Deep Reinforcement Learning Based Resource Management for DNN Inference in Industrial IoT

Weiting Zhang, *Student Member, IEEE*, Dong Yang, *Member, IEEE*, Haixia Peng, *Student Member, IEEE*, Wen Wu, *Member, IEEE*, Wei Quan, *Member, IEEE*, Hongke Zhang, *Fellow, IEEE*, and Xuemin Shen, *Fellow, IEEE*

***Abstract*—Performing deep neural network (DNN) inference in real time requires excessive network resources, which poses a big challenge to the resource-limited industrial Internet of things (IIoT) networks. To address the challenge, in this paper, we introduce an end-edge-cloud orchestration architecture, in which the inference task assignment and DNN model placement are flexibly coordinated. Specifically, the DNN models, trained and pre-stored in the cloud, are properly placed at the end and edge to perform DNN inference. To achieve efficient DNN inference, a multi-dimensional resource management problem is formulated to maximize the average inference accuracy while satisfying the strict delay requirements of inference tasks. Due to the mix-integer decision variables, it is difficult to solve the formulated problem directly. Thus, we transform the formulated problem into a Markov decision process which can be solved efficiently. Furthermore, a deep reinforcement learning based resource management scheme is proposed to make real-time optimal resource allocation decisions. Simulation results are provided to demonstrate that the proposed scheme can efficiently allocate the available spectrum, caching, and computing resources, and improve average inference accuracy by 31.4% compared with the deep deterministic policy gradient benchmark.**

***Index Terms*—DNN inference, industrial IoT, resource management, deep reinforcement learning.**

I. Introduction

With the advances in artificial intelligence (AI) technologies [2], modern industry is developing towards smart automation. Emerging industrial Internet of things (IIoT) applications powered by AI, such as intelligent manufacturing, predictive maintenance, and automatic warehousing, are being extensively applied in practical industrial systems [3], [4]. Particularly, deep neural networks (DNNs) have become an essential component for these intelligent IIoT applications [5], [6]. For automated production lines (APLs), the operating status of industrial facilities, e.g., numerically-controlled machine tools, robot arms, and automated guided vehicles, needs to be monitored in real time [7]. Based on the collected sensing data from on-site IIoT devices, DNNs can efficiently perform the condition monitoring and predict the impending failures of industrial facilities, namely *DNN inference*, which can significantly improve the production efficiency.

Manuscript received October 4, 2020; revised January 24, 2021; accepted March 14, 2021. Date of publication March 23, 2021; date of current version August 13, 2021. This work was supported in part by the National Key Research and Development Program of China under Grants 2018YFB1702000 and 2018YFE0206800, in part by the National Natural Science Foundation of China under Grant 61771040, and in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada. This paper was presented at IEEE GLOBECOM 2020 [1]. The review of this article was coordinated by Dr. Yan Zhang. *(Corresponding author: Dong Yang.)*

Weiting Zhang, Dong Yang, Wei Quan, and Hongke Zhang are with the School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing 100044, China (e-mail: 17111018@bjtu.edu.cn; dyang@bjtu.edu.cn; weiquan@bjtu.edu.cn; hkzhang@bjtu.edu.cn).

Haixia Peng, Wen Wu, and Xuemin Shen are with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo N2L 0B5, Canada (e-mail: h27peng@uwaterloo.ca; w77wu@uwaterloo.ca; sshen@uwaterloo.ca).

Digital Object Identifier 10.1109/TVT.2021.3068255

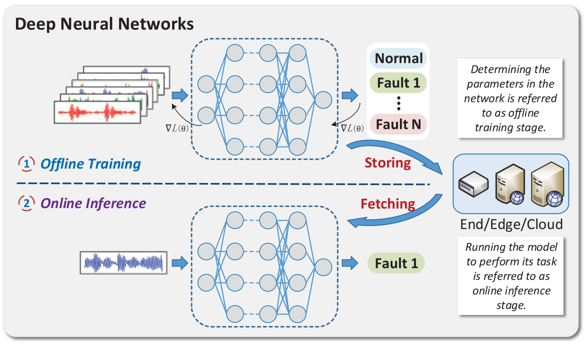
Deep Neural Networks
Normal
Fault 1
Fault N
∇L(θ)
∇L(θ)
Determining the parameters in the network is referred to as offline training stage.
1 Offline Training
Storing
2 Online Inference
Fetching
End/Edge/Cloud
Fault 1
Running the model to perform its task is referred to as online inference stage.

Fig. 1. The procedure of implementing DNN inference: (1) offline training and (2) online inference.

As shown in Fig. 1, implementing DNN inference requires two stages, namely, offline training and online inference [8]. In the offline training stage, the parameters of the DNN models are updated via iteratively training, and the well-trained DNN models are stored in a cloud server. In the online inference stage, the pre-trained DNN models can be placed at edge servers or end devices, to perform the inference tasks. With the wide deployment of intelligent IIoT applications, a large amount of inference tasks need to be processed instantly, which requires high demands on computing resources [9]. Due to the limited computation capabilities of IIoT devices, computation-intensive inference tasks can be offloaded to the remote cloud server [10], yet long-distance data transmission results in a high response latency [11]. To provide delay-sensitive computing services, multi-access edge computing (MEC) [12], arises as a novel paradigm by deploying substantial computing resources at the network edge [13]. When the pre-trained DNN models are placed at the edge servers, it is no longer necessary to transmit tasks to the cloud, thus DNN inference tasks can be performed with a low latency.

