Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

DQN inspired Joint Computing and Caching Resource Allocation Approach for Software Defined Information-Centric Internet of Things Network

FANGMIN XU[1], FAN YANG[1], SHIJIAN BAO [2], AND CHENGLIN ZHAO[1] ,

[1]Key Laboratory of Universal Wireless Communications Ministry of Education, Beijing University of Posts and Telecommunications, Beijing 100876, China
[2]China International Engineering Consulting Corporation, Beijing 100048, China

Corresponding author: Fangmin Xu (e-mail: xufm@bupt.edu.cn).

ABSTRACT With the rapid development of the Internet of Things (IoT) network, IoT devices need to perform Artificial Intelligence (AI) model to make decisions according to specifical service requirement under dynamic environment. However, the generation and usage of AI model typically requires huge amount of communication, computing and caching resource. Thus, the construction of the network and the scheduling of the limited network resources to realize the rapid generation and propagation of AI models are critical. Therefore, we propose a software-defined Information Centric-Internet of Things architecture (IC-IoT) to bring caching and computing capabilities to IoT network. Based on the proposed IC-IoT architecture, we design a jointly resource scheduling scheme to uniformly manage the computing and caching resources. The objective is to maximum the reward which consists not only short-term reward, but also long-term reward brought by caching popular AI models. The resource scheduling problem is formulated into a multi-dimensional optimization problem. A new deep Q-learning method is proposed due to the complexity and high dimension of this problem. Simulation results verify the effectiveness of software defined IC-IoT architecture and the joint resource allocation strategy.

INDEX TERMS Information Centric Network; Internet of Things; Mobile Edge Computing; Joint Optimization; Deep Q-Learning

I. INTRODUCTION

Information Centric Network (ICN) is a type of network that changes users focus from the terminal to the content, separates the content from the terminal location, and provides storage and communication services through the publish/subscribe paradigm [1]. ICN has changed the pattern of the existing IP network. It is characterised by the fact that the network carries a large amount of contents and the users main behavior is to access certain popular contents. ICN uses the request and acquire information as the basic behavior mode instead of the traditional end-to-end connection which can effectively distribute the contents. Thus, ICN can be applicable to a large number of application scenarios, especially in Internet of Things (IoT) network which needs to distribute large amount of information.

IoT applications have gradually been incorporated into all areas of the production and our daily lives. IoT is becoming more and more intelligent with the in-depth research work on related technologies. IoT devices need to make decisions by training and running Artificial Intelligence (AI) models according to environmental changes in real time [2]. However, computation capabilities of the devices are too weak to train and run the models. By offloading the complex computation tasks to Mobile Edge Computing (MEC) servers, the computation latency and network transfer load can be reduced. Therefore, edge servers need to acquire AI models and infer the models to analyze the data produced by IoT devices, then return the calculation results to the devices, so that the devices can perform related tasks intelligently.

At the same time, IoT devices generate massive data which enable the edge servers to train AI models with the data, then the AI models are distributed to the other users who need the model. For example, in Industrial Internet of Things (IIoT) scenario, a glass manufacturer wants to inspect the crack on the glass, and another glass manufacturer also wants to inspect the crack on the glass, they can use the same AI model. Thus, the AI models spread through the network in a publish/subscribe paradigm. Therefore, efficient and flexible forwarding AI models and the massive data is an important issue of current IoT research. Compared with the traditional network, ICN pays attention to the information content itself, thus it can transmit the information content efficiently. It is highly suitable for the transmission of the AI models in IoT.

Therefore, we introduce SDN paradigm to Information Centric-Internet of Things (IC-IoT) and extend the function of traditional SDN controllers. This paper proposes a Software Defined Network (SDN) based IC-IoT architecture. Since various computation and caching resources need to work together to provide service for users in this scenario, it is necessary to manage the resources in the network architecture uniformly. However, most of the existing researches only focus on computing resource allocation, and only a few researches focus on joint resource allocation [3]- [6]. Thus, we propose a resource allocation scheme by jointly considering the user requirements, model popularity, computing and caching resource status. The objective of the scheme is to maximum the reward which is consisted of short-term reward and long-term reward. Since the joint resource scheduling problem is complex and involves multiple optimization dimensions. This paper proposes Deep Q-Learning (DQL) algorithm to settle the problem.

The contributions of this paper include:

- Propose a new AI model computing and propagation application scenarios and process of IC-IoT which can be widely used in multiple IoT scenarios.
- Propose a SDN paradigm based IC-IoT network architecture, and propose a joint computing and caching resource allocation scheme based on the architecture. The scheme considers resource status, user requirements and the popularity of the AI model.
- The objective of optimization includes not only the current reward, but also the long-term reward from caching popular AI models in the network. And utilize DQL to solve the multi-dimensional problem.

We organize the rest of the paper as follows. In section 2, we introduce some related works. In section 3, we present the software-defined IC-IoT architecture and system model. In section 4, the joint resource optimization problem is formulated, with the consideration of caching and computing resources. And the problem id solved by DQL. We show the simulation settings and discuss the simulation results in section 5. Finally, conclusions are given in section 6.

II. RELATED WORKS

In the next decade or so, billions of things are expected to be put into use around the world with the support of IoT technology [2], [7]- [10]. Therefore, MEC and ICN technologies are important for processing and transmitting the huge data in a timely and efficient way. And there are many researches focus on applying MEC and ICN to IoT. Moreover, with the help of SDN technology, computing and caching resources in MEC and ICN can be centralize managed to provide higher users' quality of experience (QoE) [11].

A. INTERNET OF THINGS WITH CACHING

Caching technology has special significance for IoT. On one hand, caching is usually able to speed up data retrieval, this could bring shorter IoT task execution delays. But on the other hand, caching and related acceptance operations are very expensive in terms of processing and power consumption.

Therefore, the first question is which devices in the IoT should have caching capabilities. A simple design is to not cache in IoT devices with constrained capabilities [12]. Although it has been confirmed in [13] that network latency can be reduced even caching in an IoT node with small storage. However, this will bring some energy consumption. Therefore, we usually choose to cache content in the IoT gateway or network router [14].

