DeepEdge: A New QoE-Based Resource Allocation Framework Using Deep Reinforcement Learning for Future Heterogeneous Edge-IoT Applications

Ismail AlQerm, *Member, IEEE*, and Jianli Pan, *Senior Member, IEEE*

***Abstract*—Edge computing is emerging to empower the future of Internet of Things (IoT) applications. However, due to heterogeneity of applications, it is a significant challenge for the edge cloud to effectively allocate multidimensional limited resources (CPU, memory, storage, bandwidth, etc.) with constraints of applications' Quality of Service (QoS) requirements. In this paper, we address the resource allocation problem in Edge-IoT systems through developing a novel framework named *DeepEdge* that allocates resources to the heterogeneous IoT applications with the goal of maximizing users' Quality of Experience (QoE). To achieve this goal, we develop a novel QoE model that considers aligning the heterogeneous requirements of IoT applications to the available edge resources. The alignment is achieved through selection of QoS requirement range that can be satisfied by the available resources. In addition, we propose a novel two-stage deep reinforcement learning (DRL) scheme that effectively allocates edge resources to serve the IoT applications and maximize the users' QoE. Unlike the typical DRL, our scheme exploits deep neural networks (DNN) to improve actions' exploration by using DNN to map the Edge-IoT state to joint resource allocation action that consists of resource allocation and QoS class. The joint action not only maximize users' QoE and satisfies heterogeneous applications' requirements but also align the QoS requirements to the available resources. In addition, we develop a Q-value approximation approach to tackle the large space problem of Edge-IoT. Further evaluation shows that *DeepEdge* brings considerable improvements in terms of QoE, latency and application tasks' success ratio in comparison to the existing resource allocation schemes.**

***Index Terms*—Resource allocation, deepEdge, edge-IoT, deep reinforcement learning (DRL), quality of experience (QoE).**

I. Introduction

GROWING Internet of Things (IoT) applications such as Google Home and Amazon Echo raise the demands for cloud computing platforms for data processing. However, it is very difficult for the existing centralized cloud computing model to scale with projected large number of IoT devices and ubiquitous applications, due to the large amount of generated data to be sent over relatively long distance between IoT devices and clouds. Edge or fog computing [1], [2], [3] is considered as a potential approach to fulfill these applications demands by moving more computing, storage, and intelligence resources to the edge, which would benefit IoT applications that are delay-sensitive, bandwidth/data intensive, or that require closer intelligence. We envision a future "Edge-IoT" environment where various IoT applications could use edge computing to fulfill their resource demands and performance requirements. To enable such a vision, there are some significant challenges to overcome. On the one hand, from the demand side, a massive number of IoT devices can run heterogeneous applications with various Quality of Service (QoS) requirements and different priorities. On the other hand, from the supply side, the edge clouds are expected to dynamically allocate multidimensional resources (CPU, storage, and bandwidth) at geospatially distributed points and different levels of network hierarchy. This severely complicates the required resource allocation and scheduling algorithms. Most of the current edge computing research either focuses on resource allocation without paying attention to QoS requirements of heterogeneous applications, or optimizes specific operations such as mobile offloading, migration, placement, chaining and orchestration [4], [5], [6].

In this paper, we develop a new Edge-IoT framework named *DeepEdge* using deep reinforcement learning (DRL) that allocates resources to heterogeneous IoT applications with the goal of maximizing users' Quality of Experience (QoE). Unlike the existing resource allocation schemes in the Edge-IoT research, our proposed *DeepEdge* framework ensures IoT users' satisfaction with guaranteed heterogeneous application's QoS and accounts for the dynamic resource availability at the edge in the resource allocation decisions. The paper has the following new contributions that align with *DeepEdge* goals.

- We develop a novel QoE model that maps the applications QoS requirements to a cumulative QoE score that reflects the IoT users' satisfaction. The developed QoE model is noteworthy as it supports adjustment of QoS requirements acceptable ranges to match with the available resources at the edge. In addition, it specifies certain weight for each QoS performance metric to emphasize its impact on the overall application performance.
- We propose a novel two-stage DRL to fulfill the QoE model objectives by generating joint actions including QoS class selection which aligns applications' QoS requirements to the available resources in addition to

Manuscript received April 10, 2021; revised August 23, 2021; accepted October 19, 2021. Date of publication October 29, 2021; date of current version December 9, 2021. The work was supported by National Science Foundation (NSF) CNS core grant No. 1909520. The associate editor coordinating the review of this article and approving it for publication was H. Lutfiyya. *(Corresponding author: Ismail AlQerm.)*

The authors are with the Department of Computer Science, University of Missouri–St. Louis, St. Louis, MO 63121 USA (e-mail: alqermi@umsl.edu; pan@umsl.edu).

Digital Object Identifier 10.1109/TNSM.2021.3123959

the resource allocation action. The scheme exploits deep neural networks (DNN) to map the edge-IoT state information to resource allocation joint actions.

- The proposed DRL tackles the dimensionality problem in the heterogeneous edge-IoT environment where the size of state and action space is large. It formulates the Q-value in a form of compact representation in which it is approximated as a function of smaller set of variables.
- The proposed DRL scheme tackles the tradeoff between exploration and exploitation encountered in the DRL action generation by ranking the actions according to their Q-values to avoid the equal probability of action selection used in ϵ-greedy based exploration solutions [48].

The rest of the paper is organized as follows, the related work, its shortcomings, and the motivation for the QoE and DRL based resource allocation are presented in Section II. Section III describes *DeepEdge* system architecture, system model and QoE optimization problem formulation. The two-stage DRL-based resource allocation scheme is illustrated in Section IV. Section V presents the performance evaluation and the paper concludes in Section VI.

II. Related Work and Motivation

In this section, the related work is discussed. In addition, we present the motivation for developing QoE model that is backed by DRL for resource allocation.

A. Related Work

The potential benefits of edge computing in different network applications have been studied extensively in the recent literature. A large number of existing work has focused on edge computing either about allocation for specific applications, or optimizing some operations such as offloading, migration, and orchestration [4], [5], [6]. For offloading, many schemes have been proposed to make offloading decisions to optimize energy consumption and delay performance [7], [8], [9], [10], [11]. Some of the proposals targeted allocation of edge resources. For example, the utilization of distributive game-theoretical approaches for resource allocation in "cloud-edge" multi-level networks [12]. The authors in [9] proposed an optimization framework for energy-efficient resource allocation, by assuming that the network operator is aware of the complete information of all users' applications.

DRL has been employed for solving decision-making related problems in the context of edge computing such as computation offloading [13], [14], [15], [16], management problems in vehicular networks [17], [18], [19], [20], [21] and edge resource allocation [22], [23]. For vehicular networks, DRL has been investigated to solve several problems including resource allocation [24] and computation offloading [25], [26], [27]. For instance, the work in [28] exploited DRL to solve the problem of edge resource management by leveraging hierarchical learning architectures. In [29], the authors proposed a knowledge driven service offloading decision framework for vehicular network in which the offloading decision was formulated for multiple tasks as a long-term planning problem solved by DRL. The authors in [30] proposed a resource allocation policy for the Edge-IoT system to improve the efficiency of resource utilization using deep Q-networks (DQN). The work in [31] proposed a DQN-based resource allocation scheme, which can allocate computing and network resources to reduce the average service time. In [32], a joint optimization solution solved by actor-critic DRL was proposed for allocation of resources in fog-enabled IoT systems. The work in [33] proposed a framework for edge offloading based on DRL with latency and power consumption minimization as optimization objectives. Task offloading with a single-user edge computing system was explored in [34] where DRL was exploited to optimize the trade-off between energy consumption and slowdown of tasks in the processing queue. An online computation offloading scheme based on DQN was studied in [35] under random task arrivals. The work in [36] investigated strategies for the allocation of computational resources using DRL in edge computing networks.

Given the related work, none of the existing schemes considered awareness of multiple heterogeneous applications' demands and aligning them with the available resources at the edge. Heterogeneous IoT applications may have different requirements and characteristics. These requirements might not be fulfilled with the available resources at the edge at certain time instant, given that the edge has limited computing power comparing with the cloud computing that is of virtually unlimited computing power but the relatively high latency. The problem of satisfying users' QoE and applications' demands in multiple heterogeneous applications and dynamic IoT environment with the ability to adjust QoS requirements of applications to fit with the available resources is not addressed. None of the proposed DRL schemes for resource allocation considered using DNN to diversify action generation rather than approximation of value functions of reinforcement learning. In addition, the related work neither proposed an effective approach to tackle the problem of large state space in Edge-IoT nor effectively handled the tradeoff of exploration and exploitation in reinforcement learning. A series of typical Edge-IoT applications and their characteristics are summarized in Table I.