Except for the computing resource, caching resource is another necessity for inference task execution. When an inference task is offloaded to a node, the task can be executed only if the required pre-trained DNN model is cached in this node. However, neither an edge server nor an end device has sufficient capacity to cache all the required DNN models. Therefore, how to flexibly place the pre-trained DNN models among network nodes with limited caching resources is of significance for supporting DNN inference. In addition, spectrum resource also has a big impact on the transmission delays of inference tasks and pre-trained DNN models, thus reasonable spectrum resource allocation can provide low-latency transmission for tasks and DNN models. Hence, how to jointly manage multi-dimensional resources in IIoT networks to deliver low-latency DNN inference is critical. Moreover, except for the delay requirements, inference accuracy becomes a particularly key performance metric for intelligent IIoT applications [2]. However, due to the heterogeneity of industrial facilities, dedicated pre-trained DNN models need to be provided for inference tasks based on the dynamic service requests, which poses a great challenge to efficiently manage multi-dimensional resources in IIoT networks to provide low-latency and high-accuracy inference services.

Recently, advanced learning-based algorithms [14]–[17] can be applied to improve the decision-making efficiency. Particularly, deep reinforcement learning (DRL) approaches have been utilized to solve resource optimization problems in wireless networks [18]. Deep Q-learning (DQN) and policy gradient based algorithms [19], [20] have demonstrated the strong capability for solving the optimization problems with discrete and continuous decision variables, respectively. Moreover, unlike traditional optimization methods, DRL can effectively capture network dynamics while solving complicated optimization problems with a low time complexity, thereby improving the problem-solving efficiency [21]. Hence, DRL is a promising method to address the multi-dimensional resource management problem for DNN inference.

In this paper, aiming to maximize the average accuracy of inference tasks, we investigate how to leverage DRL to efficiently allocate multi-dimensional resources to ensure DNN inference can be completed within strict delay requirements. Specifically, an end-edge-cloud orchestration architecture is introduced to support massive inference tasks in IIoT networks. In addition, the pre-trained DNN models are flexibly placed at the end device and edge server to provide low-latency inference services. Then, a joint inference task assignment and DNN model placement problem is formulated to maximize the average inference accuracy under multi-dimensional resource constraints. To efficiently solve the optimization problem, we therefore transform the formulated problem into a Markov decision process (MDP), and then propose a twin delayed deep deterministic policy gradient (TD3) based resource management scheme to obtain the solution rapidly. Extensive simulation results are provided to demonstrate that the proposed learning-based resource management scheme can improve the average accuracy of inference tasks while satisfying the strict delay requirements. The main contributions of this paper are summarized as follows.

- We introduce an end-edge-cloud orchestration architecture to flexibly coordinate the inference task assignment and DNN model placement, thereby providing low-latency DNN inference services in IIoT networks.
- We formulate a joint task assignment and DNN placement (TADP) problem to maximize the average accuracy of massive inference tasks while satisfying the delay requirements under constrained spectrum, caching, and computing resources.
- To effectively solve the joint optimization problem, we transform the TADP problem into an MDP, and propose a DRL-based algorithm to learn the inference task assignment and DNN model placement policy and make instant resource allocation decisions for DNN inference.

The remainder of this paper is organized as follows. Section II presents related works. Section III describes the system model and problem formulation. In Section IV, the learning-based resource management scheme is proposed, including the transformation of the formulated problem. Section V presents the performance evaluations for the proposed algorithm, and the concluding remarks are provided in Section VI.

II. Related Work

A. *End, Edge, and Cloud Orchestration for DNN Inference*

Cloud computing significantly promotes the development of AI due to the powerful computing capabilities can be provided for DNN training [22]. Recently, to facilitate efficient DNN inference, the orchestration among end, edge, and cloud has attracted increasing attentions. In [23], an end-cloud synergy DNN inference method is presented for inference accelerating, where the hidden layers of DNN models are respectively processed in the mobile devices or cloud servers such that only intermediate results need to be transmitted. Moreover, Huang *et al.* [24] proposed a layer-level DNN partitioning strategy and deployed different hidden layers of DNN models among the end, edge, and cloud to distribute the computation loads. In addition to partitioning DNN models, in [25], an edge-cloud orchestration scheme is proposed, in which the shallow hidden layers and deep hidden layers of DNN models are sequentially trained in the cloud and edge servers, thereby the online inference tasks can be processed by the edge. To provide low-latency inference services for intelligent IIoT applications, Yang *et al.* [26] proposed an end-edge collaborative framework, in which a light-weight and a heavy-weight DNN models are respectively deployed at IIoT devices and MEC servers, such that the computing resources can be efficiently utilized to perform DNN inference.

In short, the reviewed works provide valuable insights to support efficient DNN inference. Different from the above schemes, our work focuses on joint multi-dimensional resource management in end-edge-cloud orchestrated IIoT networks, thereby providing low-latency inference services for intelligent IIoT applications. In our preliminary paper [1], we consider a homogeneous industrial scenario, in which a pre-trained DNN model can provide inference services for all intelligent IIoT applications. Hence, spectrum and computing resources are allocated to support the inference task assignment. As an extension of [1], in this work, we consider a heterogeneous industrial scenario where the pre-trained DNN models with differentiated structures are required to support intelligent IIoT applications. In particular, the pre-trained DNN models should be appropriately placed among network nodes to satisfy the diversified inference service requests. Thus, except for spectrum and computing resources, we further consider the caching resource to support the flexible DNN model placement.

B. DRL for Resource Management

DRL algorithms have been extensively applied to address the resource management problems due to the efficient decision-making capability. For instance, Liu *et al.* [27] proposed a DQN-based algorithm to maximize the long-term computation resource utility of MEC networks. Zhao *et al.* [28] utilized dueling double DQN to enable efficient user association in cellular networks. In [29], an asynchronous advantage actor critic based scheme has been proposed to achieve adaptive bandwidth allocation in radio access networks, thereby providing reliable transmission for video data streams. In [30], Zhang *et al.* exploited the deep weighted double Q-learning algorithm to learn a dynamic caching policy in ultra dense networks, with the objective of delay minimization. Moreover, in [31], the deep deterministic policy gradient (DDPG) algorithm has been leveraged to solve the joint resource optimization problem in vehicular networks.

