QoS Prediction and Adversarial Attack Protection for Distributed Services Under DLaaS

Wei Liang , Yuhui Li , Jianlong Xu , Zheng Qin , Dafang Zhang , Kuan-Ching Li

Abstract—Deep-learning-as-a-service (DLaaS) has received increasing attention because of its novelty as a diagram for deploying deep learning techniques. However, DLaaS still faces performance and security issues, which must be solved urgently. Given the limited computation resources and concern of benefits, distributed DLaaS systems require quality-of-service (QoS) metrics to optimize performance and reliability. However, on the one hand, new users and services are continuously joining and leaving the system, resulting in cold start issues. On the other hand, the increasing demand for robust network connections requires the model to evaluate the uncertainty. To address performance problems, we propose a deep learning-based model called embedding enhanced probability neural network, which can extract information from inside the graph structure and estimate the mean and variance values of the prediction distribution. Meanwhile, adversarial attack is a severe threat to model security under DLaaS. Thus, we investigated the service recommender system's vulnerability and propose adversarial training with uncertainty-aware loss to protect the model in noisy and adversarial environments. Extensive experimental results on a large-scale real-world QoS dataset verify the robustness and effectiveness of our model.

Index Terms—Adversarial Attacks, DLaaS, Graph Neural Network, Probability Forecast, QoS Prediction

1 Introduction

In recent years, research advancements have led to the development of the Internet-of-Things (IoT) technology. The IoT can collect and share sensory data through sensors deployed at the edge of the network [1]. We are currently witnessing the ubiquitous usage of the IoT everywhere, such as in people's daily lives and industrial applications. Meanwhile, deep learning has achieved outstanding success in a wide range of fields, such as computer vision, speech recognition, natural language processing, resource allocation [2] [3], and network management [4]. Notably, deep learning models have shown state-of-the-art performance compared to traditional machine learning models. Intelligent applications, like smart cities, smart healthcare, smart homes, and self-driving automobiles, are increasingly relying on artificial intelligence (AI), especially deep learning models. Unfortunately, IoT devices are usually low-cost devices with limited computing power and storage resources [5], making the local deployment of deep learning models nearly impossible. This problem has prompted the introduction of computation offloading strategies, like cloud-edge orchestration [6] and deep-learning-as-a-service (DLaaS). By providing model training and inference, IoT devices can perform deep learning by invoking specific web services. Many well-known cloud platforms have introduced their DLaaS programs, including IBM Watson, Microsoft Azure ML, and Amazon SageMaker. Given the high-frequency data transmission and calculation process [7], the data in the IoT are massive and sequential in nature and difficult to collect and process. Unlike local model deployment, DLaaS replaces the heavy load of data preprocessing, model training, and validation with a series of service invocations. The web services' performance directly affects the AI performance for IoT devices. Therefore, IoT devices are susceptible to network fluctuation. Moreover, the performance of web services is related to different factors. For example, low latency is preferred for model inference, whereas high throughput is best for training set uploading. For improving AI performance for IoT devices, quality-of-service (QoS)-aware service evaluation and selection are required regardless of their functionality [8].

However, retrieving QoS attributes remains a challenge. For service providers, deploying sensors for many services is expensive. For users, invoking each candidate service for QoS is unrealistic. The QoS varies due to the unstable network environment and the different network infrastructure. Thus, collaborative filtering (CF) methods in which users can upload observed QoS values to a central server and obtain candidate services have been proposed to address this issue. Through these methods, users can obtain predicted QoS values through a recommender system instead of directly accessing them. Meanwhile, the matrix factorization (MF) framework is widely adopted for CF-based QoS prediction. MF has better resistance to data sparsity than memory-based CF approaches. To enhance MF models, hybrid-based models use similarity metrics and memory-based CF to incorporate contextual information. However, the complex nature of QoS feedbacks leads to non-linear observed data. Existing approaches under linear models may not be expressive enough for user–service interaction

- *W. Liang, Y. Li, Z. Qin and D. Zhang are with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, China.*

  E-mail: weiliang99@hnu.edu.cn; 17yhli3@stu.edu.cn; zqin@hnu.edu.cn; dfzhang@hnu.edu.cn
- *J. Xu is with Computer Science Department, Colleage of Engineering,Shantou University, Shantou, China.*
  E-mail: xujianlong@stu.edu.cn
- *K. Li is with the Department of Computer Science and Information Engr. (CSIE) at Providence University, Taiwan.*
  E-mail:kuancli@pu.edu.tw

*Manuscript received *** **, 20**; revised *** **, 20**.*

and contextual information fusion. Therefore, some works have turned to the use of neural network for improving accuracy and achieving state-of-the-art performance.

The approaches mentioned above have several shortcomings. First, we argue that probability distribution is more suitable for representing prediction in the dynamic network environment. Moreover, the previously proposed methods do not fully exploit the graph structural information formed by distributed devices and services with heterogeneous network conditions to solve cold-start issues and improve accuracy. Furthermore, adversarial attacks have not been investigated in QoS prediction, resulting in a security concern on prediction credibility.

To address these issues, we proposed a deep learning-based model termed EEPrNN, which can give value prediction with confidence and robust to adversarial attacks. The main technical contributions of this study are three-folds:

1) **Enhanced Embedding.** We proposed a novel recursive defining method, term recursive residual embedding, to exploiting structural information in QoS data. In addition, we enhanced embedding module with adversarial training to defense noise and adversarial attacks. This embedding approach is either robust under data sparsity and attacks.
2) **Probability Forecast.** Here, we proposed a probability forecast framework for QoS value distribution estimation. Estimating QoS value distribution can be further adapted to many down-stream tasks, such as decision-making process(e.g., more stable service invocation experience, whether the service recommender system should select the service with a narrower confidence interval, etc.) and anomaly detection.
3) **Extensive Experiment.** We conducted a wide range of experiments on a public QoS dataset which demonstrates our model outperforms other comparison approaches and proves the effectiveness of our proposed methods.

We structure the rest of this paper as follows. Section II discusses related work. Section III presents our motivation. Section IV details our model. Section V presents the experimental results. Finally, Section VI draws the conclusion and discusses future work.

2 Related work

2.1 DLaaS in Distributed Systems

(1) Performance. Some solutions have attempted to accelerate deep learning through hardware, while others have tried at the software-level, such as model compression and highly optimized math libraries. These techniques partly addresses the energy and computing limitations of IoT devices. However, they still fail when large models are required. As a light-weighted, platform-independent, and scalable solution, DLaaS is considered a promising option. DLaaS is compatible with the current service architecture. By providing well-defined APIs, many devices with limited resources can access APIs to gain deep learning ability. (2) Security. Given that IoT devices are usually deployed in unsecure environments, security issues, such as communication security and copyright protection [9], must be urgently resolved for these devices. Deep learning has gained popularity in many fields because of its strong capability. In recent years, many studies have discovered that neural networks are vulnerable under adversarial examples. Unfortunately, this could lead to threats against IoT devices [10] because the DL model can only be black-boxed accessed. For instance, Qiu et al. [11] designed a novel adversarial attack against network intrusion detection systems in the IoT environment whose model is remotely deployed and can only be accessed through network connections. Furthermore, the neural network is fragile and vulnerable under adversarial examples [12]. For example, the model may be poisoned when a user uploads malicious data to a remote platform. To improve the secure service invocation, Liu et al. [13] applied recurrent neural network to identify web services from encrypted data transmission, thus enabling further traffic supervision and congestion control analysis and QoS guarantee. Zeng et al. [14] used augmented data as input to help in modeling against adversarial attack, thus achieving improved model security. Li et al. [15] designed and trained feature space distance and domain mismatch mechanism with an adversarial strategy before assembling them.

2.2 QoS Prediction

Memory-based CFs can be categorized into three types: user-based, service-based, and a combination of both CFs. Shao et al. [16] first applied a user-based approach to QoS prediction, which they called UPCC. A service-based approach, which is also known as IPCC, can be implemented by switching users to services. Zheng et al. [17] proposed a mixed model by integrating the UPCC and the IPCC with weight assignment. Follow-up works improved the prediction performance through similarity metrics with improved accuracy [18] [19] and the exploitation of contextual information [20] [21] [22]. Unfortunately, the major drawback with memory-based CF methods is their failure to accurately predict when data are sparse.

MF is the most widely used model-based CF framework for QoS prediction. Zheng et al. [23] proposed a probabilistic MF (PMF) for decomposing a data matrix for QoS prediction. Other methods that exploit additional side information (e.g., spatial and temporal information) have been proposed to improve accuracy. He et al. [24] presented a hierarchical MF (HMF) model to exploit geographic information, both locally and globally. Yu et al. [25] introduced bias terms for geographic location information. Furthermore, other factors that may affect the prediction accuracy of QoS values, such as IP addresses and autonomous systems [26][27]. Xu et al. [28] suggested a reputation-aware method to mitigate the impact of reliable users, while Wu et al. [27] exploited both implicit and explicit contextual information and achieved excellent performance.

Hybrid CF is a combination of memory- and model-based CF models. Zheng et al. [29] proposed NIMF in which each user and service are treated as the ensemble of their neighbors' information. Xu et al. [30] proposed user- and service-neighborhood-based MF models (UNMF and SNMF, respectively), which identify similar neighbors through similarity computation before MF. Zhang et al. [31] proposed a two-step method called CNMF, which first partitions similar users and web services via a covering-based cluster algorithm and then utilizes neighborhood information.