TABLE I
Edge-IoT Applications and Their Characteristics

Devices and Applications	Data type	Priority (1-4: H to L)	Computing Intensiveness	Data Intensiveness	Latency Sensitivity
Emergency real-time response (e.g. gunshot detection)	video/audio	1	high	high	high
VR/AR related applications	video	2 or 3	high	high	high
Home voice assistant	audio	2	medium	medium/low	high
Cognitive assistance	video/audio	2 or 3	high/medium	high/medium	medium
Building access face detection	video	3	medium	medium/low	medium
Personal Identification	audio/image/ text	3	medium/low	medium/low	medium
Home health monitoring	text	2	low	low	low
Common smart home devices	text/audio	4	medium/low	low	low
Low-level sensors	text	4	low	low	low

B. Motivation

1) Quality of Experience (QoE): QoS metrics have been utilized for long as the performance optimization objective in resource allocation proposals [37], [38]. However, they do not capture the quality perceived by users, which may result in waste of network resources. QoE concept came into practice to enable broader understanding of the impact of the network performance and complement the traditional performance measurement. In contrast to QoS, QoE not only depends on the technical performance of the system but also other factors such as contents, applications, user expectations and goals, and contexts of use. QoE is more comprehensive evaluation particularly for IoT application services as it focuses on users satisfaction reflected by application QoS guarantees through the maximization of certain quality scores. In this paper motivated by the heterogeneous applications, which directly deal with the users' perception, QoE fits the resource allocation problem as an optimization goal since it can guarantee dynamic resource allocation with users satisfaction. Thus, we develop a novel QoE estimation model for edge resource allocation with heterogeneous IoT applications that has the following characteristics: 1) It monitors the Edge-IoT environment and gathers information including QoS requirements of applications and edge resources availability. 2) It defines the QoS performance metrics that are associated with certain IoT application, and determines their impact on the application performance. 3) Moreover, it supports the adjustment of the application's QoS to align with the available resources and boosts the achieved QoE.

There are some existing works utilizing the QoE concept in the edge computing context. For example, the authors in [39] proposed a QoE-aware application placement policy that prioritizes different application placement requests according to user expectations. In [40], a framework for edge computing resource distribution was proposed with crucial security and authentication components by which it ensures the delivery of users' QoE. However, none of these proposals considered resource allocation problem for multiple heterogeneous applications as each one is focusing on specific application and mostly HTTP videos. Moreover, they do not tackle the situation when QoS requirements of the application cannot be fulfilled by the available resources at certain time.

2) Deep Reinforcement Learning: Reinforcement learning [41] such as Q-learning has become an active research area [42], [43]. It deals with agents that learn to make better decisions directly from experience through interacting with the environment. Recently, reinforcement learning was combined with deep learning techniques to develop DRL which demonstrated significant impact on various applications such as video gaming, Computer Go, and data center cooling. DRL is well-suited for the resource allocation problem in Edge-IoT given its large scale and dynamicity for the following reasons: 1) Edge-IoT systems are dynamic in the context of resource demand and resource availability varies over the time which makes it difficult to use numerical optimization to solve the resource allocation problem. DRL learns over time resource allocation actions that match with environment dynamics. 2) Resource allocation decisions made in the Edge-IoT context are highly repetitive, hence, it generates a bunch of training data for the DRL technique; 3) DRL is capable of modeling complex systems such as Edge-IoT systems as various signals can be formulated as inputs to the DNN and the output strategy can be utilized in an online stochastic environment. With continuous learning, the learning agent becomes able to optimize specific tasks under varying conditions; 4) DRL does not require any prior knowledge of the system's behavior to learn a resource allocation policy. Moreover, it can support a variety of objectives just by using different reinforcement rewards.

III. System Description

In this section, we present the proposed *DeepEdge* system model and architecture, the QoE model, and the QoE maximization problem formulation.

A. DeepEdge System Model and Architecture

The considered system model in this paper consists of multiple groups of IoT devices at one side of the network. These IoT devices demand resources from the edge of the network to support their applications in tasks processing. Each group of IoT devices runs different applications. These applications are assumed to be heterogeneous and may have distinct QoS requirements. In addition, the system model includes the edge at the other side of the network which is considered as the resource provider and manager of the Edge-IoT resource allocation. The resources are located at the edge servers where computation, memory and other resources are available. The resource allocation process is managed by a controller located at the edge. It is a centralized component that receives IoT devices' requests and allocate resources at the edge servers using DRL integrated with QoE optimization.

The proposed *DeepEdge* architecture is presented in Fig. 1. The architecture consists of multiple components that work together to achieve the resource allocation with maximum users' QoE. The architecture includes the IoT environment and the edge cloud. The IoT environment comprises multiple types of IoT devices: devices that run multiple applications and multiple devices that run the same applications such as camera surveillance. The edge cloud consists of certain number of edge servers that comprise virtual machines, memory, and computation resources. IoT and edge computing update the controller implemented at the edge with their states. For example, the IoT sends information about QoS requirements of the IoT applications' and the edge computing servers provide information about the available resources, their location, and their current load. The controller incorporates a resource allocation manager (RAM) which runs the two-stage DRL scheme and decides the resource allocation policy that maximizes the QoE and adapts applications' QoS requirements to align with the available resources. The controller receives the application and edge servers state information through the devices and servers modules respectively. The QoE model is integrated with the controller to enforce its objectives and constraints to

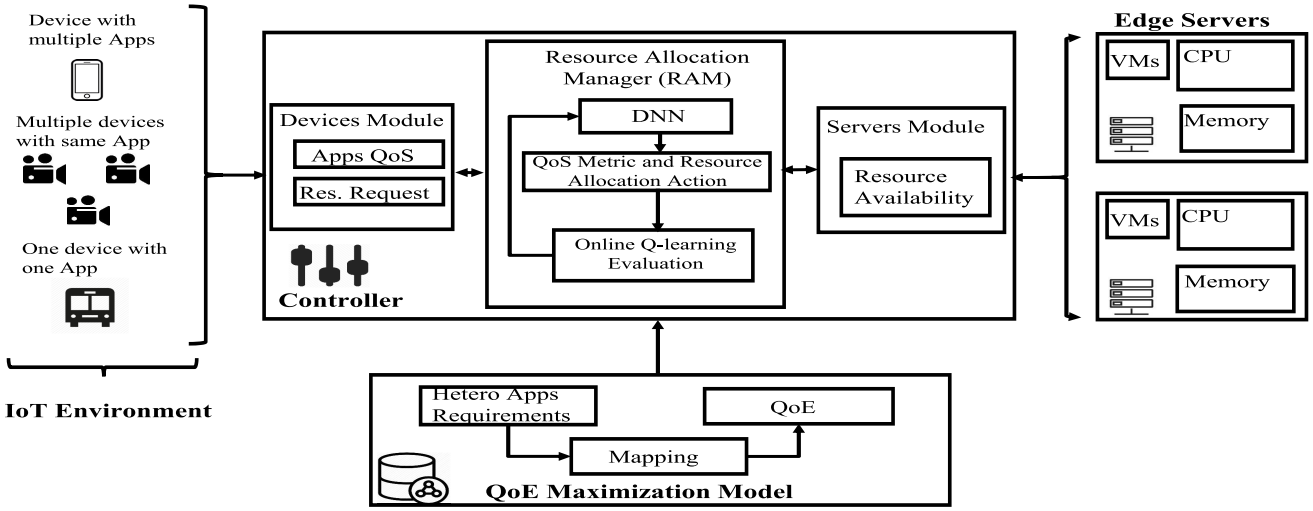

Fig. 1. *DeepEdge* Architecture.

achieve user satisfaction and efficient resources utilization in Edge-IoT with multiple heterogeneous applications.

B. The New QoE Model

The proposed QoE model aims to dynamically map the IoT applications' performance metrics such as latency into a cumulative quality score that evaluates the IoT user satisfaction. QoE is designated to be the optimization objective that *DeepEdge* exploits to drive the resource allocation decisions. The QoE formulation consists of multiple QoS performance metrics that quantify the IoT application performance. A cumulative quality score is mapped to multiple quality scores each corresponds to a QoS metric acceptable range. This range is tunable to enforce the QoS requirements to fit with the available resources. This brings a wide range of flexibility that can enhance the system resource allocation capability and maintains applications operations uninterrupted. Moreover, the QoE model incorporates the following key attributes: 1) determination of the weights of QoS metrics according to their impact on the IoT application operation; 2) QoE estimation based on the achieved QoS performance metrics of multiple heterogeneous IoT applications and the application priority that is determined according to the application type. For instance, applications with critical QoS requirements are given the highest priority.

Our QoE model is generic and can accommodate several IoT applications. We picked the following applications in this paper for demonstration purposes: emergency response, health monitoring, and personal identification. These applications comprise a broad range of QoS performance metrics, different priorities and various resource demands. Emergency response is latency sensitive with intensive data and high computation requirements while health monitoring has low latency sensitivity and requires lower data and computation. Personal identification has the least priority with moderate data intensiveness, computation and latency requirements. We denote the application type by the index ς and the IoT user (device) that runs the application by index $i \in N$. The network model assumes that one device can run single or multiple applications; or multiple devices run the same application. The QoE model considers latency (T), packet error rate (R_E), and packet loss rate (R_L) as QoS metrics for each application and assigns certain weight w for each metric to describe its impact on the application QoS. A parameter called application's QoS class α_ς is defined to represent the possible metric adjustment range. The range is evaluated using a quality score Φ which quantifies the IoT user satisfaction and contributes to the cumulative quality score. α_ς is selected to achieve the alignment of the QoS requirements of the IoT application such that they are consistent with the application's priority and the available resources at the edge. The application's priority β_ς is ranked starting from 1 to indicate the highest priority and it is assumed to be predetermined.