To sum up, the above DRL-based schemes can vastly improve the resource utilization in wireless networks. Most of which focused on optimizing different network performance metrics, such as communication delay and network throughput. However, our work considers the DNN inference services, in which inference accuracy is a key performance metric, and we focus on leveraging DRL methods to achieve efficient resource management and provide high-accuracy inference services for intelligent IIoT applications.

III. System Model and Problem Formulation

In this section, an end-edge-cloud orchestration architecture is introduced to support flexible inference task assignment and DNN model placement, and followed by the communication, caching, and computing models for resource allocation in the considered IIoT networks. Then, a joint optimization problem is formulated under multi-dimensional resource constraints.

A. End-Edge-Cloud Orchestration Architecture

We consider an AI-empowered smart factory, where numerous APLs and industrial facilities are operating, and DNN models are deployed to achieve intelligent IIoT applications, e.g., fault identification. To complete a DNN inference at one node, the node should possess the inference task and its dedicated pre-trained DNN model simultaneously. As shown in Fig. 2, we introduce an end-edge-cloud orchestration architecture composed of three layers with different functionalities and resources, i.e., end, edge, and cloud layers.

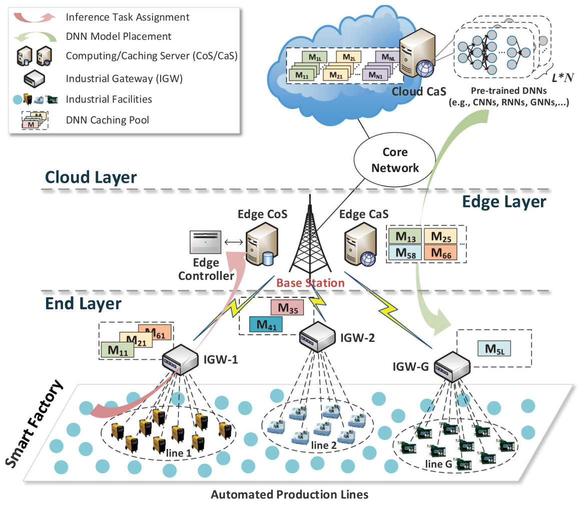

Fig. 2. The end-edge-cloud orchestration architecture in IIoT networks.

- In the cloud layer, sufficient computing and caching resources are deployed, thus DNN training is carried out in this layer, and the well-trained DNN models are stored in the cloud caching server (CCaS) as placeable intelligent services.
- In the edge layer, a pair of edge computing/caching server (ECoS/ECaS) are deployed on one base station (BS) [32], which can support the inference task execution and DNN model placement, respectively. In addition, an edge controller is installed at the ECoS which coordinates net-work nodes and observes network status in real time [33].
- In the end layer, IIoT devices collect real-time data from the APLs, and the industrial gateway (IGW) connects the massive inference tasks into the Internet. In each time slot, the edge controller makes decisions to assign inference tasks to the IGWs or ECoS, and place DNN models from the CCaS to the IGWs or ECaS.

We consider a set $\mathcal{K}_i$ of IIoT devices and a set $\mathcal{G}$ of IGWs which covers all IIoT devices, where $\mathcal{K}_i = \{1, 2, \ldots, |\mathcal{K}_i|\}$ and $i \in \mathcal{G} = \{1, 2, \ldots, |\mathcal{G}|\}$. Hence, we define a DNN inference $I_{i,k} \triangleq [W_{i,k}, M^n_{i,k}], \forall i \in \mathcal{G}, k \in \mathcal{K}_i, n \in \{0, 1\}$, where $W_{i,k}$ and $M^n_{i,k}$ respectively denote the generated inference task and the required DNN model, n denotes the accuracy level of DNN models, i.e., a light-weight DNN with low accuracy and a heavy-weight DNN with high accuracy. The details of the DNN inference definition are described in the following parts. A summary of important notations is given in Table I.

Inference Task: In the considered IIoT scenario, massive inference tasks are generated by various applications. Due to the complex industrial environment, the on-site data is susceptible to be interfered by the high-frequency noises. Hence, for IIoT device k, an inference task is defined as $W_{i,k} \triangleq (D_{i,k}, T^{max}_{i,k}, \theta_{i,k}), \forall i \in \mathcal{G}, k \in \mathcal{K}_i$, where $D_{i,k}$ (in bits) denotes the data size, $T^{max}_{i,k}$ (in seconds) denotes the maximum tolerance of task completion delay, and $\theta_{i,k} \in \{0, 1\}$ denotes the data quality. After the inference task is generated, packet detection is

TABLE I
SUMMARY OF NOTATIONS

Notation	Description
$D_{i,k}$	Task data size of IIoT device k under IGW i
$T_{i,k}^{max}$	Maximum tolerance of task completion delay
$\theta_{i,k}$	Data quality of inference task
$P_{i,k}^{n}$	Inference accuracy of each IIoT device with level n
$Q_{i,k}^{n}$	Required workload
$S_{i,k}^{n}$	Model size
R^c, R^e	Transmission rates from cloud to BS and from BS to IGWs
B, B^e	Available bandwidths from device to IGW and from IGW to BS
$\mathcal{C}^l$, $\mathcal{C}^e$	Computing capabilities of IGWs and ECoS
$\mathcal{S}^l$, $\mathcal{S}^e$	Caching capacities of IGWs and ECoS
$u_{i,k}^{ln}, u_{i,k}^{en}$	Denote whether the required DNN exists in the IGWs or ECaS
$m_{i,k}$	The binary decision variable of task assignment
$\varphi_{i,k}$	The binary decision variable of DNN model selection
$\alpha_{i,k}^{l}, \alpha_{i,k}^{e}$	The decision variables of spectrum allocation
$\beta_{i,k}^{l}, \beta_{i,k}^{e}$	The decision variables of computing resource allocation
$\omega_{i,k}^{ln}, \omega_{i,k}^{en}$	The decision variables of DNN model placement

