Learning-Based Computation Offloading for IoT Devices With Energy Harvesting

Minghui Min, *Student Member, IEEE*, Liang Xiao, *Senior Member, IEEE*, Ye Chen, Peng Cheng, *Member, IEEE*, Di Wu, *Senior Member, IEEE*, and Weihua Zhuang, *Fellow, IEEE*

***Abstract*—Internet of Things (IoT) devices can apply mobile edge computing (MEC) and energy harvesting (EH) to provide high-level experiences for computational intensive applications and concurrently to prolong the lifetime of the battery. In this paper, we propose a reinforcement learning (RL) based offloading scheme for an IoT device with EH to select the edge device and the offloading rate according to the current battery level, the previous radio transmission rate to each edge device, and the predicted amount of the harvested energy. This scheme enables the IoT device to optimize the offloading policy without knowledge of the MEC model, the energy consumption model, and the computation latency model. Further, we present a deep RL-based offloading scheme to further accelerate the learning speed. Their performance bounds in terms of the energy consumption, computation latency, and utility are provided for three typical offloading scenarios and verified via simulations for an IoT device that uses wireless power transfer for energy harvesting. Simulation results show that the proposed RL-based offloading scheme reduces the energy consumption, computation latency, and task drop rate, and thus increases the utility of the IoT device in the dynamic MEC in comparison with the benchmark offloading schemes.**

***Index Terms*—Mobile edge computing, energy harvesting, reinforcement learning, Internet of Things, computation offloading.**

Manuscript received April 22, 2018; revised October 19, 2018; accepted December 25, 2018. Date of publication January 1, 2019; date of current version February 12, 2019. This work was supported in part by the National Natural Science Foundation of China under Grants 61671396 and 91638204 and in part by the open research fund of the National Mobile Communications Research Laboratory, Southeast University (2018D08). The work of P. Cheng was supported by the National Natural Science Foundation of China under Grants 61761136012 and 61533015. The work of D. Wu was supported in part by the National Natural Science Foundation of China under Grant 61572538, in part by the Fundamental Research Funds for the Central Universities under Grant 17LGJC23, and in part by the Guangdong Special Support Program under Grant 2017TX04X148. The review of this paper was coordinated by Prof. S. De. *(Corresponding author: Liang Xiao.)*

M. Min, L. Xiao, and Y. Chen are with the Department of Information and Communication Engineering, Xiamen University, Xiamen 361005, China (e-mail: minminghuismile@gmail.com; lxiao@xmu.edu.cn; 885486739@qq.com).

P. Cheng is with the State Key Laboratory of Industrial Control Technology, Zhejiang University, Hangzhou 310027, China (e-mail: pcheng@iipc.zju.edu.cn).

D. Wu is with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China, and also with the Guangdong Key Laboratory of Big Data Analysis and Processing, Guangzhou 510006, China (e-mail: wudi27@mail.sysu.edu.cn).

W. Zhuang is with the Department Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: wzhuang@uwaterloo.ca).

Digital Object Identifier 10.1109/TVT.2018.2890685

I. Introduction

With limited computation, energy and memory resources, Internet of Things (IoT) devices such as sensors, cameras and wearable devices have a computation bottleneck to limit the support for advanced applications such as interactive online gaming and facial recognition applications [1], [2]. This challenge can be addressed by mobile edge computing (MEC) techniques [3], in which IoT devices offload the computation tasks to edge devices such as the base stations (BSs), access points (APs), laptops and even smartphones within the radio access of the IoT devices. By utilizing the computation, cache and energy resources of edge devices, computation offloading can reduce the computation latency, conserve battery resources, and even enhance security for the computation intensive IoT applications [4], [5].

Energy harvesting (EH) is another promising technique that prolongs battery lifetime and provides a satisfactory quality of experiences for IoT devices [6]–[9]. The EH module enables an IoT device to capture the ambient renewable energy such as solar radiation, wind power generation, radio-frequency (RF) signals, and kinetic human motion to supply the "greener" energy for the IoT central processing unit (CPU) and the radio transceiver.

One critical problem in the computation offloading is the selection of the edge device from all the potential radio device candidates within the radio coverage area [10]. This is further complicated by the determination of offloading rate, i.e., the quantity of computation tasks to offload to the edge device [11], [12]. An IoT device often requires a longer period of time to transmit the offloading data and receive the computation result compared with the local computation within the IoT device. This is further complicated by the chosen edge device carrying out heavy workloads and experiencing degraded radio channel fading and interference [9]. It is therefore challenging for an IoT device to optimize the offloading policy within the dynamic MEC network with time variant radio link transmission rates [13], especially with an unknown amount of the renewable energy within a given time duration.

The complications associated with offloading have recently attracted the attention of researchers. For example, the mobile offloading scheme as proposed in [14] uses the Lyapunov optimization to minimize the worst-case expected computation cost with a single known MEC server, based on the knowledge of both the transmission delay model and the local execution model. The online learning based MEC in [15] which

utilizes EH can reduce the time cost, assuming both the normal distributed renewable energy generation under the known offloading delay model and the power consumption model. The assumptions may not be practical for IoT devices, especially in dynamic MEC networks with multiple edge candidates.

In this paper, we investigate computation offloading of IoT devices with EH in a dynamic MEC network. We consider that an IoT device is connected with multiple edge devices of different communication overheads, and can utilize the EH history or model to approximately predict the future renewable energy generation trend. We present a reinforcement learning (RL) based computation offloading framework for an IoT device to select an edge device and consequently determine the proportion of the computation tasks to offload. Each offloading decision is made in accordance to the radio transmission rate of each IoT-edge link in the previous time slot, the predicted renewable energy harvesting model and the current battery level of the IoT device. The computation offloading process of the IoT device can be modeled by a Markov decision process (MDP). Consequently, reinforcement learning techniques, such as Q-learning, enable the IoT device to act as a learning agent to achieve the optimal offloading policy after a large number of offloading interactions with probability one [16].

We propose a reinforcement learning based computation offloading scheme (RLO) for an IoT device with EH without knowledge of the MEC model, the computation latency model, and the energy consumption model. The offloading policy including the edge device and the offloading rate is selected according to the current MEC state and the quality function or Q-function that is the expected long-term discount reward for each state-action pair. The IoT device evaluates the reward or utility value based on the overall latency, energy consumption, task drop loss, and the data sharing gains within each time slot. This in turn updates the Q-function according to the iterative Bellman equation [16] based on the utility and the latest offloading policy. A transfer learning method in reinforcement learning [17] exploits the offloading experiences in similar scenarios to initialize the Q-values and therefore reducing the exploration time at the initial stage in the repeated offloading process with edge devices.

Moreover, we propose a deep reinforcement learning based offloading scheme (DRLO) to improve the offloading performance for the case with a large state space. For example, an IoT device can expect a large number of feasible battery levels, a large amount of renewable energy generated in a time slot, and a large number of potential radio transmission rates corresponding to each edge device. This scheme applies deep convolutional neural network (CNN) [18] to compress the state space and accelerate the learning speed. This scheme can be implemented on IoT platforms such as Qualcomm Snapdragon 800 and Nvidia Tegra K1 that support for deep learning [19], [20].

We present the performance of RL-based offloading schemes and the convergence performance for three typical MEC scenarios, for an energy consumption model based on CPU frequencies and transmission power. The local computational time and the transmission time are considered in the analysis, and the performance bound of the proposed scheme regarding the energy consumption, computation latency, and utility from each offloading decision is provided. Simulations are performed for IoT devices with wireless power transfer to capture the ambient RF signals from a dedicated RF energy transmitter to charge the IoT battery [21], [22]. Simulation results show that the DRLO scheme saves the energy consumption of IoT devices, reduces the computation latency and decreases the task drop rate, and thus increases the utility of IoT devices, compared to the benchmark Q-learning-based offloading scheme. The main contributions of this study are summarized as follows:

1) An RL-based computation offloading scheme for IoT devices with EH is presented. This scheme enables an IoT device to select the edge device and the offloading rate according to the current battery level, the previous radio transmission rate, and the predicted EH model. Further, this scheme also enables an IoT device to achieve the optimal offloading without knowing the MEC model, the energy consumption model and computation latency model in dynamic MEC with low computation complexity;
2) We propose a deep RL-based offloading scheme to compress the state space dimension, with a transfer learning technique to accelerate the learning speed at the initial stages and improve the offloading performance for IoT devices with sufficient computation resources;
3) We provide the performance bound of the proposed RL-based offloading schemes and prove their convergence to the optimal utility.

The rest of this paper is organized as follows: Related studies are reviewed in Section II. The mobile offloading model and the EH model are presented in Section III. An RLO scheme for IoT devices with EH is proposed in Section IV and a DRLO scheme is discussed in Section V. Analysis of the performance bounds of the proposed computation offloading schemes is given in Section VI. Simulation results are discussed in Section VII, followed by the conclusion of this study in Section VIII.