With the significant progress of deep learning technology, many neural-network-based studies have been carried out. For example, Wu et al. [32] proposed a deep neural model (DNM), a framework for integrating rich contextual features, to realize multi-task QoS prediction. Xiong et al. [33] proposed a novel LSTM-based MF approach that captures the updated latent representations of users and services for online prediction. Zhou et al. [34] proposed spatio-temporal and context-aware collaborative neural models, which significantly improve the performance and gain interpretability through an attention mechanism. Xu et al. [35] proposed a neural-network-based MF approach incorporating rich context information.

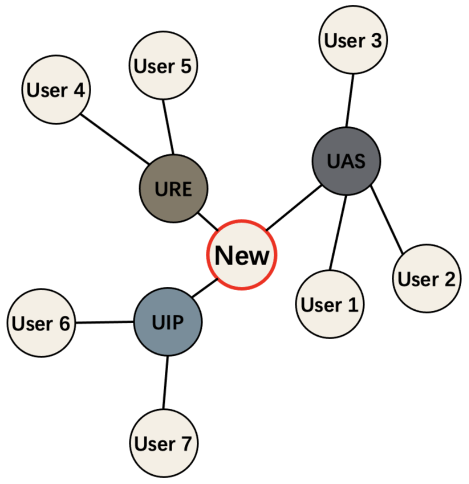
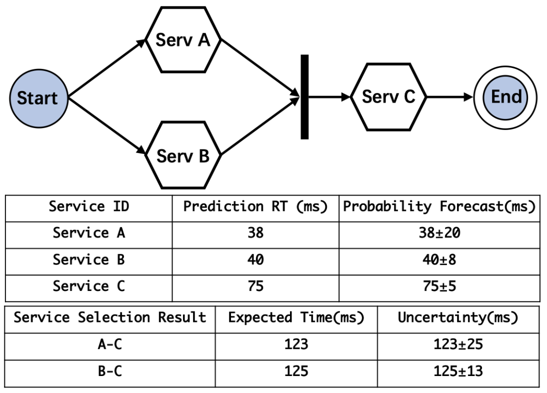


Fig. 2. Example of addressing cold start issue by neighborhood integration.

3 Motivation



Service ID	Prediction RT (ms)	Probability Forecast(ms)
Service A	38	38±20
Service B	40	40±8
Service C	75	75±5

Service Selection Result	Expected Time(ms)	Uncertainty(ms)
A-C	123	123±25
B-C	125	125±13

Fig. 1. Example of service selection. Here, we have two optional paths for service selection. path A-C shares very similar expected execution time with path B-C. However, path B-C is of significantly lower uncertainty compared to path A-C.

Figure 1 shows an example of service selection through different criteria. Previous studies on QoS prediction only focused on point forecasts without uncertainty. However, some similar users may report totally different QoS values on some services, resulting in highly uncertain prediction values. As shown in Figure 1, a service may have a low response time with high uncertainty, whereas some services may have a relatively fast response time but a low degree of uncertainty. A service selection algorithm that only focuses on the value itself may choose path A–C because it cannot predict uncertainty. In contrast, path B–C might be chosen by an uncertainty-aware algorithm as it is stable and of equal performance.

As mentioned,, the QoS prediction model regularly faces the cold-start issue. In previous works, researchers focused on addressing the cold-start issue by neighborhood integration [29][36]. On the basis of neighborhood discovery, we further explore the relation between new users and side information and start from the latter to find neighborhood users, as shown in Figure 3. In the demo, we can capture neighborhood nodes in an n-hop range not limited to 2. With enough side information, our proposed methods can always capture neighborhoods in the graph. Therefore, the cold start issue can be mitigated.

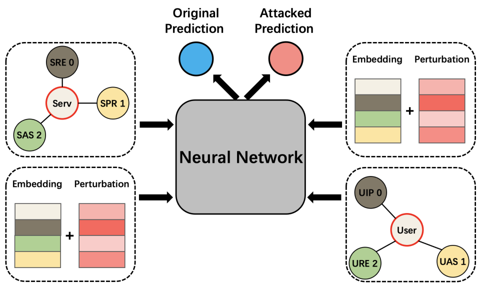


Fig. 3. Adversarial attacks on embedding module.

Adversarial attacks are a serious problem that widely exist in deep neural networks. Although adding noises in the input layer is irrational, we still naturally assume that a secure and robust recommendation model should not be sensitive to small perturbations. If random or adversarial perturbation can effectively result in worse accuracy, than the model is vulnerable to certain kinds of perturbations. Thus, adversarial attacks require urgent counteraction.

4 Methodology

4.1 Problem Description

Let $U = \{u_1, u_2, u_3, ..., u_m\}$ be a set of users and $S = \{s_1, s_2, s_3, ..., s_n\}$ be a set of services. For each $u \in U$, let $c_{u_i} = \{UAS, URE, UIP\}$ be their user side context where UAS denotes User Autonomous System, URE denotes User REgion(to be specific, their nationality) and UIP denotes User IP Address. Similiarly, for service $s \in S$, let $c_{s_j} = \{SAS, SRE, SPR, SIP\}$ be their service side context, where SAS, SRE, SPR, SIP denotes Service Autonomous System, REgion, Service Provider and Service IP Address, respectively. The context of service j invoked by user i represented by $C_{ij} = \{c_{u_i}, c_{s_j}\}$. Let interested QoS attributes be $Q = \{q_1, q_2, q_3, ..., q_p\}$. For QoS records, we use a sparse tensor $Q \in \mathbb{R}^{m \times n \times p \times t}$ to store all QoS record with different timestamps. In this paper, we assume QoS value will not change through time, therefore Q can be simplify to be a 3-dimension tensor $Q \in \mathbb{R}^{m \times n \times p}$.

Let QoS record of an invocation between service j and user i at timestamp t be a tuple :$I_{ij}^{t} =< u_i, s_j, C_{ij}, Q_{ij}, t >$.

Embedding Layer widely used in discrete mapping feature into latent representation. For convenience, we still use $u, s, uas, ure, sas, sre, spr \in \mathbf{R}^k$ to represent them in latent space where k denotes dimension of latent space.

Let f be the abstract function representing our model as we aim to predict the QoS value through the given context as well as user and service. Therefore, the probability forecast model function f can be described by the following equation:

$$[\hat{\mathbf{Q}}, \hat{\sigma^2}] = f(u_i, s_j, C_{ij}) \tag{1}$$

where $\hat{\mathbf{Q}}$ and $\hat{\sigma^2}$ denotes predicted QoS attributes and their variances, respectively.

In terms of the probability forecast model, the model predicts the parameters that determine the QoS attribute distributions and the prediction result obtained by sampling from the distribution. The probability forecast model achieves prediction through two steps:

$$\begin{aligned} &\{\theta^{(1)}, \theta^{(2)}, ..., \theta^{(k)}\} = f(u_i, s_j, C_{ij}) \\ &\hat{Q} = MonteCarloSample(\{\theta^{(1)}, \theta^{(2)}, ..., \theta^{(k)}\}) \end{aligned} \tag{2}$$

where $\theta^{(k)}$ denotes parameters for k-th QoS attribute' distribution.

4.2 Prediction Framework

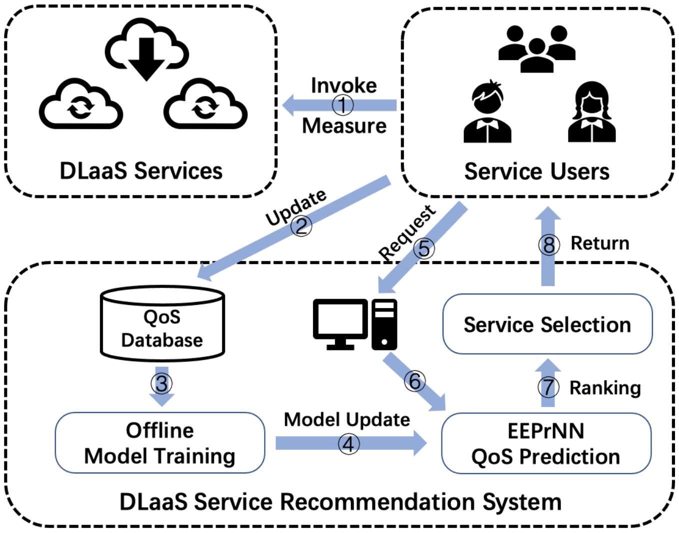


Fig. 4. The QoS prediction framework in service selection. The execution order is presented by ①-⑧.

An overview of our QoS-based cloud service selection system is shown in Figure 4, which includes three main functions:

1) **Data Collection and Update.**. For personalized prediction, we collect user-side observed QoS data. Each user collects QoS data and sends them to the server with side information on user-side c_{ui} and service-side c_{sj}. When user i invokes service j, QoS attribute Q_{ij} and timestamp t are recorded. The sampled services' invocation records will be submitted as a tuple $< u_i, s_j, C_{ij}, Q_{ij}, t >$. Then, the global user-service-timestamp tensor in the cloud database is updated. The data collection and update process corresponds to ①-②.
2) **Offline Model Training and Online Serving.** While providing prediction service, the model must be periodically retrained or optimized based on the streaming QoS data. After completing the offline training, the up-to-date model replaces the old one. The Cloud Service Recommendation System employs the latest model for prediction. The offline model training and update process corresponds to ③-④. In practice, the model is regularly updated to catch up with the latest trend.
3) **QoS-Based Service Selection.** Users request service selection suggestions via the service recommendation system server. The server predicts QoS values via our model and returns candidate service sets to users after ranking. Once the user receives the candidate lists, a service composition algorithm is executed to compose services to achieve optimal performance. The QoS-based service selection serving process corresponds to ⑤-⑧.

