patterns in trajectory data, Here we propose a model named DouLSTM-Den to predict a user's future location. As shown in Fig. 6, DouLSTM-Den comprises an LSTM layer with 3 units, a hidden LSTM layer with 2 units, and a normal dense layer with 2 hidden outputs for 2 columns. By adopting a two-layer LSTM model, we can capture more complex temporal dependencies and sequence patterns. The first LSTM layer, with 3 units, extracts lower-level temporal features. The second LSTM layer, with 2 units, builds on these lower-level features to capture higher-level temporal dependencies. This hierarchical structure enables the model to learn more intricate patterns in the data.

The DouLSTM-Den model is trained by comparing the predicted user locations with the actual locations to establish the mapping between the user's historical and future trajectory. The sequence of base stations accessed by the user is viewed as the user's historical trajectory. A user's trajectory is defined as follows:

The original trajectory data of the moving user is converted into a sequence of h positions $H_i = \{h_1, h_2, ..., h_h\}$, where H_i represents the movement trajectory of $user_i$, $h_i = \{lng_m, lat_m\}$ represents the mth longitude and latitude of $user_i$ based on time series. The current location is $H' = \{lng_t, lat_t\}$. In practice, we continuously update the trajectory by combining the current location of the user for trajectory prediction.

We predict the $t+1$th location H_{t+1} of the $user_i$ through the DouLSTM-Den model. A high-level definition of the DouLSTM-Den model can be expressed as follows:

$$H_{t+1} = f(\{H_1, H_2, ..., H_h, H_t\}) \tag{8}$$

Its technical details can be referenced in Section 3.2.

According to [34], buildings can interfere with signal transmission to mobile devices, and wood, concrete, and metal elements can obstruct service signal propagation. The environment around the edge server (e.g., dense woods, hills, and metropolitan regions with high-rise buildings, etc.) can cause significant variations of network quality in a real-world context. At the same time, the network load in various areas varies [19]. Network quality and network load conditions have a significant impact on service response time, delay, etc. [35]. It can be inferred that the QoS values of edge servers vary

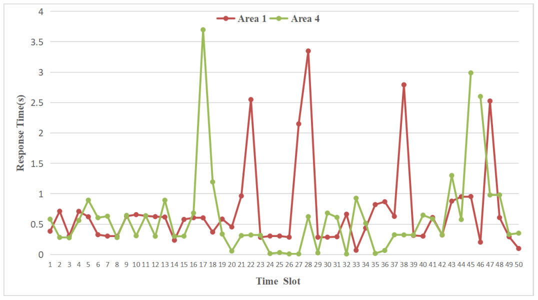


Fig. 7: Response time of a user in different areas

based on their coverage area. The coverage of the server is circular according to [36]. Accordingly, we divide the coverage area of a server into several circular rings and monitor QoS in each circular ring.

Fig.7 shows the observed response time values when a user called the same service within different coverage areas and time slots called the same service from an edge server in different time slots. This is obtained from the experimental data set. It can be seen that the QoS values of the same service at different times and locations are unequal. The response time has large fluctuations. Therefore, it is reasonable to consider time and location in the process of proactive service monitoring.

Mobile operators measure base station coverage through radio frequency planning. Base stations usually employ LPWAN technology to provide 2–5 kilometres of coverage [36]. Here we set the coverage of each edge server to 2 kilometres to imitate the service coverage offered by a real server.

The division of coverage area is expected to enhance the accuracy of QoS monitoring. To this end, the coordinates of a user and a server's locations are converted into two-dimensional plane coordinates using the Miller projection technique [37]. Next, the coverage of the server is automatically divided into circular areas by means of clustering. The Miller projection process is formulated as follows:

Let $\{lng, lat\}$ represent the longitude and latitude of a user or server. First, we convert longitude and latitude to radians. Let x_p, and y_p represent the radian value respectively, then

$$x_p = \frac{lng * \ \pi}{180} \tag{9}$$

$$y_p = \frac{lat * \ \pi}{180} \tag{10}$$

Next, we convert y_p to y_q through Miller projection, which can be realized by the following formula:

$$y_q \ = \ 1.25 \ * \ log(\tan 0.25 \ * \ \pi \ + \ 0.4 \ * \ y_p) \tag{11}$$

Finally, the converted two-dimensional plane coordinates (X, Y) is calculated by the following formula:

$$X \ = \ (W/2) \ + (W/2\pi) * x_p \tag{12}$$

$$Y \ = \ (H/2) \ - \ (H/2 * mill) * \ y_q \tag{13}$$

Where L represents the circumference of the earth, and R represents the radius of the earth, where $R = 6381.372$. When expanding the plane, the length of the X-axis equals the perimeter, which is $W = L$. The length of the Y-axis is about half the perimeter, so that $H = L/2$, and $mill$ is a constant in Miller projection.

The coverage area of the server is divided into multiple circular areas $A = (a_1, a_2, ..., a_k)$ based on the location plane coordinates of the server. The division is realized by means of clustering. This paper adopts the DBSCAN clustering method [38]. DBSCAN method is a density-based clustering algorithm that groups data points based on their proximity and density within a given epsilon radius. The density is defined in the following formula:

$$N_\epsilon(x_j) \ = \{x_j \in D | dist(x_i, x_j) \le \epsilon\} \tag{14}$$

with

$$dist(X, Y) = (\textstyle\sum_{i=1}^{n} |x_i - y_i|^2)^{1/2} \tag{15}$$

The user's distance from the server and its QoS attribute values are used as inputs to the classification algorithm for calculating the dist. If both x_i and x_j are reachable to core object samples, then x_i and x_j are densely connected. Then, the maximum density-connected sample set is defined as a class (cluster). The radius of each region from the server location is calculated by the following formula:

$$R_a^k \ = (\textstyle\sum_{i=1}^{n} s_i)/n \tag{16}$$

with

$$S = \sqrt{(\text{X-X}_{\text{bs}})^2 + (\text{Y-Y}_{\text{bs}})^2} \tag{17}$$

where S represents the distance from the user's location to the server, and $\{X, Y\}$ and $\{X_{bs}, Y_{bs}\}$ represent the two-dimensional coordinates of the user and server respectively.