The second problem is that caching all contents results in higher content redundancy and lower resource utilization. The IoT strategy should focus on improving the speed of propagation instead of long-term caching, and it is recommended to taking the content freshness and the requirements of the application into consideration. And the data replacement strategy is essential, while abandoning outdated content [15].

B. INTERNET OF THINGS WITH EDGE COMPUTING

Because IoT devices have limited computing power while need to deal with complex tasks, it has long been proposed to introduce edge computing into the IoT network. At present, edge computing has been applied in various fields of IoT.

In the smart city field, M. Li in [3] proposed the edge computing resource allocation scheme based on delay optimization, by describing the problem as partial observable Markov decision process for minimizing the network expenditure. In [4]- [6], researchers considered network, cache, and computing resources to jointly optimize the tasks computation latency in smart cities.

In the field of industrial Internet, M. Aazam et al discussed how fog provides local computing support in the IIoT environment [16]. X. Li et al proposed an IIoT adaptive transmission model based on SDN and edge computing. F. Xu et al in [17] proposed a centralized edge computing offloading strategy. The strategy brings higher processing and operation performance to the system. Q. Chao et al in [18] considered cache and compute resources together when computing tasks, and dynamically scheduled the resources to maximize the system reward in software-defined industrial internet.

Most of the existing researches on single resource allocation, only some researches focus on joint resource allocation. However, these co-optimization schemes are not applicable to our scenario. Because these co-optimization schemes only consider the short-term reward and ignore the long-term reward. However, caching popular AI models will bring huge long-term reward, since it saves the cost of recalculating the model when the model is requested in the future.

III. SYSTEM MODEL

We first introduce the proposed SDN paradigm based IC-IoT network architecture. Then the network architecture is given, including network model, cache model, and compute model.

A. PROPOSED SOFTWARE-DEFINED IC-IOT ARCHITECTURE

By considering caching and computing resources together, we propose a software-defined IC-IoT architecture. The architecture is consisted of 4 layers, which includes IoT devices, data, control, and application layer which is shown in 1. The proposed framework could be used to provide computing and caching services for IoT applications such as IIoT, smart home, intelligent city and so on.

In the IoT devices layer, for example, in IIoT scenario there are different kinds of field devices such as cameras, sensors, robot arms and band carriers. They are responsible for collecting data and execute instructions On one hand, the data which are collected by them is gathered for training AI models. On the other hand, they use AI models to make behavioral decisions, such as classify products according to the quality.

In the data layer, there are two kinds of resources, ICN routers are responsible for both routing and caching. Popular contents in the network are usually cached in the ICN routers. So that when the other users request the cached content in the future, the routers can return the results directly to users which can speed up content retrieval in the IoT network. Mobile edge gateways with stronger computing capabilities are responsible for computing tasks such as training AI models, and also inferring other AI models to analyze the data. We assume that there are C ICN routers, M MEC gateways, and U users in the system. And we use $\mathcal{C} = \{1, ..., C\}$ to present the set of ICN routers, $\mathcal{M} = \{1, ..., M\}$ to present the set of MEC gateways, and $\mathcal{U} = \{1, ..., U\}$ to present the set users.

The control layer integrates the network intelligence, which enables the system to centralized manage the cache, and compute resources dynamically.

Moreover, many applications are supported in the application layer, such as predictive maintenance, quality inspection and motion control applications in IIoT, and the intelligent electrical control, security monitoring system applications in smart home.

The workflow of this architecture is introduced here. A system user requests an AI model, such as an image identification model to inspect product quality, from an ICN router. First, the ICN router will check if this AI model is cached in the system. If the AI model is cached in the system, the ICN router will reply the user with this AI model. Otherwise, the ICN router sends the task request to SDN controller. The SDN controller runs the joint optimization algorithm and chooses a MEC server to train the model for the user. Moreover, the SDN controller decides whether to store the model in the network according to the popularity of the AI model and the computing and caching cost, and if it does, it chooses an ICN router to store the model. Finally, the ICN router forwards the matched model to the user. For a comprehensive perspective, the flow chart is shown in Fig. 2.

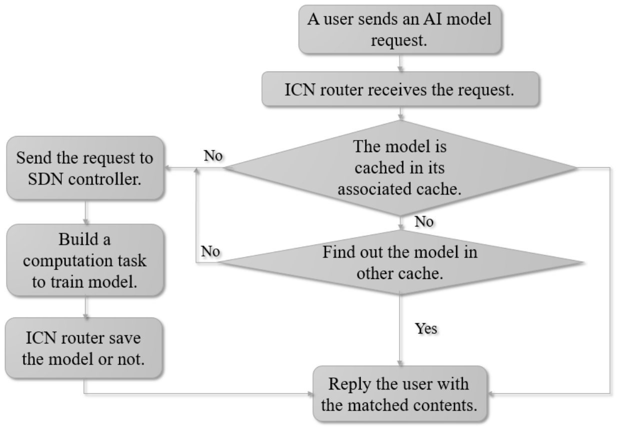
A user sends an AI model request.
ICN router receives the request.
The model is cached in its associated cache.
No
Send the request to SDN controller.
No
Find out the model in other cache.
No
Yes
Build a computation task to train model.
Yes
ICN router save the model or not.
Reply the user with the matched contents.

FIGURE 2: Workflow of the architecture.

B. CACHING MODEL

It is assumed that there are I AI models in the IC-IoT network. The models indexed by i are arranged in decreasing popularity order.The average request rate of the model i is denoted as

$$\lambda_i(t) = \frac{\lambda}{\rho i^{\alpha}}, \tag{1}$$

where the arrival of users requests is a Poisson process, the arrival rate is λ. The AI models request rate is a Zipf-like distribution. The probability of AI model i being requested is $1/\rho i^{\alpha}$, here $\rho = \sum_{i=1}^{I} 1/i^{\alpha}$. α is the Zipf slope, such that $0 < \alpha \leq 1$.