TABLE II
SAMPLE QUALITY SCORE FOR HETEROGENEOUS APPLICATIONS WITH VARIOUS REQUIREMENTS

Application	Priority	Performance Metrics	Metric Weight	Metric Class	Metric Range	Score
Emergency Response	1	Latency	0.7	1 2 3	50 ms to 100 ms 100 ms to 150 ms 150 ms to 200 ms	10 7 4
		Packet Error Rate	0.15	1 2 3	10^{-3} 10^{-2} 10^{-1}	10 8 6
		Packet Loss Rate	0.15	1 2 3	10^{-4} 10^{-3} 10^{-2}	10 8 6
Health Monitoring	2	Latency	0.1	1 2 3	150 ms to 200 ms 200 ms to 400 ms 400 ms to 600 ms	10 8 6
		Packet Error Rate	0.45	1 2 3	10^{-6} 10^{-5} 10^{-4}	10 7 5
		Packet Loss Rate	0.45	1 2 3	10^{-7} 10^{-6} 10^{-5}	10 7 4
Personal Identification	3	Latency	0.33	1 2 3	100 ms to 150 ms 150 ms to 250 ms 250 ms to 350 ms	10 8 5
		Packet Error Rate	0.33	1 2 3	10^{-3} 10^{-2} 10^{-1}	10 7 5
		Packet Loss Rate	0.33	1 2 3	10^{-3} 10^{-2} 10^{-1}	10 7 5

Table II presents the parameters of our QoE model including β_ς, α_ς, w and the corresponding quality score Φ of each metric class for the three heterogeneous applications considered in this model. The priority β_ς is specified based on how crucial

is the resource allocation for the application. The performance metric weight *w* is selected to show how sensitive the application for the corresponding metric. For example, emergency response is more sensitive to latency than to PLR or PER. The QoS class α_ς is set using DRL to maximize Φ and align to the resource availability at the edge. For instance, if the current resource request for certain application at certain time instant cannot be fulfilled due to lack of resources at the edge, the application α_ς will be altered in certain ranges that maintains the application service and fit with the available resources. The metric classes indicated in Table II show examples of the metric ranges that correspond to certain α_ς. The quality score Φ given in Table II shows how the selection of different class α_ς affects the achieved QoE. All the presented values for metric ranges and Φ are for demonstration of the QoE model functionality and how Φ is influenced by the selected on α_ς. Moreover, the values of the metrics ranges are tied to the metric weight specified. For instance, high latency weight in emergency response causes its latency ranges of different classes to be lower than other applications.

Φ is mapped to the following metrics: (*T*), (R_L) and (R_E). The latency *T* is calculated according to the link bandwidth, data size and the propagation medium. Packet loss rate R_L is evaluated according to [44] as, $R_L = \frac{MSS.\eta}{goodput.RTT}$ where *MSS* is the maximum segment size, η is a constant that incorporates the loss model and the acknowledgment strategy, goodput is the ratio of the delivered packets over the delivery completion time, and *RTT* is the round trip time. The packet error rate R_E is found according to the estimation model in [45], which relies on the link characteristics found using statistics from two distinct types of probing messages. QoE combines user experience and expectation to the edge computing system and network performance. The performance of the edge system is typically evaluated by QoS metrics. Thus, it is necessary to have qualitative relationship between QoS and QoE to be able to achieve QoE control mechanism based on QoS with maximum efficacy [46], [47]. To achieve this, we use a generic formula to correlate the variation in QoE with the achieved QoS metrics including latency, loss and error rates. The QoS metrics are represented by quality scores Φ_T, Φ_{RL}, and Φ_{RE} for latency, packet loss rate and packet error rate respectively. Each of these scores is obtained based on the application type and the selected metric class α_ς as indicated in Table II. For instance, if the resource allocation action was to select α_ς as 1 for the emergency response application which corresponds to the best range for all the QoS metrics, the quality score will be 10. The cumulative quality score achieved for each application with certain amount of resources allocated is calculated as follows,

$$\Phi_\varsigma = \sum_i \sum_j \sum_r x_{r,i,j} \left[w_1.\Phi_T + w_2.\Phi_{R_L} + w_3.\Phi_{R_E} \right] \quad (1)$$

where $x_{r,i,j}$ is the resource allocation indicator with *r* as a resource type (CPU, memory..etc.), *i* is the index of IoT device running the application, *j* is the index of the edge server providing the resources, and *w* is the weight of the performance metric. The cumulative quality score captures the impact of each of the QoS metrics on the overall performance. If the QoS metrics are below minimum thresholds, the cumulative quality score Φ_ς will be compromised. The proposed definition of QoE reflects all its impacting parameters including cumulative Φ_ς normalized score, the metric class (α_ς) and the priority (β_ς). It maps the relationship between Φ_ς and QoE according to the applications' characteristics since the applications' requirements vary from one type of application to the other. Therefore, QoE for multiple applications is modeled using an exponential mapping function to the quality score Φ_ς as follows,

$$QoE = \vartheta \sum_\varsigma \left\{ \frac{e^{\Phi_\varsigma - \alpha_\varsigma} + e^{-\Phi_\varsigma + \alpha_\varsigma}}{e^{\alpha_\varsigma} + \beta_\varsigma} + 1 \right\} \quad (2)$$

where ϑ is scaling constant selected for the mapping function. The definition in (2) is non-linear exponential monotonic mapping function, which suits our model as the performance metrics considered cannot be scaled uniformly, i.e., equal perceived performance difference does not correspond to equal numerical difference in the Φ_ς score. The considered QoS metrics including LA, PLR, and PER have exponential interdependency with user QoE in the proposed edge-IoT system. For example, when the QoE value is high, any variation in these metrics will heavily impact the QoE. However, considerable variation in these QoS metrics will not exhibit significant impact if the QoE is low. Thus, exponential mapping function is able to capture the impact of QoS metrics on QoE specifically for sensitive applications such as emergency response. In addition, different experiments in the literature demonstrated that exponential mapping outperforms other mapping functions such as linear or logarithmic [49].

C. Problem Formulation

In this section, we formulate the resource allocation optimization problem with the goal of maximizing the QoE found in (2) with consideration of all applications. QoE is rewarded when applications' QoS requirements are satisfied and thereafter the user satisfaction through achieving high QoE. The resource allocation optimization problem for QoE maximization is formulated as:

$$\max_{(x_{r,i,j}),\alpha_\varsigma} \quad QoE \quad (3)$$

$$s.t. \quad x_{r,i,j} \geq 0, \quad \forall j, r \quad (4)$$

$$\sum_j \sum_r x_{r,i,j} \leq C_j, \quad \forall i \quad (5)$$

$$T^\varsigma \leq T^{max} \quad (6)$$

$$R_E^\varsigma \leq R_E^{max} \quad (7)$$

$$R_L^\varsigma \leq R_L^{max}. \quad (8)$$

Edge server's capacity C_j is defined in constraint (5) to confirm that the allocated resources cannot exceed the server capacity. Equation (6), (7), (8) are the constraints for QoS metrics including *T*, R_E, and R_L respectively to guarantee that they will not exceed the maximum threshold. Note that ς is used to indicate the performance metric achieved for certain application. The definition of QoE in (2) is derived as a function of quality scores for each QoS metric Φ and

the resource allocation factor x. The quality scores correspond to the user satisfaction according to the achieved below the threshold QoS metrics values as in the constraints (6) to (8). Technically, we try to maximize the achievable QoE for each user under the conditions that all other users running different applications achieve maximum QoE. The mutual interest in each resource unit at certain server for all the users causes the optimization problem to be coupled across them. Moreover, the constraint in (5) makes the optimization problem coupled across all the resources of each edge server. This makes the convexity of the QoE cannot be guaranteed and hence, the optimization function in (3) becomes non-convex.

The formulated QoE optimization problem comprises the allocation of resources $x_{r,i,j}$ and the selection of the most appropriate QoS class α_ς. This will be the core of the decision-making problem solved by DRL in the next section.

IV. Two-Stage Deep Reinforcement Learning Scheme for Resource Allocation in Edge-IoT

In this section, we illustrate the two-stages DRL scheme built to allocate resources from the edge to the IoT applications. First, rationale and overview of the scheme are presented. Then, the two stages of DRL are illustrated.

A. Scheme Rationale and Overview

The DRL for resource allocation is implemented in the RAM module in the controller of *DeepEdge*. Despite the fact that DRL has a potential to solve the resource allocation problem in the Edge-IoT domain, diversity in action exploration remains a major challenge for DRL in such environment with large state/action space and sparse reward values. It is infeasible to rely on simple look up table of state/action and Q-values, i.e., it is necessary to approximate the Q-value to minimize the complexity of the scheme and account for state/action dimensionality. The sparse reward values can lead the DRL to achieve sub-optimal resource allocation policy. In addition, it is necessary to utilize multi-dimensional data and analyze it to determine the best resource allocation policy. The multi-dimensional data comprises edge server resources availability, IoT applications resource demands, and applications' QoS requirements. To tackle these challenges and leverage the multi-dimensional data for resource allocation, we build a novel two-stage DRL scheme that has the following merits: 1) It exploits DNN to enhance action exploration by mapping the Edge-IoT system state to joint actions of resource allocation and QoS class selection. Q-value of DRL is approximated as a function of smaller set of variables to tackle the large state/action space of Edge-IoT environment. This distinguishes our scheme from the typical DRL schemes which utilize DNN to approximate the value function using temporal difference and train DNN accordingly. 2) The exploration generated actions are ranked according to their Q-values to avoid the equal probability of action selection. This ranking is used to select the DNN training data and balance exploration and exploitation of actions. The balance is achieved using effective action selection probability, which is varied as a graded function of Q-value using Boltzmann distribution [50] such that the best action is given the highest selection probability. 3) It exploits information about QoS requirements of the heterogeneous applications, the resource demands, and the resources availability in actions generation; 4) The scheme generates joint actions including resource allocation with certain QoS class.