We choose the server closest to a user as the server that the user is most likely to access. We then determine the exact circular area of the user where the user will be. This is achieved by calculating the distance S between the predicted locations of the user and the edge server.

The historical QoS data of the service to be invoked by the user is extracted from all the users in the same circular area a_j of the predicted edge server. It is denoted by $T_{area_{t+1}} = \{T_{u_1}, T_{u_2}, ..., T_{u_n}\}$, where T_{u_i} represents the QoS of the service invoked by the user i. The average value of the historical QoS data is calculated to obtain the mobility-aware QoS of the service. It is denoted by $QoS_{t+1} = \sum_n^1 T_{area_{t+1}}/n$, where n is the number of the users in this area.

D. QoS monitoring based on Gaussian Hidden Bayesian classifier

The main purpose of this step is to train a Gaussian Hidden Bayes classifier based on historical data. The classifier will proactively monitor the mobility-aware QoS acquired from the last step. A Naive Bayes classifier assumes that the attribute values are independent of each other. However, it ignores the fact that there might be a dependence between QoS attribute values, leading to inaccurate classification results. Here we define a parent attribute $\pi(x_i)$ to reduce the dependence between QoS attributes. Each parent attribute represents the influence of other sample attribute values on a given attribute value. Hidden parent attributes are constructed by obtaining the mean values of the sample attribute values. For a given

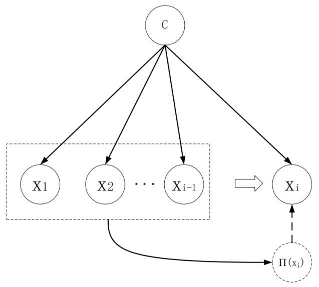


Fig. 8: Architecture of Gaussian Hidden Bayesian classifier

sample response time attribute x_i, depicted in Fig. 8, the value of the parent attribute $\pi(x_i)$ corresponds to the mean value of x_1 through x_{i-1} [22]. The improved Bayesian classifier formula can be expressed as:

$$C(X) = \underset{c_j \in C}{\arg\max} \{P(c_j) \prod_{i=1}^{n} P(x_i|\pi(x_i), c_j)\} \tag{18}$$

The Gaussian distribution is generally used to represent the class conditional probability distribution of continuous attributes. We apply Gaussian distribution to the probability distribution of continuous variables in the Bayesian classifier. The assumption of the Gaussian distribution is expressed as follows:

$$P(x_i|\pi(x_i), c_j) = N_{c_j}\left(\begin{array}{c} u_{x_i} + \rho\frac{\sigma_{x_i}}{\sigma_{\pi(x_i)}}(\pi(x_i) - u_{\pi(x_i)}), \\ \sigma_{x_i}^2(1-\rho^2) \end{array}\right) \tag{19}$$

where N_{c_j} represents the Gaussian distribution of the corresponding category c_j, u_{x_i} and $\sigma_{x_i}^2$ are the mean and variance of the sample attributes, and $u_{\pi(x_i)}$ and $\sigma_{\pi(x_i)}$ are the mean and variance of the parent attributes corresponding to the sample. The correlation coefficient between x_i and $\pi(x_i)$ is denoted by $\rho = \frac{conv(x_i,\pi(x_i))}{\sigma_{x_i}\sigma_{\pi(x_i)}}$.

In the training phase, a Gaussian hidden Bayesian classifier is constructed upon its parent attributes for each sample. The classifier is trained based on the historical data of each edge server. The spatio-temporal QoS data (i.e., the QoS data in the same circular area of a server within the same time period) is adopted as the input for the classifier. Every time a new QoS value is obtained, whether or not the QoS value satisfies the probabilistic requirements can be determined. We assume that the QoS attribute value follows the Gaussian distribution. Therefore, the determination can be implemented by the following probability density integral formula:

$$P(X < Qos\_Value) = \int_{-\infty}^{Qos\_Value} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-u)^2}{2\sigma^2}} \tag{20}$$

where μ and σ represent the mean and standard deviation of the QoS value. For example, if a QoS requirement is that the probability that the service response time is less than 2s is greater than 85%, the value of QoS_Value is 2.

In the QoS monitoring process, contains a set of user target QoS requirement vectors as $T_{QoS} = [X_1, X_2, \cdots, X_n]$, where $X_n = [x_1, x_2, \cdots, x_n]^T$ refers to the set of required QoS values of all the services called by the user n when accessing a server. The category set is $C = \{c_0, c_1\}$, where c_0 refers to a satisfactory grade and c_1 refers to an unsatisfactory grade. The posterior probabilities of c_0 and c_1 are calculated via the aforementioned process. The category with a higher posterior probability is regarded as the final monitoring result.

V. EVALUATION

A. Research Questions

The effectiveness and feasibility of the DLSTM-BRPM method are verified respectively in a simulated mobile edge environment and our created real-world data sets. The experiments aim to verify:

- RQ1: Why is the structure of the DouLSTM-Den model modelled in such a way?
- RQ2: Can QoS requirement violations be proactively monitored using DLSTM-BRPM?
- RQ3: How is DLSTM-BRPM compared with traditional monitoring methods in terms of effectiveness?
- RQ4: How does automatic partitioning affect monitoring accuracy compared to manual partitioning of server coverage?
- RQ5: How is the efficiency of our method while maintaining the accuracy of proactive monitoring?

B. Experiments on simulated data sets

1) Experiment setup

Experimental environment configuration. The TensorFlow 2.4.0 deep learning framework[2] is used to implement the proposed DouLSTM-Den model. The model is trained with a computer with Nvidia GTX1080Ti GPU. The Adam algorithm is used as the optimizer. The model is trained for 30 epochs with a batch size of 128. The initial learning rate is set to 0.001. All these parameters are optimal settings according to our experimental observation.

Data sets. This experiment involves three data sets in the experiment.