When the AI model i is well trained, it could be cached in an ICN router in this system. However, it could be replaced by other models with higher popularity. Thus, whether or not model i is stored in IC-IoT network is presented by ς_i. $\varsigma_i = 0$ means the AI model i is not stored in an ICN router; $\varsigma_i = 1$ means the AI model i is stored in the ICN router. The cache status of model i in one ICN router at time t is presented as $\varsigma_i(t)$. The caching state changes from one state to another based on a certain transition probability. And $\mathcal{J}_{\varrho_{\bar{s}}\xi_{\bar{s}}}(t)$ denotes the transition probability of $\varsigma_i(t)$ changes from $\varrho_{\bar{s}}$ to $\xi_{\bar{s}}$ at time instant t. The transition probability matrix $\Phi_i(t)$ can be presented as:

$$\Phi_i(t) = [\mathcal{J}_{\varrho_{\bar{s}}\xi_{\bar{s}}}(t)]_{2\times 2}, \tag{2}$$

where $\mathcal{J}_{\varrho_{\bar{s}}\xi_{\bar{s}}}(t) = Pr(\varsigma_i(t+1) = \xi_{\bar{s}} | \varsigma_i(t) = \varrho_{\bar{s}})$, and $\xi_{\bar{s}}, \varrho_{\bar{s}} \in \zeta$.

C. COMPUTING MODEL

There is a computation task $T_u = \{o_u, n_u, t_u\}$ from system user u, and o_u means the output size of the requested AI

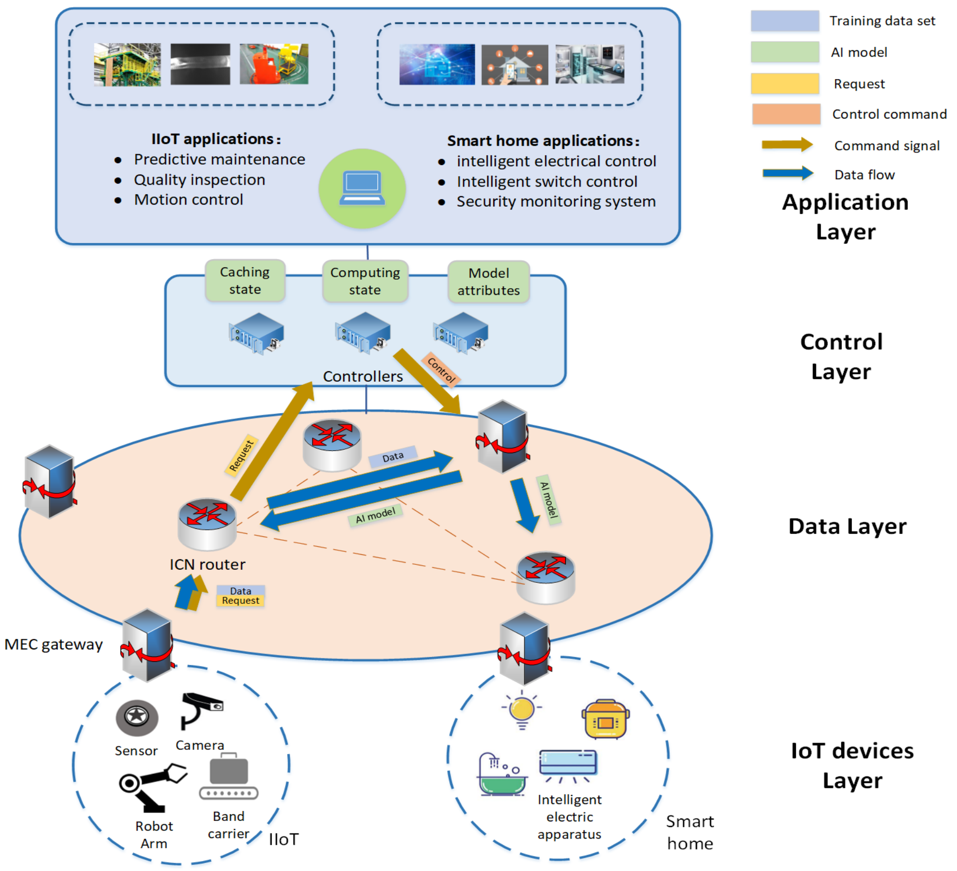

FIGURE 1: The software-defined IC-IoT architecture.

model, n_u means the needed number of CPU cycles for training this model, t_u means the maximum tolerant delay of the task.

The compute capability of MEC gateway m provided for user u is modeled as h_u^m. To discrete the value, we divide the range of h_u^m into L' intervals, i.e., $H = \{H_0, H_1, ..., H_{L'-1}\}$. Computation capability h_u^m at time t can be presented by $h_u^m(t)$ [4]. The computation capability changes from one state to another based on a certain probability. Let $\varepsilon_{\propto_{\bar{s}}\varpi_{\bar{s}}}(t)$ represent the transition probability of $h_u^m(t)$ from $\propto_{\bar{s}}$ to $\varpi_{\bar{s}}$ at time instant t. The transition probability matrix $H_u^m(t)$ is presented as:

$$H_u^m(t) = [\varepsilon_{\propto_{\bar{s}}\varpi_{\bar{s}}}(t)]_{L' \times L'}, \tag{3}$$

where $\varepsilon_{\propto_{\bar{s}}\varpi_{\bar{s}}}(t) = Pr(h_u^m(t+1) = \varpi_{\bar{s}} | h_u^m(t) = \propto_{\bar{s}})$, $\varpi_{\bar{s}}, \propto_{\bar{s}} \in H$.