II. Related Work

MEC is an effective approach to augment computation capability of mobile devices. An offloading framework is applied for offloading the computation tasks from a single mobile device to multiple edge devices, and the task allocation and CPU frequency are jointly optimized to minimize the execution latency and energy consumption [10]. Binary offloading proposed in [23] reduces the computation overhead for resource-constrained mobile devices with multi-channel wireless interference with a single edge. Resource allocation and partial offloading in a multiuser MEC network are investigated to minimize the energy consumption under a latency constraint [24]. Dynamic voltage scaling is incorporated into partial computation offloading to jointly decrease the energy consumption and latency in the allocation of the computation resources, transmission power, and offloading rates of the mobile devices [25].

Energy harvesting is a promising solution to provide perpetual energy supply for self-sustainable mobile devices. A Lyapunov optimization-based binary offloading scheme is proposed for the

EH-powered mobile devices to reduce the execution latency and computation failures with a single known MEC server [14]. For EH-powered edge servers, a learning-based online algorithm is studied for the system operator to decide the amount of workload offloaded from the edge server to the central cloud and the processing speed of the edge server [15]. The study is based on the information of core network congestion state, computation workload, and energy state information. A wireless-powered single-user mobile cloud computing system is investigated in [11], which includes the evaluation of the optimal CPU cycle frequencies for local computing and time division for offloading to adapt to the transferred power, based on the convex optimization theory. Furthermore, a binary offloading policy in a wireless-powered MEC network is proposed, which utilizes the AP to transfer RF power to the mobile device and execute the computation tasks [12]. Most of existence works are based on the known models, such as transmission delay model and energy consumption model. However, these assumptions may not be practical for IoT devices in dynamic MEC networks.

RL techniques have been applied for offloading strategies in the dynamic MEC network. An efficient RL-based resource management algorithm is proposed to optimize the on-the-fly workload offloading rate to centralized cloud and edge server, provisioning to minimize the service latency, and operational cost [15]. The computation offloading strategies include application of Q-learning techniques, i) to derive the optimal offloading rates and reduce the attack rate of smart attackers at IoT devices [26], and ii) to achieve optimal offloading and reduce response time in edge-based offloading [27]. Q-learning, Dyna-Q and post decision state techniques are applied to malware detection offloading in order to improve the detection performance without being aware of the trace generation and the channel models in dynamic radio environments [28]. A binary MEC offloading scheme named DRL in [29] uses deep RL to choose whether to offload to the edge device that serves multiple users. This can reduce the energy consumption and average computation latency as compared with both the full-offloading and non-offloading schemes. This work further improves the offloading scheme by incorporating the radio transmission rate, the harvested energy and the battery level in the state of the learning process and using transfer learning and deep learning to optimize the edge selection and the offloading rate for better computation performance.

III. System Model

An MEC network consisting of an IoT device and M edge devices is considered, as shown in Fig. 1. The IoT device can represent a smart watch and smartphone [30], which has to support computation intensive tasks such as augmented reality applications. These tasks should be processed timely to satisfy the user experience and to guide further actions locally or remotely. The IoT device is equipped with electricity storage elements and EH components, such as RF energy harvester, photovoltaic modules, and wind turbines. With energy powered by the EH component, the IoT device can either execute its computation tasks locally or offload partial or all tasks to the edge devices

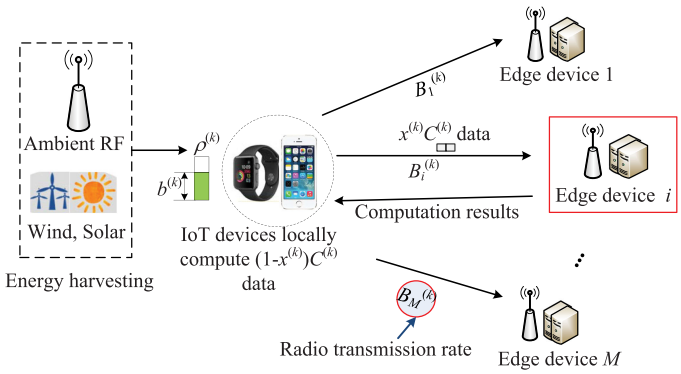


Fig. 1. MEC system with EH-powered IoT devices.

that can provide high computation performance with a virtual machine. Time is slotted with a constant slot duration, and the time slot index is denoted by k, with $k \in \{1, 2, ...\}$.

The IoT device generates computation tasks associated with a data volume of $C^{(k)}$ bits at time slot k. According to literature [31], we focus on delay-sensitive applications, with execution duration of the computation tasks $C^{(k)}$ no longer than a time slot length. The IoT device applies the computation partition scheme proposed in [32] to divide the computation tasks into N_x parts.

The IoT device selects edge device i to offload the computation tasks at time slot k based on radio link transmission rate $B_i^{(k)}$, with $1 \leq i \leq M$, and decides the offloading rate denoted by $x^{(k)}$, with $x^{(k)} \in \{l/N_x\}_{0 \leq l \leq N_x}$. More specifically, the IoT device executes the computation tasks locally if $x^{(k)} = 0$, and offloads all the computation tasks to edge device i if $x^{(k)} = 1$. The IoT device offloads $x^{(k)}C^{(k)}$ data to the selected edge device and the remainning $(1 - x^{(k)})C^{(k)}$ data are processed locally if $0 < x^{(k)} < 1$. The IoT device selects its offloading policy at time slot k denoted by $\mathbf{a}^{(k)} = [i^{(k)}, x^{(k)}]$ from all the possible policies denoted by $\mathbf{A}$. For ease of reference, our commonly used notations are summarized in Table I. The time index k in the superscript is omitted if no confusion occurs.

A. Local Computing

The CPU of the IoT device is the primary engine for local computation. The performance of CPU is controlled using the CPU-cycle frequency. Local computing for executing $(1 - x^{(k)})C^{(k)}$ input bits occurs at the IoT device. Let N denote the number of CPU cycles required for computing one input bit. Therefore, the total number of CPU cycles required for the $(1 - x^{(k)})C^{(k)}$ bits is $(1 - x^{(k)})C^{(k)}N$. By applying dynamic voltage and frequency scaling techniques [25], the IoT device can control the energy consumption for local task execution by adjusting the CPU frequency f_m for each cycle m, with $m \in \{1, 2, ..., (1 - x^{(k)})C^{(k)}N\}$. The execution latency for local computing at time slot k is denoted by $T_0^{(k)}$ and given by the following equation:

$$T_0^{(k)} = \sum_{m=1}^{(1-x^{(k)})C^{(k)}N} \frac{1}{f_m}. \tag{1}$$

TABLE I
LIST OF NOTATIONS

Symbol	Description
$x^{(k)}$	Proportion of the data offloaded to the edge device at time slot k
$B_i^{(k)}$	Radio transmission rate between the IoT device and edge device i
$\rho^{(k)}$	Amount of harvested energy
$b^{(k)}$	Battery level
P	Transmit power of the IoT device
η	Transmit power of the energy transmitter
f	Scheduled CPU-cycle frequencies for local execution
N	Number of CPU cycles required to process one bit task input
$E^{(k)}$	Energy consumption
$\mathcal{T}^{(k)}$	Computation delay
ψ	Task drop loss
W	Experience size in the CNN input sequence
H	Size of minibatch of the CNN

According to [23], the IoT device consumes $E_0^{(k)}$ energy for local computing at time slot k, with

$$E_0^{(k)} = \sum_{m=1}^{(1-x^{(k)})C^{(k)}N} \varsigma f_m^2, \tag{2}$$

where ς is the effective capacitance coefficient that depends on the chip architecture. The proposed scheme is applicable to IoT devices with the dynamic frequency and voltage scaling according to [23] and [25], and fixed CPU frequency and voltage can be viewed as a special case.

B. Computation Offloading

The IoT device sends the computation tasks to the edge device i at time slot k with the uplink radio transmission rate $B_i^{(k)}$. The transmission duration to offload $x^{(k)}C^{(k)}$ bits data to edge device i is represented by $T_i^{(k)}$, with

$$T_i^{(k)} = \frac{x^{(k)}C^{(k)}}{B_i^{(k)}}. \tag{3}$$

Based on [33], the IoT device consumes $E_i^{(k)}$ energy to offload the tasks to edge device i at time slot k, which depends on the transmit power P for offloading and the transmission duration $T_i^{(k)}$, with

$$E_i^{(k)} = \frac{x^{(k)}C^{(k)}P}{B_i^{(k)}}. \tag{4}$$

C. Energy Harvesting

The IoT device may contain RF energy harvester, photovoltaic modules and wind turbines, which can convert renewable resources (such as ambient RF signals, wind and solar) to electricity. It also contains a battery in order to balance the power supply and demand. Moreover, the harvested renewable energy can be stored in the battery to support both the local computing and computation offloading.

The amount of energy harvested at time slot k is denoted by $\rho^{(k)}$ and the total energy consumed by the IoT device at time slot k is denoted by $E^{(k)}$, with $E^{(k)} = (E_0^{(k)} + E_i^{(k)})$. The battery level at the beginning of time slot k is denoted by $b^{(k)}$, and it evolves according to the following equation:

$$b^{(k+1)} = \max\left\{0, b^{(k)} - E^{(k)} + \rho^{(k)}\right\}. \tag{5}$$

The IoT device drops the computation task, if its energy is insufficient, i.e., $b^{(k+1)} = 0$ [14].