4.3 Model Architecture Overview

Figure 5 shows the overall architecture of our model, which consists of four parts: sparse feature input layers, recursive residual embedding layers, attention networks, and task-specific layers. Input and embedding layers are responsible for transforming discrete features into dense vectors by looking up embeddings. The attention network is employed to learn interaction and correlations hierarchically. Task-specific layers contain a MeanNet and a VarianceNet, which are responsible for the output mean and variance, respectively. The variance inference network directly receives inputs from embeddings and concatenated attention output.

To summarize, our model consists of two major parts– the left part combined with attention mechanism is used to predict the mean value, the right part is for variance estimation.

4.4 Recursive Residual Embedding

4.4.1 *Construction of Relation Graph*

In this section, we elaborate on the relation graph and its construction. In real-world scenes, users have side information, which is always shared among a group of users. Both user and side information can be regarded as nodes, and the relation can be modeled as edges. On this basis, we intuitively design a recursive defining method, that is, we define a user by its context and a context by a group of users. The graph construction algorithm based on the collected record through the framework introduced above is presented in algorithm 1. The corresponding relation graph is presented in Figure 6. The graph has no self-loop because a self-loop may incur noise under cold start.

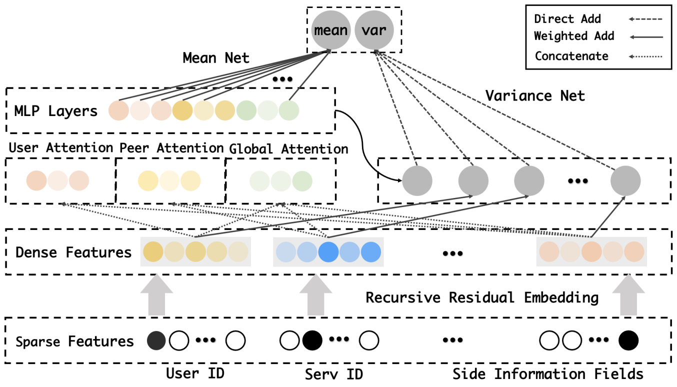


Fig. 5. Model architecture. Similar to Wide&Deep model[37], we employ a deep network with attention for implicit feature interactions for mean and a wide network for variance.

Algorithm 1: Graph Construction Algorithm

```
   Input: collected records $R$
   Output: Adjacent Matrix $A$
1  Initialize $A$ to a 0-matrix
2  Initialize a lookup table $T$
3  foreach $record \in R$ do
4      extract UID,SID,UAS,URE,UIP,SAS,SRE,SPR,SIP
        from record
5      register UID,SID...SIP if it is not in the lookup
        table $T$
6      query UID,SID...SIP's Node ID from lookup table
7      Add Link between ({UID},{UAS,URE,UIP}) to $A$
8      Add Link between ({SID},{SAS,SRE,SPR,SIP})
        to $A$
9  Transform $A$ to bidirectional graph
10 return Adjacent Matrix $A$, lookup table
```

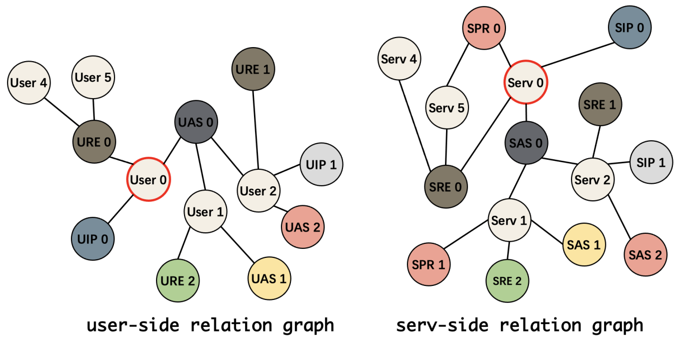


Fig. 6. The left/right part of this figure is an example of user-side/service-side relation graph(undirected graph) constructed based on users/services and their contextual information. For example, if we only consider 1-hop neighbourhood nodes, when we query the embedding of User 0, it will capture URE 0, UAS 0, UIP 0 and fused them together. Further, if we consider 2-hop neighbourhoods, User 0 will expand the receptive fields and aggregate signals from User 1,2,4,5.

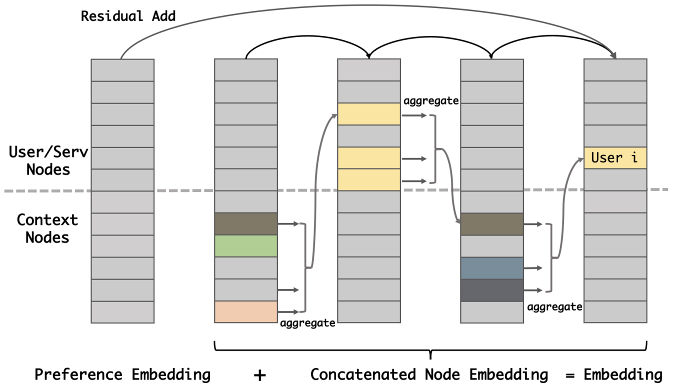


Fig. 7. Embedding Query Process.

4.4.2 Embedding Query

Embeddings are essential for a commercial recommender system and the basic components that connect the sparse features with digital representation. These components are also widely adopted in QoS-based service recommendation. In Figure 7, we present a novel embedding strategy called recursive residual embedding (RREmbedding) as the query process.

RREmbedding is parameterized by two embedding matrices, namely, the nodes embedding $\mathbf{E}_N$ and the preference embedding $\mathbf{E}_P$.. To capture the n-hop node signals on the graph, we borrow the idea of the graph convolution network [38] to aggregate information from adjacent nodes and optimize them with non-linear activation α and normalization layer $Norm$. We conduct n-order recursive residual embedding in the following manner. First, we perform message passing and aggregation:

$$\tilde{\mathbf{E}}^{(k+1)} = (\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}})\mathbf{W}^{(\mathbf{k})}\mathbf{E}^{(k)} + b^{(k)} \tag{3}$$

where where D denotes a diagnoal matrix, in which D_{ii} denotes the degree of i-th node. A represents the adjacent matrix. $\mathbf{W}^{(\mathbf{k})}$ and $b^{(k)}$ are trainable parameters of k-th layer. To accelerate the training process, we employ a normalization layer before activation function to normalize $\tilde{\mathbf{E}}^{(k+1)}$.

Further, we employ residual connection to perserve the extracted information in a deep network to alleviate over-smoothing. Therefore, we have:

$$\mathbf{E}^{(k+1)} = Norm(\alpha(\tilde{\mathbf{E}}^{(k+1)})) + \mathbf{E}_{(k)} \tag{4}$$

The final layer output, denoted as $E^{(n)}$, can be regraded as the the integration of neighborhoods. To preserve layer-wise extracted features, we concatenate output of each layer:

$$\mathbf{E}^{(neighbor)} = concat(\mathbf{E}^{(0)}, \mathbf{E}^{(1)}, ..., \mathbf{E}^{(n)}) \tag{5}$$

Nevertheless, if a group of users close to one another(e.g., in the same office, in the same university) joins the system, the users are very likely to share the same side information. In this case, the recursive embedding is insufficient to perform personalized prediction. To tackle this issue, we form another embedding matrix as personalized preference embedding. In our embedding strategy, user embedding consists of preference and neighborhood integration. Therefore, long-skip connection is added in order to learn personalized bias. Thus, we have the following:

$$\mathbf{E} = \mathbf{E}^{(neighbor)} + \mathbf{E}^{(P)} \tag{6}$$

Specifically, we applied $\mathbf{E}_N$ as $\mathbf{E}^{(0)}$ and we assign $E^{(P)}$ to 0 for newly joined nodes.

4.5 Attention Fusion Module

The attention mechanism has been widely used in deep learning as it shows effectiveness in feature selection and features fusion, such as in machine translation and recommender systems. Our work is significant in distinguishing contributions among all features as some may contain redundant information that introduce noise, which degrades the performance. A high attention weight indicates that the corresponding features have a significant contribution to the prediction results. By contrast, features assigned with low weights imply less information.

After fetching the embeddings, three representations must be built, namely, user, service, and context. These three representations are formed based on feature groups F_U,F_S,F_C with u, s, c fields, respectively. We have:

$$\begin{aligned} F_U &= [F_{U1}, F_{U2}, ..., F_{Uu}] \\ F_S &= [F_{S1}, F_{S2}, ..., F_{Ss}] \\ F_C &= [F_{C1}, F_{C2}, ..., F_{Cc}] \end{aligned} \tag{7}$$

where all features in lists above are k-dimensional dense embeddings.

