IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON AI-DRIVEN BIG DATA PROCESSING: THEORY, METHODOLOGY, AND APPLICATIONS

Received May 17, 2019, accepted July 12, 2019, date of publication August 1, 2019, date of current version August 19, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2932047

Which Channel to Ask My Question?: Personalized Customer Service Request Stream Routing Using Deep Reinforcement Learning

ZINING LIU[1], CHONG LONG[2], XIAOLU LU[3], ZEHONG HU[4], JIE ZHANG[5], AND YAFANG WANG[2]

[1]School of Software, Shandong University, Jinan 250101, China
[2]Ant Financial Services Group, Z Space, Hangzhou 310099, China
[3]School of Science, RMIT University, Melbourne, VIC 3001, Australia
[4]Alibaba Group, Hangzhou 311121, China
[5]School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798

Corresponding author: Yafang Wang (yafang.wyf@antfin.com)

This work was sponsored by the National Natural Science Foundation of China under Grant 61503217.

ABSTRACT Customer services are critical to all companies, as they may directly connect to the brand reputation. Due to a great number of customers, e-commerce companies often employ multiple communication channels to answer customers' questions, for example, Chatbot and Hotline. On one hand, each channel has limited capacity to respond to customers' requests; on the other hand, customers have different preferences over these channels. The current production systems are mainly built based on business rules that merely consider the tradeoffs between the resources and customers' satisfaction. To achieve the optimal tradeoff between the resources and customers' satisfaction, we propose a new framework based on deep reinforcement learning that directly takes both resources and user model into account. In addition to the framework, we also propose a new deep-reinforcement-learning-based routing method—double dueling deep Q-learning with prioritized experience replay (PER-DoDDQN). We evaluate our proposed framework and method using both synthetic and a real customer service log data from a large financial technology company. We show that our proposed deep-reinforcement-learning-based framework is superior to the existing production system. Moreover, we also show that our proposed PER-DoDDQN is better than all other deep Q-learning variants in practice, which provides a more optimal routing plan. These observations suggest that our proposed method can seek the tradeoff, where both channel resources and customers' satisfaction are optimal.

INDEX TERMS Deep reinforcement learning, personalized customer service, time-series data processing.

I. INTRODUCTION

The quality of customer service is crucial to a company's reputation: its quality is measured by how quickly a company responds to customers' requests and how satisfied customers are when seeking help. Obviously, the satisfaction of a customer can merely be measured using the problem-solving quality alone. In practice, a company also adopts customers' queuing time as one of the indicators for measuring satisfaction. For the purpose of shortening customers' waiting time, major companies often provided multiple communication channels for customers to choose, for example, mobile App, web-based message and the traditional hotline. Different communication channels have their own limited quota for responding to customers' requests, and also require different cognitive loads. Customers are impatient, especially when they have a request to be resolved, but they want to express their need using the least effort. The most natural communication channel is the hotline, where a customer service representative answers the call and help customers solve their problems. As a consequence, most customers prefer the hotline channel and it often leads to a long waiting time, especially during peak hours. Therefore, the obvious problem occurred is the imbalanced workloads of each channel.

The associate editor coordinating the review of this manuscript and approving it for publication was Zhanyu Ma.

How can we solve this problem to make customers satisfied, and how can we help companies optimally allocate limited resources are the key questions. There are a lot of other factors that impact customer acceptance rates, e.g., the design of user interface, the privacy issues of the question, etc. All of them can affect user's choice. This paper focus only on the routing problem at first: redistributing the right customers to non-hotline, and reducing the burden of a round of dialogue for unsuitable customers. In the experiment result section we can see the significant improvements after applying our routing model. We will consider other factors in our future work.

The customer service system produces about 50000 data every day in the e-commerce company. Existing systems adopted by many companies are rule-based to deal with big data issues, which are easy to implement as most of the business rules are already defined. However, they are not flexible and often end up with a large set of complicated rules. Also, using a rule-based system doesn't aim to seek a balance between optimality of resource allocation and customer satisfaction, but purely for the low-cost implementation purpose. We carefully explore and evaluate these drawbacks in this paper, and show that although rule-based systems may be an efficient way to handle customer requests stream, it is far from making both customers and the company satisfactory.

Motivated by the recent success in deep reinforcement learning [1], we propose a deep-reinforcement-learning based framework to perform the customer service requests routing. The proposed framework is in sharp contrast to the rule-based system, as our framework directly captures: (i) customers' preference and (ii) each channels future traffic. Customers have different preferences over different communication channels, and a simple routing method may result in low satisfaction. For example, if the problem is not so urgent, a student may be willing to leave messages on the offline service desk if the current hotline is busy; someone who is doubtful about the chatbot may be stick to the hotline channel, regardless the waiting time. Recommending customers their preferrable surrogate channel is essential as it affects the overall customers' experience. Another key to solving the allocation problem is to be able to predict the channel's traffic over the next time window. The requests data stream often comes in a high speed, especially in peak hours and a mass corporation, a system can fail miserably if it ignores the time-series feature of the request stream. For example, if our prediction suggests that the hotline is not busy in the next time window we can then let customers be served using the hotline if they prefer to; similarly if our prediction shows the hotline's capacity will be exceeded in the next time window, we may try to route customers' request to alternative channels if possible. Hence, the framework we proposed in this paper is to seek tradeoffs among the channel capacity, user preference and the predicted traffic of the current channel.

Our framework consists of three major components: (i) customer profiling module, (ii) flow forecasting module and (iii) an agent that suggests a customer to the most suitable channel. The first component is used to model the *environment* in our reinforcement learning framework and the last one is the *agent*. We make use of users' attribute to infer their preference over different channels, so that our system can perform personalized routing according to each individual's preference. As we mentioned, the customer requests stream is a time-series data and it is crucial if we can predict the volume of each channel of a certain time point. In this paper, we evaluate several techniques for time-series prediction and give a recommended approach. Our framework is built based on deep reinforcement learning method. Despite of using existed deep Q-learning and its variants, we also propose the **d**ouble **d**ueling DQN with **p**rioritized **e**xperience **r**eplay (PER-DoDDQN) method, which combines the strength from both double DQN (DoDQN) [2] and dueling DQN (DDQN) [3].

We evaluate our proposed framework and method using synthetic and real customer service data collected from a large financial technology company. We propose several evaluation metrics and evaluate our solution effectiveness from three aspects: (i) channel congestion, (ii) customers' acceptance rate of the recommended alternatives and (iii) the channel utilization. Experiments demonstrate the substantial routing effectiveness gains can be achieved – both channel resources and customers satisfaction reach an optimal state – using our proposed new routing framework is more effective than the existing system, and our proposed PER-DoDDQN is more effective than DQN variants.

Our contribution can be concluded in three-fold:

1) We model the customer service requests routing problem using deep reinforcement learning, considering both channel resources and customers' satisfaction.
2) We propose the double dueling deep Q-learning with prioritized experience replay method to solve the routing problem, which achieves that better performance than its counterparts in practice.
3) We perform an extensive evaluation using both real and synthetic data to demonstrate the practical value of our proposed methods.

The rest of this paper is organized as follows. The background and our problem definition will be introduced in Section II. After that, we will present our system, including the user model and the routing approach based on reinforcement learning in Section III, together with experimental results and analysis in Section IV. Then related technical work about reinforcement learning and traffic forecasting will be provided in Section V. Finally, the conclusions and future work will be provided in Section VI.

II. BACKGROUND

In this section, we provide the background of routing application in detail, including application scenario and the baseline system.