As a special case, the RF-enabled wireless energy transfer technology in [21] is considered, in which the IoT device is powered with the dedicated wireless power transmitter that provides continuous and stable microwave energy over the air. Let $\upsilon \in (0, 1)$ be the energy conversion efficiency, $\eta^{(k)}$ denote the transmission power, $d^{(k)}$ represent the distance between the energy transmitter and the IoT device at time slot k, $\tau \geq 2$ denote the path loss factor exponent, and G denote the combined gain of the RF energy transmitter antenna and the IoT antenna. According to [21], the amount of the harvested energy of the IoT device in time slot k is given by

$$\rho^{(k)} = \upsilon\eta^{(k)}\left(d^{(k)}\right)^{-\tau} G. \tag{6}$$

The amount of energy harvested at time slot k can be estimated by the IoT device according to the power harvested history and the modeling method in [34]. The estimated amount of the generated energy is denoted by $\hat{\rho}^{(k)}$. The estimation error regarding $\rho^{(k)}$ is denoted by $\triangle^{(k)}$, which follows a uniform distribution with the maximum estimation error ζ, and is given by

$$\triangle^{(k)} = \rho^{(k)} - \hat{\rho}^{(k)} \sim \zeta U(-1, 1). \tag{7}$$

IV. RL-BASED COMPUTATION OFFLOADING

In the dynamic computation offloading process, the IoT device selects its proportion of data to offload to an edge device based on the system state, consisting of the previous radio link transmission rate, the predicted amount of harvested renewable energy, and the current battery level. Noteworthy, the next system state observed by the IoT device depends only on the current state and actions, and not on the past history of the process. Therefore, the computation offloading process can be viewed as an MDP, in which the Q-learning technique, a model-free and widely used RL technique, can derive the optimal policy without knowledge of the MEC model, energy consumption model, and computation latency model.

We propose an RL-based offloading scheme RLO that uses the transfer learning method [17] to initialize the learning parameters. More specifically, the Q-values are initialized with the offloading experience in similar environments, i.e., a number of indoor or outdoor networks with typical MEC deployments given the same energy harvesting source. The RLO scheme reduces random explorations at the initial stage of the dynamic

computation offloading process, and thus accelerates the learning speed.

In each time slot, according to the EH model such as (6), the IoT device can estimate the amount of harvested energy $\hat{\rho}^{(k)}$, observe its current battery level $b^{(k)}$ and the radio link data rates to the M edge devices in the last time slot (i.e., $B_1^{(k-1)}, ..., B_M^{(k-1)}$), which are used to formulate the state denoted by $\mathbf{s}^{(k)}$, with $\mathbf{s}^{(k)} = [B_1^{(k-1)}, ..., B_M^{(k-1)}, \hat{\rho}^{(k)}, b^{(k)}]$.

The IoT device selects edge device i and the rate of computation tasks to offload, i.e., $\mathbf{a}^{(k)} = [i^{(k)}, x^{(k)}]$, based on state $\mathbf{s}^{(k)}$, and applies the ε-greedy policy with $0 < \varepsilon \leq 1$ to avoid staying in the local maximum. More specifically, the offloading policy that achieves the highest expected long-term utility or Q-function denoted by $Q(\mathbf{s}^{(k)}, \mathbf{a})$ for current state is selected with a high probability $1 - \varepsilon$, while each of other offloading strategies is selected with a low probability $\varepsilon/((N_x + 1)\, M)$ to encourage explorations.

The IoT device offloads computation tasks associate with the $x^{(k)}C^{(k)}$ data to edge device i at time slot k, and locally processes the remaining tasks. When the edge device completes processing the tasks on behalf of the IoT device, it sends the computation results back to the IoT device. It is assumed that the output of the computation results is of small size, thus the transmission delay between the edge device and the IoT device is negligible. The IoT device then evaluates the data sharing gain, measures the total energy consumption and the results of the processed tasks, and measures the computation latency represented by $\mathcal{T}^{(k)}$ which depends on the local execution latency $T_0^{(k)}$ and the transmission delay to the selected edge device $T_i^{(k)}$, i.e., $\mathcal{T}^{(k)} = \max\{T_0^{(k)}, T_i^{(k)}\}$.

The IoT device fails to perform the computation task in case of insufficient energy, i.e., $b^{(k+1)} = 0$. The task drop loss denoted by ψ is defined as the cost if a computation task fails. The indicator function denoted by $\mathbf{I}(\varpi)$ equals 1 if ϖ is true and 0 otherwise. Let β and μ be the weighting parameters of energy consumption and computation latency of the IoT device, respectively. The utility of the IoT device to select edge device i at time slot k is denoted by $U_i^{(k)}(x)$, which depends on the data sharing gains, the task drop loss, the energy consumption, and the overall computation latency, and is given by:

$$U_i^{(k)}(x) = xC^{(k)} - \psi \mathbf{I}\left(b^{(k+1)} = 0\right) - \beta E^{(k)} - \mu \mathcal{T}^{(k)}. \quad (8)$$

The system state transits to the next state $\mathbf{s}^{(k+1)}$ if the IoT device offloads $x^{(k)}C^{(k)}$ data to the selected i-th edge device at state $\mathbf{s}^{(k)}$. According to this offloading experience, denoted by $\left(\mathbf{s}^{(k)}, \mathbf{a}^{(k)}, U^{(k)}, \mathbf{s}^{(k+1)}\right)$, the IoT device updates the Q-function of the state-action pair $\left(\mathbf{s}^{(k)}, \mathbf{a}^{(k)}\right)$ in the following manner:

$$Q\left(\mathbf{s}^{(k)}, \mathbf{a}^{(k)}\right) \leftarrow (1-\alpha)\, Q\left(\mathbf{s}^{(k)}, \mathbf{a}^{(k)}\right) + \alpha\left(U^{(k)} + \gamma \max_{\mathbf{a}' \in \mathbf{A}} Q\left(\mathbf{s}^{(k+1)}, \mathbf{a}'\right)\right), \quad (9)$$

where the learning rate $\alpha \in (0, 1]$ is the weight of the current offloading experience. The discount factor $\gamma \in [0, 1]$ denotes the myopic view of the IoT device regarding the future reward. The computation offloading scheme is summarized in Algorithm 1. This scheme can achieve the optimal computation offloading policy via trial-and-error over sufficient time slots.

Algorithm 1: RL-Based Computation Offloading Scheme.

Initialize C, f, N, $B_1^{(0)}, ..., B_M^{(0)}$, $\hat{\rho}^{(1)}$, $b^{(1)}$, α, γ and ε
Set $\mathbf{Q} \leftarrow \overline{\mathbf{Q}}$ according to the transfer learning technique
for $k = 1, 2, 3, ...$ **do**
  Estimate the amount of harvested energy $\hat{\rho}^{(k)}$
  Observe the radio transmission rate, $B_1^{(k-1)}, ..., B_M^{(k-1)}$, and the current battery level $b^{(k)}$
  $\mathbf{s}^{(k)} = \left[B_1^{(k-1)}, ..., B_M^{(k-1)}, \hat{\rho}^{(k)}, b^{(k)}\right]$
  Select $\mathbf{a}^{(k)} = [i^{(k)}, x^{(k)}] \in \mathbf{A}$ via ε-greedy policy
  Offload $x^{(k)}C$ data to the selected i-th edge device
  Observe the the battery level
  Evaluate the energy consumption, computation latency and task drop loss
  Evaluate $U^{(k)}$ via (8)
  Update $Q(\mathbf{s}^{(k)}, \mathbf{a}^{(k)})$ via (9)
end for

The learning speed of the RLO scheme increases with the dimension of the action-state space, which increases with the number of feasible IoT battery levels, the amount of renewable energy generated in a time slot, and the number of potential radio transmission rates corresponding to each edge device, yielding serious performance degradation.

V. Deep RL-Based Computation Offloading

Herein, we propose a deep RL-based offloading scheme DRLO to improve the computation performance. This scheme utilizes a deep CNN to compress the state space and accelerate the convergence speed of offloading process in Algorithm 1. Fig. 2 illustrates that the deep CNN is a nonlinear approximator of the Q-value for each offloading policy.

Algorithm 2 summarizes the DRLO scheme, which extends the system state $\mathbf{s}^{(k)}$ as in Algorithm 1 to the offloading experience sequence at time slot k denoted by $\boldsymbol{\varphi}^{(k)}$, to accelerate the learning speed and improve the computation performance of the IoT device. More specifically, the experience sequence consists of the current system state $\mathbf{s}^{(k)}$ and the previous W system state-action pairs, i.e.,

$$\boldsymbol{\varphi}^{(k)} = \left(\mathbf{s}^{(k-W)}, \mathbf{a}^{(k-W)}, ..., \mathbf{s}^{(k-1)}, \mathbf{a}^{(k-1)}, \mathbf{s}^{(k)}\right). \quad (10)$$

If the number of offloading experiences is less than W, i.e., $k \leq W$, the IoT device randomly selects an offloading policy from action set $\mathbf{A}$, as shown in Algorithm 2.