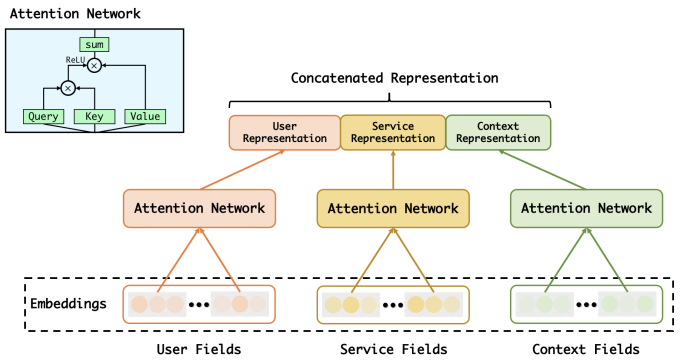


Fig. 8. The architecture of Attention Fusion Module.

The attention mechanism enables us to discriminate the importance of distinguishing factors before compressing them into a single representation. Here, we apply the self-attention [39] mechanism to capture pair-wise interactions. The attention network first uses linear transformation functions (f, g, h) to transfers x to query Q_x, key K_x, and value V_x. Then, output attention map Y is computed by matrix multiplication:

$$Y = ReLU(\frac{\mathbf{Q_x}\mathbf{K_x}^T}{\sqrt{k}})\mathbf{V_x} \tag{8}$$

where ReLU[40] is a widely used activation function in deep learning. Our attention module can automatically select the most suitable weight for computing the compress representation with limited extra parameters. Then, we perform concatenation for feature fusion because it preserves all information. The attention fusion module is illustrated in Figure 8.

User- and service-representations capture local and personalized features as they only focus on factors from their side. However, given that service invocation requires dual-side participation, context representation should include both sides' contextual factors, mainly because the invocation environment is essential in identifying the network path to generate accurate predictions. For example, the UAS may be strongly related to service autonomous systems because they represent the network path between the user and the service. Furthermore, different invocations may focus on different features that represent the path. For some users–services path, nationality determines their experience, whereas for users who share the same service provider with services may mainly determine by service provider.

4.6 Probability Forecast

Ideally, prediction methods should yield an uncertainty or confidence interval around estimation. However, neural networks do not naturally provide this capability. Therefore, we add branched networks for mean and variance. Denoting the value of QoS-specific attributes (e.g., throughput and response time) as z_{ij}, our goal is to encourage the model to output a prediction value μ and a variance σ^2, which can reflect how uncertain the prediction is. Representing service and user context as c_{ij}, where i and j respectively denote the i-th user and j-th service, the entire model function can be described as follows:

$$[\hat{\mu}, \hat{\sigma}^2] = f(c_{ij}, u_i, s_j) \tag{9}$$

The mean value, which can be regarded as the predicted value, is calculated via a three-layered perceptron. It received the concatenated attention output and transform it into prediction mean. The mean value is computed as:

$$\begin{aligned} \mathbf{z}_1 &= ReLU(Norm(\mathbf{W}_1\mathbf{X} + b_1)) \\ \mathbf{z}_2 &= ReLU(Norm(\mathbf{W}_2\mathbf{z}_1 + b_2)) \\ \mu &= W_3\mathbf{z}_2 + b_3 \end{aligned} \tag{10}$$

Uncertainty is essential in evaluating the confidence of the model. Considering that interpretation is also a critical factor in recommender systems, we design a flattening-style network, which directly targets feature embeddings and the hierarchical attention output and computes the estimated variance. The uncertainty can be formulated as follows:

$$\sigma^2 = \sum_0 n\mathbf{W}^s\mathbf{x}^n + b_s \tag{11}$$

where $\mathbf{x}^n$ denotes embedding or attention output. We transform each input $\mathbf{x}^n$ to score by $\mathbf{W}^s$ and b_s, which represent the parameters of score function. By adding them, we get σ^2, and the score indicates the contribution to uncertainty.

Noted that, our model provided a framework that is not only limited to predefined distribution. It supports many other distributions(e.g., negative binomial, t-distribution, normal distribution) by restoring parameters through mean and variance.

4.7 Multi-Task Learning

Many attributes must be considered in selecting an optimal service from the candidate list. Different QoS attributes are computed through common side information, and some are highly correlated. Low latency may indicate high throughput, as a low level of delay indicates that the communication tunnel is running correctly. To fully utilize the correlation between multiple attributes, we duplicate task specific layers that consist of a MeanNet and a VarianceNet.

4.8 Adversarial Loss Function

To estimate model parameters, an objective function must be specified. The following loss functions are commonly used: mean absolute error (MAE) and mean square error (MSE). However, they are fit for regression tasks but not for probability prediction tasks because a task has multiple outputs. The outputs are interdependent, which must be explicitly considered.

In our model, we assumed the aleatoric uncertainty subjected to Gaussian Distribution and the corresponding loss function is given by [41]:

$$\mathcal{L} = \min_{\Theta} \sum_{i=1}^{N} \frac{(z_{ij} - \hat{\mu_{ij}})^2}{2\sigma^2} + \frac{1}{2} log\sigma^2 + \lambda||\Theta||^2 \tag{12}$$

where $\hat{\mu_{ij}}$ σ^2 are outputs of our model, z_{ij} is the label. Regularization terms are applied to the model's trainable parameters Θ.

We do not need any extra labels to learn uncertainty. In this objective function, the model is encouraged to output a large variance when the prediction error is large. Such a variance cannot be minimized to zero because this can lead to increased loss. The model learns how to reach a balance in the training process.

Adversarial examples are severe threat to neural networks. We uses adversarial perturbation proposed in [42] to generate adversarial sample Δr and our new loss function is presented as:

$$\tilde{\mathcal{L}} = \mathcal{L} + \lambda \mathcal{L}(\Theta; x + \Delta r) \tag{13}$$

5 Experiments

In this section, we estimate our model with extensive experiments. Specifically, we aim to answer:

RQ1 How does our model compare to baseline methods? To what extent our model improves the performance?

RQ2 Does our model vulnerable to adversarial attacks? How robust is our model under attacks after adversarial training?

RQ3 How do the hyper-parameters affect the performance?

RQ4 How robust is the model in cold start scene? What can learn from attention mechanism?

RQ5 What can learn from probability forecast? What can affect prediction uncertainty?

5.1 Experimental Settings

5.1.1 Dataset

We selected the WS-DREAM dataset for this experiment. WS-DREAM[43] dataset #1 contains a QoS matrix of 339 users and 5825 services, and includes two QoS attributes, RT and TP. The basic statistics of this dataset are listed in Table 1.

TABLE 1
Statistics of Dataset#1

Attribute	Value
Users	339
Services	5825
Response-Time Matrix Density	94.9%
Throughput Matrix Density	92.7%
Response Time(s)	0.001-19.9
Throughput(kbps)	0.004-1000
User Autonomous Systems	137
Service Autonomous Systems	822
User Country	31
Service Country	71

5.1.2 Data Preprocessing

Given that users did not invoke this service, we removed entries marked as -1 from the QoS data matrix. The dataset's statistics (Table 1) indicate that the observed matrix is dense, but in the actual scene, the matrix could be sparser. For a better simulation of the actual scene's performance, we use low matrix densities from 2.5% to 10% with a stepping value of 2.5%. We randomly take 2.5% of the data from the data matrix as the training set and the remaining part as the test set. Subsequently, we generate all matrix subscripts and samples after random permutation.

5.1.3 Evaluation Metrics

We evaluate the model performance of ours in comparison with other existing approaches by using the following metrics:

- **MAE** (Mean Absolute Error). It is commonly used in evaluating the performance in recommender systems.
$$MAE = \frac{\sum |Q_{ij} - \hat{Q}_{ij}|}{N_{sample}} \tag{14}$$
- **RMSE**(Root Mean Squared Error). RMSE is used to illustrate the degree of dispersion of the sample. For non-linear fittings, the smaller the dispersion, the better the RMSE.
$$RMSE = \sqrt{\frac{\sum (Q_{ij} - \hat{Q}_{ij})^2}{N_{sample}}} \tag{15}$$
- **MRE**(Median Relative Error). MRE is used to take the median value of pair-wise error. Similar to RMSE, it can reflect whether the majority of relative errors distribute in a wide range.

where Q_{ij} denotes ground truth QoS value and N_{sample} means the count of test sample in above metrics. We choose these metrics as: **1)** QoS values are of large variance. Thus, it is better to focus on relative error metrics that reflect prediction robustness, i.e., MRE. **2)** As it is widely used in evaluating performance in QoS prediction fields, we also included MAE and RMSE for comparison.

5.1.4 Baselines

We compare the following models with the proposed model, to demonstrate the prediction performance of the proposed model. In terms of neighborhood-based methods, PMF uses neighborhood information for the QoS matrix. We select Biased MF and HMF as the representative methods for location-based methods. For context-aware models, we choose CSMF and DNM. DNM is also a neural-network-based method.