A. APPLICATION BACKGROUND

Modern business often provides several communication channels for customers convenience, ranging from traditional

call center service, online chatbot, to mobile APP. Besides the traditional call center (or hotline) services, other channels may be served by a mixture of automatic chatbot and human customer representatives. Apparently, the hotline service has the least capacity to deal with customers' request, while it is the most preferred one by a majority of the customers, due to its low communication cost. Other channels face a similar trade-off. For example, it is easier for users to access mobile Apps for issuing customer service requests, but it requests a lot of cognitive effort for users to describe their information need precisely and the interaction is often less efficient compared to using the hotline service. While most of the time the mobile APP may not be preferred, for simple requests, customers' may resolve them more efficient by selecting or browsing pre-selected questions. Similarly, for the online interface, customers are required to input their requests precisely, which may prevent customers from using this channel. However, compared to the mobile APP and the hotline service, the web interface provides a way for customers to interact with each other, which may help to reduce the workload of customer service representatives to some extent. When a large number of customers are making requests to one channel, the channel's capacity may be exceeded, and customers need to wait a long time. In this case, most customer services platforms will re-direct users to other channels, however, users have a large chance to reject such recommendation.

According to our application scenario, we need to propose a system that seeks tradeoffs between the channel capacity and customers' satisfaction, where the satisfaction is measured using the acceptance rate and waiting time. Let n be the total number of communication channels, each of which has a capacity of $c_i (1 \leq i \leq n)$. Assume all customers will use the customer service platforms. Each customer has a user profile $\mathbf{u} = \langle f_0, f_1, \ldots, f_k \rangle$, where f_i represents a value of i-th attribute. We consider at time t, each channel will also have a request flow, referred to as e_t. The utilization rate v_i to represent the percentage of the free resources of the current channel, relative to the channel capacity. Formally, the routing problem is:

Problem 1: Customer Service Routing Problem. Given a user and the request, and the current capacity of each channel c_i, the problem is to recommend user a communication channel i, such that: (i) the overall users' satisfaction rate is maximized; (ii) the overall utilization rates of communication channels are maximized.

B. BASELINE CUSTOMER SERVICE ROUTING SYSTEM

To compared with our proposed RL-based algorithm, we consider a real customer service routing system as an example, shown in Figure 1. This is a real product system adopted by a large financial technology company, which is designed based on business rules. The system considers two communication channels: self-service and hotline, where self-service let customers solve simple requests and the hotline is the traditional call-based customer service.

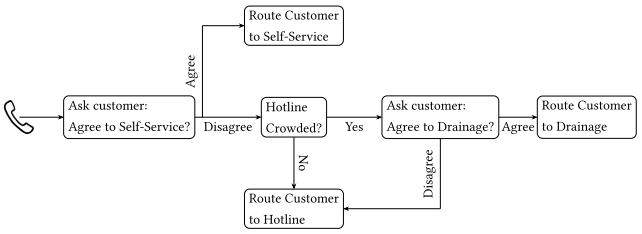


FIGURE 1. Existing customer service request routing system, which is designed based on business rules.

The system works as follows: when a customer calls in, the system always asks whether the customer would like to use a self-service, as the hotline may be congested and the customer's question may be simple. If the customer agrees, he or she will be routed to the self-service channel, otherwise, the system will then make a decision depending on the current queue length of the hotline channel. A customer will always be re-directed to hotline service when there are no people waiting in the queue. However, when the service channel is busy and there is a queue, the system will randomly select a set of customers to ask if they are willing to switch to the drainage channel. The exact number of customers who will be chosen to ask for a second preference depends on the current length of the hotline queue – if the queue is long, then more customers will be selected than a short queue.

From the description, we can observe that, existing rule-based customer service routing system strictly stick to the rule that always respects customers' choices, regardless of the current channel state. Customers can choose to stay wait until the hotline is available, or switch to other provided channels. We analyze the real data collected from the production system, and find that only 20.1% customers agreed to accept the self-service channel in the current customer service routing system. A detailed analysis and comparison will be provided in Section IV.

Obviously, existing systems only provide a simple interface for users to express their interest, rather than performing an effective routing operation. Letting customers choose their preferred channel almost always lead to the congestion in the hotline channel, as a result, customers will need to wait for a long time until be served; the other channels will be wasted in this situation. In this paper, we want to mitigate this extremely imbalanced resources utilization, so we propose a new routing framework that aims at finding optimal plans for both the company and customers.

III. PERSONALIZED REQUESTS ROUTING FRAMEWORK

As we described in Definition 1, the customer request routing system needs to find an optimal state between channel resources and customers' satisfaction. We first describe our proposed routing framework and then our proposed **d**ouble **d**ueling DQN with **p**rioritized **e**xperience **r**eplay (PER-DoDDQN) model.

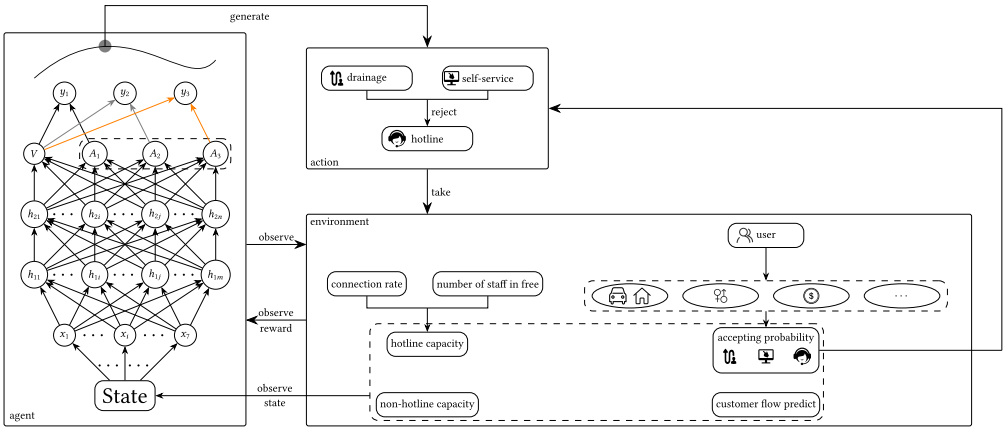


FIGURE 2. Overview of the proposed routing framework.

A. FRAMEWORK DESCRIPTION

We show an overview of our proposed customer service routing system in Figure 2, which is based on deep reinforcement learning. In the proposed framework, there are two important components: environment and agent.

The environment is shown in the right bottom pane of Figure 2, which consists of a channel model (the left-hand side in the pane) and a user model (the right-hand side in the pane). The channel model is used to estimate the volume of request flow, as a part of the state that the agent can observe. If we estimate that a channel will have a large incoming request in the next time window, our routing framework should avoid allocating more customers to that channel. Since the customer service request show a strong temporal correlation, we formulate the problem of channel requests predication as a time-series prediction problem. Moreover, as in peak hours, the data stream may come in with a high volume and velocity, so pre-estimating the number of the incoming requests helps the system to make a better routing decision. All flow forecasting algorithm described in Section V-B can potentially be applied to this problem. After a careful evaluation, we choose the `XGBoost` as it reveals the best performance in the real scenario. We will describe the detailed evaluation in Section IV.

Another important element of the environment is the user model, describing customers' preference over different communication channels. Intuitively, customers preferences vary on all possible communication channels. If the recommended channel is not preferred by a customer, then it is highly likely the customer will reject the channel routing, which leads to a longer waiting queue of a popular channel, for example, the hotline channel. We build the user model using a neural networks, which has three hidden layers with size 128, 256 and 128 respectively. For each user, we input their attributes into the user model and output their acceptance probability of each communication channel. All routing models in this paper use the same architecture shown in Figure 3.

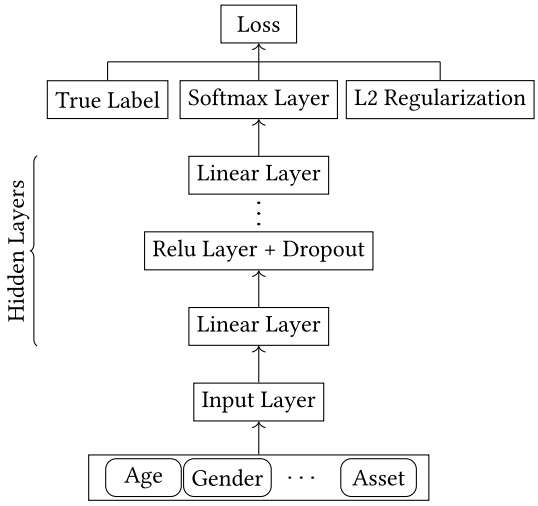


FIGURE 3. The user model.