The training time T_u at MEC gateway m could be expressed as:

$$t_u^m = \frac{n_u}{h_u^m(t)}. \tag{4}$$

The computation latency must be smaller than the maximum tolerant latency of the task when ignoring the transmission latency. Thus,

$$s.t. \frac{n_u}{h_u^m(t)} < t_u \tag{5}$$

Thus the computation rate is:

$$CompR_u^m(t) = a_u^m(t)\frac{o_u}{t_u^m} = a_u^m(t)\frac{h_u^m(t)o_u}{n_u}, \tag{6}$$

where $a_u^m(t)$ means whether user u uses MEC gateway m for training the AI model. Let $a_u^m(t) = 1$ denote MEC gateway m is assigned to user u for training AI model, and $a_u^m(t) = 1$ denote MEC gateway m is not assigned to user u for training AI model.

Moreover, the use of the MEC gateways can not exceed their computation capabilities.

$$s.t. \sum_{u \in \mathcal{U}} a_u^m(t) o_u \le O_m, \tag{7}$$

where O_m is the maximum output data size of AI models can be trained on MEC gateway m concurrently.

Thus, considering cache, and compute resources together in IC-IoT will bring a complex system, the resource scheduling problem in this system is difficult to be solved by conventional algorithms.

IV. PROBLEM FORMULATION

The resources scheduling problem in IC-IoT network is formulated into a DQL process, then the system state, controller action, system reward are put forward.

A. SYSTEM STATE

The system state consists caching status, and computing status. Thus, system state $S(t)$ at time t is represented as:

$$S(t) = \begin{bmatrix} \Gamma_u^1(t) & \Gamma_u^2(t) & ... & \Gamma_u^c(t) & ... & \Gamma_u^C(t) \\ h_u^1(t) & h_u^2(t) & ... & h_u^m(t) & ... & h_u^M(t) \end{bmatrix}, \quad (8)$$

where $\Gamma_u^c(t) = [\varsigma_1(t), \varsigma_2(t), ..., \varsigma_i(t), ..., \varsigma_I(t)]$, $\varsigma_i(t) \in \{0, 1\}$.

B. CONTROLLER ACTION

The controller should decide whether or not the requested AI model should be cached in the system and which ICN router should cache the model. Moreover, the controller need to determine which MEC gateway should train the AI model for user u. Thus, controller action $a_u(t)$ at time instant t could be presented as:

$$a_u(t) = \{CaA_u(t), CompA_u(t)\}, \quad (9)$$

where $CaA_u(t)$, and $CompA_u(t)$ mean:

1) $CaA_u(t) = [CaA_u^0(t), CaA_u^1(t), ..., CaA_u^C(t)]$, where $CaA_u^0(t)$ represents cache the model or not. $CaA_u^0(t) = 1$ means that the AI model is not popular and the network refuses to store the AI model. $CaA_u^0(t) = 0$ means the AI model will be stored. And $CaA_u^c(t), \forall c \in \mathcal{C}$ means whether or not the AI model requested by user u is stored in ICN router c. $CaA_u^c(t) \in \{0, 1\}$, where $CaA_u^c(t) = 0$ means the model is not stored in ICN router c, otherwise $CaA_u^c(t) = 1$. Moreover, only one ICN router store the requested AI model, thus $\sum_{c\in\mathcal{C}} CaA_u^c = 1, \forall u \in \mathcal{U}$.

2) $CompA_u(t) = [CompA_u^1(t), ..., CompA_u^M(t)]$, where $CompA_u^m, \forall m \in \mathcal{M}$ means whether or not the model requested by user u is trained by MEC gateway m. And $CompA_u^m(t) \in \{0, 1\}$, where $CompA_u^m(t) = 0$ means the model is not trained in MEC gateway m, otherwise $CompA_u^m(t) = 1$. Moreover, only one MEC gateway train the AI model for the user u, thus $\sum_{m\in\mathcal{M}} CompA_u^m = 1, \forall u \in \mathcal{U}$.

C. SYSTEM REWARD

The optimization goal of this paper is to maximize the system revenue, so that the system can get the highest profit when it is dealing with the same task. Therefore, the total profit is equal to the difference between the income and the expenditure of the system.

Since the cached AI model may be requested by other network users in the future, which could bring profits. Thus, the reward of this model is consisted of short-term reward and long-term reward.

Short-term reward is the difference between each user's payment and the electricity cost of the system for serving the user. The network consumes electricity when MEC gateway training the model and ICN router caching the model. The decision on whether or not to cache the AI model determines whether or not the system consumes energy due to caching in this time slot.

Long-term reward is obtained if the SDN controller decides to cache the AI model. Thus, the MEC gateway does not need to recalculate the model when the other users will request the model in the future. And energy consumption cost by recalculation could be saved. Thus, the reward function is denoted as:

$$R_u(t) = r_u^s(t) + wr_u^l(t). \quad (10)$$

where w is a weight to balance the short-term and long-term reward. w could be set small when we focus more on short-term reward, and could be set large if we focus more on long-term reward.

Firstly, we consider specifically about short-term reward. The system need to pay for electricity usage caused by caching in ICN router c and computing in MEC gateway m, the electricity fee is denoted as η_m per killowatt-hour. Meanwhile, the system charge the user u for the fee of training the AI model, which is represented by ϕ_u per million CPU cycle. Thus, short-term reward is denoted as:

$$\begin{aligned} r_u^s(t) = & R_u^{pay} - \sum_{c\in\mathcal{C}} E_{u,c}^{cahche} - \sum_{m\in\mathcal{M}} E_{u,m}^{comp} \\ = & \phi_u n_u - \sum_{c\in\mathcal{C}} CaA_u^c(t)\eta_m o_u w_c \\ & - \sum_{m\in\mathcal{M}} CompA_u^m(t)\eta_m n_u e_m \\ = & \phi_u n_u - \eta_m(\sum_{c\in\mathcal{C}} CaA_u^c(t) w_c o_u \\ & + \sum_{m\in\mathcal{M}} CompA_u^m(t) e_m n_u). \end{aligned} \quad (11)$$

where w_c denotes the electricity consumption for caching one bit content,and e_m denotes the electricity consumption of CPU running a cycle.