TABLE 2
Basic information of the methods in comparison

Method	Category	Context-Aware	Multi-Task
xPCC	Memory Based	No	No
PMF	Matrix Factorization	No	No
NIMF	Matrix Factorization	No	No
BiasedMF	Matrix Factorization	Yes	No
CSMF	Matrix Factorization	Yes	No
DeepFM	Neural Network	Yes	No
NeuMF	Neural Network	No	No
DNM	Neural Network	Yes	Yes
EEPrNN	Neural Network	Yes	Yes

1) **xPCC** [16], [44], [17] are methods based on Pearson Correlation Coefficient.
2) **PMF** [23] performs a probabilistic model in matrix factorization.
3) **NIMF** [29] is a hybrid methods combining matrix factorization with similarity computation via PCC.
4) **BiasedMF** [25] is a matrix factorization method with bias terms used to quantify the location impact.
5) **CSMF** [27] is a context-sensitive matrix factorization model that exploits interactions of users-to-services and environment-to-environment.
6) **DeepFM** [45] is a variant of Wide&Deep model combining the power of factorization machines and deep learning in a new neural network architecture.
7) **NeuMF** [46] generalized matrix factorization under NCF framework by modeling with non-linearities through a multi-layer perceptron to learn the user-item interaction function.
8) **EEPrNN** is the model we proposed in this paper with **Pr**obability Forecast mechanism. This model uses no alternative masked training.

5.1.5 Parameters Setting

We implement our model in PyTorch 1.4 with Python 3.7 and deploy it in a server with Ryzen 9 process, 32GB memory, and two NVIDIA GTX 2080Ti GPUs. In terms of model settings, we select 64 as the hidden dimension for embedding. For the gradient descent algorithm, we select the adaptive moment estimation [47] algorithm (Adam). The hidden units of MLPs(see Figure 5) are set to {256, 128, 128}. The initial learning rate and weight decay (L2 Regularization) are set to 0.002 and 0.001, respectively . For the training process, we set the batch size to 256.

5.2 Performance Evaluation (RQ1)

According to Table 3 and Table 4, we have the following observations:

- Generally, factorization-based methods are superior to memory-based methods in all metrics, which can be verified by comparing xPCC with xMF. The relative difference is rather significant when the data is sparser, indicating that the factorization-based methods are more effective in coping with data sparsity than the other methods.
- Through the comparison of PMF, BiasMF, and CSMF, we can summarize that introducing more side information into the network increases the prediction accuracy. This finding proves that contextual factors contain useful information for model inference.
- Neural network-based methods, DNM, and our model significantly outperform MF-based methods, such as CSMF, BiasMF, and PMF, confirming that deep learning techniques provide us with powerful non-linearity approximation ability for QoS prediction.
- Compared with other methods without modeling the relationship between features in a graph structure, EEPrNN achieves MAEs of 0.418 and 16.117 on RT and TP, respectively, with improvements of 11.25% and 21.11% against the best of baselines when the matrix density is 2.5%. As the data become sparser, our model's leading edge increases. This phenomenon demonstrates that our model can effectively deal with data sparsity, which is vital in real-world service recommendation. Furthermore, the proposed embedding strategy can be used for neighbor discovery to capture the signals to mitigate the low accuracy under extreme data sparsity.

5.3 Adversarial Security (RQ2)

We investigate our model under adversarial attacks by adding adversarial perturbation to different part of the embeddings. The performance degradation is given as below:

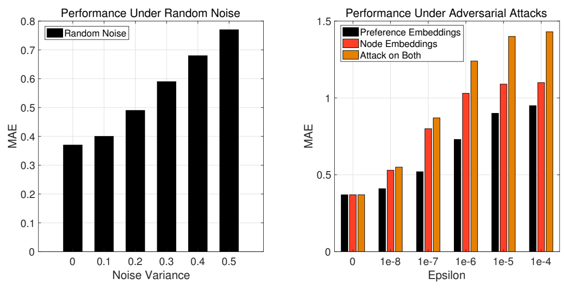


Fig. 9. Performance degradation brought by noise and adversarial examples.

According to Figure 9, we summarized:

- The origin model is sensitive to noise and the prediction performance is greatly decreased with stronger noise.
- RREmbeddings is high vulnerable to adversarial attack even under weak adversarial perturbations. Our experiments show that the adversarial attack target on "Node Embeddings" is more dangerous than the attack on "Preference Embeddings".

Through the table above, we can know that, if there isn't any defense to adversarial examples, the whole model will be destroyed. Therefore, adversarial training is necessary to counteract with this issues.

From Figure 10 and Table 5, we learned that:

TABLE 3
Performance Comparisons of QoS Prediction Models on Response Time

Response Time												
Method	Density = 2.5%			Density = 5%			Density =7.5%			Density = 10%		
	MAE	MRE	RMSE	MAE	MRE	RMSE	MAE	MRE	RMSE	MAE	MRE	RMSE
UPCC	0.709	0.777	1.467	0.640	0.665	1.380	0.588	0.567	1.339	0.556	0.523	1.309
IPCC	0.755	0.626	1.657	0.637	0.619	1.399	0.615	0.607	1.367	0.596	0.591	1.343
UIPCC	0.690	0.744	1.460	0.625	0.633	1.368	0.578	0.548	1.327	0.549	0.509	1.298
PMF	0.713	0.753	1.844	0.570	0.515	1.537	0.516	0.453	1.398	0.486	0.426	1.318
NIMF	0.673	0.624	1.738	0.555	0.487	1.477	0.509	0.449	1.361	0.479	0.428	1.291
BiasMF	0.689	0.900	1.543	0.590	0.672	1.385	0.534	0.548	1.304	0.502	0.503	1.253
CSMF	0.558	0.446	1.435	0.492	0.403	1.308	0.468	0.388	1.289	0.444	0.379	1.218
DeepFM	0.478	0.267	1.491	0.414	0.214	1.350	0.389	0.185	1.293	0.367	0.171	1.260
NeuMF	0.471	0.400	1.362	0.415	0.281	1.302	0.367	0.185	1.273	0.354	0.171	1.220
DNM	0.499	0.333	1.464	0.437	0.235	1.396	0.400	0.200	1.362	0.362	0.184	1.275
EEPrNN-ST	0.448	0.257	1.398	0.399	0.194	1.301	0.377	0.195	1.240	0.363	0.186	1.201
EEPrNN	0.418	0.222	1.320	0.372	0.190	1.250	0.362	0.171	1.229	0.352	0.169	1.189
Improve vs Baseline	11.25%	16.85%	3.08%	10.14%	11.21%	3.99%	1.36%	7.57%	3.46%	0.56%	1.17%	2.38%

TABLE 4
Performance Comparisons of QoS Prediction Models on Throughtput

Throughput												
Method	Density = 2.5%			Density = 5%			Density =7.5%			Density = 10%		
	MAE	MRE	RMSE	MAE	MRE	RMSE	MAE	MRE	RMSE	MAE	MRE	RMSE
UPCC	31.759	0.812	67.816	27.209	0.673	60.907	24.493	0.580	57.176	22.605	0.513	54.522
IPCC	31.781	0.601	73.172	27.077	0.598	62.930	26.415	0.590	61.318	26.182	0.593	60.353
UIPCC	30.547	0.725	68.556	26.024	0.636	60.996	24.462	0.589	58.238	23.479	0.559	56.271
PMF	24.409	0.410	72.546	19.117	0.309	58.661	16.921	0.275	52.447	15.761	0.260	48.951
NIMF	25.401	0.449	70.253	18.885	0.306	56.123	16.411	0.262	50.239	15.108	0.241	47.055
BiasMF	29.614	0.759	72.498	23.694	0.571	59.447	23.257	0.599	55.851	22.324	0.575	52.641
CSMF	20.431	0.400	58.016	16.934	0.324	49.837	15.501	0.311	45.706	14.997	0.284	43.460
DeepFM	21.142	0.330	64.193	17.007	0.252	52.781	14.898	0.214	46.811	14.026	0.200	45.066
NeuMF	23.928	0.397	63.952	15.833	0.236	49.775	13.999	0.199	45.341	13.042	0.189	43.076
DNM	21.004	0.339	67.232	17.078	0.251	58.186	14.598	0.210	50.854	13.934	0.194	48.208
EEPrNN-ST	16.308	0.320	52.062	13.191	0.181	43.214	12.095	0.159	40.937	11.486	0.155	38.850
EEPrNN	16.117	0.217	51.500	12.539	0.192	40.952	11.944	0.165	39.619	11.378	0.156	38.146
Improve vs Baseline	21.11%	34.24%	11.23%	20.80%	18.64%	17.72%	14.68%	17.08%	12.62%	12.76%	17.46%	11.44%

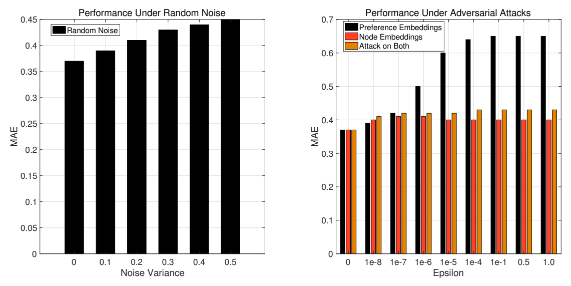


Fig. 10. Performance degradation brought by noise and adversarial examples after adversarial training.

TABLE 5
Performance degradation brought by noise and adversarial examples.