The agent which takes actions based on observed states is shown in the leftmost pane in Figure 2. It takes the state observed from the environment as the input and then a neural network is used to recommend a channel to the customer. If a customer accepts our recommendation, then a successful routing is performed, which indicates we may reduce the workload of some bottleneck channels to some extent and also the user is willing to accept a channel switch. Note that, in our framework, all components can be configured with the most suitable algorithm, and we will show an empirical comparison among different configurations in Section IV.

B. THE ROUTING MODEL

The overall routing model is based on deep reinforcement learning, and more specifically, the DQN variants. They approximate $q_\theta(s, a)$, which means the value of "state"(s) and "action"(a) pairs with the deep learning. The algorithms optimize the policy according to the TD-error:

$r + \gamma \max_{a'} q(s', a') - q(s, a)$ and are updated as the following equation

$$q(s, a) = q(s, a) + \alpha(r + \gamma \max_{a'} q(s', a') - q(s, a)). \quad (1)$$

where r represents the immediate reward, and s' and a' represents the next state and the next possible actions. All DQN variants are based on the above ideas and will be described in detail in Section V.

We start to describe how we formulate the three key elements (action, state and reward function) in any reinforcement learning model, and then we describe our proposed PER-DoDDQN in Algorithm 1.

1) ACTION

The system aims to learn a policy that can determine which channel should be recommended to users, so the action here is to select a channel among n candidates, where n is determined by the total number of channels in the real application. We represent the action as a, $(a \in \{1, 2, 3, \ldots, n\})$.

2) STATE

A state is expected to express the customer's channel preference, and channel feasibility of handling furthermore requests. So the state includes: the customers' preferences over different channels $\mathbf{u}$, the channel capacity $\mathbf{c}$ and the channel's future request flow traffic $\hat{\mathbf{e}}_t$. The entire state is represented as $\mathbf{s} = \langle \mathbf{u}, \hat{\mathbf{e}}_t, \mathbf{c} \rangle$.

3) REWARD FUNCTION

There are duel reward aspects we need to consider: from the customers' perspective and from the channel capacity, based on which the reward is:

$$R = g_{a,t} - \lambda_1 \cdot ReLu\left(-\min(\mathbf{c}_t - \lambda_3 \cdot \hat{\mathbf{e}}_{t+1})\right) - \lambda_2 \cdot (ReLu(\min(\mathbf{c}_t - \lambda_3 \cdot \hat{\mathbf{e}}_{t+1})))^2, \quad (2)$$

where $\lambda_1 \gg \lambda_2$ and λ_3 is a smoothing parameter, making reward put more emphasis on the current capacity.

We use $g_{a,t}$ be the reward that is given to the user's current action at time t, indicating if we make a good recommendation from a customer's perspective. When a user accepts the recommendation, $g_{a,t} = 1$; otherwise $g_{a,t} = -1$. Therefore, $g_{a,t}$ can guide the system learning towards the channel that a customer prefers.

The second and third terms in Equation (2) is designed according to bottleneck channels, which is the channel has the least remaining capacity and favored by the majority, for example, the hotline channel. As the bottleneck channel may easily be congested and intuitively, customers' satisfaction will also be largely affected by the waiting time in the queue – even more important than the preference. To guide the system learn to avoid such case, we also add an extra quadratic punishment item to our reward function.

Algorithm 1 Personalized Customer Service Routing (**PER-DoDDQN**)

Input: Customer service flow, Channel capacity $C \leftarrow C_0 = (c_1, c_2, \ldots, c_n)$, User attributes, period K, *done_num*
Build $\mathbf{u}(u_1, u_2, \ldots, u_n)$ ▷ User model, and bottleneck channel's probability is 1.
$D \leftarrow \{\}, \theta \leftarrow 0, \bar{\theta} \leftarrow 0$ ▷ Initialize **PER** buffer D, and parameters
$A^* \leftarrow \{\}$, terminal_state$\leftarrow$ False ▷ Keep tracking customers' actions
$T \leftarrow$ *length*(*training dataset*)
for $t \leftarrow 1$ **to** T **do**
  $c_{id} \leftarrow$ current customer's id ▷ Get current customers' id
  $\mathbf{s}_t \leftarrow \langle \mathbf{u}, \hat{e}, C \rangle, a_t \leftarrow -1$
  if $t > 1$ **then**
    store transition$(\mathbf{s}_{t-1}, a_{t-1}, R_t, s_t)$ in $\mathcal{D}$ with weight and update $\mathcal{D}$ [4]
  end if
  select action $\mathbf{a}_t$ according to ϵ-greedy [5], following policy π_θ [2] [3], let $g_{a,t} \leftarrow 1$
  if customer reject the selected action **then**
    reassign the customer to bottleneck channel, $g_{a,t} \leftarrow -1$.
  end if
  $A^*[c_{id}] \leftarrow a_t, C[a_t] \leftarrow C[a_t] - 1$ ▷ Reduce channel capacity
  Compute the immediate reward R_{t+1} using Equation 2
  if a customer c_i finishes request **then**
    $C[A^*[c_i]] \leftarrow C[A^*[c_i]] + 1$
  end if
  if $\min(C) <$ *done_num* **then**
    s_t is terminal state
    reset the channel capacity:$C \leftarrow C_0$
  end if
  For the transition $(\mathbf{s}_{t-1}, a_{t-1}, R_t, \mathbf{s}_t)$ sampled from $\mathcal{D}$ according to [4], compute loss :

$$\mathcal{L} = \begin{cases} (R_t - q_\theta(\mathbf{s}_{t-1}, a_{t-1}))^2, \text{if } \mathbf{s}_t \text{ is terminal state} \\ (R_t + \gamma q_{\bar{\theta}}(\mathbf{s}_t, a_t) - q_\theta(s_{t-1}, a_{t-1}))^2, \text{otherwise} \end{cases} \quad (3)$$

  Where q_θ is computed based on Eq 16
  Minimize loss by gradient descent
  $\bar{\theta} \leftarrow \theta$ if $t \bmod K \equiv 0$
end for

4) PERSONALIZED CUSTOMER SERVICE ROUTING ALGORITHM

We describe the work flow using our proposed PER-DoDDQN in Algorithm 1. The system takes channel capacity, user profile and the request flow series as input, output a channel recommendation for each customer. As our focus is neither customer profiling nor flow forecasting, we simplify the two module description in our presentation.

TABLE 1. Parameter values obtained on the training set. Note we set the memory size to 2,000, and the batch size is 320. For all models, we early terminate the current episode when the hotline channel has a congestion of 100 requests.

Parameter	PER-DoDDQN				DoDDQN	DQN	DoDQN	DDQN
	full	e	u	noter				
λ_1	0.900	0.800	0.500	0.900	0.900	0.800	0.900	0.900
λ_2	0.015	0.020	0.015	0.015	0.015	0.015	0.020	0.015
λ_3	0.300	–	0.300	0.300	0.300	0.300	0.300	0.300
γ	0.500	0.300	0.700	0.500	0.500	0.700	0.500	0.500

Note that, the choice of algorithm used to solve the problem should be made according to the real application data. For incoming request stream, we first obtain the customer id and current channel capacity at the moment. We then initialize the state at the current time point in Line 8. If this is not the starting point and we've already made some actions, we store our previous action and current state into the replay buffer $\mathcal{D}$, in Line 10. We then select the current optimal action for the current customer and update the channel capacity from Line 16 to Line 20. Note that, a customer can choose to reject and accept our suggestion, and we need to update $g_{a,t}$ accordingly. If a customer rejects our suggestion, we always put them into the bottleneck channel, which definitely will be accepted. From Line 25 to Line 26, we describe the loss function used in our proposed PER-DoDDQN method, in which it combines both DQN and DDQN. As we've mentioned, we adopt prioritized experience replay, so the mini-batch sampled from replay buffer is based on weights computed in Equation 17. Note that when the environment reaches the terminal state, the TD-error is liable to suddenly increase. Therefore, we use prioritized experience replay so that the transition with the terminal state can be trained more times, speeding up the model convergence.