TABLE II
THE RELATIONSHIP BETWEEN PRE-TRAINED DNNS AND DATA QUALITY

DNNs / Data Quality	Light-weight ($n=0$)	Heavy-weight ($n=1$)
Noiseless Data ($\theta_{i,k}=0$)	$(\mathcal{P}_{UL}, S_{i,k}^0, Q_{i,k}^0)$	$(\mathcal{P}_{UH}, S_{i,k}^1, Q_{i,k}^1)$
Noisy Data ($\theta_{i,k}=1$)	$(\mathcal{P}_{DL}, S_{i,k}^0, Q_{i,k}^0)$	$(\mathcal{P}_{DH}, S_{i,k}^1, Q_{i,k}^1)$

performed first. Here, if the data is noiseless, $\theta_{i,k}=0$, otherwise $\theta_{i,k}=1$. Note that, the data size of a task is the same regardless of $\theta_{i,k}=0$ or 1.

Pre-trained DNN Model: We define a pre-trained DNN model $M_{i,k}^n \triangleq (P_{i,k}^n, Q_{i,k}^n, S_{i,k}^n), \forall i \in \mathcal{G}, k \in \mathcal{K}_i, n \in \{0,1\}$, where $P_{i,k}^n, Q_{i,k}^n$, and $S_{i,k}^n$ represent the inference accuracy, the required workload, and the model size, respectively. Considering the heterogeneity of industrial facilities and the limited resources of IIoT networks, the system provides each task with two accuracy levels' dedicated DNN models, which can be selected to perform the corresponding inference tasks according to the available resources. Here, $n=0$ represents a light-weight DNN model which can provide low-accuracy inference services, and $n=1$ represents a heavy-weight DNN model which can provide high-accuracy inference services. Both levels' DNN models have the same neural network structure, such as fully-connected, recurrent, and convolutional structures, yet are with different number of hidden layers. In addition, the number of hidden layers of heavy-weight DNN models is more than light-weight DNN models. Notably, more hidden layers generally lead to lower approximation errors, such that achieving higher inference accuracy. For each inference task, both accuracy levels of DNNs can be selected to complete the DNN inference, yet a more accurate DNN model occupies more computing and caching resources. Table II shows the relationship among the inference accuracy, the required workload, and the model size. For simplicity, if $\theta_{i,k}=1$, then $P_{i,k}^0=\mathcal{P}_{DL}$ and $P_{i,k}^1=\mathcal{P}_{DH}$; otherwise, $P_{i,k}^0=\mathcal{P}_{UL}$ and $P_{i,k}^1=\mathcal{P}_{UH}$. Note that $\mathcal{P}_{UL}<\mathcal{P}_{UH}, \mathcal{P}_{DL}<\mathcal{P}_{DH}, \mathcal{P}_{DL}<\mathcal{P}_{UL}$, and $\mathcal{P}_{DH}<\mathcal{P}_{UH}$. In addition, $S_{i,k}^0<S_{i,k}^1$ and $Q_{i,k}^0<Q_{i,k}^1$.

At the beginning of each time slot, the edge controller makes the following decisions: 1) where the task should be assigned to, namely task assignment decision; 2) which pre-trained DNN model should be selected and whether it is cached at the corresponding nodes, namely DNN model placement decision; and 3) how much spectrum, computing, and caching resources should be allocated, namely resource allocation decisions. In each time slot, we assume that each IIoT device generates one task, so there are $N=\sum_{i=1}^{G}|\mathcal{K}_i|$ tasks need to be processed by their dedicated pre-trained DNN models. Considering the limited resources of IGWs, the tasks have to be collaboratively performed within the end-edge-cloud orchestration architecture, thereby completing inference tasks within the strict delay requirements.

B. Communication Model

Due to the long distance between IIoT devices and BS, massive sensing data of IIoT devices is aggregated by IGWs. The IGWs further access the aggregated data by cellular network techniques that can support reliable long-distance data transmission. For communication between IIoT devices and IGWs, we consider the industrial wireless sensor network technique [34]. For communication between IGWs and BS, we consider orthogonal frequency division multiple access technique to transmit inference tasks from the IGWs to the BS. Both networks are equipped with the available bandwidth B and B^e, respectively. The task $W_{i,k}$ in each time slot can only be assigned to one node, i.e., IGW i or ECoS. Let $m_{i,k}\in\{0,1\}$ denote the binary decision variable of task assignment. Here, if $m_{i,k}$ is 1, $W_{i,k}$ is accordingly assigned to IGW i, otherwise to ECoS. We assume the communication between each IIoT device and each IGW is assigned an orthogonal channel, i.e., no interference exists in the network. Thus, the spectrum efficiency for IIoT devices is given by

$$\gamma_{i,k}=\log_2\left(1+\frac{p_{i,k}|h_{i,k}|^2}{\sigma^2}\right), \forall i\in\mathcal{G}, k\in\mathcal{K}_i. \qquad (1)$$