Fig. 2 shows that the experience sequence, $\boldsymbol{\varphi}^{(k)}$, is reshaped into an $n_0 \times n_0$ square matrix and then input into the CNN with weights denoted by $\theta^{(k)}$. The CNN consists of two convolutional (Conv) layers and two fully connected (FC) layers. The first Conv layer consists of m_1 different filters. Each filter has size

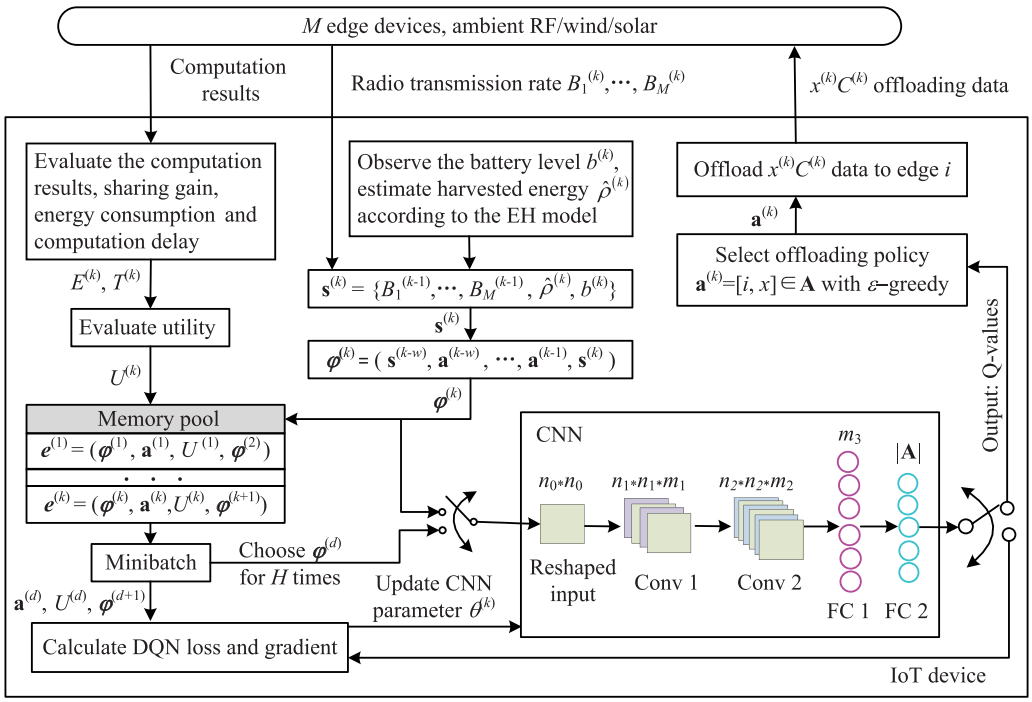


Fig. 2. Illustration of the deep RL-based computation offloading scheme for IoT devices.

$n_1 \times n_1$ and uses stride s_1. The output of the first Conv is then passed through a rectified linear unit (ReLU) given in [18] as the activation function. The second Conv layer includes m_2 different filters. Each filter has size $n_2 \times n_2$ and stride s_2, and uses the same linear rectifier. The outputs of the 2nd Conv layer are flattened to a vector and then sent to the two FC layers. The first FC layer involves m_3 rectified linear units, and the second FC layer provides $Q\left(\boldsymbol{\varphi}^{(k)}, \mathbf{a}; \theta^{(k)}\right)$ to estimate the Q-values for all $|\mathbf{A}|$ computation offloading strategies at the current system sequence $\boldsymbol{\varphi}^{(k)}$.

To make a tradeoff between exploitation and exploration, the IoT device selects the offloading policy, $\mathbf{a}^{(k)} \in \mathbf{A}$, based on the output of the CNN according to the ε-greedy policy [35], and then sends $x^{(k)}C^{(k)}$ data to edge device i.

After receiving the processed results from the edge device, the IoT device evaluates the data sharing gains, the current battery level, the energy consumption and the overall latency to obtain its reward or utility $U^{(k)}$ via (8). The DRLO scheme also updates a quality function Q for each action-state pair in the dynamic computation offloading process, given by

$$Q\left(\boldsymbol{\varphi}^{(k)}, \mathbf{a}\right) = \mathbb{E}_{\boldsymbol{\varphi}'}\left[U^{(k)} + \gamma \max_{\mathbf{a}'\in\mathbf{A}} Q\left(\boldsymbol{\varphi}', \mathbf{a}'\right) | \boldsymbol{\varphi}^{(k)}, \mathbf{a}\right] \quad (11)$$

where $\boldsymbol{\varphi}'$ is the next state sequence based on the selection of offloading policy $\mathbf{a}^{(k)}$ at state $\boldsymbol{\varphi}^{(k)}$.

The experience replay utilizes the memory pool as shown in Fig. 2 to store the offloading experiences. The offloading experience that the IoT device obtained at time slot k is denoted by $\mathbf{e}^{(k)} = \left(\boldsymbol{\varphi}^{(k)}, \mathbf{a}^{(k)}, U^{(k)}, \boldsymbol{\varphi}^{(k+1)}\right)$, and the memory pool is $\mathcal{D}$, with $\mathcal{D} = \left\{\mathbf{e}^{(1)}, ..., \mathbf{e}^{(k)}\right\}$. According to the experience replay technique, the IoT device randomly selects an experience from $\mathcal{D}$ to update the CNN weight $\theta^{(k)}$. The stochastic gradient descent algorithm is applied, in which the mean-squared error between the network's output and the target optimal Q-value is minimized with the minibatch updates. The loss function is given by [18]

$$L(\theta^{(k)}) = \mathbb{E}_{\boldsymbol{\varphi}, \mathbf{a}, U, \boldsymbol{\varphi}'}\left[\left(R - Q\left(\boldsymbol{\varphi}, \mathbf{a}; \theta^{(k)}\right)\right)^2\right] \quad (12)$$

where the target optimal Q-function R is given by

$$R = U + \gamma \max_{\mathbf{a}'\in\mathbf{A}} Q\left(\boldsymbol{\varphi}', \mathbf{a}'; \theta^{(k-1)}\right). \quad (13)$$

This process repeats H times, and $\theta^{(k)}$ is then updated according to these randomly selected experiences.

Similar to Algorithm 1, the transfer learning method is applied to initialize the CNN parameters in the DRLO scheme rather than initializing them randomly to accelerate the learning speed. The IoT device stores emulational experience $\{\boldsymbol{\varphi}^{(k)}, \mathbf{a}^{(k)}, U^{(k)}, \boldsymbol{\varphi}^{(k+1)}\}$ in the database $\mathbb{E}$, and the resulting CNN weights θ^* based on ξ experiences are used to initialize $\theta^{(0)}$ as shown in Algorithm 2.

VI. Performance Evaluations

The optimal offloading policy is analyzed for a static scenario, which can be used as the performance upper bound to evaluate the RL-based offloading scheme in the dynamic process over sufficient time slots. The time index k in the superscript is omitted in the following for clarity. The IoT device selects the optimal offloading policy $\mathbf{a}^* = [i^*, x^*]$, with x^* of computation tasks offloaded to edge device i^* to maximize the utility of the IoT device. According to [9] and [14], the IoT device has the same local CPU-cycle frequency f during the learning process and takes N CPU cycles to perform a single computation task.

Algorithm 2: Deep RL-Based Computation Offloading Scheme.

Initialize C, f, N, $B_1^{(0)}, ..., B_M^{(0)}$, $\hat{\rho}^{(1)}$, $b^{(1)}$, W, H, and $\mathcal{D} = \emptyset$
$\theta^{(0)} \leftarrow \theta^*$
for $k = 1, 2, 3, ...$ **do**
  Estimate the amount of harvested energy $\hat{\rho}^{(k)}$
  Observe the radio transmission rate, $B_1^{(k-1)}, ..., B_M^{(k-1)}$, and the current battery level $b^{(k)}$
  $\mathbf{s}^{(k)} = \left[B_1^{(k-1)}, ..., B_M^{(k-1)}, \hat{\rho}^{(k)}, b^{(k)}\right]$
  if $k \leq W$ **then**
    Select $\mathbf{a}^{(k)} = [i^{(k)}, x^{(k)}] \in \mathbf{A}$ at random
  else
    Form the experience sequence $\boldsymbol{\varphi}^{(k)} = \left(\mathbf{s}^{(k-W)}, \mathbf{a}^{(k-W)}, \mathbf{s}^{(k-W+1)}, ..., \mathbf{s}^{(k)}\right)$
    Set $\boldsymbol{\varphi}^{(k)}$ as the input of the CNN
    Observe the output of the CNN to obatain $Q\left(\boldsymbol{\varphi}^{(k)}, \mathbf{a}; \theta^{(k)}\right)$
    Select $\mathbf{a}^{(k)} \in \mathbf{A}$ via ε-greedy policy
  end if
  Offload $x^{(k)}C$ data to the selected i-th edge device
  Observe the battery level
  Evaluate the energy consumption, computation latency and task drop loss
  Evaluate $U^{(k)}$ via (8)
  $\mathcal{D} \leftarrow \mathcal{D} \bigcup \left(\boldsymbol{\varphi}^{(k)}, \mathbf{a}^{(k)}, U^{(k)}, \boldsymbol{\varphi}^{(k+1)}\right)$
  for $d = 1, 2, 3, ..., H$ **do**
    Select $\left(\boldsymbol{\varphi}^{(d)}, \mathbf{a}^{(d)}, U^{(d)}, \boldsymbol{\varphi}^{(d+1)}\right) \in \mathcal{D}$ at random
    Calculate R via (13)
  end for
  Update the CNN weights with $\theta^{(k)}$ by minibatch gradient descent
end for