IV. EXPERIMENTS

We describe experimental evaluation in this section, including results on both synthetic and real datasets.

A. EXPERIMENTAL SETUP

1) REAL-WORLD DATASET

Our real data is sampled from June 15, 2018 to October 1, 2018 from customer service log provided by a big financial technology company. In total, there are 4898143 customer requests when congestion happens. For each customer, we consider seven features, which consists of gender, age, residential province, household, car, assets and credit limit. These features are preprocessed, and mapped to integer values if the original feature is categorical.

2) SYNTHETIC DATASET

For improving the robustness of RL model, we simulate data based on the real-world Dataset. Firstly, we extract 10 days data from the real-world dataset, then we simulate the length of these sessions sampled from the uniform distribution, $U(1, 6)$ minutes. Then we synthesize the probability of customer with R package SynthPop [6], with the cart method. Finally, we counted customer flow and added Gaussian noises to the flow data as the customer flow prediction data. We get 487,354 simulator customer service data from 15 June 2018 to 24 June 2018. We use 50,000 data from simulator dataset as the testing dataset.

It must be noted that the immediate reward according to Equation 2 is considered as the labels in supervised machine learning because there are no real labels in the synthetic dataset and real-world dataset.

3) PARAMETER SETTINGS

As there is a high percentage of noises in the real dataset, we train all models using the synthetic dataset, and apply the optimal parameter settings to the real dataset. For each configuration and the dataset, we list the parameter settings in Table 1.

We select these parameters of RL models when the parameters fulfill these two conditions: (i) the loss of agent converging and (ii) the reward increment when the action is selected by the agent. These models converge with about 9000 iterations in simulator datasets, so the model is learned after the experience replay is updated 50 times.

4) BASELINES AND CONFIGURATIONS

In order to empirically show the effectiveness of our system, we consider the following variants: (i) deep Q-Learning model variants and (ii) state variants. So, in addition to our proposed system, double dueling DQN with prioritized replay (PER-DoDDQN), we consider the nature DQN (DQN), the dueling DQN (DDQN), a plain double DQN model (DoDQN) and the double dueling DQN (DoDDQN) without prioritized replay. In order to show the impact of our channel model, customer flow prediction model and terminal state, we also show results of which the state is $\langle \mathbf{u}, \mathbf{c} \rangle$, $\langle \mathbf{c}, \hat{\mathbf{e}} \rangle$ and removing the terminal state, which we use PER-DoDDQN-e, PER-DoDDQN-u and PER-DoDDQN-noter to represent, respectively. All of our algorithms are implemented using `tensorflow`.

5) TRADITIONAL MACHINE LEARNING ALGORITHMS

In this paper, we also provide experiments by comparing traditional machine learning algorithms (KNN [7], CNNs [8], [9] and SVM [7]). In this paper, the Neural Network consists of two convolutional layers and two fully

connected layers. LSTM cannot be applied in this task as the single message($< \mathbf{u}, \mathbf{c}, \hat{\mathbf{e}} >$) has no sequence information.

6) EVALUATION METRICS

To detailly evaluate these models in simulator dataset, We designed six metrics. Where N represents the total number of consultations in the test set in the next six metrics. *Congestion Rate* (CCR) means the percentage of data when congestion happens;

$$CCR = \frac{\sum_{i=1}^{N} \vec{1}_{C<0}}{N}, \tag{4}$$

where $C < 0$ means that there are people waiting for service.$\vec{1}_{C<0}$ represents whether the congestion occurs when people calling. If there are customers in the waiting queue, $\vec{1}_{C<0} = 1$, otherwise, $\vec{1}_{C<0} = 0$.

Average Congestion Level (AC) means the average degree of congestion in simulator dataset;

$$AC = \frac{\sum_{i=1}^{N} Relu(-\vec{\mathbf{C}})}{N}, \tag{5}$$

where $Relu(-\vec{\mathbf{C}})$, $\vec{\mathbf{C}} = \{C_i\}$ represents the length of the waiting queue of channel i. $C_i < 0$ means there number of customers in the waiting queue of channel i, $Relu(-\vec{\mathbf{C}})$ means if there are customers in waiting queue, $Relu(C_i) = abs(C_i).C_i > 0$ means there are C_i capacity in channels, so $Relu(-C_i) = 0$

Peak congestion (PC) means the minimum total capacity.

$$PC = max(Relu(-\vec{\mathbf{C}})), \tag{6}$$

PC refects the most congested degree, also represents the maximum length of the waiting queue.

And *AFR* means the average idle degree of the catering staff.

$$AFR = \frac{\sum_{i=1}^{N} Relu(\vec{\mathbf{C}})}{N}, \tag{7}$$

where $Relu(\vec{\mathbf{C}})$ represents the remaining capacity of channels. If there are remaining capacity of channels, $Relu(\vec{\mathbf{C}}) = \vec{\mathbf{C}}$, otherwise $Relu(\vec{\mathbf{C}}) = 0$.

Self-service

Acceptance Rate (SP) means the percentage of customers who accepted the switch-to-self-service suggestions;

$$SP = \frac{\sum_{i=1}^{N} \vec{1}_{sp}}{N}, \tag{8}$$

where $\vec{1}_{sp}$ represents whether the customer accepts the self-service channel. $\sum_{i=1}^{N} \vec{1}_{sp}$ the total number of customers who accept the self-service channels.

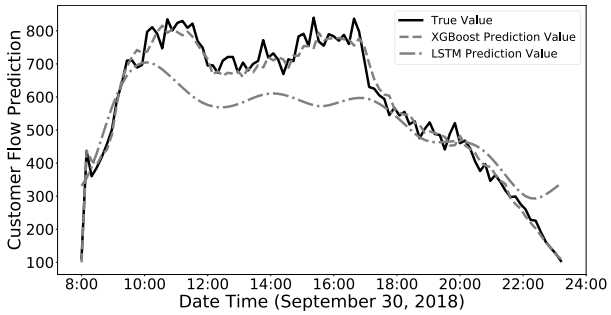


FIGURE 4. Comparison of algorithms on the flow forecasting task. The solid black line is the true value, the dash is result estimated using `XGBoost` and the dash dot shows the result computed using LSTM.

Drainage Acceptance Rate (DP) means the percentage of customers who accepted the switch-to-app suggestions;

$$DP = \frac{\sum_{i=1}^{N} \vec{1}_{dp}}{N}, \tag{9}$$

where $\vec{1}_{dp}$ represents whether the customer accepts the target drainage channel. $\sum_{i=1}^{N} \vec{1}_{dp}$ means the total number of customers who accept the drainage channels.

Besides, we also calculate the rewards of all RL models for this task, and then normalize these rewards.

When evaluating on the real dataset, we are additionally interested in *Routing Rate*, which means the number of customers who agree to be routed to other channels outside hotline, divided by the total number of customers. These metrics are used to evaluate the reinforcement learning algorithms and traditional machine learning.

B. CUSTOMER FLOW FORECASTING

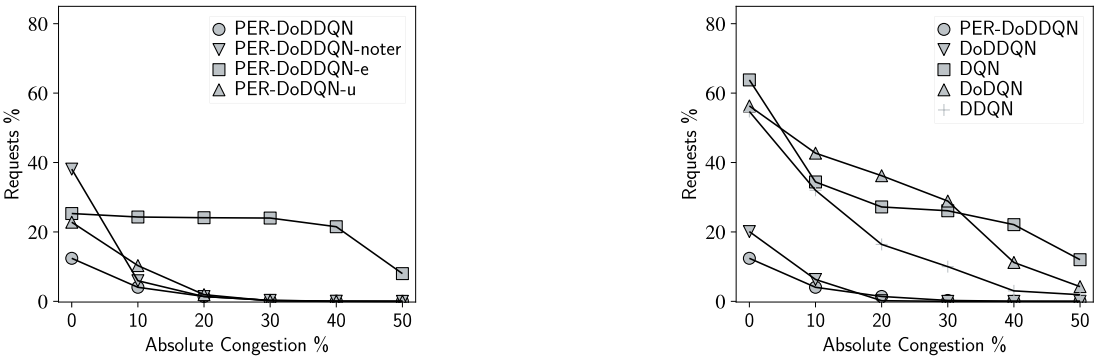
The flow prediction is verified as an important factor to the model in the simulation results listed in the previous section. It is directly related to the performance of the system.