Similarly, $\gamma_{i,k}^e=\log_2(1+\frac{p_{i,k}^e|h_{i,k}^e|^2}{\sigma^2})$ represents the spectrum efficiency for IGWs. Here, $p_{i,k}$ and $p_{i,k}^e$ denote the transmission powers of IIoT device k and IGW i, $|h_{i,k}|^2$ and $|h_{i,k}^e|^2$ denote the gains of the IIoT device-IGW and IGW-BS channels, and σ^2 denotes the power of the Gaussian noise. Hence, the transmission rate for device k is calculated by

$$R_{i,k}=\alpha_{i,k}^l B\gamma_{i,k}, \forall i\in\mathcal{G}, k\in\mathcal{K}_i. \qquad (2)$$

Similarly, $R_{i,k}^e=\alpha_{i,k}^e B^e\gamma_{i,k}^e$ represents the transmission rate for IGW i. IIoT device k and IGW i are allocated a fraction $\alpha_{i,k}^l$ of bandwidth B and a fraction $\alpha_{i,k}^e$ of bandwidth B^e, respectively, where $0\le\alpha_{i,k}^l,\alpha_{i,k}^e\le 1$, $\sum_{i\in\mathcal{G},k\in\mathcal{K}_i} m_{i,k}\alpha_{i,k}^l\le 1$, and $\sum_{i\in\mathcal{G},k\in\mathcal{K}_i}(1-m_{i,k})\alpha_{i,k}^e\le 1$. Thus, the transmission delay for offloading an inference task from IIoT device k to the connected IGW i and from device k to the BS can be calculated by

$$t^l_{i,k} = \frac{D_{i,k}}{R_{i,k}}, \forall i \in \mathcal{G}, k \in \mathcal{K}_i, \tag{3}$$

and

$$t^e_{i,k} = \frac{D_{i,k}}{R_{i,k}} + \frac{D_{i,k}}{R^e_{i,k}}, \forall i \in \mathcal{G}, k \in \mathcal{K}_i, \tag{4}$$

respectively. Here, we ignore the transmission delay on wired communications between the BS and ECoS due to the wired connection. In addition, the size of the inference result is much smaller than that of the task [26], hence, the time consumption on returning the inference result to the APLs can be neglected in this paper.

C. Caching Model

As depicted in Section III-A, all the pre-trained DNN models are stored in the CCaS. In addition, the IGWs and ECaS are respectively equipped with $\mathcal{S}^l$ and $\mathcal{S}^e$ caching spaces for caching DNN models. During the system operation, the pre-trained DNN models are placed to the IGWs or ECaS from the CCaS according to the requests. For each inference task, a heavy-weight and a light-weight DNN models can be provided for inference services. In each time slot, the system selects an appropriate DNN model for each inference task. Hence, we define $\varphi_{i,k} \in \{0,1\}, \forall i \in \mathcal{G}, k \in \mathcal{K}_i$, to represent which level of DNN model is selected, where $\varphi_{i,k} = 1$ indicates a heavy-weight DNN and $\varphi_{i,k} = 0$ indicates a light-weight DNN. Note that heavy-weight DNN models have higher inference accuracy, yet occupying more caching resources.

Let binary variables $u^{l_n}_{i,k}, u^{e_n}_{i,k} \in \{0,1\}, \forall i \in \mathcal{G}, k \in \mathcal{K}_i, n \in \{0,1\}$ denote whether the required DNN model exists in the IGWs or ECaS. Taking $u^{l_0}_{i,k}$ as an example, $u^{l_0}_{i,k} = 0$ indicates that the required light-weight DNN does not exist in the corresponding IGWs, otherwise is the opposite. If an inference task is assigned to a node and an appropriate pre-trained DNN model is selected, the edge controller then checks the corresponding caching pools to make sure whether the required DNN has been stored. If $u^{l_n}_{i,k}$ or $u^{e_n}_{i,k}$ is 0, a DNN model placement request of a node in low layers is sent to the superior nodes, and then the required DNN models can be transmitted to the node that sent the request. Note that the request is unidirectional, namely, the required DNN models can only be transmitted from CCaS to ECaS, from CCaS to IGW, or from ECaS to IGW. Hence, the transmission delay of $M^n_{i,k}$ from CCaS to ECaS is given by

$$\tau^{c_n}_{i,k} = \frac{S^n_{i,k}}{R^c}, \forall i \in \mathcal{G}, k \in \mathcal{K}_i. \tag{5}$$

Similarly, $\tau^{e_n}_{i,k} = \frac{S^n_{i,k}}{R^e}$. Thus, $\tau^{l_n}_{i,k} = \tau^{c_n}_{i,k} + \tau^{e_n}_{i,k}$, where $\tau^{e_n}_{i,k}$ and $\tau^{l_n}_{i,k}$ represent the transmission delays from ECaS to IGW, and from CCaS to IGW, respectively. In addition, R^c and R^e denote the transmission rates from cloud to the BS and from BS to the IGWs, respectively.

Let binary decision variables $\omega^{l_n}_{i,k}, \omega^{e_n}_{i,k} \in \{0,1\}, \forall i \in \mathcal{G}, k \in \mathcal{K}_i$ denote whether cache the requested DNN models in the corresponding caching pools after finishing the inference tasks, where $n = 0$ or 1 represents the caching decision for a light-weight or heave-weight DNN model. Taking $\omega^{l_n}_{i,k}$ as an example, $\omega^{l_n}_{i,k} = 1$ indicates that the DNN models can be cached in the corresponding IGWs to provide instant inference services for the following tasks, otherwise is the opposite. With these caching decisions, the pre-trained DNN models can be flexibly placed at the end device and edge server, and the caching resources of network nodes can be utilized more reasonably.