- Data Set 1 is the Shanghai Telecom data set[3]. This data set includes the geographic location information of 3,233 base stations and 611,507 service calling records, including start time and end time of service calls, server addresses (latitude and longitude), and user IDs.
- Data Set 2 is a real-world Web service quality data set released by the Chinese University of Hong Kong[4]. This data set includes the response time of 4,500 Web services called by 142 users in 64 different time slices (15 minutes per time slice).
- Data Set 3 is a simulated verification data set. The verification data set is generated according to users' QoS requirements in the experiment. The verification data is used to verify the effectiveness of the proposed method. For example, if the QoS requirement is that the probability that the response time of the service is less than 3.6s is greater than 80%, we inject more than 20% exceptional response time (i.e. greater than 3.6s) samples in a certain range of the original samples as the verification data.

[2] $https://github.com/tensorflow/tensorflow/tree/v2.4.0$
[3] $http://sguangwang.com/TelecomDataset.html$
[4] $http://wsdream.github.io/dataset/wsdream_dataset2.html$

TABLE I: Sample edge QoS data set

Response time	x_1	x_2	x_3	...	x_m
S_1	5.982	2.131	0.854	...	1.285
S_2	0.228	0.262	0.366	...	0.222
S_3	0.237	0.273	0.376	...	0.232
...	...	...	...	...	...
S_n	6.777	0.263	0.173	...	0.13

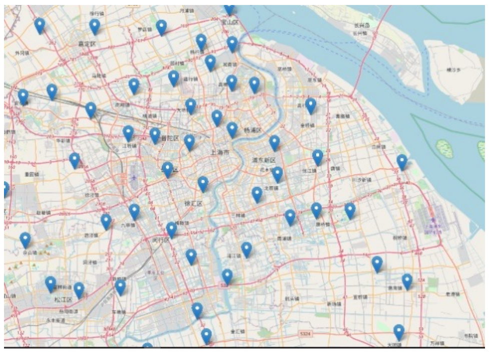


Fig. 9: Distribution of the edge servers

Data set fusion. The mobile edge environment needs to consider users' location changes. The traditional data set cannot meet such a requirement. Therefore, we employ the data fusion method [39], [40] to create a data set with dynamic user locations. A simulated mobile edge environment is constructed by fusing Data Set 1 and Data Set 2. The data fusion process is detailed as follows:

(1) QoS data partitioning. We assume that the services (with QoS) in the same geographic locations (i.e. latitude and longitude values) belong to the same edge servers. In this way, the existing data sets are merged to generate an edge QoS data set. Table I is an example of the edge service QoS data set, where S represents the edge server, and x represents the sample data stream of the edge server. This data set contains the geographic locations of 60 edge servers, the movement trajectories of 160 users (see Table II), and QoS (i.e. response time and throughput) values of 5,085 service calls.

(2) Edge data set formation. The locations of edge servers are determined by applying K-means clustering on the data set. A cluster centroid is regarded as an edge server. The other points (i.e. locations of base stations) in the same cluster are regarded as the historical users' locations covered by the edge server. Hereby users' trajectories are decided. We set k to 60 according to each server's coverage area (i.e. a circle with a radius of approximately 2 kilometres according to [41] and the distribution of the users' locations. Fig. 9 shows the locations of some edge servers.

Comparison method. We compare DLSTM-BRPM with the following state-of-the-art service quality monitoring methods to verify the superiority of DLSTM-BRPM. These include ghBSRM [22], wBSRM [14] and IgS-wBSRM [21].

2) Experimental results

Model Structure Comparison. To answer RQ1, we set up an experiment to verify if the prediction performance of the DouLSTM-Den model with the current structure is better

TABLE II: Statistics of user trajectories

Distance(km)	0-2	2-4	4-6	6-8	8-30
Number of users	31	57	51	14	7

than the models with other structures. The structure of a neural network model usually poses a significant impact on the performance of the model. Therefore, a design-of-experiment (DOE) method [42] [43] is performed to determine the best structure of the proposed model. This method allows each experiment to use the same hyper-parameter settings for training and testing, except for the structural parameters. Consequently, the model structure can be tuned.

We conduct a performance comparison between the proposed DouLSTM-Den model and the variations of the model with other structures. We follow a standard 8:2 ratio to assign the training and testing sets. The moving paths of 128 users are selected as the training set to train this model. The validation set contains the moving paths of the other 32 users. The metrics of *MAE* (Mean Absolute Error), *RMSE* (Root Mean Square Error) and R^2 are selected to evaluate the prediction accuracy of the models. Their mathematical formulae are expressed as follows:

$$MAE = \frac{\sum |u_i - u_i'|}{N} \tag{21}$$

$$RMSE = \sqrt{\frac{\sum (u_i - u_i')^2}{N}} \tag{22}$$

$$R^2 = 1 - \frac{\sum_i (u_i' - u_i)^2}{\sum_i (u_i' - u_i)^2} \tag{23}$$

where u_i is the true coordinate value of user i, u_i' is the predicted coordinate value of user i, and N is the number of predicted users. The lower the RMSE and MAE values, the higher the model's prediction accuracy. The higher the R^2 value, the better the model's fitting degree with the actual need. Table III shows the compared models and their performance upon those metrics. L_i refers to an LSTM layer with i units. D_i represents a dense layer with i hidden outputs in the output layer. We investigate several combinations between these two layers. For example, L_3 + D_3 refers to a 3-unit LSTM layer followed by a dense layer with 3 hidden outputs in the output layer. Each experiment is trained and tested with the same hyperparameter setting. It can be seen from Table III that our model based on L_3 + L_2 + D_2 achieves the best prediction results. In addition, we observed that the performance of the model is decreased when the number of network layers is increased. Therefore, we can conclude that the DouLSTM-Den model based on the structure of L_3 + L_2 + D_2 is more appropriate for user trajectory prediction in this experimental environment.

Our next experiment is to verify if the model can converge with the increasing number of epochs. Fig. 10 shows the loss curve of the DouLSTM-Den model on the training data set and the verification data set. It can be seen that this model can converge after 13 epochs of training.