The long-term reward is defined as expected return within the next k time slots. $\mathcal{J}_{\varrho_{\bar{s}}\varrho_{\bar{s}}}(t)$ is probability that model i remains cached. Thus, the long-term reward is denoted as:

$$r_u^l(t) = \eta_m n_u e_m CaA_u^0 \frac{1 - \mathcal{J}_{\varrho_{\bar{s}}\varrho_{\bar{s}}}(t)^k}{1 - \mathcal{J}_{\varrho_{\bar{s}}\varrho_{\bar{s}}}(t)} \frac{\lambda}{\rho i^\alpha} \quad (12)$$

.

Thus, by considering short-term and long-term reward together, the reward function is denoted as:

$$\begin{aligned} R_u(t) = & r_u^s(t) + wr_u^l(t) \\ = & \phi_u n_u + w\eta_m n_u e_m CaA_u^0 \frac{1 - \mathcal{J}_{\varrho_{\bar{s}}\varrho_{\bar{s}}}(t)^k}{1 - \mathcal{J}_{\varrho_{\bar{s}}\varrho_{\bar{s}}}(t)} \frac{\lambda}{\rho i^\alpha} \\ & - \eta_m(\sum_{c\in\mathcal{C}} CaA_u^c(t) w_c o_u + \sum_{m\in\mathcal{M}} CompA_u^m(t) e_m n_u) \end{aligned}$$

. (13)

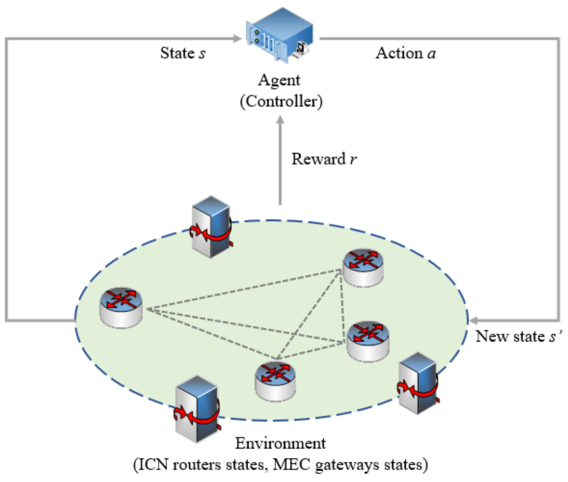

FIGURE 3: Agent-environment iteration paradigm.

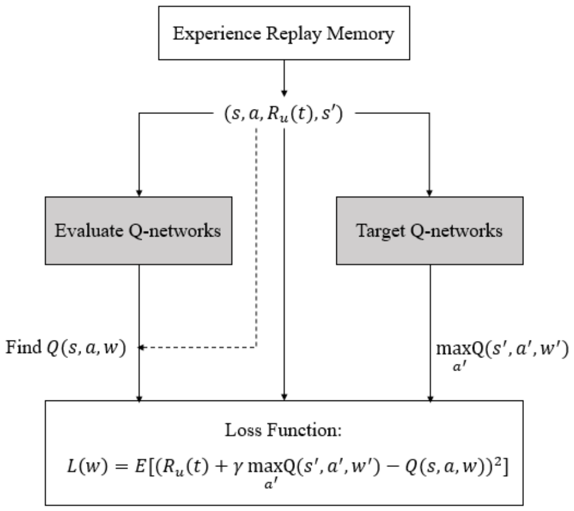

FIGURE 4: The presentation of the loss function.

D. DEEP Q-LEARNING ALGORITHM

In the proposed software defined IC-IoT, the controllers visualize and schedule the the caching resource from ICN routers, and the computing resource from MEC gateways. Due to the following reasons, it is difficult to solve the resource allocation problem in this system with tradition algorithms:

- High complexity and dimensional system.
- Need to learn the regularity of the system environment.
- Need to make decisions while considering both current and long-term reward.

Therefore, we use deep Q-learning (DQL) to choose optimal action which brings maximum reward function.

Fig. 3 displays the interaction between controller and system environment in Q-learning. In each episode, the controller can percept current resources status $S(t)$, i.e., the status of caching resource, and computing resource. Based on a given policy, the controller chooses an action $a_u(t)$ in current environment, such as, it assigns a MEC gateway m to the user u for training the AI model, decides whether to cache the model i and assigns an ICN router for caching the model. Then the environment changes to $S(t+1)$, and the system obtains the reward $R_u(t)$.

The policy function is denoted as π, where $\pi(a_u(t)|S(t))$ presents the probability of choosing action $a_u(t)$ under environment state $S(t)$. Action-state value function $Q^{\pi}(s,a)$ is used to evaluate the expect reward of a policy in Q-learning which is denoted as:

$$Q^{\pi}(s,a) = E^{\pi}[\sum_{k=0}^{\infty} \gamma^k R_u(t+k+1)|S_t = s, a_u(t) = a], \tag{14}$$

where $E^{\pi}[*]$ is the mathematical expectation with state transition probability matrix and policy π. $\gamma \in [0,1]$ is the discount factor to balance current reward and long-term reward.

Q-learning is a temporal-difference learning algorithm, and the evaluation method of $Q(s,a)$ is denoted as:

$$Q(s,a) \leftarrow Q(s,a) + \alpha(R_u(t) + \gamma \max_{a'} Q(s',a') - Q(s,a)), \tag{15}$$

where the learning rate $\alpha \in (0,1]$. $Q(s,a)$ values are recorded in a Q table, and agent usually chooses the action with largest $Q(s,a)$ value.

Since the system state and the controller action in our system is large. It is such difficult to record $Q(s,a)$ in a Q-table. So, neural networks are used to estimate $Q(s,a)$, we use w to denote the parameters of neural networks, the parameters set w include weights and biases. And the neural networks are trained to enable $Q(s,a,w) \approx Q(s,a)$ [19]. In each learning iteration, we train the neural networks to minimize loss, and use the neural networks to estimate real $Q(s,a)$ which is shown in Fig 4 and loss function is expressed as follows:

$$L(w) = E[(R_u(t) + \gamma \max_{a'} Q(s',a',w') - Q(s,a,w))^2]. \tag{16}$$

Two innovations are proposed to make DQL more efficient:

1) Fixed target neural networks. There are two sets of independent neural networks with the same structure. In each episode, one changes parameters to decrease the loss function, called evaluated Q-networks. Another set of neural networks are temporary fixed, and updates parameters with evaluated Q-networks every some steps, called target Q-networks. Then, the algorithm could be trained in a stable manner.