D. Computing Model

The IGWs and ECoS are equipped with the computing capabilities of $\mathcal{C}^l$ and $\mathcal{C}^e$ (CPU cycles per second), respectively. As depicted in Table II, a heavy-weight DNN model achieves high inference accuracy while consuming more computing resource, and a light-weight DNN is the opposite. Let $\beta^l_{i,k}$, and $\beta^e_{i,k} (0 \le \beta^l_{i,k}, \beta^e_{i,k} \le 1)$ denote the fractions of computing resources allocated to the inference tasks, respectively. Thus, the delay for inference task executing at IGWs can be calculated by

$$c^l_{i,k} = \frac{Q^n_{i,k}}{\beta^l_{i,k} \mathcal{C}^l}, \forall i \in \mathcal{G}, k \in \mathcal{K}_i. \tag{6}$$

Similarly, $c^e_{i,k} = \frac{Q^n_{i,k}}{\beta^e_{i,k} \mathcal{C}^e}$ for task executing at ECoS. In addition, the decision variables of computing resources are constrained by $\sum_{i \in \mathcal{G}, k \in \mathcal{K}_i} m_{i,k} \beta^l_{i,k} \le 1$ and $\sum_{i \in \mathcal{G}, k \in \mathcal{K}_i} (1 - m_{i,k}) \beta^e_{i,k} \le 1$.

Accordingly, the inference task completion delay at IGWs and ECoS can be calculated by

$$\begin{aligned} d^l_{i,k} = \; & \max\{t^l_{i,k}, (1 - u^{l_n}_{i,k}) u^{e_n}_{i,k} \tau^{e_n}_{i,k} + (1 - u^{l_n}_{i,k}) \cdot \\ & (1 - u^{e_n}_{i,k}) \tau^{l_n}_{i,k}\} + c^l_{i,k}, \forall i \in \mathcal{G}, k \in \mathcal{K}_i, \end{aligned} \tag{7}$$

and

$$d^e_{i,k} = \max\{t^e_{i,k}, (1 - u^{e_n}_{i,k}) \tau^{c_n}_{i,k}\} + c^e_{i,k}, \forall i \in \mathcal{G}, k \in \mathcal{K}_i, \tag{8}$$

respectively. Here, for $d^l_{i,k}$ or $d^e_{i,k}$, due to the inference task and the required DNN model are simultaneously transmitted to a node, thus the maximum value between the transmission delays of inference task and DNN model is considered as the corresponding total transmission delay.

Taking two cases into consideration, the total time for completing the inference task $W_{i,k}$ can be calculated by

$$d_{i,k} = m^l_{i,k} d^l_{i,k} + m^e_{i,k} d^e_{i,k}, \forall i \in \mathcal{G}, k \in \mathcal{K}_i. \tag{9}$$

Once the DNN selecting decision $\varphi_{i,k}$ is determined and the data quality $\theta_{i,k}$ is detected, the task can be executed and then each IIoT device obtains an inference accuracy, i.e.,

$$\begin{aligned} U_{i,k} = \; & (1 - \theta_{i,k}) \left[\varphi_{i,k} \mathcal{P}_{UH} + (1 - \varphi_{i,k}) \mathcal{P}_{UL}\right] + \\ & \theta_{i,k} [\varphi_{i,k} \mathcal{P}_{DH} + (1 - \varphi_{i,k}) \mathcal{P}_{DL}], \forall i \in \mathcal{G}, k \in \mathcal{K}_i. \end{aligned} \tag{10}$$

Note that the corresponding accuracy can be obtained only when the task is completed within the delay requirement.

E. Problem Formulation

In the considered IIoT networks, we deploy an end-edge-cloud orchestration architecture to support DNN inference, in which the inference tasks of IIoT devices are assigned to the IGWs and ECoS, while the pre-trained DNN models are placed to the IGWs and ECaS from CCaS. In this paper, we focus on enabling high-accuracy DNN inference in IIoT networks. As previously described, inference accuracy is a key performance metric for DNN-based IIoT applications. Only the task is completed within strict delay requirements, the system then can obtain the corresponding accuracy. Enabling the tasks to be reasonably assigned among network nodes to obtain more inference accuracies is important to improve the overall system performance. Therefore, aiming at maximizing the average accuracy of inference tasks, while satisfying the delay requirements of tasks and the constraints of spectrum, caching, and computing resources, we formulate the optimization problem as follows:

$$\max_{m,\boldsymbol{\varphi},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\omega}} \frac{1}{N}\sum_{i=1}^{G}\sum_{k=1}^{|\mathcal{K}_i|} U_{i,k} \tag{11}$$

and subject to:

$$C1: \ d_{i,k} \le T_{i,k}^{\max}, \forall i \in \mathcal{G}, k \in \mathcal{K}_i,$$

$$C2: \ m_{i,k} \in \{0,1\}, \forall i \in \mathcal{G}, k \in \mathcal{K}_i,$$

$$C3: \ \varphi_{i,k} \in \{0,1\}, \forall i \in \mathcal{G}, k \in \mathcal{K}_i,$$

$$C4: \ \sum_{i\in\mathcal{G},k\in\mathcal{K}_i} m_{i,k}\alpha_{i,k}^l \le 1, \ \sum_{i\in\mathcal{G},k\in\mathcal{K}_i}(1-m_{i,k})\alpha_{i,k}^e \le 1,$$
$$0 \le \alpha_{i,k}^l, \alpha_{i,k}^e \le 1,$$

$$C5: \ \sum_{i\in\mathcal{G},k\in\mathcal{K}_i} m_{i,k}\beta_{i,k}^l \le 1, \ \sum_{i\in\mathcal{G},k\in\mathcal{K}_i}(1-m_{i,k})\beta_{i,k}^e \le 1,$$
$$0 \le \beta_{i,k}^l, \beta_{i,k}^e \le 1,$$