Proposition 1: The optimal offloading policy of the IoT device is $\mathbf{a}^* = [i^*, 1]$, if

$$B_{i^*} \geq \max\left\{\frac{\mu + \beta P}{\beta \varsigma N f^2 + 1}, \frac{P}{\varsigma N f^2}\right\} \tag{14}$$

where

$$i^* = \arg\max_{1 \leq i \leq M} B_i. \tag{15}$$

Proof: By (8), $\forall B_i \leq xf/\left((1-x)N\right)$, we have

$$\frac{\partial U_i}{\partial B_i} = \frac{\beta P + \mu}{B_i} \geq 0, \tag{16}$$

and $\forall B_i \geq xf/\left((1-x)N\right)$, we have

$$\frac{\partial U_i}{\partial B_i} = \frac{\beta P}{B_i} \geq 0 \tag{17}$$

indicating that the utility of IoT device, U_i, is maximized at $i^* = \arg\max_{1 \leq i \leq M} B_i$.

By (8) and (14), for given $NB_{i^*}/\left(NB_{i^*} + f\right) < x \leq 1$, we have

$$\begin{aligned} U_{i^*}(1) &= C\left(1 - \frac{\beta P + \mu}{B_{i^*}}\right) - \psi \mathbf{I}\left(b - \frac{PC}{B_{i^*}} + \rho = 0\right) \\ &\geq \left(\beta \varsigma N f^2 - \frac{\beta P + \mu}{B_{i^*}} + 1\right) xC - \beta \varsigma C N f^2 \\ &\quad - \psi \mathbf{I}\left(b + \left(\varsigma N f^2 - \frac{P}{B_{i^*}}\right) xC - \varsigma C N f^2 + \rho = 0\right) \\ &= U_{i^*}(x). \end{aligned} \tag{18}$$

Similarly, if $0 \leq x \leq NB_{i^*}/\left(NB_{i^*} + f\right)$, we have

$$\begin{aligned} U_{i^*}(1) &\geq \left(1 + \beta \varsigma N f^2 - \frac{\beta P}{B_{i^*}} + \frac{\mu N}{f}\right) xC \\ &\quad - \psi \mathbf{I}\left(b + \left(\varsigma N f^2 - \frac{P}{B_{i^*}}\right) xC - \varsigma C N f^2 + \rho = 0\right) \\ &\quad - \beta \varsigma C N f^2 - \frac{\mu C N}{f} = U_{i^*}(x). \end{aligned} \tag{19}$$

Thus, we prove $\mathbf{a}^* = [i^*, 1]$ is the optimal offloading policy of the IoT device. ■

Remark 1: If the IoT device has a good radio channel to at least one edge device as indicated in (14), the IoT device offloads all the computation tasks to the edge device with the largest radio link rate as indicated in (15) to maximize its utility due to the low computation latency and energy consumption.

In the dynamic computation offloading process, the IoT device selects its proportion of data to offload to an edge device based on the state of the current system. The next system state observed by the IoT device is independent of the previous states and actions. Therefore, the computation offloading process can be viewed as an MDP.

Theorem 1: If (14) and (15) hold for a sufficient long time to offload, the RL-based offloading schemes as presented in Algorithms 1 and 2 can achieve the utility of the IoT device given by

$$U = C\left(1 - \frac{\beta P + \mu}{B_{i^*}}\right) - \psi \mathbf{I}\left(b - \frac{PC}{B_{i^*}} + \rho = 0\right). \tag{20}$$

Proof: If (14) and (15) hold, we have the optimal offloading policy $\mathbf{a}^* = [i^*, 1]$. By (1)-(5) and (8), we have (20). According to [16], the proposed schemes can eventually derive the optimal offloading policy of the IoT device $\mathbf{a}^* = [i^*, 1]$ in the MDP via trial-and-error. Therefore, if (14) and (15) hold, the proposed schemes can achieve the utility (20) after a sufficient long iterative time. ■

Corollary 1: If (14) and (15) hold for a sufficient long time to offload, the IoT device with the RL-based offloading schemes as presented in Algorithms 1 and 2 can achieve the computation latency and energy consumption given by $\mathcal{T} = C/B_{i^*}$ and $E = PC/B_{i^*}$.

Remark 2: The IoT device applies the RL-based offloading scheme without knowledge of the MEC model and the exact energy consumption and computation latency model in the dynamic offloading process, which can achieve the optimal

computation performance. In this scenario, the radio channel between the IoT device and edge devices are in a good condition. By trail-and-error, the IoT device offloads all the computation tasks to the edge device to optimize its performance. In this case, the computation latency and energy consumption are positively linearly related to the generated task size C.

Proposition 2: The optimal offloading policy of the IoT device is $\mathbf{a}^* = [i^*, 0]$, if

$$B_{i^*} \leq \frac{\beta P f}{f + \beta \varsigma N f^3 + \mu N} \tag{21}$$

where

$$i^* = \arg \max_{1 \leq i \leq M} B_i. \tag{22}$$

Proof: The proof is similar to Proposition 1. ■

Remark 3: If all the edge devices experiences severe channel fading as indicated in (21), the IoT device processes all the computation tasks locally to avoid wasting the transmission power.

Theorem 2: If (21) and (22) hold for a sufficient long time to offload, the RL-based offloading schemes given in 1 and 2 can achieve the utility of the IoT device given by

$$U = -\psi \mathbf{I}\left(b - \varsigma C N f^2 + \rho = 0\right) - CN\left(\beta \varsigma f^2 + \frac{\mu}{f}\right). \tag{23}$$

Proof: The proof is similar to Theorem 1. ■

Corollary 2: If (21) and (22) hold for a sufficient long time to offload, the IoT device with the RL-based offloading schemes as presented in Algorithms 1 and 2 can achieve the computation latency and energy consumption given by $\mathcal{T} = CN/f$ and $E = \varsigma C N f^2$.

Remark 4: In this scenario, all the edge devices experiences severe channel fading. By trial-and-error, the IoT device processes all the computation tasks locally to maximize its utility due to the high transmission delay and energy consumption.

Proposition 3: The optimal offloading policy of the IoT device is

$$\mathbf{a}^* = \left[i^*, \frac{N B_{i^*}}{N B_{i^*} + f}\right] \tag{24}$$

if

$$\min\left\{\frac{\beta P + \mu}{\beta \varsigma N f^2 + 1}, \frac{P}{\varsigma N f^2}\right\} \geq B_{i^*} \geq \frac{P}{\varsigma N f^2} \tag{25}$$

where

$$i^* = \arg \max_{1 \leq i \leq M} B_i. \tag{26}$$

Proof: The proof is similar to Proposition 1. ■

Remark 5: If the best edge device experiences modest fading as indicated in (25), the IoT device processes partial computation tasks locally and chooses the edge device with the best link state as indicated in (26).

Theorem 3: If (25) and (26) hold for a sufficient long time to offload, the RL-based offloading schemes as presented in Algorithms 1 and 2 can achieve utility of the IoT device given by

$$\begin{aligned} U = & -\psi \mathbf{I}\left(b + \frac{CN^2 \varsigma f^2 B_{i^*} - CNP}{N B_{i^*} + f} - CN \varsigma f^2 + \rho = 0\right) \\ & - CN\left(\beta \varsigma f^2 + \frac{\beta \varsigma N f^2 B_{i^*} + B_{i^*} + \mu - \beta P}{N B_{i^*} + f}\right). \end{aligned} \tag{27}$$

Proof: The proof is similar to Theorem 1. ■

Corollary 3: If (25) and (26) hold, the IoT device with the RL-based offloading schemes as presented in Algorithms 1 and 2 can achieve the computation latency and energy consumption given by

$$\mathcal{T} = \max\left\{\frac{CN}{f}\left(1 - \frac{N B_{i^*}}{N B_{i^*} + f}\right), \frac{CN}{N B_{i^*} + f}\right\} \tag{28}$$

$$E = \varsigma C N f^2 \left(1 - \frac{N B_{i^*}}{N B_{i^*} + f}\right) + \frac{CNP}{N B_{i^*} + f}. \tag{29}$$

Remark 6: In this case, by trial-and-error, the IoT device processes partial computation tasks locally and offloads the remaining computation tasks to the edge device to tradeoff the energy consumption and computation latency, in order to maximize its utility in (27). The computation latency and energy consumption as indicated in (28) and (29) is positively related to the generated task size C, and the utility is negatively related to C.