Accuracy. To further answer RQ1, we set up an experiment to verify if the DouLSTM-Den model can accurately predict a

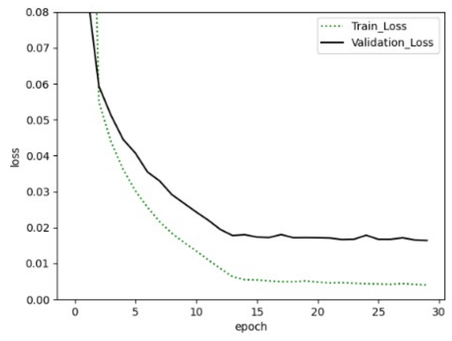


Fig. 10: Curves of loss value

TABLE III: Comparison of prediction accutacy of each model

model	MSE	MAE	R^2
L_2	$1.1e^{-3}$	$7.949e^{-3}$	$8.466e^{-1}$
L_3 + L_2	$1.0e^{-3}$	$6.797e^{-3}$	$9.081e^{-1}$
L_3 + D_2	$1.4e^{-3}$	$7.706e^{-3}$	$8.91e^{-1}$
L_3+L_2+D_2	$\mathbf{8.3e^{-4}}$	$\mathbf{3.769e^{-3}}$	$\mathbf{9.79521e^{-1}}$
L_3+L_2+L_2+L_2	$8.4e^{-4}$	$3.838e^{-3}$	$9.74129e^{-1}$

user's trajectory. We use the fused data set for the model testing. The test set contains 160 users' locations. The prediction accuracy is verified by comparing the actual users' coordinates and the predicted users' coordinates.

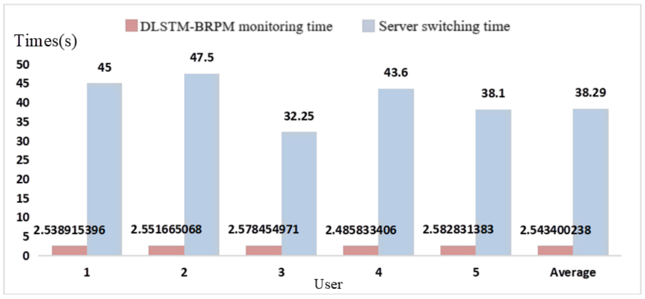
We demonstrate the average absolute error rates of 5 randomly selected users and all the users in the test set in Table IV. It can be seen that the average absolute error of the difference between the predicted and actual positions for the five users is less than 0.02. In addition, the overall error rate is also relatively low (less than 0.05). It shows that the model can accurately predict users' locations in the next moment based on the users' historical movement trajectory. Therefore, we can conclude that the prediction results of the DouLSTM-Den model can describe the users' future trajectories.

Feasibility. To answer RQ2, we set up an experiment to assess the feasibility of the proposed method. We verify whether our approach can detect abnormal service states before users access new edge servers. We choose two realistic scenarios of user movement with different speeds: 1) a simulated scenario that a user is driving, and 2) a simulated scenario that a user is riding a high-speed train. The experiment assumes that a group of 160 users call services when driving a car and taking a high-speed train respectively. We assume that the speed of the vehicle is $72km/h$ and the speed of the train is $300km/h$. The monitoring time $t_{DLSTM-BRPM}$ of our approach mainly contains two parts: the time $t_{DouLSTM}$ to obtain the mobility-aware QoS attribute value based on the DouLSTM-Den model, and the time t_{mon} to monitor the QoS using the Bayesian classifier. The estimated time t_{tra} required for a user to access a new edge server is obtained by calculating the distance between two edge servers divided by the speed.

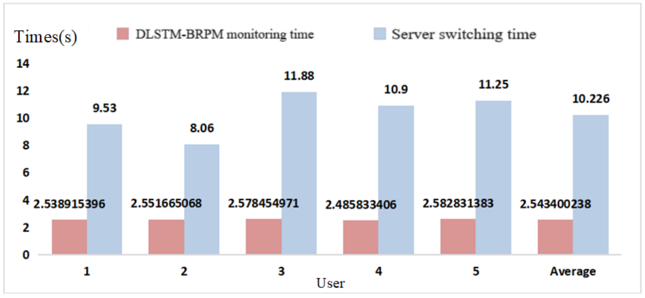
Fig. 11(a) and Fig. 11(b) respectively show the time needed for proactive monitoring and connecting to a new edge server for 5 randomly selected users and all the users when driving and taking high-speed trains respectively. We can draw a conclusion that our approach can efficiently complete the proactive service monitoring before users access new edge servers. This would provide more time for servers to make decisions if service anomalies occur.

TABLE IV: Comparison of predicted and actual coordinate

	Real coordinate	Predicted coordinate	Absolute Error
u1	(31.1616,121.5273)	(31.1625,121.5266)	(0.0009,0.0007)
u2	(31.1067,121.3989)	(31.1096,121.3876)	(0.0029,0.0113)
u3	(30.9546,121.3338)	(30.9439,121.3245)	(0.0107,0.0093)
u4	(31.1314,121.4297)	(31.1512,121.4200)	(0.0198,0.0097)
u5	(31.3140,121.5098)	(31.3230,121.4912)	(0.0090,0.0186)
All users			(0.0433,0.0496)



(a)



(b)

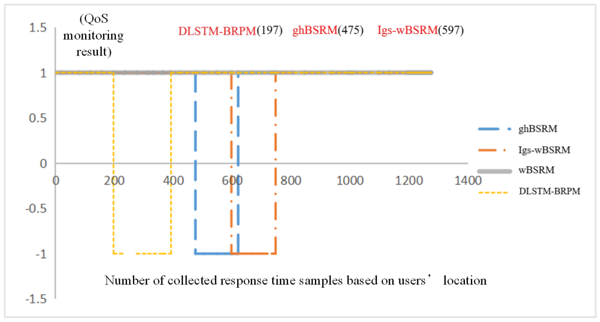
Fig. 11: Time consumption comparison between proactive service monitoring ($t_{DLSTM-BRPM}$) and server switching (t_{tra}) when (a) driving cars and (b) taking high-speed trains.

Effectiveness Comparison. To answer RQ3, we establish an experiment to verify whether the proposed proactive monitoring method can more quickly and accurately detect service exceptions before users call the services. Since response time and throughput are two typical types of QoS attributes, we selected both as monitoring attributes to validate the foundational applicability of the approach. We compared the proposed approach with the three baseline approaches mentioned above.