$$C6: \ \sum_{i\in\mathcal{G}}\sum_{k\in\mathcal{K}_i} m_{i,k}[\varphi_{i,k}S_{i,k}^1 + (1-\varphi_{i,k})\,S_{i,k}^0] \le \mathcal{S}^l,$$
$$\sum_{i\in\mathcal{G}}\sum_{k\in\mathcal{K}_i}(1-m_{i,k})[\varphi_{i,k}S_{i,k}^1 + (1-\varphi_{i,k})\,S_{i,k}^0] \le \mathcal{S}^e,$$

$$C7: \ \omega_{i,k}^{l_0}, \omega_{i,k}^{l_1}, \omega_{i,k}^{e_0}, \omega_{i,k}^{e_1} \in \{0,1\}, \forall i \in \mathcal{G}, k \in \mathcal{K}_i,$$

where $U_{i,k}$ is given by Eq. (10) and N is the number of inference tasks in each time slot. In addition, the optimization variables consist of five components: 1) $m = \{m_{i,k}\}$ is the task assignment decision matrix for assigning inference tasks to IGWs and ECoS; 2) $\boldsymbol{\varphi} = \{\varphi_{i,k}\}$ is the matrix describing the multi-level DNN selection decisions; 3) $\boldsymbol{\alpha} = \{\alpha_{i,k}^l, \alpha_{i,k}^e\}$ is the matrix describing spectrum resources allocated to inference tasks by the IGWs and BS; 4) $\boldsymbol{\beta} = \{\beta_{i,k}^l, \beta_{i,k}^e\}$ is the matrix describing computing resources allocated to inference tasks by the IGWs and ECoS; and 5) $\boldsymbol{\omega} = \{\omega_{i,k}^{l_0}, \omega_{i,k}^{l_1}, \omega_{i,k}^{e_0}, \omega_{i,k}^{e_1}\}$ is the matrix describing whether cache the requested DNN models in the corresponding nodes. For constraints, $C1$ first guarantees that the entire time cost of DNN inference cannot exceed the maximum delay tolerance $T_{i,k}^{max}$ of $W_{i,k}$. $C2$ and $C3$ ensure that the inference task can only be executed at one node and only one pre-trained DNN model can be selected in each time slot. Moreover, $C4 - C6$ indicate that the resource allocation decisions should be made with the available bandwidth (B and B^e), computing ($\mathcal{C}^l$ and $\mathcal{C}^e$), and caching ($\mathcal{S}^l$ and $\mathcal{S}^e$) resources, respectively. $C7$ indicates whether the requested DNN models can be stored in the corresponding nodes or not. By solving the formulated problem, the pre-trained DNNs can be flexibly placed and selected based on the delay requirements and available multi-dimensional resources, thereby effectively supporting DNN inference.

IV. DRL-Based Resource Management Scheme

According to the Eq. (11), the formulated TADP problem is a mix-integer programming problem, which is difficult to be solved by traditional optimization methods. Moreover, due to the strict delay requirements of inference tasks, the problem should be solved quickly. To this end, we propose a DRL-based scheme to achieve efficient resource management, thereby obtaining the solution rapidly. In this section, we first transform the formulated problem into an MDP, and then propose a learning-based algorithm to solve the formulated optimization problem efficiently.

A. Problem Transformation

Considering the mix-integer variables of the formulated optimization problem and the strict delay requirements of inference tasks, instead of using traditional optimization methods, we present a learning-based algorithm for efficient resource management to achieve DNN inference in IIoT networks [35]. To rapidly obtain the optimal decisions under stochastic environment, we first transform the formulated problem into an MDP, and then adopt a DRL-based approach to solve it [36]. The framework of DRL is composed of the agent and environment. As shown in Fig. 2, the agent is coordinated by the edge controller installed at the ECoS, and everything beyond that is deemed as the environment. The key components of the DRL method, i.e., the observed state, action, and reward, are summarized as follows:

1) State: At each time slot, IIoT devices and network nodes periodically send the task information $\{D_{i,k}, T_{i,k}^{max}, \theta_{i,k}\}$ and caching pool state information $\{u_{i,k}^{l_0}, u_{i,k}^{l_1}, u_{i,k}^{e_0}, u_{i,k}^{e_1}\}$ to the ECoS. Thus, the agent can obtain the state space $\mathcal{S}$ of the considered IIoT system. Then, the state vector at time slot t, $s(t) \in \mathcal{S}$, can be defined as

$$s(t) = \{D_{i,k}(t), T_{i,k}^{max}(t), \theta_{i,k}(t), u_{i,k}^{l_0}(t), u_{i,k}^{l_1}(t), u_{i,k}^{e_0}(t), u_{i,k}^{e_1}(t)\}, \forall i \in \mathcal{G}, k \in \mathcal{K}_i. \tag{12}$$

2) Action: According to the observed environment states, the agent makes decisions based on the policy π. The decisions, including the task assignment $m_{i,k}$, multi-levels DNN selection $\varphi_{i,k}$, resource allocation ratios of spectrum $\{\alpha_{i,k}^l, \alpha_{i,k}^e\}$ and computing $\{\beta_{i,k}^l, \beta_{i,k}^e\}$, and DNN model placement $\{\omega_{i,k}^{l_0}, \omega_{i,k}^{l_1}, \omega_{i,k}^{e_0}, \omega_{i,k}^{e_1}\}$, made by the agent at time slot t,

$a(t) \in \mathcal{A}$, can be described as

$$a(t) = \{m_{i,k}(t), \varphi_{i,k}(t), \alpha^{l}_{i,k}(t), \alpha^{e}_{i,k}(t), \beta^{l}_{i,k}(t), \beta^{e}_{i,k}(t), \omega^{l_0}_{i,k}(t), \omega^{l_1}_{i,k}(t), \omega^{e_0}_{i,k}(t), \omega^{e_1}_{i,k}(t)\}, \forall i \in \mathcal{G}, k \in \mathcal{K}_i, \tag{13}$$

where $a(t) \in \mathcal{A}$ is with $N \times 10$ dimensions.