The computation complexity of DRLO denoted by Γ mostly relies on the CNN structure. Let m_0 denote the number of the input channels of the CNN, and M_l denote the size of the output features of Conv l. According to [36], the size of the output features of Conv 1 and that of the Conv 2 are $(n_0 - n_1)/s_1 + 1$ and $(n_0 - n_1)/(s_1 s_2) - (n_2 - 1)/s_2 + 1$, respectively. Thus we have the following results:

Theorem 4: The computational complexity of the DRLO scheme is given by

$$\Gamma = \mathcal{O}\left(m_1 n_2^2 m_2 \left(\frac{n_0 - n_1}{s_1 s_2} - \frac{n_2 - 1}{s_2} + 1\right)^2\right). \tag{30}$$

Proof: The computational complexity of the FC layers is not involved because it is sufficiently small compared with the Conv layers. According to the CNN architecture in [18], we have

$$m_1 n_2^2 m_2 \left(\frac{n_0 - n_1}{s_1 s_2} - \frac{n_2 - 1}{s_2} + 1\right)^2 \geq n_1^2 m_1 \left(\frac{n_0 - n_1}{s_1} + 1\right)^2.$$

Thus, according to [37], we have

$$\begin{aligned} \Gamma &= \mathcal{O}\left(\Sigma_{l=1}^{2} m_{l-1} n_l^2 m_l M_l^2\right) \\ &= \mathcal{O}\left(n_1^2 m_1 \left(\frac{n_0 - n_1}{s_1} + 1\right)^2 \right. \\ &\quad \left. + m_1 n_2^2 m_2 \left(\frac{n_0 - n_1}{s_1 s_2} - \frac{n_2 - 1}{s_2} + 1\right)^2\right) \\ &= \mathcal{O}\left(m_1 n_2^2 m_2 \left(\frac{n_0 - n_1}{s_1 s_2} - \frac{n_2 - 1}{s_2} + 1\right)^2\right). \end{aligned} \tag{31}$$

This completes the proof. ■

Remark 7: The computation complexity of the DRLO increases with the number of the CNN filters and the filter size of the Conv layers and decreases with the filter strides. On the other hand, the computation performance of the DRLO improves with the number of CNN filters that represents the features of the MEC system and a shorter stride that captures more computation offloading details. The parameters in Algorithm 2 such as the number of filters and filter strides have to be chosen as a trade-off between the computational complexity and the computation performance.

VII. Simulation Results

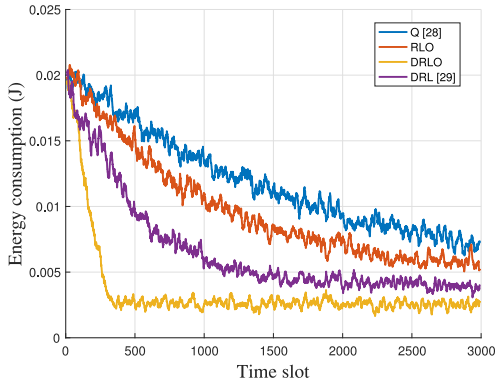
In this section, the performance of the proposed RL-based computation offloading schemes for the IoT device is evaluated in the dynamic network with three edge devices. Simulations were performed for an IoT device powered with RF energy harvester that aims to provide real time emergency care and healthcare advice. The computation tasks are generated at a speed of 120 kb/s, with each bit takes 1000 CPU cycles to process according to [24]. The IoT device uses the energy harvesting model given by (6) to estimate the amount of the renewable energy generated by the EH module in this time slot. The energy harvesting efficiency υ is 0.51, and the RF transmitter generates 3 W equivalent isotropically radiated energy power. The combined gain of the RF energy transmitter antenna and IoT antenna is 6 dB and the path loss is 2. The distance $d^{(k)}$ is modeled as a Markov chain with $\Pr\left(d^{(k+1)}=m|d^{(k)}=n\right)=d_{mn}$, $\forall\, m,n\in\{1,3,5,7,9\}$ m, according to [21].

The estimation error of the IoT device regarding the amount of harvested energy is assumed to follow a uniform distribution with $\triangle^{(k)}\sim 0.1U(-1,1)$. The IoT device uses the transmit power 0.5 W, and has the effective capacitance coefficient 10^{-11} according to [23]. The IoT device uniformly quantizes the battery level $0\le b^{(k)}\le 7.4$ Wh into 21 levels and the harvested energy $0\le\rho^{(k)}\le 1.7\times10^{-3}$ Wh into 6 levels as a tradeoff between the exploration cost and offloading performance.

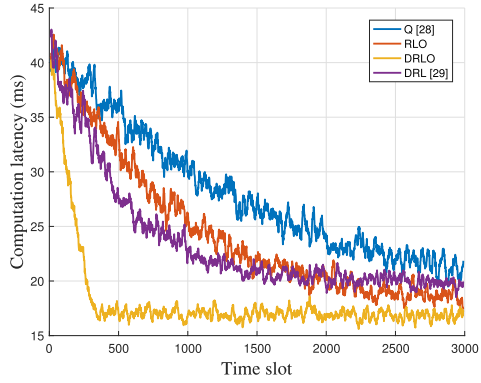
In the simulations, the radio transmission rate of the IoT-edge 1 link is modelled as a Markov chain with $\Pr(B_1^{(k+1)}=m|B_1^{(k)}=n)=B_1^{mn}$, $\forall\ m,n\in\{3,4,5,6,7\}$ Mbps. Similarly, the radio transmission rates for the 2^{nd} (or 3^{rd}) edge device is modelled as an independent Markov chain with $\Pr(B_2^{(k+1)}=m|B_2^{(k)}=n)=B_2^{mn}$ (or $\Pr(B_3^{(k+1)}=m|B_3^{(k)}=n)=B_3^{mn}$), $\forall\ m,n\in\{8,9,10,11,12\}$ Mbps (or {5, 6, 7, 9, 10} Mbps). Unless otherwise specified, a typical time slot lasts 1 s, $\psi=10$, $\beta=0.7$ and $\mu=1$.

In the learning algorithm, we set $\alpha=0.9$, $\gamma=0.5$, $\varepsilon=0.1$, $W=10$, and $H=16$, and the CNN parameters of the DRLO scheme are selected and listed in Table II, to achieve good offloading performance according to the experiments not presented here. We define the task drop rate as the fraction of failed computation tasks. The simulations evaluate three benchmark scheme, including DRL in [29], the Q-learning based offloading scheme in [28], and the non-offloading scheme.

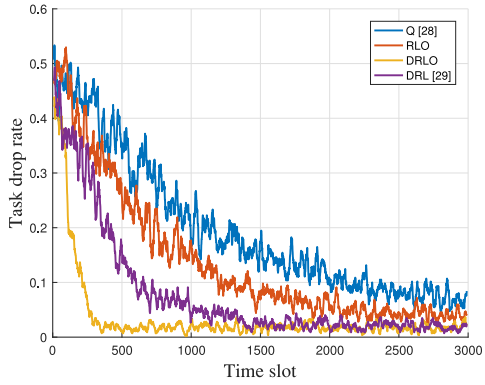
Fig. 3 shows that the DRLO scheme achieves the optimal computation offloading performance after convergence, which



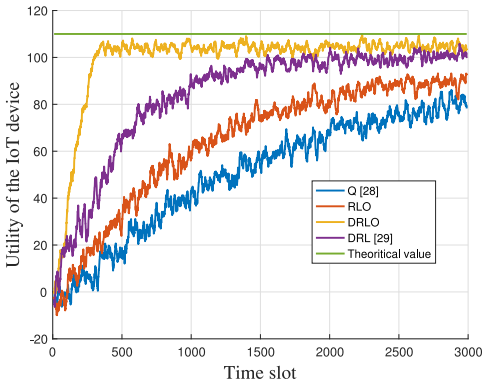
(a) Energy consumption



(b) Computation latency



(c) Task drop rate



(d) Utility of the IoT device

Fig. 3. Performance of the IoT computation offloading over time.