As mentioned in section V-B, **LSTM** algorithm [10] and `XGBoost` algorithm [11] are widely used in this area, so they are used and compared in our experiments.

Our pre-processing steps are as follows: we split the entire flow for every 10 minute and count the flow volume, this results in a total number of 10,271 time points; In order to predict the volume in the next time window, the forecasting makes use of historical data in the past 24 hours (144 points). All methods are trained on the 80% of the data and tested on the rest.

We follow [12] to implement the **LSTM** method, which consists of two layers. The algorithm runs for 40 epochs with batch size 200. A dropout rate of 0.8 is employed to avoid overfitting. The `XGBoost` ends after 40 rounds training, with the learning rate 0.1; the maximal tree depth is 6; and the colsample-tree is 0.7. We set an equal weight of 0.1 to both L1 and L2 regularization when training `XGBoost`.

The final prediction results in Figure 4. We can observe a better performance of using `XGBoost` than using **LSTM** [13].



(a) Effect of PER-DoDDQN configurations

(b) Effectiveness comparison of DQN variants

FIGURE 5. Evaluation results on the hotline channel, measured by absolute congestion percentage, which is average queue length relative to the channel's original capacity.

TABLE 2. Synthetic results. The numbers in bold are the best performance on the metric.

	PER-DoDDQN				DoDDQN	DQN	DoDQN	DDQN
	full	e	u	noter				
CCR	**0.124**	0.253	0.228	0.383	0.201	0.638	0.523	0.547
AC	**-1.123**	-11.693	-2.404	-2.196	-1.648	-15.925	-15.342	-9.179
PC	-44	-67	-39	-42	**-26**	-92	-77	-77
AFR	19.606	20.955	30.775	10.383	33.625	**8.921**	9.360	10.541
DP	0.258	0.325	0.245	0.307	0.321	**0.327**	0.299	0.304
SP	0.400	0.373	0.395	0.392	0.457	0.419	0.385	**0.478**
Rewards	**0.912**	0.465	0.649	0.767	0.872	-1.853	-1.334	-0.373

For a better understanding of the performance difference, we evaluate both algorithms using commonly used metrics, accuracy and root-mean-square-error (RMSE):

$$\text{RMSE} = \sqrt{\sum_{i=1}^{T} \frac{1}{T}(y_i - \hat{y}_i)^2}, \tag{10}$$

where T is the total time points, y_i is the true value and $\hat{y}_i$ is the predict value. We define the accuracy with a small error-tolerance range: -8% to 15%. For example, when actually there are 100 customer requests, we regard prediction that falls between 92 and 115 are correct.

Evaluation results further confirm the observations made in Figure 4: `XGBoost` is more effective, which has an accuracy of 0.803 and RMSE of 40.445; while LSTM has a lower accuracy and a higher RMSE, which are 0.726 and 81.234, respectively.

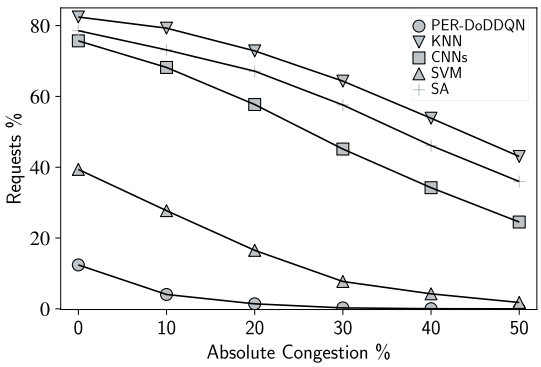


FIGURE 6. The results of PER-DoDDQN and other machine learning(eg.heuristic algorithm) algorithms on simulated data.

C. ROUTING RESULTS ANALYSIS

We show the results of various configurations on both real and synthetic data in this section.

Note that, as in the real application, the bottleneck channel is the hotline, we will directly use the "hotline" channel in results analysis. It takes 35.228 seconds training PER-DoDDQN's model every 1000 batches and the training convergences after training 4000 batches.

1) RESULTS ON SYNTHETIC DATA

We show evaluation results on the synthetic dataset in table 2 and figure 5. As the hotline channel is the main concern in the real production system, we first focus on the absolute congestion percentage of this particular channel, and show results in Figure 5. We show the comparison of our proposed PER-DoDDQN in Figure 5(a). As we can see, our environment model is crucial for finding the optimal routing plan: the method PER-DoDDQN-e, which doesn't have flow estimation is consistently worse than the others. Without user modeling also hurt the performance, but only slightly on the synthetic data. We then compare our method to other DQN variants in Figure 5(b). The trend is clear – our proposed PER-DoDDQN is the best among all. Also, we can see that the prioritized experience replay plays an important role in the model, without which the performance can be degraded a lot.

TABLE 3. Synthetic results of PER-DoDDQN and other machine learning algorithms. The numbers in bold are the best performance on the metric.

	PER-DoDDQN				KNN	CNNs	SVM	SA
	full	e	u	noter				
CCR	**0.124**	0.253	0.228	0.383	0.824	0.757	0.394	0.786
AC	**-1.123**	-11.693	-2.404	-2.196	-45.085	-30.486	-7.900	-38.773
PC	-44	-67	**-39**	-42	-161	-140	-82	-148
AFR	19.606	20.955	30.775	10.383	**9.937**	12.146	23.104	14.386
DP	0.258	0.325	0.245	0.307	0.276	**0.352**	0.273	0.341
SP	**0.400**	0.373	0.395	0.392	0.396	0.392	0.394	0.386

TABLE 4. Evaluation results on the real dataset. The baseline is based on business rules from the product system. Besides the metrics used previously, the routing rate (RR) and routing number (RN) is also used. Note that RN is shown in the unit of $\times 10^6$.

	PER-DoDDQN				DoDDQN	DQN	DoDQN	DDQN	Baseline
	full	e	u	noter					
DP	0.341	0.282	0.277	0.341	0.340	0.326	0.328	**0.342**	0.021
SP	0.411	0.399	0.393	0.408	0.437	**0.471**	0.449	0.408	0.201
RR	**0.390**	0.265	0.343	0.385	0.276	0.221	0.263	0.273	0.216
RN	**1.908**	1.297	1.680	1.889	1.350	1.082	1.288	1.338	1.056
Rewards	**2.638**	-0.430	-0.523	-0.441	-0.293	-0.306	-0.312	-0.333	–

We then consider the overall evaluation using proposed metrics, in Table 2. Among all evaluation metrics, the CCR is the most important one, and the proposed PER-DoDDQN is the best across all configurations; this trend also holds when evaluating using AC. DoDDQN shows a better performance when considering the peak congestion, and it achieves a similar but worse performance than its PER variant. Using a simple DQN is the best if only drainage percentage or average free rate, however, it simply means a non-optimal plan is shown. Further, results on both DoDDQN and DDQN suggest that our proposed PER-DoDDQN is more advanced than considering any of the components alone.

We compare RL models with standard heuristics algorithms, including Simulated annealing (SA), CNN and Baselines(rule-based system) which can be viewed as heuristic methods. As shown in Figure 6 and Table 3, the traditional machine learning (SA, SVM, KNN, CNN) perform worse than the RL algorithms, especially in the metrics CCR, AC and PC. Because the long term gains are not considered in these models. The RL framework takes into account not just immediate reward but also the impact of the selected action in the future. But other ML models recommend channels to the customers solely on the current state.

Note that the average idle degree of the catering staff(AFR) can be very low when the customers are more likely to be assigned to hotline regardless of the capacity of channels. And the percentages of customers who accepted the switch-to-self-service suggestions (SP) and switch-to-app suggestions(DP) are independent of the future state. Supervised machine learning methods only care about the current state, so it can perform similarly to or even perform better than the RL models in one or two of the above three metrics, but obviously it cannot change the fact that it cannot beat RL algorithms among all the metrics in general.