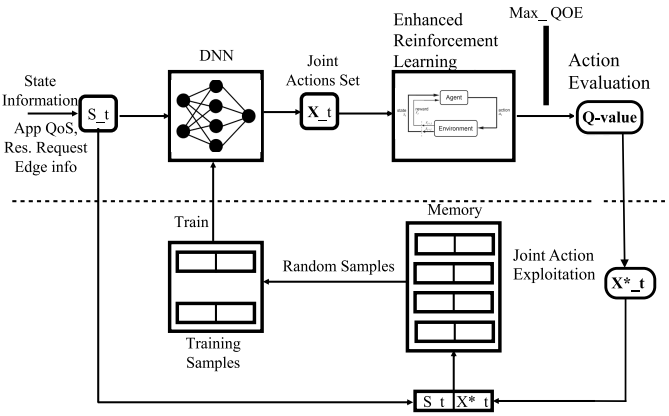

Fig. 2. *DeepEdge* two-stage DRL scheme overview.

Our DRL scheme consists of two stages: 1) Action exploration and evaluation. 2) Action exploitation and DNN training. In the first stage, we employ DNN to generate joint resource allocation and QoS class selection actions. After the generation of the joint actions, reinforcement learning is engaged to evaluate the joint actions and select the ones that have the maximum Q-value, which is defined based on the achieved QoE described earlier in the system model. In the second stage, the joint actions with the highest Q-value during exploration are exploited and stored in a replay memory. The memory is used to train the DNN and update its parameter such that the actions generated in the next iteration are improved. The overview of the two-stage DRL mechanism is shown in Fig. 2. More specifically, in the first stage, the scheme generates joint actions based on the DNN current action policy π_{θ_t}, parameterized by θ_t which is the weight that connects the hidden neurons in DNN. Then, the generated actions during exploration are evaluated using the proposed approximated Q-value.

In the second stage, the best joint actions are selected among the actions generated with certain action selection probability. The state and the corresponding selected joint action with the highest Q-value (S_t, X_t^*) is added to a replay memory. The action policy at DNN is updated by fetching a batch of training samples to train the DNN. After training, DNN updates its weighing parameter θ_t to θ_{t+1} and the action policy $\pi_{\theta_{t+1}}$. The new action policy $\pi_{\theta_{t+1}}$ will be exploited in the next iteration to generate joint actions at $t + 1$. Such a reinforcement learning iteration allows the DNN to continuously improve the quality of the actions generated.

B. Deep Reinforcement Learning Stages

In this section, we illustrate the two stages of DRL for resource allocation joint action generation.

1) First Stage (Action Exploration and Evaluation): In this stage, the DNN receive the Edge-IoT state S_t information at time t defined as the IoT applications resource demand y_i, the QoS requirements and the resources available at the edge $S_t = \{y_i, R_L^{max}, R_E^{max}, T^{max}, C_j\}$. According to the current action policy denoted as $\pi_{\theta_t} : \{S_t\} \rightarrow \mathbf{X}^t$, a set of joint actions is generated by DNN and denoted by a mapping f_{θ_t} as follows,

$$\mathbf{X}^t = f_{\theta_t}(S_t) \tag{9}$$

where $\mathbf{X}_t = \{X_k^t, k = 1, 2, \ldots, K\}$, and $X_k^t = \{x_{r,i,j}^t, \alpha_\varsigma^t\}$ is the kth entry of $\mathbf{X}_t$. Each entry in $\mathbf{X}_t$ is a joint action and is assumed to be continuous. The universal approximation theorem claims that if hidden layers have large number of hidden neurons and a proper activation function is applied at the neurons, they will be sufficient to approximate any continuous mapping f [51]. We exploit ReLU as an activation function [52] of the hidden layers, where the output b and input v of a neuron are related by $b = max\{v, 0\}$. In the output layer, we use sigmoid activation function as $b = 1/(1 + e^{-v})$. It is necessary to map the set of joint actions $\mathbf{X}_t$ to a discrete action set such that the actions can be evaluated by the reinforcement learning Q-value function. We employ typical K-nearest-neighbors (KNN) algorithm [53] to do the mapping. After obtaining the candidates discrete joint actions from KNN, the performance of these actions is evaluated using reinforcement learning. The action evaluation is conducted based on the QoE optimization objective defined in (3).

We assume that the Edge-IoT environment evolves as a discrete-time Markov decision process (DTMDP). The maximization problem in (3) falls within the domain of a DTMDP. In order to find the optimal action policy, we define a DTMDP that associates an action to every Edge-IoT state, a state transition and a reward function. The state transitions and actions occur at discrete time epochs. *DeepEdge* controller monitors the Edge-IoT state S_t in current epoch t and generates discrete joint actions $\mathbf{X}_t$, which are found using DNN. A reward function is generated for each joint action X_t at the end of the epoch. The reward function R_t is selected to be the *QoE* defined in (2). The formal expression for the DTMDP is given as (S, X, T, R), where $T : S \times X \times S' \rightarrow [0, 1]$ is a state transition probability function. Ultimately, the objective of DRL integrated with DTMDP is to find an optimal joint action X_t that maximizes the QoE in (2). The Q-value of the reinforcement learning is exploited to evaluate the joint action is defined as the current expected reward plus a future discounted reward as follows,

$$Q^*(S_t, X_t) = E\left[R(S_t, X_t) + \varphi \max_{X' \in \mathbf{X}_t} Q^*\left(S_t', X_t'\right)\right] \tag{10}$$

where $\varphi \in (0, 1]$ is the discount factor. The optimal Q-value $Q^*(S_t, X_t)$ is updated by the change in the Q-value according to the transition from state S_t to state S_t' under the action X_t at epoch t as follows,

$$Q^{t+1}(S_t, X_t) = \left(1 - \mu^t\right) Q(S_t, X_t) + \mu \left[R(S_t, X_t) + \varphi \max_{X' \in \mathbf{X}} Q\left(S_t', X_t'\right)\right] \tag{11}$$

where $\mu \in [0, 1]$ is the learning rate. Reinforcement learning is a stochastic approximation method that solves the Bellman's optimality equation associated with the DTMDP. It does not require state transition probability model as it converges with probability one to a solution if $\sum_{t=1}^{\infty} \varphi^t$ is infinite, $\sum_{t=1}^{\infty} (\varphi^t)^2$ is finite, and all state/action pairs are visited infinitely often [54].

One of the main shortcomings of using Q-value for action evaluation in the dynamic Edge-IoT environment is the large state space. It is not feasible to use state/action tables and find the corresponding Q-value in such environment for action evaluation. Thus, it is necessary to approximate the Q-value. This approximation reduces the complexity of the system and enhances its convergence. Thus, we approximate the Q-value as a function of a smaller set of variables in which Q-value utilizes a countable state space S^* using the function $Q' : S^* \times X$. This function is referred as a function approximator. The vector $\rho = \{\rho^p\}_{p=1}^P$ is exploited to approximate the Q-value by minimizing the metric of difference between $Q^*(S_t, X_t)$ and $Q'(S_t, X_t, \rho)$ for all $(S_t, X_t) \in S^* \times X$. Thus, the approximated Q' value is formalized as, $Q'(S_t, X_t, \rho) = \sum_{p=1}^P \rho^p \psi_p(S_t, X_t) = \rho \psi^T(S_t, X_t)$ where T denotes the transpose operator and the vector $\psi(S_t, X_t) = [\psi_p(S_t, X_t)_{p=1}^P]$ with a scalar function $\psi_p(S_t, X_t)$ that is identified as the basis function (BF) over $S^* \times \chi$, and $\rho^p (p = 1, \ldots, P)$ are the associated weights. We use Stochastic Gradient Descent (SGD) method to update the weights. The Q-value update rule in (11) is redefined as follows,

$$\rho_{t+1}\psi^T(S_t, X_t) = \left\{\left(1 - \mu^t\right)\rho_t \psi^T(S_t, X_t) + \mu^t \left[R(S_t, X_t) + \varphi \max_{X' \in \mathbf{X}} \rho_t \psi^T\left(S_t', X_t'\right)\right]\right\} \times \psi(S_t, X_t) \tag{12}$$

where the gradient is a vector of partial derivatives with respect to the elements of ρ_t.