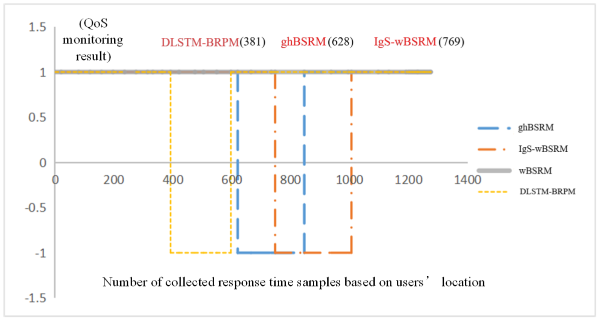
Data Set 3 is used for the experiment. Data Set 3 contains two QoS attributes, i.e., response time and throughput. First, we extract the QoS values of 2000 services to train a Gaussian hidden Bayes classifier. We then inject 200 exceptional samples with a response time of 3s in the ranges of [200,400] and [400,600] of 1000 test samples (i.e. services). We also inject 200 exceptional samples with a throughput of 1.0kbps in the ranges of [600,800] and [800,1000] of the test samples. These exceptional samples are used to verify whether the proactive monitoring method can detect service exceptions in advance.

The response time requirement is that the probability that the service response time is less than 2s is greater than

85%. The throughput requirement is the probability that the maximum throughput of the service is not less than 1.83kbps and greater than 80%. These probability values mentioned above are determined by analyzing the distribution of the data set.

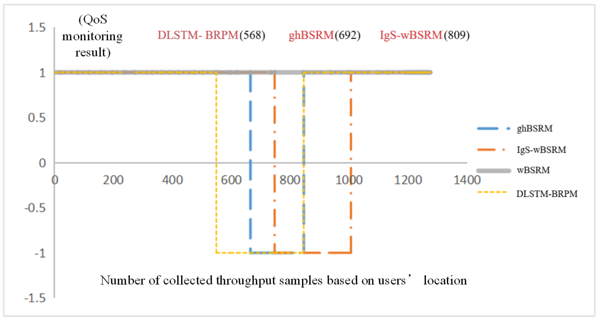


(a)

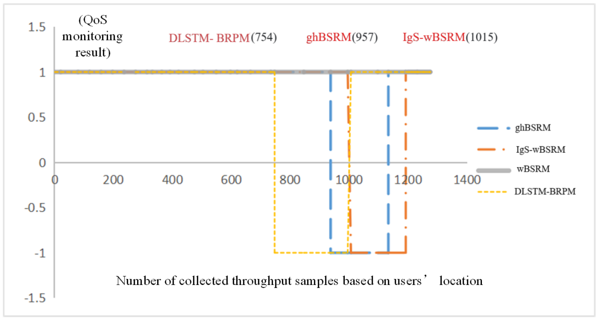


(b)

Fig. 12: Result of response time monitoring on: (a) Exceptional samples injected in [200,400] and (b) Exceptional samples injected in [400,600].



(a)



(b)

Fig. 13: Result of throughput monitoring on: (a) Exceptional samples injected in[600,800] and (b) Exceptional samples injected in[800,1000].

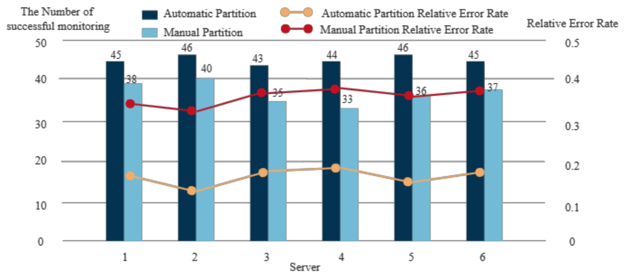


Fig. 14: Comparison of manual and automatic partition-based monitoring results

Fig. 12(a), Fig. 12(b), Fig. 13(a) and Fig. 13(b) respectively show the monitoring results of the exceptional samples injected in different intervals. The abscissa represents the number of samples that a monitoring method can obtain based on the test set (i.e., 1400 services accessed by 32 users). The ordinate represents the monitoring result, where 1 represents normal, and -1 represents abnormal. The number of samples required for each method to monitor the abnormality of the service status is marked at the top of the diagram. It can be seen that the proposed proactive monitoring method (i.e. DLSTM-BRPM) needs the lowest numbers of samples to detect the service exceptions. ghBSRM detects the service exceptions quicker than IgS-wBSRM. However, ghBSRM makes misjudgments in some cases. For example, we insert wrong samples in the interval [200, 400] in Fig. 12(a). The ghBSRM method only detects exceptions at 475 samples for the first time. In this regard, ghBSRM only considers contextual information for monitoring current QoS other than QoS data predicted to conduct active monitoring. In addition, wBSRM cannot monitor these service anomalies. This results from the fact that it does not take into account the mobility of users and the impact of other samples when making decisions for the current monitoring samples. In general, it can be seen that the prediction results of DLSTM-BRPM are more consistent with the injected exceptions. The experimental results verify the effectiveness of the proposed proactive monitoring method in the mobile edge environment.

Division mode comparison. To answer RQ4, we conduct an experiment to verify the effect of automatic partitioning on the accuracy of monitoring results. Firstly, we compare the error rates of predicted QoS (i.e. response time) between the automatic and manual partitioning methods. Second, we compare the impact of space-time-aware QoS on the monitoring results (i.e., whether the service requests at that location can satisfy the user demand) with the two partitioning approaches. We use the DBSCAN algorithm to implement automatic partitioning. The manual partitioning divides the server coverage into five areas with the respective radius of [400,800,1200,1600,2000]. This is due to the uniform distribution of historical requests observed in the coverage area. Fig.14 shows the monitoring results of 50 requests under six randomly selected servers respectively.