The relative decrease in MAE after applying random noise								
	var=0.1		var=0.2		var=0.3		var=0.4	
Dataset	w/o Adv	Adv	w/o Adv	Adv	w/o Adv	Adv	w/o Adv	Adv
RT	-8.10%	-5.40%	-32.40%	-10.80%	-59.40%	-16.20%	-86.50%	-18.90%
TP	-21.50%	-0.29%	-66.60%	-10.20%	-109.30%	-16.00%	-150.90%	-21.20%

The relative decrease in MAE after applying adversarial perturbation								
	eps=1e-8		eps=1e-7		eps=1e-6		eps=1e-5	
Dataset	w/o Adv	Adv	w/o Adv	Adv	w/o Adv	Adv	w/o Adv	Adv
RT	-47.70%	-9.90%	-136.10%	-12.60%	-233.90%	-12.30%	-277.80%	-14.20%
TP	-70.90%	-6.80%	-198.90%	-9.40%	-314.00%	-11.40%	-367.90%	-17.35%

- Adversarial training is an effective way to counteract adversarial attacks. The model gains can remain high accuracy under adversarial attacks with slight performance loss.
- Adversarial training is not only work for adversarial examples, but also enable model to deal with random noise.

5.4 Impact of Hyper-Parameters (RQ3)

To respond to RQ2, we conduct a series of experiments to investigate our model's main hyper-parameter.

5.4.1 *Impact of Dimensionality*

The embedding dimension determines the number of latent factors to represent features. To investigate how dimensionality can affect the prediction performance, we change dimension k from 16 to 128 (multiplied by two each time) and let the matrix density be 2.5%, 5%, 7.5%, and 10%, respectively. The results are presented in Table 6. Generally,, the performance is improved when the dimension and the matrix density increase. When the dimension is 64, the best performance is achieved in most scenes. Although no trend indicates how the performance changes with the dimension, a small dimension is recommended because a large dimension implies high computation complexity (as the complexity of our model is linearly related with dimension) with high risks of overfitting.

5.4.2 *Impact of Embedding Depth*

As RREmbedding plays a pivotal role in EEPrNN, we conduct experiments to investigate its impact on performance. We start by exploring the influence of depth to check whether EEPrNN can benefit from stacking graph convolution layers. Then, we study how the aggregator affects the performance. Depth n means the capture of n-hop nodes as neighbors, while the aggregator refers to how we can aggregate neighborhood information. Given that our relation graphs are anti-reflexive, we only select aggregators

TABLE 6
Performance comparisons with different dimension and in different matrix density

Dimensionality: Response Time

Method	Density = 2.5%		Density = 5%		Density =7.5%		Density = 10%	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
16.000	0.421	1.351	0.381	1.264	0.366	1.223	0.355	1.193
32.000	0.418	1.331	0.390	1.295	0.370	1.249	0.355	1.206
64.000	0.418	1.320	0.372	1.2502	0.364	1.225	0.352	1.189
128.000	0.432	1.358	0.382	1.263	0.372	1.247	0.353	1.189

Dimensionality: Throughtput

Method	Density = 2.5%		Density = 5%		Density =7.5%		Density = 10%	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
16.000	15.855	51.325	12.819	42.126	11.983	39.912	11.209	37.444
32.000	15.481	49.781	12.828	42.003	12.029	39.940	11.314	38.103
64.000	16.117	51.499	12.539	40.952	11.995	39.831	11.378	38.146
128.000	15.938	50.964	12.948	42.294	11.850	39.877	11.247	37.744

TABLE 7
The impact on performance of n-hop neighbourhood.

Aggregator	Hop=0		Hop=1		Hop=2		Hop=3	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
Mean			0.3867	1.2734	0.3795	1.2718	0.3721	1.2502
Pool	0.4093	1.2991	0.3884	1.2944	0.3831	1.261	0.3772	1.2511
GCN			0.3889	1.279	0.3798	1.2659	0.3822	1.2707

that allow zero in-degrees, namely, mean, GCN, and pool, and 0-hop means that conventional embedding is adopted.

According Table 7, we have the following observations:

- In most cases, increasing the depth of RREmbedding substantially enhances the accuracy. RREmbedding-2 and RREmbedding-3 are better than RREmbedding1, which takes first-order neighbors only. This finding implies that a 1-hop neighborhood is insufficient to generate embedding as users only aggregate users' profiles accurately. In contrast, 2-hop neighborhoods can improve the performance because, for each user, they can aggregate other users' information through a shared user profile (user $\rightarrow$ profile $\rightarrow$ user). %RREmbedding-1 defines a user by aggregate contextual factors, and user groups represent contextual factors. RREmbedding-2 extracts further information.
- Different types of aggregators show varied performance. The GCN- and Pool-aggregators employ linear transformation to process signals, whereas the Mean aggregator takes the average of neighborhoods. However, the latter exhibits effectiveness, suggesting that the node features' transformation might be unnecessary, indicating that users who share the same contextual factors may be very similar to each other.

5.5 Cold Start Evaluation (RQ4)

We investigated the performance under different types of cold start to stimulate the real-world scenes:

- (1) New users join and invoke known services(denoted as **U**ser **C**old **S**tart).
- (2) New services join and known users invoke them(denoted as **Serv**ice **C**old **S**tart).
- (3) New users and services join, and new users invoke a new services (denoted as Both CS), which is an extreme cold start case of recommender system as only side information can be utilized.
- (4) All users and services are known, which is the baseline for comparison. In this experiment, the matrix density is fixed to 5%, and the proportion of cold start users/services is set to 10%.

TABLE 8
Performance Comparisons of in Cold Start Scenes

Type	RT		TP	
	MAE	RMSE	MAE	RMSE
User CS	0.407	1.303	13.601	44.250
Serv CS	0.433	1.319	18.092	56.677
Both CS	0.460	1.441	19.811	58.477
None CS	0.372	1.252	12.539	40.952

Through Table 8, we can summarize:

- When service recommender system encounters with cold start services/users, the performance drops.
- Serv CS can be more harmful to the prediction performance than User CS, maybe because QoS attributes are more sensitive to services than to users. A reasonable explanation for this phenomenon is that, to some extent, both response time and throughput are determined by the status of the server.
- For new users and new services QoS prediction, our model can only make predictions based on their known neighbors without their preference, thereby producing a high error. Although neighborhoods are similar, they are still incapable of replacing preference embedding.

In our work, the attention mechanism, as an efficient feature selection method, is used to assign weights on generated user/service-side representation and context representation. In the model's inference process, the weights assigned to different features change dynamically. To analyze the influence of three self-attention mechanisms from a macro perspective, we randomly choose 10000 records and obtain the average of the weights of the features. We test the attention mechanism under four circumstances (see subsection 5.5) to present how it changes with different inputs through heatmaps. In the following figures, the vertical axis represents the Query, while the horizontal axis represents the Key. Thus, each column/row expresses the degree of interest in other features when calculating a specific feature, thus representing the second-order interaction between features.

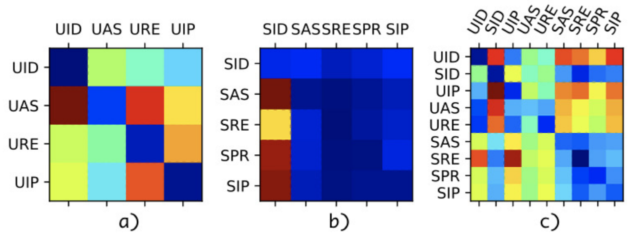


Fig. 11. Attention map in non cold start scene.

According to Figure 11, in the non-cold-start scene, user-side representation mainly relies on the UAS and the URE. However, service-side attention focuses on the SID column rather than the interactions between contexts, thus indicating that the factors are relatively independent. All factors are essential for SID, indicating that the network environments are critical for service representation. Although complex, the process mainly focuses on the interactions between the user-side and service-side contexts for the global attention map. These feature combinations indicate that the network path plays a role in the performance of invocation.

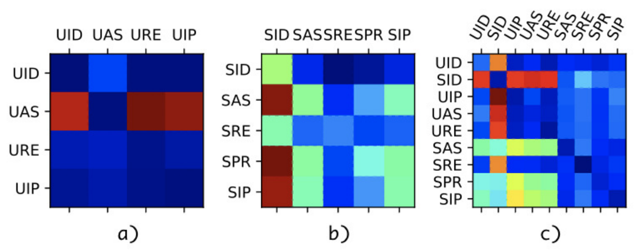


Fig. 12. Attention map when new users join.

According to Figure 12, when a user is regarded as an unknown one, user-side representation highly relies on their context, which comprises first-order neighbors. Compared to Figure 11c, Figure 12c shows that the model dynamically changes in focus and becomes more reliant on service-side factors. This finding suggests that service can determine user experience to a great extent.

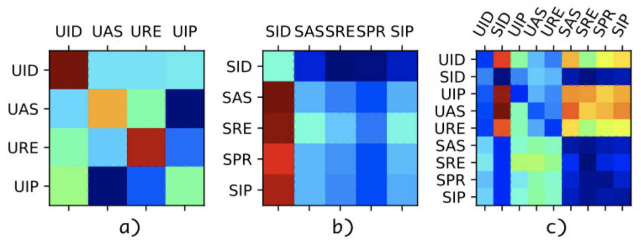


Fig. 13. Attention map when new services join.

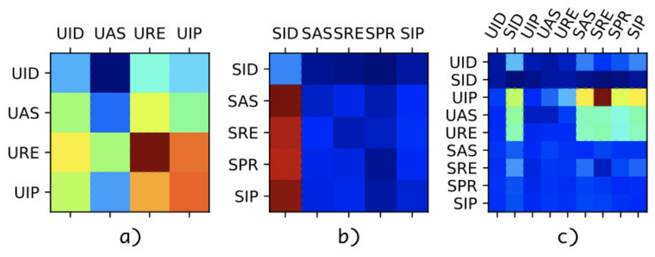


Fig. 14. Attention map when users and services are unknown.