2) Second Stage (Action Exploitation and DNN Training): The action with the highest Q-value X_t^* must be exploited among other actions of state S_t and added to the replay memory to train the DNN. The replay memory will be populated with state/action pairs that have the highest Q-value over certain number of iterations. The action exploitation is accomplished through determination of action policy (π_ς^t), which is defined as the probability of selection of action X_t at state S_t. It corresponds to the set of actions with the highest Q-value. The attainment of this policy is tied to resolving the exploration vs. exploitation tradeoff. Exploration aims to look for new joint actions so it does not only utilize the actions known to achieve high Q-value. Exploitation is the process of using the good actions available. The most common method to balance exploration and exploitation is to use the ϵ-greedy selection [48], where ϵ is the portion of the time that a learning agent takes a randomly selected action instead of taking the action that is most likely to maximize its reward given the actions available. However, ϵ-greedy selects equally among the available actions, i.e., the worst action is likely to be chosen as the best one. In order to overcome this issue, we develop a new method in which the action selection probabilities are varied as

a graded function of Q-value. The best joint action is given the highest selection probability while others are ranked according to their Q-values. Boltzmann distribution [50] is adopted to achieve this ranking. The action selection probability at epoch t is given as follows,

$$\pi_{\varsigma}^{*}(S_t, X_t) = \frac{e^{Q(S_t,X_t)/\tau}}{\sum_{X' \in \mathbf{X}} e^{Q(S_t',X_t')/\tau}} \tag{13}$$

where τ is a positive parameter which can take high value and this indicates that the actions probabilities nearly equal. In case τ has low values, this indicated a big difference in selection probabilities for actions with different Q-values. This action selection probability is updated after Q-value approximation as follows,

$$\pi_{\varsigma}^{*}(S_t, X_t) = \frac{e^{\rho_t \psi^T(S_t,X_t)/\tau}}{\sum_{X \in \mathbf{X}} e^{\rho_t \psi^T(S_t,X_t)/\tau}}. \tag{14}$$

The selected state/actions pairs are added to the memory at each epoch and utilized later to train the DNN. This improves the upcoming joint actions that will be generated by the DNN in the future epoch. To achieve this, *DeepEdge* maintains an initially empty memory of limited capacity. At the t-th epoch, a new training data (S_t, X_t^*) is added to the memory. When the memory is full, the newly generated data sample replaces the oldest one. The experience replay technique [42], [55] is utilized to train the DNN using the stored data samples. After certain number of epochs when there is enough data to train the DNN, we randomly select a group of training data samples $\{(S_\upsilon, X_\upsilon^*)|\upsilon \in \Upsilon_t\}$ from the memory, where Υ is the set of selected time indices. The DNN parameter θ_t is updated using Adam algorithm [56] which targets minimization of the average cross-entropy loss $L(\theta_t)$ defined as follows,

$$L(\theta_t) = -\frac{1}{|\Upsilon_t|} \sum_{\upsilon \in \Upsilon_t} \Big[(X_\upsilon^*)^T \log f_{\theta_t}(S_\upsilon) + (1 - X_\upsilon^*)^T \log\big(1 - f_{\theta_t}(S_\upsilon)\big) \Big] \tag{15}$$

where $|\Upsilon_t|$ is the size of Υ_t, T denotes the transpose operator, and the log function is the element-wise logarithm operation for a vector. We start the training step when the number of samples is larger than half of the memory size. Eventually, the DNN learns the best joint action for each state (S_t, X_t^*). Thus, it becomes smarter and continuously improves its produced joint action.

The two-stage DRL for resource allocation procedure is presented in Algorithm 1. The algorithm acquires the Edge-IoT state information which includes QoS requirements, resource demand and edge servers resources capacity information. It starts by initializing the DNN with certain parameter θ. The DNN generates the joint actions. The output of DNN is converted to discrete format and then received by the approximated reinforcement learning to evaluate the generated actions by the DNN. The actions with the highest Q-value are exploited according to the probability in (14) and used to populate the dedicated memory of DNN. After certain number of epoch, a sample of state/action pairs is fetched from the memory and used for DNN training and updating θ using Adam algorithm.

Algorithm 1 Two-Stage Deep Reinforcement Learning Algorithm to Solve Resource Allocation in *DeepEdge*

Require: Network state S_t which include QoS requirements of the application, resource demands edge servers resources capacity at each epoch t

Ensure: Joint action for resource allocation and QoS metric class $X_t = \{X_t^*, \alpha_\varsigma^t\}$

BEGIN
Initialize the DNN with random parameters θ_t and empty replay memory
Set iteration number m and the training interval Ω
for (t=1 to m) **do**
  Generate a set of joint actions $\mathbf{X}_t = f_{\theta_t}(S_t)$
  Use KNN to convert the continuous set of actions into a discrete set
  Run Approximated reinforcement learning to evaluate the action for resource allocation that must satisfy $X_t^* = \max_X \rho\psi^T(S_t, X_t)$
  Exploit actions according (14)
  Update the memory by adding (S_t, X_t^*)
  if $\Omega = 1$ **then**
    Uniformly select a group of data samples $\{(S_\upsilon, X_\upsilon)|\upsilon \in \Upsilon_t\}$ from the memory
    Train the DNN with $\{(S_\upsilon, X_\upsilon)|\upsilon \in \Upsilon_t\}$ and update θ_t using Adam algorithm
  end if
end for
END

The complexity of the proposed two-stages DRL is found based on the number of edge servers *J*, the number of available resources of certain type *r*, and the number of devices that demand the resources *N*. The implementation of the DRL algorithm considers different application and scenarios. It associates actions generation for the device with the available resources and edge servers. The computation complexity of the action exploration stage of the DRL is $O(JN^r)$ operations. The complexity of the exploitation and training stage is $O(m\Omega)$ according to the number of epoch m and the training interval Ω. The memory requirements to store the samples for DNN training is $N^{(r*J)}$. Exploration and exploitation are achieved with the merit of the approximated Q-value $O(Q'\theta_t(S_t, X_t, \rho))$ instead of the typical Q-value in the traditional Q-learning. The computation complexity of our proposed two-stages DRL is acceptable given the achieved performance and in comparison with the traditional Q-learning which has a an exponential computational complexity of $O(N^{J*r})$. The traditional Q-learning may only achieve maximum achievable QoE by searching all possible combinations of state/action/rewards. Consequently, it requires more number of operations and its computation complexity escalates in an exponential pattern.

V. Performance Evaluation

We evaluate the performance of the proposed *DeepEdge* for resource allocation in Edge-IoT with respect to the average

TABLE III
SYSTEM PARAMETERS

Parameter	Value
Symbol time	4.5 μs
Frame length	600 symbols
Block-length of uplink	200 symbols
CPU cycles	3×10^9 cycles/s
Learning rate μ	0.5
Discount factor φ	0.9
Number of hidden Layers	2

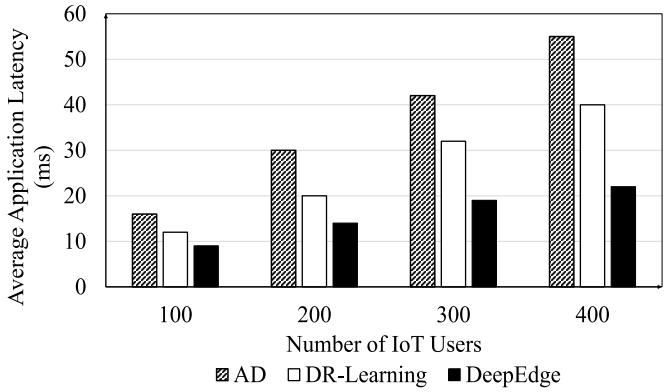

Fig. 3. Latency for emergency response application.

application's latency, achieved users' QoE, and the average application tasks success ratio.

A. Evaluation Setup

We consider a network that consists of 10 edge servers uniformly distributed in the network. Each server is equipped with a 3-core CPU where the CPU cycle frequency of each core is 3×10^9 cycles per second. The frame length is 600 symbols where the time of one symbol is 4.5 μs. The block-length of uplinks are all assumed to be equal to 200 symbols. The number of IoT users is assumed as $N \in [100, 400]$, randomly distributed within the network. The bandwidth available for sharing is set to 10 MHz. Applications' latency requirement and data size, as well as the corresponding CPU cycles, are determined by the specific IoT application type. We consider the three applications described in the system model and their corresponding QoS requirements. The DRL parameters and rest of simulation parameters are presented in Table III. The application task data size is set as a uniform distribution, [2, 8] MB, and corresponding CPU cycles is variable.

B. Application Latency Evaluation

We evaluate the performance of *DeepEdge* in terms of the average encountered latency for certain application by varying the number of IoT users starting from 100 users where 50% of the users run the evaluated application and 50% run the other two applications with 25% of the users for each. Fig. 3, 4, and 5 present the average latency for the three applications: emergency response, health monitoring and personal identification respectively with variable number of IoT users. The achieved latency is compared to the resource allocation schemes: DQN-based (AD) [31] and actor-critic (DR-Leanring) [32] presented in the related

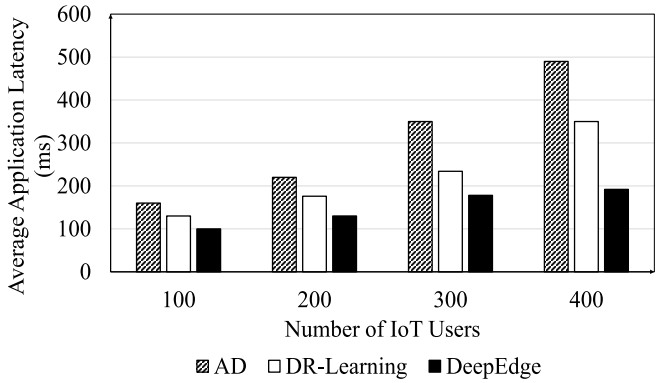

Fig. 4. Latency for health monitoring application.