12) Experience replay memory. DQL records experiences in $(s, a, R_u(t), s')$ format in a replay memory with fixed-sized. DQL randomly selects batch of them for training the neural networks.This makes the controller learn from past experiences stored in the memory.

Thus, the process of DQL approach is shown in Fig 5.

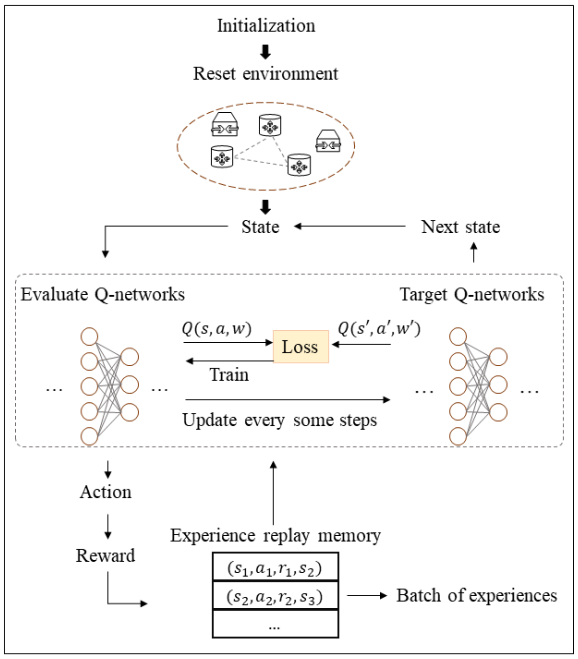

FIGURE 5: The workflow of DQL.

V. SIMULATION AND RESULTS ANALYSIS

A. SIMULATION SETTINGS

We use a CPU-based server, which has 8GB 2666 MHz DDR4 memory, 2GHz Intel Core i7, and 256GB drive. Python 3.6.4 with Tensorflow 1.8.0 is used to provide software environment. Note that we use a seven-layer neural networks in the simulation.

We assume the caching status of one AI model in ICN routers could be cached ($\varsigma_i = 1$), and non-existence ($\varsigma_i = 0$), the transition probability matrix is set as:

$$\Phi = \begin{bmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{bmatrix}. \qquad (17)$$

In a similar way, the computation capability states of MEC gateways could be strong (with the computation rate $h_u^m(t) = 50$), medium ($h_u^m(t) = 10$), and weak ($h_u^m(t) = 1$), the transition probability matrix is set as:

$$E = \begin{bmatrix} 0.6 & 0.3 & 0.1 \\ 0.1 & 0.6 & 0.3 \\ 0.3 & 0.1 & 0.6 \end{bmatrix}. \qquad (18)$$

The rest parameter setting in the simulation are presented in Table 1.

In order to compare the performance, there are four simulation schemes:

- DQL based joint optimization scheme (considering cache and compute resources).
- DQL based computing scheme (considering compute resource only).
- DQL based caching scheme (considering cache resource only).

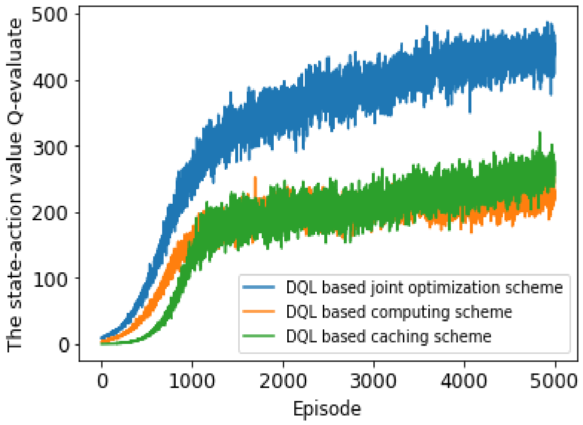

FIGURE 6: State-action value of different schemes.

- Static allocation (static cache, and compute resource allocation).

B. SIMULATION RESULTS

Fig. 6 presents the connection between the state-action value and the training episodes in different schemes. The convergence of state-action value indicates the convergence of the proposed algorithm. We compare the convergence situation of difference schemes in Fig. 6. ϵ-greedy linearly decrease from 1 to 0.1 in 1000 training episodes and then remains stable. With the increasing of training episodes, till about the 1000th episode, Q-evaluate remains stable, which shows good convergence characteristic of the algorithm. Note that, the algorithm converges under different transition probability matrices.

Fig. 7 presents the changing trend of learning loss under DQL based joint optimization scheme in the training process. It also indicates the convergence characteristic of the proposed algorithm. It can be seen that the curve does not decline smoothly, because the input data in DQN is changed step by step, and different data will be obtained according to the learning situation. When training steps reaches 5000, the loss of DQL based joint optimization scheme starts to decrease gradually which means the algorithm eventually converges.

Fig.8 shows the reward performance of different schemes when the required CPU-cycle changes from 5 to 10 million cycles. The revenue of each mode increases, although the increasing number of computing cycles brings higher computing energy consumption. Since the amount of fees paid by users are proportional to the number of CPU cycles required by their computing tasks. However, it can be seen that compared with the other schemes, the DQL based joint optimization scheme has the largest reward. Moreover, it has the largest growth rate of reward. And the DQL based computing scheme has the lowest benefit, even lower than static allocation scheme. Since it gives up huge long-term reward when the cost of energy for caching a model is relatively small, and it is not sensible.

TABLE 1: Setting of simulation parameters.