TABLE 5. Evaluation results on the real dataset with machine learning algorithms. The baseline is based on business rules from the product system. RR is Routing rate and RN is routing number. Note that RN is shown in the unit of $\times 10^6$.

	PER-DoDDQN				KNN	CNNs	SVM	SA
	full	e	u	noter				
DP	**0.341**	0.282	0.277	0.341	0.275	0.329	0.290	0.329
SP	**0.411**	0.399	0.393	0.408	0.395	0.342	0.390	0.340
RR	**0.390**	0.265	0.343	0.385	0.166	0.109	0.242	0.119
RN	**1.908**	1.297	1.680	1.889	0.812	0.535	1.182	0.580

2) COMPARISON ON THE REAL DATASET

We show experimental results on the real dataset, with the real product system in use as the baseline. Note that, we use the results of `XGBoost` discussed in Section IV-B when modeling environment, in all of deep reinforcement learning method. Among all 4,898,143 records in our dataset, when considering the baseline model, around 984,526 (20.1%) of customers agree to be routed to self-service, and only 104,820 (2.14%) customers are successfully routed to drainage channel.

As suggested by Table 4, our proposed deep reinforcement learning framework outperforms baseline greatly. Among all methods PER-DoDDQN, works the best in absolute numbers of customers assigned to non-hotline if there is a congestion: it successfully reassigns 1.908 million (39.0%) customers to non-hotline channels. Note that, in real practice, the last tow metric routing rate and routing numbers are the most important indicators; and the PER-DoDDQN is the best on both. Compare to the synthetic data in Table 2, we can see that the PER-DoDDQN is the optimal configuration across all methods. For example, the DP results of PER-DoDDQN is almost the best on the real dataset, while this is not the case on the synthetic dataset. A similar observation can be made when considering SP metric. When a user model is

built and employed in the framework, we can see that it is of great importance in our model – without which the model performance will decrease largely. The same observation on the importance of flow forecasting can also be seen from the effectiveness score of PER-DoDDQN-e in Table 4.

Comparing with RL frameworks, the other machine learning algorithms perform poorly in real-world dataset. The number of customers redistributed to other channels by other machine learning algorithms is far less than that by RL models. As shown in Table 5, among all supervised algorithms, SVM performs best in these two metrics (RR and RN). But It only redistributes 1.182 million customers to non-hotline channels, slightly higher than baseline. For DP and SP, supervised algorithms perform worse than PER-DoDDQN.

V. RELATED WORK

In this section, we will introduce two aspects of technical background in detail, including deep reinforcement learning in Section V-A and flow forecasting in Section V-B.

A. DEEP REINFORCEMENT LEARNING

Reinforcement learning (RL) is widely used in many application systems, such as network communication [14], object detection [15], digital image steganalysis [16] and edge computing [17]. It is a hot area of machine learning concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative rewards by optimizing the policy [1]:

$$\pi_\theta(s, a) = \underset{\pi}{\operatorname{argmax}}\, E(\sum_{t=0}^{\infty} \gamma^t r_t; \theta)$$

where γ is the discounted factor with the immediate reward greater. Unlike supervised learning which requires labeled training data, or unsupervised one without labels, reinforcement learning learns from the environment through ''interaction'': it will observe the environment all around, and summarize them into a 'state'(S_t). Together with two strategies of exploration and exploitation, the action a_t will be selected. After that action, the agent will observe a new state S_{t+1}, and get the rewards R_t of the new environment [18]. R_t will be used to update the agent's strategies. After trying a large number of the above steps, the agent will optimize its strategies to adapt the environment. Reinforcement learning also defines some 'terminal states'. The learning process will reset a new episode when the agent has reached a terminal state. Now there are two types of reinforcement: value-based models and policy-based models [1].

Value-based reinforcement learning algorithms aim at learning the state-action value function (or Q-function), by minimizing the Temporal-Difference error(often referred to as TD-error). The Q value function [1] can be defined as Equation 11:

$$q_\theta(s, a) = E(G_t | S_t = s, A_t = a; \theta) = E(r + \gamma E_{a'}(q_\theta(s', a'))), \tag{11}$$

Note that $q_\theta(s, a; \theta)$ is the value function when the agent selects the action a in the state s, and s' is the new state of the environment. Besides, γ is the discounted future reward factor, and r is the immediate reward. Value-based algorithms can only deal with enumerable action space. The following four value-based methods are widely used.

1) Q-LEARNING

The earlier classic RL algorithm is Q-learning [19], which first generates Q table and R table, then update Q table during training, and therefore it only works in discrete state and action. R table is initialized to the immediate rewards of state-action pairs. While the Q values of the state-action pairs are stored in Q table. Due to Equation 11, the Q table is updated during training as follows:

$$q(s, a) = q(s, a) + \alpha(r + \gamma \max_{a'} q(s', a') - q(s, a)). \tag{12}$$

2) DEEP Q-LEARNING

As one of the most important branches of machine learning, deep learning has developed rapidly in recent years and has been successfully applied in many areas. Reference [20] use RICNN for object detection in sensing images. Reference [21] model the background with deep autoencoders for object detection. Reference [22] propose a unified co-salient framework with two highlights. Reference [23] propose SP-MIL framework for co-saliency detection. Reference [24] propose a fusion algorithm of the image feature and the text feature extracted from two separate networks for image classification. Reference [25] analyze DNA methylation data by deep learning.

Deep Learning is able to be coupled with reinforcement learning, which greatly enhances the performance of RL. One of the successful model is deep Q-learning (DQN), which uses deep learning method to approximate Q table, along with two improvements, experience replay [26] and two separate neural networks [27]. One successful application of DQN is computer games. Reference [27] mentioned that DQN has already reached the human level in 49 games of Atari 2600 game series. DQN is the improvement of Q-learning, minimizing the TD-error:

$$\mathcal{L} = (r + \gamma \max_{a'} q_{\bar{\theta}}(s', a') - q_\theta(s, a))^2, \tag{13}$$

where $q_{\bar{\theta}}$ is used to evaluate the target value and q_θ is used to evaluate the current value, and q_θ will be assigned to $q_{\bar{\theta}}$ at regular intervals. For simplicity, we refer to $(r + \gamma \max_{a'} q_{\bar{\theta}}(s', a'))$ as $\hat{y}_i$ in the following introduction. DQN learns the policy by gradient descent, and the gradient of the loss is written as Equation 14:

$$\nabla_\theta \mathcal{L} = E_{s,a,r,s'}((\hat{y}_i - q_\theta(s, a))\nabla_\theta q_\theta(s, a)). \tag{14}$$

3) DOUBLE DQN

It has improved DQN through selecting the action before evaluating Q value, which can reduce the chance of overestimations [2]. The loss function (TD-error) of Double

DQN is modified into Equation 15, but the learning method is the same as nature DQN.

$$\mathcal{L} = (r + \gamma q_{\bar{\theta}}(s', \arg\max_{a'} q_\theta(s', a')) - q_\theta(s, a))^2. \quad (15)$$

4) DUELING DQN

The Q value in dueling DQN consists of two parts, value function, V, and advantage function, A. The dueling DQN can learn which states are (or are not) valuable, without having to learn how the action effects the state [3]. And The Q value in dueling DQN can be written as follows:

$$q_{\theta,\alpha,\beta}(s, a) = V_{\theta,\beta}(s) + A_{\theta,\alpha}(s, a) - \frac{1}{|\mathcal{A}|}\sum_{a'} A_{\theta,\alpha}(s, a'), \quad (16)$$

where θ is the sharing parameter of V and A, while β and α are the private parameters of V and A respectively.