It can be seen that the average error rate (0.162) of the QoS predicted by the automatic partitioning is smaller than that (0.358) of the manual partitioning. The average number

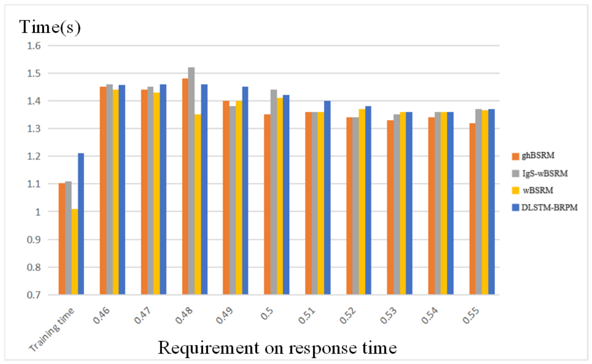


Fig. 15: Average time consumption

of correct monitoring results of the former is 9 higher than that of the latter. In addition, the average error rate of all the servers based on the automated partitioning and the manual partitioning is 0.175 and 0.347, respectively. The average number of correct monitoring QoS results of all the servers based on the automatic and the manual partitioning is 45 and 36, respectively.

Since the regions are divided by the clustering algorithm, the users in each region have similar geographical locations and similar QoS. As a result, the QoS obtained upon the predicted locations appears to be more accurate.

Performance. To answer RQ5, we set up an experiment to verify if the average time consumption (i.e., the time of training and monitoring per service) of DLSTM-BRPM is shorter than the three aforementioned baseline methods. We separately record the time required for each method to train 2000 samples under different QoS (i.e. response time) thresholds. We then measure the time required for these monitoring methods to complete the monitoring of 3000 samples under several different QoS thresholds and obtain the average time consumed for each sample. The time of the monitoring method under the same quality requirements can effectively reflect its efficiency.

The training time and the monitoring time of the candidate methods under different QoS thresholds are visualized in Fig.15. It can be seen that the monitoring time of DLSTM-BRPM is not higher than the baseline methods on the premise that its monitoring performance is significantly improved. This can be attributed to the time taken by this method to obtain spatio-temporal QoS data and predict a user's trajectory. In addition, the training cost of DLSTM-BRPM is slightly higher (approximately 0.1s) than the other baseline methods. This is considered to be acceptable given that the training is only run one time for each user in an edge server.

C. Evaluation in a real-world edge environment

1) Data set description: We conduct experiments in a small-scale real-world edge environment to further evaluate the feasibility of the proposed method. 6 edge servers are established on the campus (with a total area of 1077 acres) of Hohai University in Nanjing, China. Each edge server has a signal coverage of 50m radius (see Fig. 17). 20 students are recruited to access up to 510 services when moving among the coverage areas of those servers. Fig. 16 shows the number of

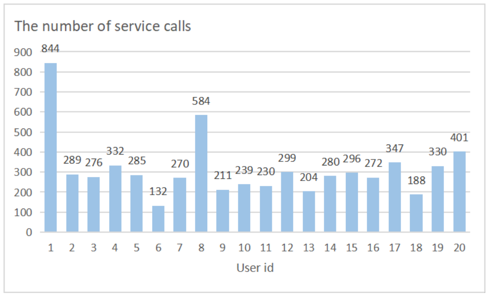


Fig. 16: The number of service calls of each user

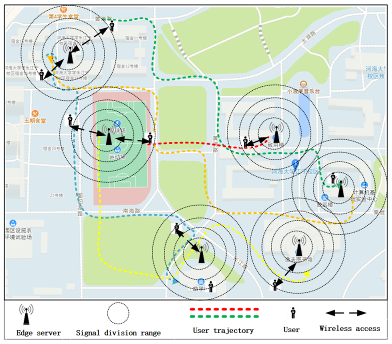


Fig. 17: Mobile edge environment in Hohai University

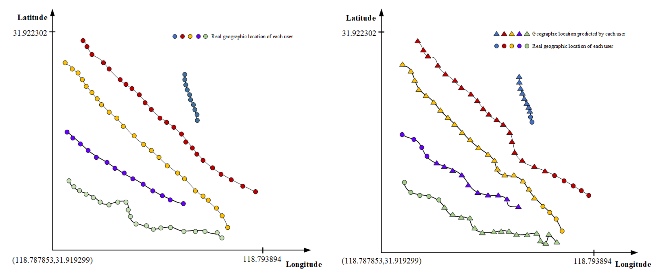
services accessed by each user. We collect the users' service access information, including User IDs, Server Addresses, Access Time, Response Time (ms) and Service Transmission Bytes, from the 6 edge servers and conduct the experiments.

2) Experimental results: Fig.18(a) shows the actual trajectories of 5 randomly selected users. Fig.18(b) shows the partial actual trajectories and the predicted trajectory of these users, where a circle represents a user's real position and a triangle represents the user's predicted position. It can be seen from Fig.18(b) that the trajectory prediction is not predicted from the initial position. In this regard, the DouLSTM-Den model needs some training data for accurate prediction. It can be seen that the predicted locations are relatively close to the actual locations. The average absolute error of the prediction for all the 20 users is less than 0.0012. This can be viewed as within an acceptable range.

We also verify the accuracy of proactive monitoring. Fig. 19(a) visualizes the monitoring result of 93 services invoked by 5 users selected from the previous population. Fig. 19(b) demonstrates the proactive monitoring result of the services, where the black icon indicates that the QoS meets a user requirement, and the white icon indicates the opposite case. It is obvious that most service statuses can be proactively tracked and the accuracy of the monitoring results is ensured. The error rate (i.e. the ratio that the monitoring results are opposite to the actual service status) for the 20 users is 6.83%. The results show that our method can effectively monitor most of the service status in advance.

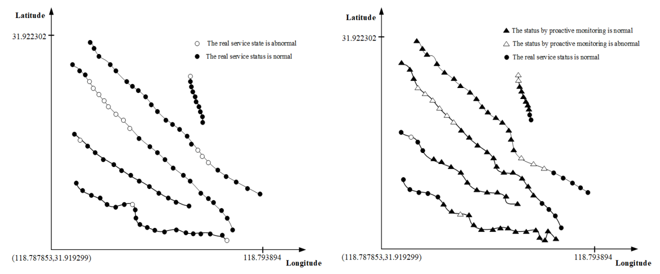
VI. Conclusion

This paper presents a proactive QoS monitoring in MEC environment based on the DouLSTM-Den model and a Gaussian hidden Bayes classifier. Experiments are conducted on both



(a) (b)

Fig. 18: Comparison between (a) actual trajectories and (b) predicted trajectories of 5 randomly selected users on the data set of Hohai University



(a) (b)

Fig. 19: Comparison between (a) actual monitoring results and (b) proactive monitoring results of 5 randomly selected users on the data set of Hohai University

simulated and real data sets, and results show the effectiveness and feasibility of the proposed method. For future work, the following tasks will be considered: i) we will design solutions to accurately predict users' multi-lag moving paths; ii) we will improve this method to adapt to multivariate QoS monitoring; iii) we will consider user privacy protection when designing future proactive QoS monitoring methods.