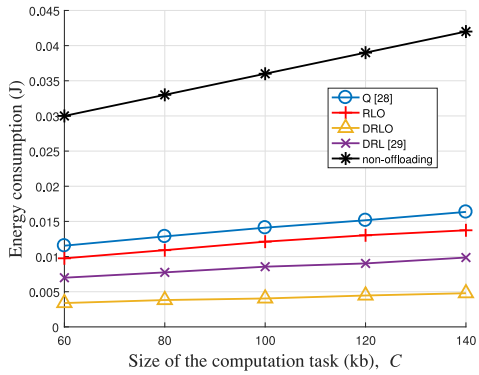
TABLE II
CNN PARAMETERS

Layer	Conv 1	Conv 2	FC 1	FC 2
Input	6×6	$4 \times 4 \times 20$	360	180
Filter size	3×3	2×2	/	/
Stride	1	1	/	/
# filters	20	40	180	$\|\mathbf{A}\|$
Activation	ReLU	ReLU	ReLU	ReLU
Output	$4 \times 4 \times 20$	$3 \times 3 \times 40$	180	$\|\mathbf{A}\|$

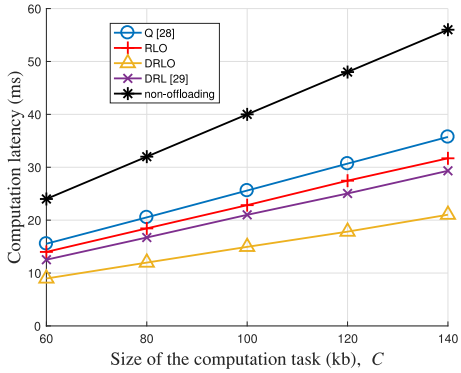
is in close agreement with the analysis results in Theorem 1. For example, the utility of the IoT device almost converges to the performance bound, given by (20), if the simulation settings satisfy (14) and (15). In this case, the IoT device tends to offload all the computation tasks to the edge device to maximize the system performance.

Moreover, the RLO scheme outperforms the Q-learning offloading scheme, by having a lower energy consumption, shorter computation latency, and lower task drop rate. For instance, the RLO scheme reduces the energy consumption by 28.6%, the computation latency by 16.7%, and the task drop rate of the IoT device by 25.0%, compared to the Q-learning scheme at time slot 1000. The computation offloading with DRLO further improves the performance, e.g., it reduces the energy consumption by 75.0%, computation latency by 32.6%, and the task drop rate of the IoT device by 85.6%, at time slot 1000, compared to the RLO scheme. That is attributed to the fact that the DRLO algorithm, an extension of Q learning, compresses the learning state space by using CNN to accelerate the learning process and enhance the computation offloading performance. If the interaction time is sufficiently long enough, the RLO scheme can also converge to the optimal theoretical results. We can also see that, the proposed DRLO scheme outperforms the DRL scheme in [29]. For instance, it reduces the energy consumption by 58.3%, the computation latency by 26.7%, the task drop rate of the IoT device by 55.5%, and thus increases the utility by 14.6%, at time slot 1000. That is because the DRLO scheme incorporates the EH technique and more system state to improve the computation performance.

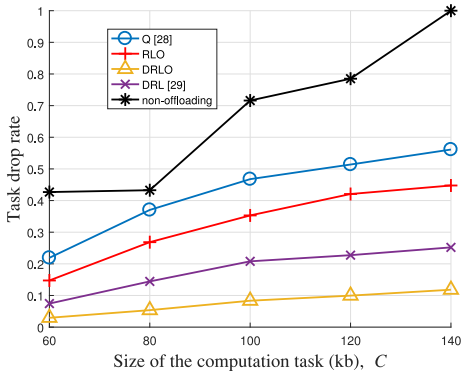
The offloading performance averaged over first 1500 time slot in the dynamic offloading is studied. Fig. 4 shows the relationship between the system performance and the size of the computation task C. Clearly, the energy consumption, computation latency, and task drop rate of the IoT device increase with the amount of computation task in MEC. For instance, if the amount of the computation task changes from 60 kb to 140 kb, the energy consumption, computation latency and task drop rate of the IoT device with DRLO scheme increase by 33.3%, 58.5% and 81.8%, respectively. In the dynamic game with $C = 100$ kb computation tasks, the DRLO outperforms the RLO with 68.1% lower energy consumption, 33.3% shorter computation latency, and 75.7% lower task drop rate, and outperforms the non-offloading scheme with 89.1% lower energy consumption, 62.5% shorter computation latency, and 88.1% lower task drop rate.



(a) Energy consumption



(b) Computation latency



(c) Task drop rate

Fig. 4. Average performance of the IoT computation offloading for the given size of computation task.

Fig. 5 exhibits the impact of the RF transmission power η of the energy transmitter in the wireless powered MEC network, with η increases from 1 W to 3 W. The task drop rate decreases with the increase in the RF transmission power of the energy transmitter. For instance, if the RF transmission power changes from 1 W to 3 W, the task drop rate of the DRLO scheme decreases by 90.8%. In the dynamic mobile offloading process with the RF transmission power $\eta = 2$ W, the task drop rate of the DRLO is 65.7% less than that of RLO, which is 58.6% and 90.1% lower than that of the DRL and non-offloading scheme, respectively.

As shown in Table III, the DRLO requires 345 MB memory most of the time, which can be supported by IoT platforms, such as Qualcomm Snapdragon 800 and Nvidia Tegra K1 according

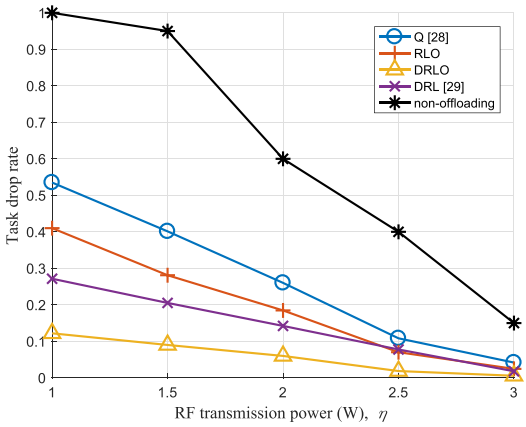


Fig. 5. Task drop rate of the IoT computation offloading for given RF transmission power, η.

TABLE III
COMPUTATION OVERHEAD OF THE RL-BASED OFFLOADING SCHEMES

Scheme	Memory size (MB)	Policy selection time (ms)	Convergency time (time slot)
DRLO	345	8.3	400
RLO	168	0.4	> 3000

to [19]. On the other hand, the IoT devices with constrained computational resources that cannot support deep learning can use RLO to select the offloading policy with significantly reduced computation complexity. For example, the RLO scheme saves 95.2% of the time to select the offloading policies in a time slot compared with the DRLO.

VIII. CONCLUSION

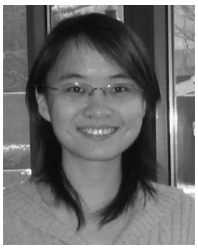
Energy harvesting powered mobile edge computing is a promising approach to improve the computation capabilities for self-sustainable IoT devices. In this paper, we have presented the RL-based computation offloading framework for IoT devices with EH to achieve the optimal offloading policy without awareness of the MEC model, the computation latency and energy consumption model. An RLO scheme has been proposed for IoT devices with low complexity, which utilizes the transfer learning technique to accelerate the learning speed. We have also presented a DRLO scheme that uses CNN to compress the state space in the learning process. Furthermore, the conditions for both "fully offloading" and "locally processing" are provided, and the performance bounds of the proposed RL-based offloading after convergence under three typical scenarios are studied. Simulations have been performed for the IoT device with RF signal-based wireless power transfer to verify the theoretical results, indicating that the DRLO scheme generates the best policy, and outperforms the DRL scheme with 58.3% lower energy consumption, 26.7% less computation latency, and 55.5% lower task drop rate. We will implement the RL-based offloading schemes and evaluate their performance via experiments in our future work.