5) EXPERIENCE REPLAY AND PRIORITIZED EXPERIENCE REPLAY

The replay buffer is often adopted in the reinforcement learning process, in order to reduce the correlation of the data. Experienced replay is adopted by [26], it randomly samples transitions from previous training in order to make the data be subject to stationary function, and to make the neural network in DQN converge easier. Randomly sampling transitions from the replay buffer may hurt the performance of the algorithm as these transitions are not equally weighted. A straightforward method is to use a biased sampling method, in which the sampling probability is proportional to the TD-error – transitions with higher TD-errors are more likely to be sampled. Based on this intuition, [4] proposed prioritized experience replay. Further, in order to reduce the time spent in sorting these samples, these transitions in prioritized experienced replay are stored using the SumTree data structure [4]. Let p_j be TD-error, the weight w_i of the i-th transition, (s_i, a_i, r_i, s_i'), is computed using:

$$w_i = p_i^\alpha / \sum_j (p_j^\alpha). \quad (17)$$

6) POLICY BASED METHOD

Policy gradient algorithm is the classical policy based algorithm. Compared to value based algorithm, the policy gradient algorithm directly maximizes the expectations of the state value function. Policy based reinforcement learning can also be applied to continuous action space [28].

7) ACTOR-CRITIC ALGORITHM (AC)

In the previous RL models, the agent can only be updated after an episode in the policy gradient algorithm. Actor-Critic (AC) algorithm solves this problem by combining policy gradient algorithm with deep Q-learning ingeniously. A3C [29], DDPG [30], PPO [31] and ACKTR [32] are all the new development of RL based actor-critic framework. The policy based methods can be used in continuous action space. DQN algorithms are suitable to deal with discrete action space. Also through the explanation of the above-mentioned content, the RL methods based Actor Critic framework, which have the complex structure, are difficult to converge. Therefore, we prefer to choose DQN to solve our routing task in the next sections, according to the characteristics of the problem.

8) THE LATEST PROGRESS

Reference [33] propose a method based on neural network and combined with reinforcement learning to process the scarce data or the task which changes quickly. Reference [34] illustrates the problem of overestimation exists in the actor-critic framework, and proposes a algorithm based on double DQN to limit overestimation. Reference [35] proposes a algorithm called soft actor-critic based on the maximum entropy framework to improve the convergence and reinforcement learning stability. Reference [36] mainly studied the effect of baselines dependent of state-action, especially in this continuous control tasks.

9) DISCUSSION

So far, we've summarized some popular value-based deep reinforcement learning methods. The DQN proposed by [26] is the first successful integration of the deep neural network and reinforcement learning, which benefits a lot from using experienced replay. It can work on the continuous state space, which is a result of using the deep neural network as the function approximator. However, the DQN suffers from overestimations, as pointed out by [2]. To solve the problem, [2] proposed double DQN (DoDQN), which addresses the problem by modifying the updating formula. The dueling DQN (DDQN) improves by separating Q-value into state value and action advantage, which improves the converging rate [3]. In our framework, DQN and all its variants can be applied, and we empirically compare their performance in this application scenario.

The policy based methods can be used in continuous action space. DQN algorithms are suitable to deal with discrete action space. Also through the explanation of the above-mentioned content, the RL methods based Actor Critic framework, which have the complex structure, are difficult to converge. Therefore, we prefer to choose DQN to solve our routing task in the next sections, according to the characteristics of the problem.

B. FLOW FORECAST

In our proposed framework, one of the important parts is to predict the future flow of each channel, which we model it as a time-series prediction problem. A time series is a series of data points indexed (or listed or graphed) in time order [37]. Examples of time series are heights of ocean tides and counts of sunspots. In our task, customer requests flow can be viewed as a time series, of which the volume may change temporally. The problem of time series prediction is a research topic for many years, which aims at predicting the value at a given time point. Time-series prediction problem is often addressed by

using machine learning models, include Random Forest [38], GBRT [39], `XGBoost` [40], and Long Short-Term Memory (LSTM) [12].

The Random forest is an ensemble algorithm with multiple parallel trees to reduce the time spent in training and testing [41]. Whereas, random forest algorithm often suffers from overfitting because of the inevitable problem that data may be full of noise. The GBRT trained with residual data is a kind of tree-based boosting algorithm [42], so it only supports serial execution. The improved `XGBoost` greatly improves GBRT and can run in parallel with the block structure [43].

Deep learning has been demonstrated the success in many application scenarios, and also in the time-series prediction. The superiority of applying LSTM [12] to series prediction can hardly be generalized to all types of data. Cautions need to be made when making decisions on whether to adopt the method in the flow forecast problem. For example, [44] use LSTM to predict Mackey-Glass series and the Santa Fe FIR laser emission series, and observe that LSTM does not seem to consistently be the best on all series, especially on simpler time series prediction tasks, it may be less effective. Reference [44] suggest to use LSTM only when simpler traditional methods fail. All previously discussed methods can be applied to our problem, and we will provide empirical evaluation in Section IV.

VI. CONCLUSION AND FUTURE WORK

We formulate the classic customer request routing problem into an optimization problem by considering both channel resources and customers' satisfaction. To address the real problem, we proposed a novel framework, which is based on the deep reinforcement learning method. In addition to the framework, we also propose a new routing method by combining DDQN and DoDQN methods. Extensive experiments on both real and synthetic data show that our proposed framework greatly improves the existing system and our proposed PER-DoDDQN method is the best configuration.

In the future work, we plan to further improve our method from the following perspectives: (i) improve our user profiling by understanding users' description of requests, instead of considering attributes alone; (ii) we plan to incorporate real-time features into the proposed PER-DoDDQN model, for a better model of the environment; (iii) we also plan to generalize our model to more routing or dispatching related problems.

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[2] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. AAAI*, Mar. 2016, pp. 2094–2100.

[3] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. De Freitas, "Dueling network architectures for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, Dec. 2016, pp. 1995–2003.

[4] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized sequence experience replay," 2015, *arXiv:1905.12726*. [Online]. Available: https://arxiv.org/abs/1905.12726

[5] M. Wunder, M. L. Littman, and M. Babes, "Classes of multiagent Q-learning dynamics with epsilon-greedy exploration," in *Proc. ICML*, 2010, pp. 1167–1174.

[6] B. Nowok, G. M. Raab, and C. Dibben, "synthpop: Bespoke creation of synthetic data in R," *J. Stat. Softw*, vol. 74, no. 11, pp. 1–26, Oct. 2016.

[7] J. Wang, J. Wang, Y. Wu, J. Wang, H. Zhu, M. Lin, and J. Wang, "A machine learning framework for resource allocation assisted by cloud computing," *IEEE Netw.*, vol. 32, no. 2, pp. 144–151, Mar. 2017.

[8] B. Mao, F. Tang, Z. M. Fadlullah, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "A novel non-supervised deep-learning-based network traffic control method for software defined wireless networks," *IEEE Wireless Commun.*, vol. 25, no. 4, pp. 74–81, Aug. 2018.

[9] F. Zhu, Z. Ma, X. Li, G. Chen, J.-T. Chien, J.-H. Xue, and J. Guo, "Imagetext dual neural network with decision strategy for small-sample image classification," *Neurocomputing*, vol. 328, Aug. 2018.

[10] D. Qin, J. Yu, G. Zou, R. Yong, Q. Zhao, and B. Zhang, "A novel combined prediction scheme based on CNN and LSTM for urban $pm_{2.5}$ concentration," *IEEE Access*, vol. 7, pp. 20050–20059, 2019.

[11] D. Zhang, L. Qian, B. Mao, C. Huang, B. Huang, and Y. Si, "A data-driven design for fault detection of wind turbines using random forests and XGboost," *IEEE Access*, vol. 6, pp. 21020–21031, 2018.

[12] A. Azzouni and G. Pujolle, "A long short-term memory recurrent neural network framework for network traffic matrix prediction," 2017, *arXiv:1705.05690*. [Online]. Available: https://arxiv.org/abs/1705.05690

[13] Z. Ma, H. Yu, W. Chen, and J. Guo, "Short utterance based speech language identification in intelligent vehicles with time-scale modifications and deep bottleneck features," *IEEE Trans. Veh. Technol.*, vol. 68, no. 1, pp. 121–128, Jan. 2019.