Acknowledgment

This work is funded by the National Natural Science Foundation of China under Grant No.62272145 and No.U21B2016, the Jiangsu Province Higher Education Basic Science (Natural Science) Research General Program (Number: 513), the Natural Science Research Start-up Foundation of Recruiting Talents of Nanjing University of Posts and Telecommunications (Grant No. NY223166), and the Australian Research Council's Discovery Projects funding scheme (DP220101823).

References

[1] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching and communications," *Ieee Access*, vol. 5, pp. 6757–67jg79, 2017.

[2] K. Chan, I. Poernomo, H. Schmidt, and J. Jayaputera, "A model-oriented framework for runtime monitoring of nonfunctional properties," in *Quality of Software Architectures and Software Quality*, pp. 38–52, Springer, 2005.

[3] I. Lee, O. Sokolsky, J. Regehr, *et al.*, "Statistical runtime checking of probabilistic properties," in *International Workshop on Runtime Verification*, pp. 164–175, Springer, 2007.

[4] A. Bouguettaya, M. Singh, M. Huhns, Q. Z. Sheng, H. Dong, Q. Yu, A. G. Neiat, S. Mistry, B. Benatallah, B. Medjahed, M. Ouzzani, F. Casati, X. Liu, H. Wang, D. Georgakopoulos, L. Chen, S. Nepal, Z. Malik, A. Erradi, Y. Wang, B. Blake, S. Dustdar, F. Leymann, and M. Papazoglou, "A service computing manifesto: The next 10 years," *Commun. ACM*, vol. 60, p. 64–72, mar 2017.

[5] H. Billhardt, R. Hermoso, S. Ossowski, and R. Centeno, "Trust-based service provider selection in open environments," in *Proceedings of the 2007 ACM symposium on Applied computing*, pp. 1375–1380, 2007.

[6] D. Gunter, B. Tierney, K. Jackson, J. Lee, and M. Stoufer, "Dynamic monitoring of high-performance distributed applications," in *Proceedings 11th IEEE International Symposium on High Performance Distributed Computing*, pp. 163–170, IEEE, 2002.

[7] D. A. Menasce, "QoS issues in web services," *IEEE internet computing*, vol. 6, no. 6, pp. 72–75, 2002.

[8] L. Baresi and S. Guinea, "Towards dynamic monitoring of WS-BPEL processes," in *International Conference on Service-Oriented Computing*, pp. 269–282, Springer, 2005.

[9] F. Raimondi, J. Skene, and W. Emmerich, "Efficient online monitoring of web-service SLAs," in *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, pp. 170–180, 2008.

[10] L. Zeng, H. Lei, and H. Chang, "Monitoring the QoS for web services," in *International Conference on Service-Oriented Computing*, pp. 132–144, Springer, 2007.

[11] L. Shao, J. Zhang, Y. Wei, J. Zhao, B. Xie, and H. Mei, "Personalized qos prediction forweb services via collaborative filtering," in *Ieee international conference on web services (icws 2007)*, pp. 439–446, IEEE, 2007.

[12] L. Grunske, "An effective sequential statistical test for probabilistic monitoring," *Information and Software Technology*, vol. 53, no. 3, pp. 190–199, 2011.

[13] Y. Zhu, M. Xu, P. Zhang, W. Li, and H. Leung, "Bayesian probabilistic monitor: A new and efficient probabilistic monitoring approach based on bayesian statistics," in *2013 13th International Conference on Quality Software*, pp. 45–54, IEEE, 2013.

[14] P. Zhang, Y. Zhuang, H. Leung, W. Song, and Y. Zhou, "A novel QoS monitoring approach sensitive to environmental factors," in *2015 IEEE International Conference on Web Services*, pp. 145–152, IEEE, 2015.

[15] T. Wei, P. Zhang, H. Dong, H. Jin, and A. Bouguettaya, "Mobility-aware proactive qos monitoring for mobile edge computing," in *Service-Oriented Computing: 20th International Conference, ICSOC 2022, Seville, Spain, November 29–December 2, 2022, Proceedings*, pp. 134–142, Springer, 2022.

[16] P. Zhang, Z. Su, Y. Zhu, W. Li, and B. Li, "WS-PSC monitor: a tool chain for monitoring temporal and timing properties in composite service based on property sequence chart," in *International Conference on Runtime Verification*, pp. 485–489, Springer, 2010.

[17] L. Grunske and P. Zhang, "Monitoring probabilistic properties," in *Proceedings of the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, pp. 183–192, 2009.

[18] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Collaborative web service qos prediction via neighborhood integrated matrix factorization," *IEEE Transactions on Services Computing*, vol. 6, no. 3, pp. 289–299, 2012.

[19] Q. Zhou, H. Wu, K. Yue, and C.-H. Hsu, "Spatio-temporal context-aware collaborative QoS prediction," *Future Generation Computer Systems*, vol. 100, pp. 46–57, 2019.

[20] Y. Zhang, Z. Zheng, and M. R. Lyu, "WSPred: A time-aware personalized QoS prediction framework for Web services," in *2011 IEEE 22nd international symposium on software reliability engineering*, pp. 210–219, IEEE, 2011.

[21] P. Zhang, H. Jin, Z. He, H. Leung, W. Song, and Y. Jiang, "Igs-wbsrm: A time-aware web service qos monitoring approach in dynamic environments," *Information and software technology*, vol. 96, pp. 14–26, 2018.

[22] P. Zhang, Y. Zhang, H. Dong, and H. Jin, "Mobility and Dependence-aware QoS Monitoring in Mobile Edge Computing," *IEEE Transactions on Cloud Computing*, 2021.