The joint analysis of Figure 13 and Figure 14 shows that when the service is unknown, the attention map of the user side becomes complicated, indicating that the model attempts to preserve more information from the user side. In both cases, the model tries to pay more attention to features that can identify network paths.

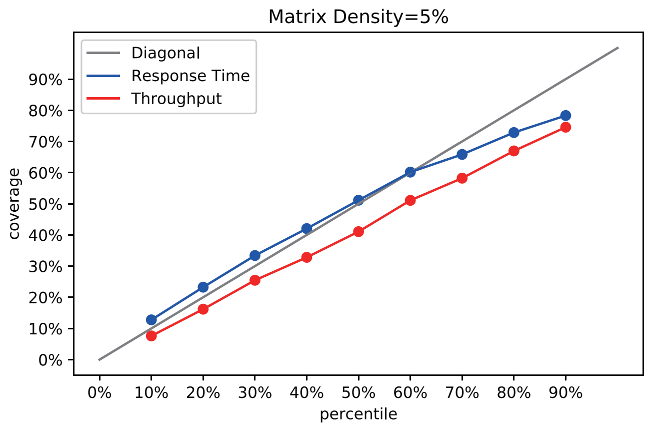


Fig. 15. Prediction Quantile Accuracy.

5.6 Probability Forecast Evaluation (RQ5)

5.6.1 Quantile Analysis

We evaluate how accurate our probability forecast model is, and the result is presented in Figure 15. For an ideally calibrated model, the coverage rate equals the percentile, corresponding to the diagonal line. Generally,, our model can generate more accurate predictions on response time than on throughput. In terms of throughput, the coverage rate, that is, the real value falling into the corresponding confident interval, falls behind the diagonal after 60%, while the response time is consistently lower than the diagonal. This result indicates different difficulty for prediction, the response time is harder to predict. A reasonable explanation is that response time may be sensitive to the server load and the underlying network within the same contexts, while throughput is sensitive to the server's status.

5.6.2 Uncertainty Analysis

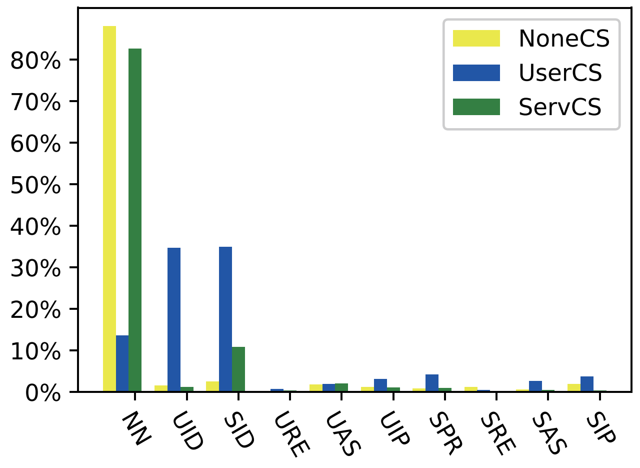


Fig. 16. Uncertainty contribution in three scenes.

We visualize the weights of uncertainty sources in both cold-start and non-cold-start scenes to analyze the source of uncertainty. According to Figure 16, in non-cold-start and service cold-start scenes, the uncertainty contributions are weak, except for NN, indicating that the uncertainty mainly comes from the neural networks and not from the embeddings. This phenomenon can be attributed to the insufficient neural network training due to data sparsity. When new users join, the uncertainty source mainly comes from UID and SID, maybe because the user's preference embedding contains specific information for user and service interaction. Without this, it is hard to predict user and service behaviors. NN contributes the most to the uncertainty for service cold start, followed by SID, while UID contributes the least. This phenomenon indicates that the user has a pivotal role in the entire system, and QoS prediction is highly personalized.

6 Conclusion

Under the DLaaS environment, providing credible and accurate QoS values for users is necessary to obtain optimal deep-learning services. However, previously proposed QoS prediction approaches often fall short of solving the cold-start problem and the uncertainty estimation and are highly vulnerable to adversarial attacks. On the basis of these challenges, we propose a novel QoS prediction approach called EEPrNN, which is robust against noise and secure against adversarial attacks. EEPrNN employs recursive residual embedding techniques to find neighbors inside relation graphs to mitigate the cold-start issue. In addition, an uncertainty estimation method is also proposed to demonstrate how confident the model is terms of prediction and how it can quickly locate the source of uncertainty. Moreover, we investigated the influence of noise and adversarial attacks on QoS prediction and propose adversarial training with uncertainty-aware loss. The experiments conducted on a large-scale distributed service dataset and the subsequent empirical analysis prove the proposed model's effectiveness.

The proposed work presents an initial attempt in mining structural information in QoS prediction tasks. Although we achieved good performance, the high time complexity may hinder the application of the proposed model. Thus, we intend to simplify the model and optimize its performance by carefully designing the graph operation in our future work. We also intend to utilize temporal features to form a spatio-temporal aware model to satisfy the online QoS prediction's dynamic nature. Focus will also be given to enhancing the scalability and meeting the industrial requirement of large-scale distributed service recommendations.

Acknowledgments

This research was financially supported by the National Natural Science Foundation of China (No.61702318 and No. 62072170), the Fundamental Research Funds for the Central Universities (531118010527), 2020 Li Ka Shing Foundation Cross-Disciplinary Research Grant (No.2020LKSFG08D) , and in part by 2019 Guangdong province special fund for science and technology ("major special projects + task list") project (No. 2019ST043).

References

[1] W. Liang, J. Long, K.-C. Li, J. Xu, N. Ma, and X. Lei, "A fast defogging image recognition algorithm based on bilateral hybrid filtering," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 0, no. ja. [Online]. Available: https://doi.org/10.1145/3391297

[2] W. Liang, W. Huang, J. Long, K. Zhang, and D. Zhang, "Deep reinforcement learning for resource protection and real-time detection in iot environment," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6392–6401, 2020.

[3] D. Bowen, P. Hao, W. Senzhang, B. Md. Zakirul Alam, W. Lihong, G. Qiran, L. Lin, and L. Jing, "Deep irregular convolutional residual lstm for urban traffic passenger flows prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 3, pp. 972–985, 2020.

[4] H. Wang, Y. Wu, G. Min, and W. Miao, "A graph neural network-based digital twin for network slicing management," *IEEE Transactions on Industrial Informatics*, vol. PP, pp. 1–1, 2020.

[5] W. Liang, S. Xie, D. Zhang, X. Li, and K. Li, "A mutual security authentication method for rfid-puf circuit based on deep learning," *ACM Transactions on Internet Technology*, pp. 1–20, 2020.

[6] Y. Wu, "Cloud-edge orchestration for the internet-of-things: Architecture and ai-powered data processing," *IEEE Internet of Things Journal*, pp. 1–1, 2020.

[7] W. Liang, L. Xiao, K. Zhang, M. Tang, D. He, and K. C. Li, "Data fusion approach for collaborative anomaly intrusion detection in blockchain-based systems," *IEEE Internet of Things Journal*, pp. 1–1, 2021.

[8] D. Tsesmetzis, I. Roussaki, and E. Sykas, "QoS-aware service evaluation and selection," *Eur. J. Oper. Res.*, vol. 191, no. 3, pp. 1101–1112, 2008.

[9] W. Liang, D. Zhang, X. Lei, M. Tang, K. Li, and A. Zomaya, "Circuit copyright blockchain: Blockchain-based homomorphic encryption for ip circuit protection," *IEEE Transactions on Emerging Topics in Computing*, pp. 1–1, 2020.

[10] H. Qiu, Q. Zheng, T. Zhang, M. Qiu, and J. Lu, "Towards secure and efficient deep learning inference in dependable iot systems," *IEEE Internet of Things Journal*, vol. PP, no. 99, pp. 1–1, 2020.

[11] H. Qiu, T. Dong, T. Zhang, J. Lu, and M. Qiu, "Adversarial attacks against network intrusion detection in iot systems," *IEEE Internet of Things Journal*, vol. PP, no. 99, pp. 1–1, 2020.

[12] M. Qiu and H. Qiu, "Review on image processing based adversarial example defenses in computer vision," in *2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*, May 2020, pp. 94–99.

[13] X. Liu, J. You, Y. Wu, T. Li, L. Li, Z. Zhang, and J. Ge, "Attention-based bidirectional gru networks for efficient https traffic classification," *Information Sciences*, vol. 541, pp. 297–315, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S002002552030445X

[14] Y. Zeng, H. Qiu, G. Memmi, and M. Qiu, "A data augmentation-based defense method against adversarial attacks in neural networks," in *Algorithms and Architectures for Parallel Processing*, M. Qiu, Ed. Cham: Springer International Publishing, 2020, pp. 274–289.

[15] Y. Li, Y. Song, L. Jia, S. Gao, and M. Qiu, "Intelligent fault diagnosis by fusing domain adversarial training and maximum mean discrepancy via ensemble learning," *IEEE Transactions on Industrial Informatics*, vol. PP, no. 99, pp. 1–1, 2020.

[16] 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)*, July 2007, pp. 439–446.

[17] Z. Zheng, M. Hao, 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, 2011.

[18] H. Sun, Z. Zheng, J. Chen, and M. R. Lyu, "Personalized web service recommendation via normal recovery collaborative filtering," *IEEE Transactions on Services Computing*, vol. 6, no. 4, pp. 573–579, 2013.