Symbols	Values	Implications
η_m	7.23 Cent/KWH	Paid-price for consuming one killowatt-hour electricity
ϕ_u	10 Cent/Mcycles	Charging-price of utilizing per Mcycles computation resource.
w_c	0.5J	The electricity consumption for caching one bit content.
e_m	1J	The electricity consumption for running one CPU cycle.
o_u	2.5 Mbits	The content size.
$\frac{1}{\rho i^{\alpha}}$	0.3	The probability of AI model being requested.
n_u	5 Mcycles	The required CPU cycles for training the requested model.
λ	3	The arrival rate of the users requests.
α	0.0001	The learning rate [20].

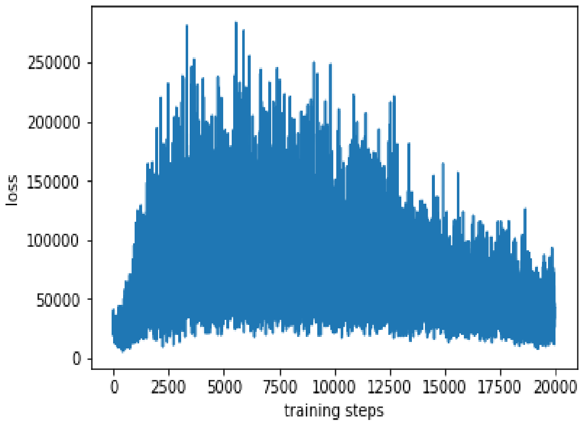

FIGURE 7: Learning loss of DQL based joint optimization scheme.

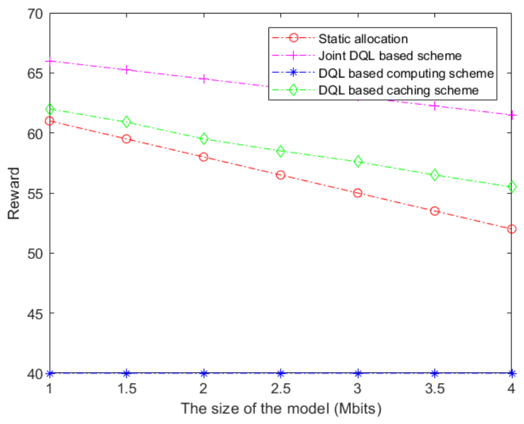

FIGURE 9: The expected reward versus the size of model o_u under different schemes.

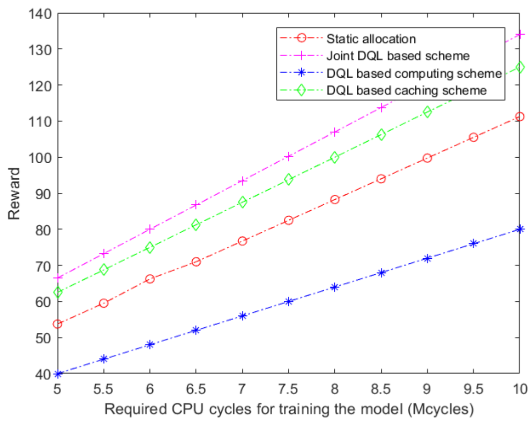

FIGURE 8: The expected reward versus the required CPU cycles n_u under different schemes.

In Fig.9, we simulate the reward performance of different schemes when the model size changes from 1Mbits to 4Mbits. Because caching model is a behavior of the system for gaining long-term reward, users do not need to pay extra fees to the system for caching the model. Therefore, with the data volume of the model increasing, the system gains less and less reward. For models with large enough data volumes, caching the models in the system is not a good strategy. It it obvious that the proposed algorithm can bring the highest reward which shows the advantage of joint optimization scheme.

In Fig.10, we simulate the reward performance of different schemes when the probability of requesting model changes from 0 to 0.6. From the figure, we can see among the DQL based joint optimization scheme, DQL based caching scheme, and static allocation schemes, the joint resource allocation scheme always brings higher reward. Compare with DQL based computing scheme, the DQL based joint optimization scheme could gain higher reward when the model requesting probability is higher than about 0.025. However, when the probability is too small, it has a little smaller reward than the DQL based computing scheme. Since it has a probability of 0.1 to randomly selects actions which leads to a relatively reduction of the reward. However, overall, the DQL based joint optimization scheme performs better compared with the other schemes.

In Fig.11,we simulate the reward performance of different schemes when the arriving rate of the network changes from 1 task/time slot to 6 tasks/time slot. From the figure, we can see among the DQL based joint optimization scheme, DQL based caching scheme, and static allocation schemes, the joint resource allocation scheme always brings higher reward. And the reward increases with the arriving rate.

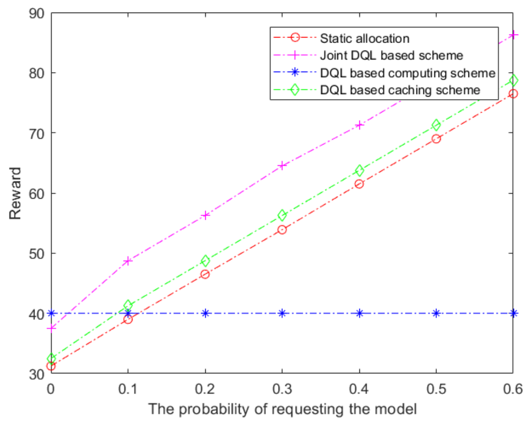

FIGURE 10: The system reward versus the AI model popularity under different schemes.

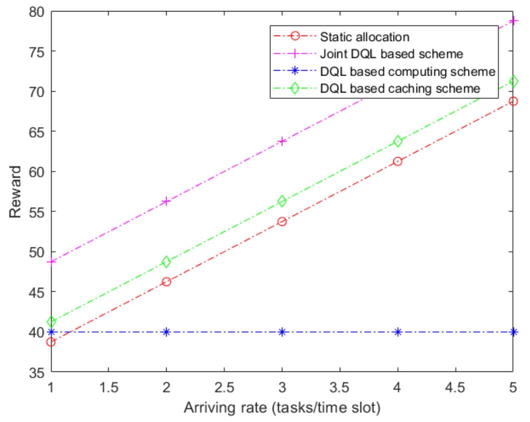

FIGURE 11: The system reward versus the task arriving rate λ under different schemes.