[23] H. Gao, W. Huang, and Y. Duan, "The cloud-edge-based dynamic reconfiguration to service workflow for mobile ecommerce environments: a QoS prediction perspective," *ACM Transactions on Internet Technology (TOIT)*, vol. 21, no. 1, pp. 1–23, 2021.

[24] A. Moustafa and M. Zhang, "Towards proactive web service adaptation," in *International Conference on Advanced Information Systems Engineering*, pp. 473–485, Springer, 2012.

[25] F. Tommasi, V. De Luca, and C. Melle, "QoS monitoring in real-time streaming overlays based on lock-free data structures," *Multimedia Tools and Applications*, vol. 80, no. 14, pp. 20929–20970, 2021.

[26] M. Najm, M. Patra, and V. Tamarapalli, "An Adaptive and Dynamic Allocation of Delay-sensitive Vehicular Services in Federated Cloud," in *2021 International Conference on COMmunication Systems & NETworkS (COMSNETS)*, pp. 97–100, IEEE, 2021.

[27] F. Shahid, A. Zameer, and M. Muneeb, "Predictions for COVID-19 with deep learning models of LSTM, GRU and Bi-LSTM," *Chaos, Solitons & Fractals*, vol. 140, p. 110212, 2020.

[28] K. Smagulova and A. P. James, "A survey on LSTM memristive neural network architectures and applications," *The European Physical Journal Special Topics*, vol. 228, no. 10, pp. 2313–2324, 2019.

[29] R. van de Schoot, S. Depaoli, R. King, B. Kramer, K. Märtens, M. G. Tadesse, M. Vannucci, A. Gelman, D. Veen, J. Willemsen, *et al.*, "Bayesian statistics and modelling," *Nature Reviews Methods Primers*, vol. 1, no. 1, pp. 1–26, 2021.

[30] D. J. Schad, M. Betancourt, and S. Vasishth, "Toward a principled Bayesian workflow in cognitive science.," *Psychological methods*, vol. 26, no. 1, p. 103, 2021.

[31] E. Daxberger, A. Kristiadi, A. Immer, R. Eschenhagen, M. Bauer, and P. Hennig, "Laplace Redux-Effortless Bayesian Deep Learning," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[32] J. Yu, M. Bai, G. Wang, and X. Shi, "Fault diagnosis of planetary gearbox with incomplete information using assignment reduction and flexible naive bayesian classifier," *J. Mech. Sci. Technol*, vol. 32, no. 1, pp. 37–47, 2018.

[33] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," vol. 5, pp. 450–465, 2018.

[34] H. S. Karanja, S. Misra, and A. Atayero, "Impact of mobile received signal strength (rss) on roaming and non-roaming mobile subscribers," *Wireless Personal Communications*, pp. 1–18, 2023.

[35] R. B. Sørensen, D. M. Kim, J. J. Nielsen, and P. Popovski, "Analysis of latency and mac-layer performance for class a lorawan," *IEEE Wireless Communications Letters*, vol. 6, no. 5, pp. 566–569, 2017.

[36] F. Turčinović, G. Šišul, and M. Bosiljevac, "Lorawan base station improvement for better coverage and capacity," *Journal of Low Power Electronics and Applications*, vol. 12, no. 1, p. 1, 2022.

[37] 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.

[38] D. Deng, "Dbscan clustering algorithm based on density," in *2020 7th International Forum on Electrical Engineering and Automation (IFEEA)*, pp. 949–953, 2020.

[39] F. Castanedo, "A review of data fusion techniques," *The scientific world journal*, vol. 2013, 2013.

[40] 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.

[41] Y. Aslan, S. Salman, J. Puskely, A. Roederer, and A. Yarovoy, "5G Multi-User System Simulations in Line-of-Sight With Space-Tapered Cellular Base Station Phased Arrays," in *2019 13th European Conference on Antennas and Propagation (EuCAP)*, pp. 1–5, 2019.

[42] H. Zhou, D. Ren, H. Xia, M. Fan, X. Yang, and H. Huang, "AST-GNN: An attention-based spatio-temporal graph neural network for Interaction-aware pedestrian trajectory prediction," *Neurocomputing*, vol. 445, pp. 298–308, 2021.

[43] R. Chai, A. Tsourdos, A. Savvaris, S. Chai, Y. Xia, and C. P. Chen, "Multiobjective overtaking maneuver planning for autonomous ground vehicles," *IEEE transactions on cybernetics*, 2020.

ShunHui Ji received the B.S. degree in computer science and technology and the Ph.D. degree in computer software and theory from Southeast University, Nanjing, China, in 2008 and 2015, respectively. She is currently an Associate Professor at the College of Computer and Software, Hohai University, Nanjing, China. Her research interests include service computing, cloud computing, software modeling, analysis, testing, and verification.

Jiajia Li received the bachelor's degree in Software Engineering from Zhengzhou University of Light Industry, Zhengzhou, China in 2019. She is currently working toward the MS degree with the College of Computer and Software, Hohai Unversity, Nanjing, China. Her current research interests include service computing and data mining.

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*.

Ting Wei is a master student in the College of Computer and Software, Hohai University, Nanjing, China. She received her bachelor degree in Digital Media Technology from North University of China, Taiyuan, China in 2019. Her current research interests include service computing and data mining.

Hai Dong is a Senior Lecturer in the School of Computing Technologies at RMIT University, Melbourne, Australia. He received a PhD from Curtin University, Perth, Australia. His primary research interests include Service-Oriented Computing, Edge Computing, Blockchain, Cyber Security, Machine Learning, and Data Science. He is a senior member of the IEEE.

Pengcheng Zhang received the Ph.D. degree in computer science from Southeast University in 2010. He is currently a full professor with the College of Computer and Software, Hohai University, Nanjing, China. His research interests include software engineering, services computing and data science. He has published 70 papers in premiere or famous computer science journals and conferences. He was the co-chair of IEEE AI Testing 2019 conference.

Athman Bouguettaya 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 or has been on the editorial boards of several leading journals. He is a Fellow of the IEEE and a Distinguished Scientist of the ACM.