[19] K. K. Fletcher and X. F. Liu, "A collaborative filtering method for personalized preference-based service recommendation," in *2015 IEEE International Conference on Web Services*, 2015, pp. 400–407.

[20] M. Tang, T. Zhang, J. Liu, and J. Chen, "Cloud service qos prediction via exploiting collaborative filtering and location-based data smoothing," *Concurrency & Computation Practice & Experience*, vol. 27, no. 18, pp. 5826–5839, 2016.

[21] X. Chen, X. Liu, Z. Huang, and H. Sun, "Regionknn: A scalable hybrid collaborative filtering algorithm for personalized web service recommendation," in *2010 IEEE international conference on web services*. IEEE, 2010, pp. 9–16.

[22] M. Tang, Y. Jiang, J. Liu, and X. Liu, "Location-aware collaborative filtering for qos-based service recommendation," in *2012 IEEE 19th international conference on web services*. IEEE, 2012, pp. 202–209.

[23] Z. Zheng and M. R. Lyu, "Personalized reliability prediction of web services," *Acm Transactions on Software Engineering Methodology*, vol. 22, no. 2, pp. 12.1–12.25, 2013.

[24] P. He, J. Zhu, Z. Zheng, J. Xu, and M. R. Lyu, "Location-based hierarchical matrix factorization for web service recommendation," in *2014 IEEE international conference on web services*. IEEE, 2014, pp. 297–304.

[25] D. Yu, Y. Liu, Y. Xu, and Y. Yin, "Personalized qos prediction for web services using latent factor models," in *2014 IEEE International Conference on Services Computing*, June 2014, pp. 107–114.

[26] Y. Xu, J. Yin, S. Deng, N. N. Xiong, and J. Huang, "Context-aware qos prediction for web service recommendation and selection," *Expert Systems with Applications*, vol. 53, pp. 75–86, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417416000208

[27] H. Wu, K. Yue, B. Li, B. Zhang, and C.-H. Hsu, "Collaborative qos prediction with context-sensitive matrix factorization," *Future Generation Computer Systems*, vol. 82, pp. 669–678, 2018.

[28] J. Xu, Z. Zheng, and M. R. Lyu, "Web service personalized quality of service prediction via reputation-based matrix factorization," *IEEE Transactions on Reliability*, vol. 65, no. 1, pp. 28–37, 2016.

[29] 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.

[30] Y. Xu, J. Yin, Z. Wu, D. He, and Y. Tan, "Reliability prediction for service oriented system via matrix factorization in a collaborative way," in *2014 IEEE 7th International Conference on Service-Oriented Computing and Applications*, 2014, pp. 125–130.

[31] Y. Zhang, K. Wang, Q. He, F. Chen, S. Deng, Z. Zheng, and Y. Yang, "Covering-based web service quality prediction via neighborhood-aware matrix factorization," *IEEE Transactions on Services Computing*, pp. 1–1, 2019.

[32] H. Wu, Z. Zhang, J. Luo, K. Yue, and C. Hsu, "Multiple attributes qos prediction via deep neural model with contexts," *IEEE Transactions on Services Computing*, pp. 1–1, 2018.

[33] R. Xiong, J. Wang, Z. Li, B. Li, and P. C. Hung, "Personalized LSTM Based Matrix Factorization for Online QoS Prediction," *Proc. - 2018 IEEE Int. Conf. Web Serv. ICWS 2018 - Part 2018 IEEE World Congr. Serv.*, vol. 63, pp. 34–41, 2018.

[34] Q. Zhou, H. Wu, K. Yue, and C. H. Hsu, "Spatio-temporal context-aware collaborative QoS prediction," *Futur. Gener. Comput. Syst.*, vol. 100, pp. 46–57, 2019. [Online]. Available: https://doi.org/10.1016/j.future.2019.05.024

[35] J. Xu, L. Xiao, Y. Li, M. Huang, Z. Zhuang, T.-H. Weng, and W. Liang, "Nfmf: neural fusion matrix factorisation for qos prediction in service selection," *Connection Science*, pp. 1–16, 2021.

[36] Z. Chen, L. Shen, F. Li, and D. You, "Your neighbors alleviate cold-start: On geographical neighborhood influence to collaborative web service qos prediction," *Knowledge-Based Systems*, vol. 138, no. DEC.15, pp. 188–201, 2017.

[37] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, R. Anil, Z. Haque, L. Hong, V. Jain, X. Liu, and H. Shah, "Wide &amp; deep learning for recommender systems," in *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, ser. DLRS 2016. New York, NY, USA: Association for Computing Machinery, 2016, p. 7–10. [Online]. Available: https://doi.org/10.1145/2988450.2988454

[38] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2017.

[39] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds., 2017, pp. 5998–6008. [Online]. Available: https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[40] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence*

and Statistics, ser. Proceedings of Machine Learning Research, G. Gordon, D. Dunson, and M. Dudík, Eds., vol. 15. Fort Lauderdale, FL, USA: PMLR, 11–13 Apr 2011, pp. 315–323. [Online]. Available: http://proceedings.mlr.press/v15/glorot11a.html

[41] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds., 2017, pp. 5574–5584. [Online]. Available: https://proceedings.neurips.cc/paper/2017/hash/2650d6089a6d640c5e85b2b88265dc2b-Abstract.html

[42] T. Miyato, A. M. Dai, and I. Goodfellow, "Adversarial training methods for semi-supervised text classification," 2017.

[43] 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.

[44] G. Linden, B. Smith, and J. York, "Amazon. com recommendations: Item-to-item collaborative filtering," *IEEE Internet computing*, vol. 7, no. 1, pp. 76–80, 2003.

[45] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "Deepfm: A factorization-machine based neural network for CTR prediction," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, C. Sierra, Ed. ijcai.org, 2017, pp. 1725–1731. [Online]. Available: https://doi.org/10.24963/ijcai.2017/239

[46] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. Chua, "Neural collaborative filtering," in *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, R. Barrett, R. Cummings, E. Agichtein, and E. Gabrilovich, Eds. ACM, 2017, pp. 173–182. [Online]. Available: https://doi.org/10.1145/3038912.3052569

[47] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

Wei Liang received the Ph.D. degree in computer science and technology from Hunan University, China, in 2013. He was a Postdoctoral Scholar with Lehigh University, Bethlehem, P A, USA, during 2014–2016. He is currently an Associate Professor with the College of Computer Science and Electronic Engineering, Hunan University. He has authored or coauthored more than 110 journal/conference papers such as IEEE T ransactions on Industrial Informatics, IEEE T ransactions on Emerging T opics in Computing, IEEE T ransactions on Computational Biology and Bioinformatics, and IEEE Internet of Things Journal. His research interests include blockchain security technology, networks security protection, embedded system and hardware IP protection, fog computing, and security management in wireless sensor networks (WSN).

Yuhui Li is a graduate student of Hunan University. His research interests include service computing, distributed computing and blockchain security.

Jianlong Xu received a Ph.D.degree from the South China University of Technology in 2013 and then studied as a postdoctoral fellow at Shenzhen Research Institute, the Chinese University of Hong Kong. He is currently a lecturer in the College of Engineering, Shantou University, China. His research interests include service computing, Internet of Things, and distributed computing.

Zheng Qin received his PhD degree in computer science from Chongqing University, China, in 2001. He was a visiting scholar at Michigan State University from 2010 to 2011. He is a full professor and vice dean in the College of Computer Science and Electronic Engineering, Hunan University, China. He is the director of the Hunan Key Laboratory of Big Data Research and Application, and the vice director of the Hunan Engineering Laboratory of Authentication and Data Security, Changsha, China. His research interests include blockchain, data science, information security, and software engineering.

Dafang Zhang received the Ph.D. degree in applied mathematics from Hunan University, Changsha, China, in 1997.,He is currently a Professor with the College of Computer Science and Electronic Engineering, Hunan University. He was a Visiting Fellow with Regina University, Regina, SK, Canada, during 2002–2003, and a Senior Visiting Fellow with Michigan State University, East Lansing, MI, USA, in 2013. He has authored or coauthored more than 230 journal/conference papers and principal investigator (PI) for more than 30 large scale scientific projects. His research interests include dependable systems/networks, network security, network measurement, hardware security, and IP protection.

Kuan-Ching Li is currently a Distinguished Professor of the Department of Computer Science and Information Engr. (CSIE) at Providence University, Taiwan. He received the Licenciatura in Mathematics, and MS and Ph.D. degrees in Electrical Engineering from the University of Sao Paulo (USP), Brazil, in 1994, 1996 and 2001, respectively. Prior to joining PU in 2003, he was a post-doc scholar at the University of California-Irvine (UCI), USA. He is a recipient of awards and funding support from a number of agencies and high-tech companies, as also received chair and distinguished professorships from universities in China and other countries. He has been actively involved in many major conferences and workshops in program/general/steering conference chairman positions and as a program committee member and has organized numerous conferences related to high-performance computing and computational science and engineering. His research interests include GPU/manycore computing, Big Data, and cloud. Besides publication of numerous research papers and articles, he is co-author/co-editor of several technical professional books published by CRC Press, Springer, McGraw-Hill and IGI Global. He is a senior member of the IEEE, a life member of the TACC, a member of the AAAS, and a fellow of the IET.