However, when the arriving rate is too low, static allocation scheme which always chooses to cache the content in certain ICN router gains smaller reward than the DQL based computing scheme.

CONCLUSION

This paper propose a software-defined IC-IoT network architecture and proposed a specific application scenarios of IC-IoT, namely the publishing and subscription scenarios of the AI models. Then we propose a joint resource allocation scheme to dynamically manage and arrange the caching and computing resources based on the proposed architecture. The network always selects actions that bring the highest reward to the system. The reward is consisted of short-term reward and long-term reward. We formulate the resource scheduling problem as a multi-dimensional optimization problem and utilize deep Q-learning algorithm to settle the problem. Evaluation results prove that the scheme is effective and convergent in different scenarios. Some future work is underway to consider cloud edge collaboration or collaboration between edges for training the model in order to better promote the proposed approach.

COMPETING INTERESTS

The authors declare that they have no competing interests.

ACKNOWLEDGMENT

The project is supported by the New Model Application Intelligent Manufacturing Projects of MIIT and the Key Program of the National Natural Science Foundation of China under Grant 61431008.

REFERENCES

[1] G. Xylomenos et al., "A Survey of Information-Centric Networking Research," in IEEE Communications Surveys & Tutorials, vol. 16, no. 2, pp. 1024-1049, Second Quarter 2014.
[2] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of Things (IoT): A vision, architectural elements, and future directions,” Future Gener. Comput. Syst., vol. 29, no. 7, pp. 1645–1660, 2013.
[3] M. Li, P. Si and Y. Zhang, "Delay-Tolerant Data Traffic to Software-Defined Vehicular Networks With Mobile Edge Computing in Smart City," in IEEE Transactions on Vehicular Technology, vol. 67, no. 10, pp. 9073-9086, Oct. 2018.
[4] Y. He, F. R. Yu, N. Zhao, V. C. M. Leung, and H. Yin, “Software-defined networks with mobile edge computing and caching for smart cities: A big data deep reinforcement learning approach,” IEEE Commun. Mag., vol. 55, no. 12, pp. 31–37, Dec. 2017.
[5] C.Wang, Y. He, F. R. Yu, Q. Chen, and L. Tang, “Integration of networking, caching and computing in wireless systems: A survey, some research issues and challenges,” IEEE Commun. Surveys Tut., vol. 20, no. 1, pp. 7–38, Jan.–Mar. 2018.
[6] Q. Chen, F. R. Yu, T. Huang, R. Xie, J. Liu, and Y. Liu, “An integrated framework for software defined networking, caching and computing,” IEEE Network, vol. 31, no. 3, pp. 46–55, May 2017.
[7] M. Amadeo et al., “Information-centric networking for the Internet of Things: Challenges and opportunities,” IEEE Netw. Mag., vol. 30, no. 2, pp. 92–100, Mar./Apr. 2016.
[8] S. Sarkar, S. Chatterjee, and S. Misra, “Assessment of the suitability of fog computing in the context of Internet of Things,” IEEE Trans. Cloud Comput., to be published, doi: 10.1109/TCC.2015.2485206.
[9] Y.-S. Jeong, N. Chilamkurti, and L. J. G. Villalba, “Advanced technologies and communication solutions for Internet of Things,” Int. J. Distrib. Sensor Netw., vol. 10, no. 5, p. 3, May 2014.
[10] S. Bera, S. Misra, and M. S. Obaidat, “Mobility-aware flow-table implementation in software-defined IoT,” in Proc. IEEE GLOBECOM, Washington, DC, USA, Dec. 2016, pp. 1–6.
[11] H. Kim and N. Feamster, “Improving network management with software defined networking,” IEEE Commun. Mag., vol. 51, no. 2, pp. 114–119, Feb. 2013.
[12] A. Lindgren et al., “Applicability and Trade-Offs of Information-Centric Networking for Eff cient IoT,” IETF Internet Draft, Jan. 2015.
[13] E. Baccelli et al., “Information Centric Networking in the IoT: Experiments with NDN in the Wild,” ACM Conf. Information-Centric Networking, 2014.
[14] S. Arshad, M. A. Azam, M. H. Rehmani and J. Loo, "Recent Advances in Information-Centric Networking based Internet of Things (ICN-IoT)," in IEEE Internet of Things Journal.
[15] S. Vural et al., “In-Network Caching of Internet-of-Things Data,” Proc. IEEE ICC, 2014.
[16] M. Aazam, S. Zeadally and K. A. Harras, "Deploying Fog Computing in Industrial Internet of Things and Industry 4.0," in IEEE Transactions on Industrial Informatics, vol. 14, no. 10, pp. 4674-4682, Oct. 2018.
[17] F. Xu, H. Ye, F. Yang and C. Zhao, "Software Defined Mission-Critical Wireless Sensor Network: Architecture and Edge Offloading Strategy," in IEEE Access, vol. 7, pp. 10383-10391, 2019.
[18] C. Qiu, F. R. Yu, H. Yao, C. Jiang, F. Xu and C. Zhao, "Blockchain-Based Software-Defined Industrial Internet of Things: A Dueling Deep Q-Learning Approach," in IEEE Internet of Things Journal.

[19] C. Qiu, H. Yao, R. Yu, F. Xu and C. Zhao, "Deep Q-learning Aided Networking, Caching, and Computing Resources Allocation in Software-Defined Satellite-Terrestrial Networks," in IEEE Transactions on Vehicular Technology.

[20] Trishul M Chilimbi, Yutaka Suzue, Johnson Apacible, and Karthik Kalyanaraman. Project Adam: Building an efficient and scalable deep learning training system. In OSDI, volume 14, pages 571-582, 2014.