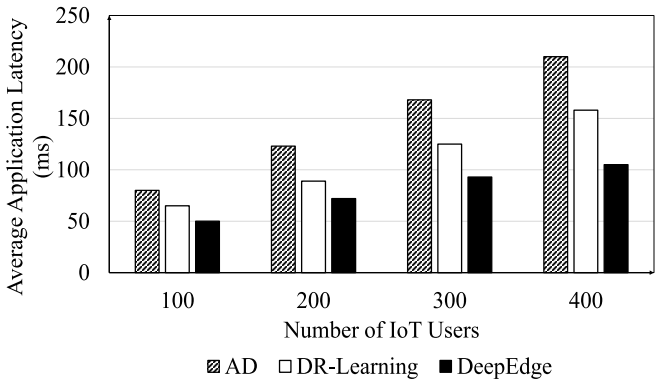

Fig. 5. Latency for personal identification application.

TABLE IV
EVALUATIONS OF DEEPEDGE VS OPTIMAL EXHAUSTIVE SEARCH

Application	Exhaustive Latency (ms)	DeepEdge Latency (ms)
Emergency Response	1.123 ms	1.165 ms
Health Monitoring	13.24 ms	13.386 ms
Personal Identification	5.878 ms	6.053 ms

work. We increase the number of users from 100 to 400 by step of 100 to show the change of latency. The results show that our proposed *DeepEdge* achieves the best result in terms application latency comparing with the other schemes. Moreover, the latency is maintained low in comparison to other schemes even with large number of users involved.

In addition, we compare the performance of *DeepEdge* in terms of latency against the optimal exhaustive search resource allocation for the three applications: emergency response, health monitoring and personal identification. Exhaustive search requires searching through all the possible resource allocation possibilities. It is impractical in the considered edge-IoT applications where the search becomes complicated and consumes significant time as the system scale grows in terms of the numbers of IoT devices, edge server and edge resources. Table IV presents the recorded latency for 20 devices which is a small number given the mentioned applications. We only notice a minimal difference in the latency for different applications between *DeepEdge* and the optimal scheme given the small number of devices.

TABLE V
EVALUATIONS OF DEEPEDGE RESOURCE ALLOCATION SCENARIOS

Scenario	Number of users	Application	QoS requirements (max)			QoS Achieved		
			LA	PLR	PER	LA	PLR	PER
First (Single Application)	100	Emergency Response	50 ms	10^{-2}	10^{-2}	39.31 ms	0.00648	0.00425
Second (Application Heterogeneity)	200	Emergency Response	50 ms	10^{-2}	10^{-2}	41.652 ms	.00741	0.00524
		Health Monitoring	180 ms	10^{-4}	10^{-4}	100.486 ms	0.0000845	0.0000654
Third (Users and Application Heterogeneity)	300	Emergency Response	50 ms	10^{-2}	10^{-2}	43.652 ms	.00827	0.00664
		Health Monitoring	180 ms	10^{-4}	10^{-4}	111.954 ms	0.0000886	0.0000732
	400	Emergency Response	50 ms	10^{-2}	10^{-2}	45.147 ms	.00894	0.00712
		Health Monitoring	180 ms	10^{-4}	10^{-4}	118.784 ms	0.0000912	0.0000792
		Personal Identification	100 ms	10^{-3}	10^{-3}	69.857 ms	0.000787	0.000839

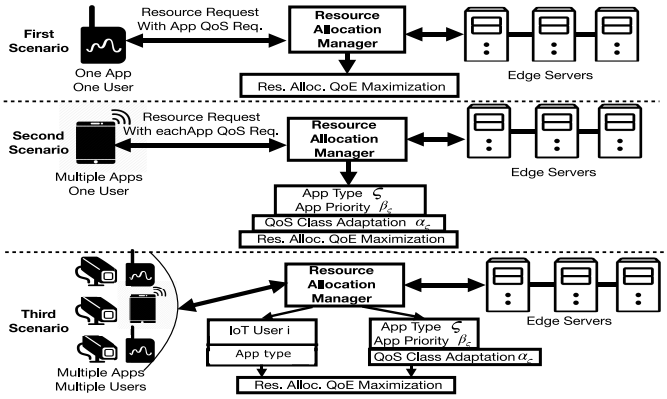

Fig. 6. *DeepEdge* resource allocation scenarios.

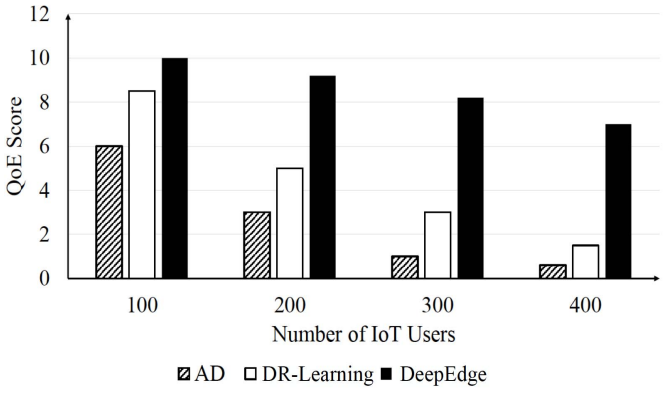

Fig. 7. QoE for multiple applications scenarios.

C. Evaluation of Various DeepEdge Resource Allocation Scenarios

In this subsection, we discuss and evaluate multiple scenarios of how *DeepEdge* operates to perform resource allocation with QoE maximization. Fig. 6 depicts the scenarios of resource allocation for multiple heterogeneous applications. For the first scenario, it has 100 IoT users which run emergency response application with high QoS requirements. The resources requests of emergency response application are sent to the RAM in the controller. The request is processed through the two-stage DRL by selecting the most appropriate QoS class α_ς and allocate edge resources accordingly. In the second scenario (application heterogeneity), it is assumed that each one of 200 IoT user runs two applications (emergency response and personal identification), which lets the controller treat all the IoT users the same. The RAM here receives requests from the same user but for multiple applications. It recognizes the application type ς, identifies the applications priority β_ς and analyzes their QoS requirements. Then, it enforces the application QoS class adaptation starting with the lower priority application. For example, the QoS class of the personal identification application that has the lowest priority will be adapted first through proper election of its α_ς. The two-stage DRL allocates the resources for both applications with the goal of maximizing the QoE in (3). In the third scenario (users and application heterogeneity), we present two evaluation examples: First, there are 300 heterogeneous IoT users of which 100 users are running emergency response, 100 users for health monitoring and 100 with two applications emergency response and health care monitoring. In the second example: there are 400 IoT users of which 100 users emergency response, 100 users health monitoring, 100 users with personal identifications, and 100 users running the three applications simultaneously. All the users report their requests along with the QoS requirements of applications to the RAM at the controller. All the requests are sorted according to the users index *i* and application type ς. Then, the RAM allocates resources to these applications with consideration of application priority β_ς and resource availability at the edge. These parameters are exploited by the two-stage DRL to adapt QoS class $\alpha_{i,\varsigma}$ and allocate resources accordingly with the goal to maximize the joint QoE for all users and satisfaction of their applications. Table V presents the specifications of the three scenarios, QoS metrics requirements and the average achieved metrics by *DeepEdge* for each application. We observe that *DeepEdge* always maintains the QoS metrics below the specified threshold even in the most complicated setting of the third scenario.

Moreover, QoE is evaluated with consideration of the different scenarios presented in Table V to demonstrate *DeepEdge* capability to tackle the heterogeneity of IoT applications in resource allocation. The QoE function derived in (2) is exploited as an evaluation metric to demonstrate the merit of the proposed two-stages DRL against other DRL schemes: the DQN-based scheme (AD) [31] and the actor critic scheme (DR-Learning) [32]. However, the QoE function for the AD and DR-Learning schemes is calculated using the quality score of the application latency only (not including quality scores for PLR and PER) as it is the only QoS metric they considered as an optimization goal. The average QoE is plotted in Fig. 7. Fig. 7 shows that *DeepEdge* outperforms both schemes as they lack the capability of handling multiple applications

TABLE VI
EVALUATIONS OF RUNTIME FOR ALL SCHEMES IN DIFFERENT SCENARIOS

Scenario	Number of IoT Users	Runtime (s)		
		AD	DR-Learning	DeepEdge
First (Single Application)	100	2.154	1.656	1.325
Second (Application Heterogeneity)	200	4.946	3.783	2.913
Third (Users and Application Heterogeneity)	300	10.285	7.794	5.312
	400	15.23	12.062	7.845

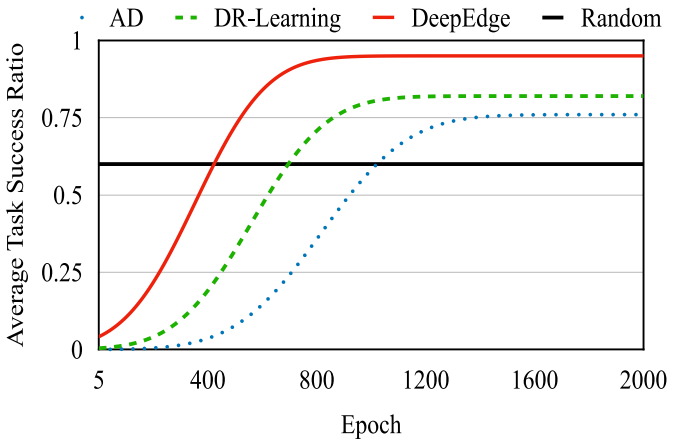

Fig. 8. Average task success ratio convergence.