[14] Z. Xu, J. Tang, J. Meng, W. Zhang, Y. Wang, C. H. Liu, and D. Yang, "Experience-driven networking: A deep reinforcement learning based approach," in *Proc. IEEE Conf. Comput. Commun.*, Honolulu, HI, USA, Apr. 2018, pp. 1871–1879.

[15] Y. Rao, J. Lu, and J. Zhou, "Attention-aware deep reinforcement learning for video face recognition," in *Proc. IEEE Int. Conf. Comput. Vis.*, Venice, Italy, Oct. 2017, pp. 3951–3960.

[16] D. Hu, S. Zhou, Q. Shen, S. Zheng, Z. Zhao, and Y. Fan, "Digital image steganalysis based on visual attention and deep reinforcement learning," *IEEE Access*, vol. 7, pp. 25954–25935, 2019.

[17] Q. Qi, J. Wang, Z. Ma, H. Sun, Y. Cao, L. Zhang, and J. Liao, "Knowledge-driven service offloading decision for vehicular edge computing: A deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 4192–4203, May 2019.

[18] R. Marcus and O. Papaemmanouil, "Deep reinforcement learning for join order enumeration," 2018, *arXiv:1803.00055*. [Online]. Available: https://arxiv.org/abs/1803.00055

[19] B. J. A. Kröse, "Learning from delayed rewards," *Robot. Auton. Syst.*, vol. 15, no. 4, pp. 233–235, 1995. doi: 10.1016/0921-8890(95)00026-C.

[20] G. Cheng, P. Zhou, and J. Han, "Learning rotation-invariant convolutional neural networks for object detection in VHR optical remote sensing images," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 12, pp. 7405–7415, Dec. 2016.

[21] J. Han, D. Zhang, X. Hu, L. Guo, J. Ren, and F. Wu, "Background prior-based salient object detection via deep reconstruction residual," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 8, pp. 1309–1321, Aug. 2015.

[22] D. Zhang, J. Han, C. Li, J. Wang, and X. Li, "Detection of co-salient objects by looking deep and wide," *Int. J. Comput. Vis.*, vol. 120, no. 2, pp. 215–232, 2016.

[23] D. Zhang, D. Meng, and J. Han, "Co-saliency detection via a self-paced multiple-instance learning framework," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 5, pp. 865–878, May 2017. doi: 10.1109/TPAMI.2016.2567393.

[24] F. Zhu, Z. Ma, X. Li, G. Chen, J.-T. Chien, J.-H. Xue, and J. Guo, "Image-text dual neural network with decision strategy for small-sample image classification," *Neurocomputing*, vol. 328, pp. 182–188, Feb. 2019.

[25] Z. Si, H. Yu, and Z. Ma, "Learning deep features for dna methylation data analysis," *IEEE Access*, vol. 4, pp. 2732–2737, 2016.

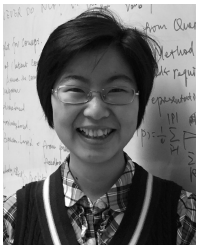
[26] I.-A. Hosu and T. Rebedea, "Playing atari games with deep reinforcement learning and human checkpoint replay," 2016, *arXiv:1607.05077*. [Online]. Available: https://arxiv.org/abs/1607.05077

[27] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, and S. Petersen, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.

[28] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural Netw.*, vol. 21, no. 4, pp. 682–697, May 2008.
[29] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. 33nd Int. Conf. Mach. Learn.*, New York, NY, USA, Jun. 2016, pp. 1928–1937.
[30] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*. [Online]. Available: https://arxiv.org/abs/1509.02971
[31] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*. [Online]. Available: https://arxiv.org/abs/1707.06347
[32] Y. Wu, E. Mansimov, S. Liao, R. B. Grosse, and J. Ba, "Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation," 2017, *arXiv:1708.05144*. [Online]. Available: https://arxiv.org/abs/1708.05144
[33] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "Meta-learning with temporal convolutions," 2017, *arXiv:1707.03141*. [Online]. Available: http://arxiv.org/abs/1707.03141
[34] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," (2018), *rXiv:1802.09477*. [Online]. Available: https://arxiv.org/abs/1802.09477
[35] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," 2018, *arXiv:1801.01290*. [Online]. Available: https://arxiv.org/abs/1801.01290
[36] G. Tucker, S. Bhupatiraju, S. Gu, R. E. Turner, Z. Ghahramani, and S. Levine, "The mirage of action-dependent baselines in reinforcement learning," 2018, *arXiv:1802.10031*. [Online]. Available: https://arxiv.org/abs/1802.10031
[37] T. M. Martinetz, S. G. Berkovich, and K. J. Schulten, "'Neural-gas' network for vector quantization and its application to time-series prediction," *IEEE Trans. Neural Netw.*, vol. 4, no. 4, pp. 558–569, Jul. 1993.
[38] L. Lin, F. Wang, X. Xie, and S. Zhong, "Random forests-based extreme learning machine ensemble for multi-regime time series prediction," *Expert Syst. Appl.*, vol. 83, pp. 164–176, Oct. 2017.
[39] G. Barta, G. Nagy, G. Papp, and G. Simon, "Forecasting framework for open access time series in energy," in *Proc. IEEE Int. Energy Conf. (ENERGYCON)*, Apr. 2016, pp. 1–6.
[40] W. Wang, Y. Shi, G. Lyu, and W. Deng, "Electricity consumption prediction using xgboost based on discrete wavelet transform," *DEStech Trans. Comput. Sci. Eng.*, Oct. 2017.
[41] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
[42] J. H. Friedman, "Stochastic gradient boosting," *Comput. Statist. Data Anal.*, vol. 38, no. 4, pp. 367–378, 2002.
[43] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," 2016, *arXiv:1603.02754*. [Online]. Available: https://arxiv.org/abs/1603.02754
[44] F. A. Gers, D. Eck, and J. Schmidhuber, "Applying LSTM to time series predictable through time-window approaches," in *Proc. ICANN*, Aug. 2001, pp. 669–676.

ZINING LIU is currently pursuing the master's degree with the School of Software, Shandong University. From July 2017 to July 2018, he was an Intern with the Chinese Academy of Sciences. His research interests include reinforcement learning and personalized recommendation systems.

CHONG LONG received the Ph.D. degree from the Department of Computer Science and Technology, Tsinghua University, China, in 2010. He was with Yahoo!, Amazon, and Hulu, respectively. He is currently an Algorithm Expert with the Ant Financial Services Group. His research interests include natural language processing, information retrieval, information extraction, and machine learning.



XIAOLU LU received the Ph.D. degree in computer science from RMIT University, in 2018, where she is currently a Postdoctoral Research Fellow with the Department of Information Technology. Her research interests include the area of information retrieval, focusing on search evaluation and trade-offs between effectiveness and efficiency.

ZEHONG HU received the Ph.D. degree from Nanyang Technological University, Singapore. He is currently an Algorithm Expert with the Alibaba Group. His research interests include multiagent systems and mechanism design. His papers have been published by top conferences, including AAAI, IJCAI, and NIPS.

JIE ZHANG received the Ph.D. degree from the Cheriton School of Computer Science, University of Waterloo, Canada, in 2009. He is currently an Associate Professor with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. He is also an Associate with the Singapore Institute of Manufacturing Technology (SIMTech). During his Ph.D. study, he held the prestigious NSERC Alexander Graham Bell Canada Graduate Scholarship rewarded for top Ph.D. students across Canada. He was a recipient of the Alumni Gold Medal at the 2009 Convocation Ceremony. The Gold Medal is awarded once a year to honor the top Ph.D. graduate from the University of Waterloo. His papers have been published in top journals and conferences and won several best paper awards. He is also active in serving research communities.

YAFANG WANG received the Ph.D. degree from the Database and Information Systems Group, Max Planck Institute for Informatics, Germany, in February 2013, where she was a Postdoctoral Researcher, until September 2013. From October 2013 to March 2017, she was an Associate Professor with the School of Computer Science and Technology, Shandong University. Since March 2017, she has been a Staff Algorithm Engineer with Ant Financial (Alibaba Group). Her research interests include knowledge harvesting, semantic analytics, and reinforcement learning.