REFERENCES

[1] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surv. Tut.*, vol. 19, no. 4, pp. 2322–2358, Aug. 2017.
[2] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen, "Vehicular fog computing: A viewpoint of vehicles as the infrastructures," *IEEE Trans. Veh. Technol.*, vol. 65, no. 6, pp. 3860–3873, Jun. 2016.
[3] J. Feng, Z. Liu, C. Wu, and Y. Ji, "AVE: Autonomous vehicular edge computing framework with ACO-based scheduling," *IEEE Trans. Veh. Technol.*, vol. 66, no. 12, pp. 10660–10675, Dec. 2017.
[4] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *IEEE Trans. Wireless Commun.*, vol. 11, no. 6, pp. 1991–1995, Jun. 2012.
[5] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 854–864, Jun. 2016.
[6] S. Sudevalayam and P. Kulkarni, "Energy harvesting sensor nodes: Survey and implications," *IEEE Commun. Surv. Tut.*, vol. 13, no. 3, pp. 443–461, Jul.–Sep. 2011.
[7] D. Mishra *et al.*, "Smart RF energy harvesting communications: Challenges and opportunities," *IEEE Commun. Mag.*, vol. 53, no. 4, pp. 70–78, Apr. 2015.
[8] D. Mishra, S. De, and D. Krishnaswamy, "Dilemma at RF energy harvesting relay: Downlink energy relaying or uplink information transfer?" *IEEE Trans. Wireless Commun.*, vol. 16, no. 8, pp. 4939–4955, Aug. 2017.
[9] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 17, no. 3, pp. 1784–1797, Dec. 2017.
[10] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Trans. Commun.*, vol. 65, no. 8, pp. 3571–3584, Apr. 2017.
[11] C. You, K. Huang, and H. Chae, "Energy efficient mobile cloud computing powered by wireless energy transfer," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 5, pp. 1757–1771, May 2016.
[12] S. Bi and Y.-J. A. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *IEEE Trans. Wireless Commun.*, vol. 17, no. 6, pp. 4177–4190, Jun. 2018.
[13] F. Lin, C. Chen, N. Zhang, X. Guan, and X. Shen, "Autonomous channel switching: Towards efficient spectrum sharing for industrial wireless sensor networks," *IEEE Internet Things J.*, vol. 3, no. 2, pp. 231–243, Apr. 2016.
[14] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3590–3605, Sep. 2016.
[15] J. Xu, L. Chen, and S. Ren, "Online learning for offloading and autoscaling in energy harvesting mobile edge computing," *IEEE Trans. Cogn. Commun. Netw.*, vol. 3, no. 3, pp. 361–373, Jul. 2017.
[16] R. S. Sutton and A. G. Barto, *Reinforcement learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.
[17] H. Zuo, G. Zhang, W. Pedrycz, V. Behbood, and J. Lu, "Fuzzy regression transfer learning in Takagi–Sugeno fuzzy models," *IEEE Trans. Fuzzy Syst.*, vol. 25, no. 6, pp. 1795–1807, Dec. 2017.
[18] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Jan. 2015.
[19] N. D. Lane, S. Bhattacharya, P. Georgiev, C. Forlivesi, and F. Kawsar, "An early resource characterization of deep learning on wearables, smartphones and internet-of-things devices," in *Proc. Int. Workshop Internet Things Appl.*, New York, NY, USA, Nov. 2015, pp. 7–12.
[20] N. D. Lane, P. Georgiev, and L. Qendro, "DeepEar: Robust smartphone audio sensing in unconstrained acoustic environments using deep learning," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, Osaka, Japan, Sep. 2015, pp. 283–294.
[21] S. Bi, C. K. Ho, and R. Zhang, "Wireless powered communication: Opportunities and challenges," *IEEE Commun. Mag.*, vol. 53, no. 4, pp. 117–125, Apr. 2015.
[22] X. Lu, P. Wang, D. Niyato, D. I. Kim, and Z. Han, "Wireless networks with RF energy harvesting: A contemporary survey," *IEEE Commun. Surv. Tut.*, vol. 17, no. 2, pp. 757–789, Nov. 2015.
[23] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 12, no. 9, pp. 4569–4581, Aug. 2013.
[24] C. You, K. Huang, H. Chae, and B. H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, Mar. 2017.

[25] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4268–4282, Aug. 2016.
[26] L. Xiao, C. Xie, T. Chen, H. Dai, and H. V. Poor, "A mobile offloading game against smart attacks," *IEEE Access*, vol. 4, pp. 2281–2291, May 2016.
[27] W. Zhang, Y. Hu, Y. Zhang, and D. Raychaudhuri, "SEGUE: Quality of service aware edge cloud service migration," in *Proc. IEEE Int. Conf. Cloud Comput. Technol. Sci.*, Luxembourg City, Luxembourg, Dec. 2016, pp. 344–351.
[28] L. Xiao, Y. Li, X. Huang, and X. Du, "Cloud-based malware detection game for mobile devices with offloading," *IEEE Trans. Mobile Comput.*, vol. 16, no. 10, pp. 2742–2750, Oct. 2017.
[29] J. Li *et al.*, "Deep reinforcement learning based computation offloading and resource allocation for MEC," in *Proc. IEEE Wireless Commun. Netw. Conf.*, Barcelona, Spain, Jun. 2018, pp. 1–6.
[30] N. Lu, N. Cheng, N. Zhang, X. Shen, and J. W. Mark, "Connected vehicles: Solutions and challenges," *IEEE Internet Things J.*, vol. 1, no. 4, pp. 289–299, May 2014.
[31] O. Munoz, A. Pascual-Iserte, and J. Vidal, "Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading," *IEEE Trans. Veh. Technol.*, vol. 64, no. 10, pp. 4738–4755, Oct. 2015.
[32] W. Liu, J. Cao, L. Yang, L. Xu, X. Qiu, and J. Li, "AppBooster: Boosting the performance of interactive mobile applications with computation offloading and parameter tuning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 6, pp. 1593–1606, Nov. 2017.
[33] 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.
[34] H. Wang and J. Huang, "Incentivizing energy trading for interconnected microgrids," *IEEE Trans. Smart Grid*, vol. 9, no. 4, pp. 2647–2657, Jul. 2018.
[35] E. R. Gomes and R. Kowalczyk, "Dynamic analysis of multiagent Q-learning with ε-greedy exploration," in *Proc. ACM Annu. Int. Conf. Mach. Learn.*, Montreal, QC, Canada, Jun. 2009, pp. 369–376.
[36] C. C. T. Mendes, V. Frémont, and D. F. Wolf, "Exploiting fully convolutional neural networks for fast road detection," in *Proc. IEEE Int. Conf. Robot. Automat.*, Stockholm, Sweden, May 2016, pp. 3174–3179.
[37] K. He and J. Sun, "Convolutional neural networks at constrained time cost," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Boston, MA, USA, Jun. 2015, pp. 5353–5360.

Minghui Min received the B.S. degree in automation from Qufu Normal University, Rizhao, China, in 2013, and the M.S. degree in control theory and control engineering from Shenyang Ligong University, Shengyang, China, joint training with Shenyang Institute of Automation, Chinese Academy of Sciences, Shengyang, China, in 2016. She is currently working toward the Ph.D. degree with the Department of Communication Engineering, Xiamen University, Xiamen, China. Her research interests include network security and wireless communications.



Liang Xiao (M'09–SM'13) received the B.S. degree in communication engineering from Nanjing University of Posts and Telecommunications, Nanjing, China, in 2000, the M.S. degree in electrical engineering from Tsinghua University, Beijing, China, in 2003, and the Ph.D. degree in electrical engineering from Rutgers University, New Brunswick, NJ, USA, in 2009. She was a Visiting Professor with Princeton University, Virginia Tech, and University of Maryland, College Park. She is currently a Professor with the Department of Communication Engineering, Xiamen University, Xiamen, China. She was an Associate Editor for the IEEE TRANSACTIONS INFORMATION FORENSICS AND SECURITY and a Guest Editor for the IEEE JOURNAL OF SELECTED TOPICS IN SIGNAL PROCESSING. She was the recipient of the Best Paper Award for 2016 INFOCOM Big Security WS and 2017 ICC.

Ye Chen received the B.S. degree in communication engineering from Xiamen University, Xiamen, China, in 2017, where he is currently working toward the M.S. degree with the Department of Communication Engineering. His research interests include network security and wireless communications.

Peng Cheng received the B.E. degree in automation and the Ph.D. degree in control science and engineering in 2004 and 2009, respectively, both from Zhejiang University, Hangzhou, China. He is currently a Professor with the College of Control Science and Engineering, Zhejiang University. His research interests include networked sensing and control, cyber-physical systems, and control system security. He is as Associate Editor for the IEEE TRANSACTIONS ON CONTROL OF NETWORK SYSTEMS, *Wireless Networks*, *International Journal of Communication Systems*, and a Guest Editor for the IEEE TRANSACTIONS ON SIGNAL AND INFORMATION PROCESSING OVER NETWORKS. He serves/served as the TPC Co-Chair of the IEEE IOV 2016 Local Arrangement Chair for the IEEE MobiHoc 2015 and the Publicity Co-Chair for the IEEE MASS 2013.

Di Wu (M'06–SM'17) received the B.S. degree from the University of Science and Technology of China, Hefei, China, in 2000, the M.S. degree from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2003, and the Ph.D. degree in computer science and engineering from the Chinese University of Hong Kong, Hong Kong, in 2007. He was a Postdoctoral Researcher with the Department of Computer Science and Engineering, Polytechnic Institute of New York University, Brooklyn, NY, USA, from 2007 to 2009, advised by Prof. K. W. Ross. He is currently a Professor and the Assistant Dean with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China. His research interests include cloud computing, multimedia communication, Internet measurement, and network security. He was the recipient of the IEEE INFOCOM 2009 Best Paper Award.

Weihua Zhuang (M'93–SM'01–F'08) has been with the Department of Electrical and Computer Engineering, University of Waterloo, Canada, since 1993, where she is a Professor and a Tier I Canada Research Chair in Wireless Communication Networks. She is a Fellow of the Royal Society of Canada, the Canadian Academy of Engineering, and the Engineering Institute of Canada. She was the Editor-in-Chief for the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY (2007–2013), the Technical Program Chair/Co-Chair of IEEE VTC Fall 2017 and Fall 2016, and the Technical Program Symposia Chair of the IEEE Globecom 2011. She is an elected member in the Board of Governors and VP Publications of the IEEE Vehicular Technology Society. She was an IEEE Communications Society Distinguished Lecturer (2008–2011). She was the recipient of the 2017 Technical Recognition Award from the IEEE Communications Society Ad Hoc and Sensor Networks Technical Committee, one of 2017 ten N2Women (Stars in Computer Networking and Communications), and a co-recipient of several Best Paper awards from IEEE conferences.