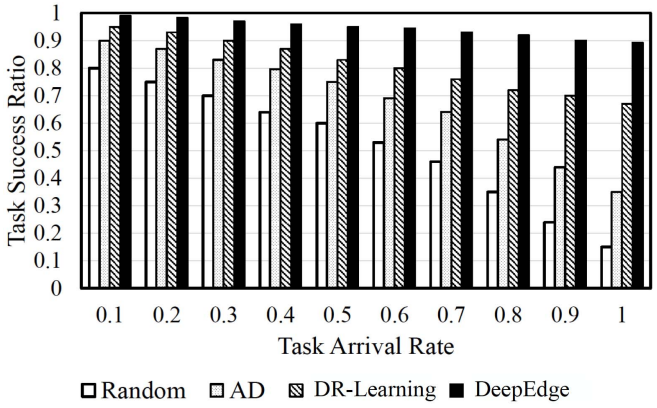

Fig. 9. Average task success ratio vs. arrival rate.

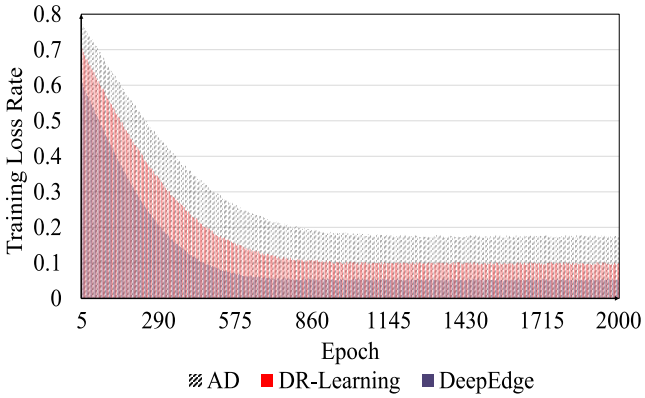

Fig. 10. Normalized Training Loss rate.

running on large number of devices. We notice that in the first scenario, other schemes achieve comparable performance as only one application is running. The QoE decreases as the number of users increases which is expected as the competition between users for resources increases. However, the drop of QoE in *DeepEdge* as the number of users increased with variety of applications is not significant in comparison to the other schemes.

In addition, we compare the system runtime for each scenario settings for all schemes. Table VI presents the runtime in seconds for each scheme in each scenario that correspond to the achieved QoE in Fig. 7. We notice that *DeepEdge* records the lowest runtime in comparison to other resource allocations schemes in all scenarios with considerable difference in the most complicated scenario with 400 devices. Thanks to the enhanced design of the developed two stages DRL.

D. Task Success Ratio

Another evaluation factor considered in this paper is the task success ratio, which is the ratio of the application's tasks with satisfied QoS requirements to the total number of running application's tasks. We adopt the settings of the second scenario in Table V. Fig. 8 presents the average task success ratio of the proposed *DeepEdge* with average application's resources request rate of 0.5. It is observed that the performance is improved gradually with learning as the system becomes familiar with the environment and capable to make better resource allocation decisions. Moreover, we evaluate the task success ratio against the variable task arrival rates in Fig. 9. It shows that *DeepEdge* outperforms other allocation schemes and maintains the success ratio above 0.9 regardless of the increase in the task arrival rates. Other schemes' success ratios fall dramatically as the tasks arrival rate evolves.

E. Convergence and Training Evaluation and Discussion

We evaluate the performance of the DNN utilized in *DeepEdge* in terms of the training losses. The evaluation shows the training quality of DNN in *DeepEdge* as the resource allocation proceeds. We plot the training loss rate of our proposed DRL in Fig. 10 and compare it to other allocation schemes. The training loss rate gradually decreases and stabilizes at around 0.04. We clearly notice that DRL developed in DeepEdge converges faster and with lower training loss rate comparing to the DQN in AD [31] and actor-critic in DR-Learning [32]. The convergence speed of *DeepEdge* is evaluated in terms of the achieved normalized QoE with respect to the number of epoch as in Fig. 11. We observe that the moving average QoE of *DeepEdge* gradually converges to the maximum. Specifically, the achieved average QoE exceeds 0.98 and the variance gradually decreases to zero as iteration becomes larger. We adopt the settings of the second scenario in Table IV for the evaluation of training losses and convergence.

The evaluation of *DeepEdge* shows that it outperforms other resource allocation schemes. The reason for that is the consideration of multiple heterogeneous applications in the proposed QoE model, which aims to guarantee IoT users satisfaction through fulfillment of different applications' QoS

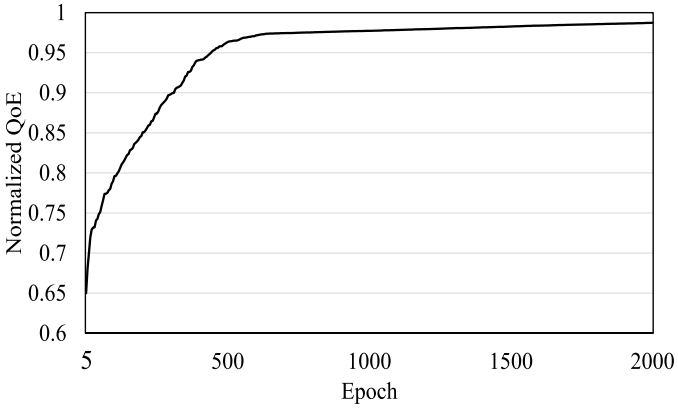

Fig. 11. Normalized QoE for *DeepEdge*.

requirements. The consideration of aligning the IoT applications' requirements with the available resources at the edge has a tremendous contribution to the achieved performance. In addition, the developed two-stage DRL adds the following advantages to *DeepEdge*. 1) It benefits from historical actions to foster the framework experience. 2) It generates joint actions and enhances the diversity of actions at the exploration stage using DNN. 3) The Q-value approximation reduces the complexity of the system which can be noticed at the convergence speed in comparison to other schemes.

VI. Conclusion

Edge computing comes into practice as a potential solution to tackle the IoT applications resource demanding issue in a fast manner. Resource allocation in the context of edge computing becomes important as there can be many heterogeneous IoT applications competing for limited resources at the edge. The paper has tackled the resource allocation problem in Edge-IoT environment in a way that fulfills the IoT applications' requirements and maximizes IoT users' satisfaction. We developed the *DeepEdge* framework which comprises a novel QoE model to quantify the IoT users satisfactions based on the QoS requirements of applications. *DeepEdge* employs a novel two-stage DRL scheme which learns by reinforcement resource allocation policy that maximizes users' QoE, and tunes the application requirements to align with the available edge resources. Moreover, *DeepEdge* exploits DNN to generate joint actions and utilizes historical allocation decision to improve its generated actions and expedite the system convergence. Evaluation results demonstrate *DeepEdge*'s capability in optimizing users QoE and maintaining task success ratio at the maximum.

References

[1] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[2] J. Pan and J. McElhannon, "Future edge cloud and edge computing for Internet of Things applications," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 439–449, Feb. 2018.

[3] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. 1st MCC Workshop Mobile Cloud comput.*, 2012, pp. 13–16.

[4] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[5] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan, and K. K. Leung, "Dynamic service migration in mobile edge-clouds," in *Proc. IEEE IFIP Netw. Conf. (IFIP Networking)*, 2015, pp. 1–9.

[6] R. Ranjan, B. Benatallah, S. Dustdar, and M. P. Papazoglou, "Cloud resource orchestration programming: Overview, issues, and directions," *IEEE Internet Comput.*, vol. 19, no. 5, pp. 46–56, Sep./Oct. 2015.

[7] L. Liu, Z. Chang, X. Guo, S. Mao, and T. Ristaniemi, "Multi-objective optimization for computation offloading in fog computing," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 283–294, Feb. 2018.

[8] X. Sun and N. Ansari, "Latency aware workload offloading in the cloudlet network," *IEEE Commun. Letter.*, vol. 21, no. 7, pp. 1481–1484, Jul. 2017.

[9] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 1, no. 2, pp. 89–103, Jun. 2015.

[10] X. Lyu, H. Tian, C. Sengul, and P. Zhang, "Multiuser joint task offloading and resource optimization in proximate clouds," *IEEE Trans. Veh. Technol.*, vol. 66, no. 4, pp. 3435–3447, Apr. 2017.

[11] X. Chen, W. Li, S. Lu, Z. Zhou, and X. Fu, "Efficient resource allocation for on-demand mobile-edge cloud computing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 9, pp. 8769–8780, Sep. 2018.

[12] H. Zhang, Y. Xiao, S. Bu, D. Niyato, F. R. Yu, and Z. Han, "Computing resource allocation in three-tier IoT Fog networks: A joint optimization approach combining stackelberg game and matching," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1204–1215, Oct. 2017.

[13] L. Huang, X. Feng, C. Zhang, L. Qian, and Y. Wu, "Deep reinforcement learning-based joint task offloading and bandwidth allocation for multi-user mobile edge computing," *Digit. Commun. Netw.*, vol. 5, no. 1, pp. 10–17, 2019.

[14] J. Li, H. Gao, T. Lv, and Y. Lu, "Deep reinforcement learning based computation offloading and resource allocation for MEC," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, 2018, pp. 1–6.

[15] F. D. Vita, D. Bruneo, A. Puliafito, G. Nardini, A. Virdis, and G. Stea, "A deep reinforcement learning approach for data migration in multi-access edge computing," in *Proc. ITU Kaleidoscope Mach. Learn. 5G Future (ITU K)*, 2018, pp. 1–8.

[16] J. Wang, J. Hu, G. Min, W. Zhan, Q. Ni, and N. Georgalas, "Computation offloading in multi-access edge computing using a deep sequential model based on reinforcement learning," *IEEE Commun. Mag.*, vol. 57, no. 5, pp. 64–69, May 2019.

[17] M. Li, J. Gao, L. Zhao, and X. Shen, "Deep reinforcement learning for collaborative edge computing in vehicular networks," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, no. 4, pp. 1122–1135, Dec. 2020.

[18] Y. Liu, H. Yu, S. Xie, and Y. Zhang, "Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 11, pp. 11158–11168, Nov. 2019.

[19] G. Qiao, S. Leng, S. Maharjan, Y. Zhang, and N. Ansari, "Deep reinforcement learning for cooperative content caching in vehicular edge computing and networks," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 247–257, Jan. 2020.

[20] Q. Luo, C. Li, T. H. Luan, and W. Shi, "Collaborative data scheduling for vehicular edge computing via deep reinforcement learning," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9637–9650, Oct. 2020.

[21] Q. Qi and Z. Ma, "Vehicular edge computing via deep reinforcement learning," 2019, *arxiv:1901.04290*.

[22] D. Zeng, L. Gu, S. Pan, J. Cai, and S. Guo, "Resource management at the network edge: A deep reinforcement learning approach," *IEEE Netw.*, vol. 33, no. 3, pp. 26–33, May/Jun. 2019.

[23] X. Liu, Z. Qin, and Y. Gao, "Resource allocation for edge computing in IoT networks via reinforcement learning," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Shanghai, China, 2019, pp. 1–6.

[24] J. Zhao, M. Kong, Q. Li, and X. Sun, "Contract-based computing resource management via deep reinforcement learning in vehicular fog computing," *IEEE Access*, vol. 8, pp. 3319–3329, 2020.

[25] Z. Ning, P. Dong, X. Wang, J. Rodrigues, and F. Xia, "Deep reinforcement learning for vehicular edge computing: An intelligent offloading system," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 6, p. 60, 2019.

[26] W. Zhan, C. Luo, J. Wang, G. Min, and H. Duan, "Deep reinforcement learning-based computation offloading in vehicular edge computing," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Waikoloa, HI, USA, 2019, pp. 1–6.

[27] M. Khayyat, I. A. Elgendy, A. Muthanna, A. S. Alshahrani, S. Alharbi, and A. Koucheryavy, "Advanced deep learning-based computational offloading for multilevel vehicular edge-cloud computing networks," *IEEE Access*, vol. 8, pp. 137052–137062, 2020.
[28] H. Peng and X. Shen, "Deep reinforcement learning based resource management for multi-access edge computing in vehicular networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 4, pp. 2416–2428, Oct.–Dec. 2020.
[29] Q. Qi *et al.*, "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.
[30] X. Xiong, K. Zheng, L. Lei, and L. Hou, "Resource allocation based on deep reinforcement learning in IoT edge computing," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 1133–1146, Jun. 2020.
[31] J. Wang, L. Zhao, J. Liu, and N. Kato, "Smart resource allocation for mobile edge computing: A deep reinforcement learning approach," *IEEE Trans. Emerg. Topics Comput.*, vol. 9, no. 3, pp. 1529–1541, Jul.–Sep. 2021.
[32] Y. Wei, F. R. Yu, M. Song, and Z. Han, "Joint optimization of caching, computing, and radio resources for fog-enabled IoT using natural actor-critic deep reinforcement learning," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2061–2073, Apr. 2019.
[33] H. Zhang, W. Wu, C. Wang, M. Li, and R. Yang, "Deep reinforcement learning-based offloading decision optimization in mobile edge computing," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, 2019, pp. 1–7.
[34] H. Meng, D. Chao, and Q. Guo, "Deep reinforcement learning based task offloading algorithm for mobile-edge computing systems," in *Proc. 4th Int. Conf. Math. Artif. Intell. (ICMAI)*, 2019, 90–94.
[35] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Performance optimization in mobile-edge computing via deep reinforcement learning," 2018, *arXiv:1804.00514*.
[36] T. Yang, Y. Hu, M. C. Gursoy, A. Schmeink, and R. Mathar, "Deep reinforcement learning based resource allocation in low latency edge computing networks," in *Proc. Int. Symp. Wireless Commun. Syst. (ISWCS)*, Lisabon, Portugal, Aug. 2018, pp. 1–5.
[37] Y. Xiao, M. Noreikis, and A. Ylä-Jaäiski, "QoS-oriented capacity planning for edge computing," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2017, pp. 1–6.
[38] Z. Ye, S. Mistry, A. Bouguettaya, and H. Dong, "Long-term QoS-aware cloud service composition using multivariate time series analysis," *IEEE Trans. Services Comput.*, vol. 9, no. 3, pp. 382–393, May/Jun. 2016.
[39] R. Mahmud, S. Srirama, K. Ramamohanarao, and R. Buyya, "Quality of experience (QoE)-aware placement of applications in Fog computing environments," *J. Parallel Distrib. Comput.*, vol. 132, no. 3, pp. 190–203, Oct. 2019.
[40] Y. Lu, M. Motani, and W.-C. Wong, "A QoE-aware resource distribution framework incentivizing context sharing and moderate competition," *IEEE/ACM Trans. Netw.*, vol. 24, no. 3, pp. 1364–1377, Jun. 2016.
[41] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.
[42] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.
[43] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *Proc. 33rd Int. Conf. Mach. Learn.*, Jun. 2016, pp. 1928–1937.
[44] S. Basso, M. Meo, A. Servetti, and J. De Martin, "Estimating packet loss rate in the access through application-level measurements," in *Proc. ACM SIGCOMM Workshop Meas. Stack (W-MUST)*, 2012, pp. 7–12.
[45] B. Han and S. Lee, "Efficient packet error rate estimation in wireless networks," in *Proc. 3rd Int. Conf. Testbeds Res. Infrastruct. Develop. Netw. Commun.*, 2007, pp. 1–9.
[46] K. Nagin, A. Kassis, D. Lorenz, K. Barabash, and E. Raichstein, "Estimating client QoE from measured network QoS," in *Proc. 12th ACM Int. Conf. Syst. Storage (SYSTOR)*, 2019, p. 188.
[47] T. Hoßfeld, P. E. Heegaard, L. Skorin-Kapov, and M. Varela, "Fundamental relationships for deriving QoE in systems," in *Proc. 11th Int. Conf. Qual. Multimedia Exp. (QoMEX)*, 2019, pp. 1–6.
[48] M. Tokic, "Adaptive ε-greedy exploration in reinforcement learning based on value differences," *Advances in Artificial Intelligence* (Lecture Notes in Computer Science), vol. 6359. Heidelberg, Germany: Springer, 2010.
[49] D. D. Hora, A. Asrese, V. Christophides, R. Teixeira, and D. Rossi, "Narrowing the gap between QoS metrics and Web QoE using above-the-fold metrics," in *Passive and Active Measurement*. Cham, Switzerland: Springer, 2018.
[50] A. D. Tijsma, M. M. Drugan, and M. A. Wiering, "Comparing exploration strategies for q-learning in random stochastic mazes," in *Proc. IEEE Symp. Series Comput. Intell. (SSCI)*, Dec. 2016, pp. 1–8.
[51] S. Marsland, *Machine Learning: An Algorithmic Perspective*. New York, NY, USA: CRC Press, 2015.
[52] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn. (ICML)*, Jun. 2010, pp. 807–814.
[53] K. Fukunaga and P. M. Narendra, "A branch and bound algorithm for computing k-nearest neighbors," *IEEE Trans. Comput.*, vol. 100, no. 7, pp. 750–753, Jul. 1975.
[54] C. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, no. 3, pp. 279–292, 1992.
[55] J. Lin, "Reinforcement learning for robots using neural networks," School Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, USA, Rep. CMU-CS-93-103, 1993.
[56] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2015, pp. 1–15.

Ismail AlQerm (Member, IEEE) received the Ph.D. degree in computer science from the King Abdullah University of Science and Technology (KAUST) in 2017. He is a Postdoctoral Research Associate with the Department of Computer Science, University of Missouri–St. Louis. His research interests include edge computing, resource allocation in IoT networks, developing machine learning techniques for resource allocation in wireless networks, and software defined radio prototypes. He was among the recipients of KAUST Provost Award. He is a member of ACM.

Jianli Pan (Senior Member, IEEE) received the M.S. degree in information engineering from the Beijing University of Posts and Telecommunications, China, and the M.S. and Ph.D. degrees from the Department of Computer Science and Engineering, Washington University at St. Louis, USA. He is currently an Associate Professor with the Department of Computer Science, University of Missouri–St. Louis, St. Louis, MO, USA. His current research interests include Internet of Things, edge computing, machine learning, cybersecurity, and smart energy. He is an Associate Editor for *IEEE Communication Magazine* and IEEE ACCESS.