3) Reward: Once action $a(t)$ is taken by the environment based on the observed state $s(t)$, the agent then will obtain a reward. In the training stage, the policy π is iterated according to the obtained reward until achieving the convergence. To maximize the accuracy of DNN inference, we therefore define the reward for inference tasks at time slot t, i.e.,

$$r(t) = \begin{cases} \frac{1}{N}\sum_{i=1}^{G}\sum_{k=1}^{|\mathcal{K}_i|} U_{i,k}(t), & \text{if } C1 - C7 \text{ are satisfied,} \\ 0, & \text{otherwise,} \end{cases} \tag{14}$$

where $U_{i,k}(t)$ is the accuracy of inference task $W_{i,k}$ in time slot t. The reward function indicates that only the appropriate resources are allocated and the delay requirement are satisfied, simultaneously, the agent then can obtain the system reward. Otherwise, the task cannot be completed, thereby $r(t) = 0$. For notational simplicity, the time index t is abbreviated as the subscript in the following context.

B. Proposed TD3-Based Algorithm

In the considered IIoT networks, it is difficult to model the environment state accurately due to the massive inference requests and the unknown caching pool state transitions. Moreover, the high-dimensional state and action spaces caused by the network scale (i.e., G and N) and multi-dimensional resources, also result in intractable convergence performance [31]. In recent years, actor-critic methods, such as DDPG [37], have obtained considerable achievements in solving optimization problems with large action spaces or continuous decision variables [20]. TD3, as an enhanced version of DDPG, effectively addresses the function approximation error to reduce overestimation bias and avoid suboptimal policy updates, thereby improving the learning speed and performance in continuous control domains [38]. In this paper, we propose a TD3-based algorithm to solve the transformed MDP problem.

For the proposed TD3-based algorithm, the goal of the agent is to find an optimal policy π^* with the maximum average inference accuracy. During the learning stage, the agent randomly samples a minibatch experiences from $\mathcal{D}_r$. Taking the q-th experience $\{s_q, a_q, r_q, s_{q+1}\}$ as an example, the actor makes TADP actions $a = \pi_{\boldsymbol{\theta}}(s_q)$ and $a' = \pi_{\boldsymbol{\theta}'}(s_{q+1})$ via the evaluation and target networks with parameters $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$. Then, the evaluation critics evaluate a and a_q with parameters $\boldsymbol{\omega}_j, j \in \{1,2\}$, and the target critics evaluate a' with parameter $\boldsymbol{\omega}'_j$, to calculate the $Q(s_q, a_q; \boldsymbol{\omega}_j)$ and $Q(s_{q+1}, \pi_{\boldsymbol{\theta}'}(s_{q+1}); \boldsymbol{\omega}'_j)$, respectively. Thus, the target Q-values are given by

$$y_q = r_q + \gamma \min_{j=1,2} Q(s_{q+1}, \pi_{\boldsymbol{\theta}'}(s_{q+1}) + \epsilon; \boldsymbol{\omega}'_j), \tag{15}$$

where $\epsilon \sim clip(\mathcal{N}(0,\rho), -c, c)$ denotes the added noise, c denotes the clip bound of ϵ, and $\gamma \in [0,1]$ denotes a discounting factor. Thus, the loss function of the evaluation critics can be expressed by

$$L(\boldsymbol{\omega}_j) = \frac{1}{2D_b}\sum_{q\in\mathcal{D}_b}\left(y_q - \min_{j=1,2} Q(s_q, a_q; \boldsymbol{\omega}_j)\right)^2, \tag{16}$$

Algorithm 1: TD3-Based Algorithm for Multi-Dimensional Resource Management.

```
1  Initialize actor network $\pi_{\boldsymbol{\theta}}(s)$ and critic networks $Q_{\boldsymbol{\omega}_1}$, $Q_{\boldsymbol{\omega}_2}$
     with parameters $\boldsymbol{\theta}$, $\boldsymbol{\omega}_1$, $\boldsymbol{\omega}_2$;
2  Initialize target networks with weights $\boldsymbol{\theta}'$, $\boldsymbol{\omega}'_1$, $\boldsymbol{\omega}'_2$;
3  Initialize experience replay buffer $\mathcal{D}_r$;
4  for $episode = 1$ to $E$ do
5      Initialize environment, and receive initial state $s_1$;
6      for step $t = 1$ to $T$ do
7          Select action with the current policy $a_t \sim \pi_{\boldsymbol{\theta}}(s_t) + \epsilon, \epsilon \sim \mathcal{N}(0,\rho)$;
8          ▷ DNN caching pool updating
9          Discretize action $m_{i,k}, \varphi_{i,k}, \omega^{l_0}_{i,k}, \omega^{l_1}_{i,k}, \omega^{e_0}_{i,k}, \omega^{e_1}_{i,k}$;
10         for IGWs $i = 1$ to $G$ do
11             for IIoT devices $k = 1$ to $|\mathcal{K}_i|$ do
12                 • Step 1: Compute remaining caching space
13                 if $u^{l_0}_{i,k} = 1$ then
14                     $S^l[i] = S^l[i] + S^n_{i,k}$
15                 • Step 2: Compute transmission delay $\tau_{i,k}$
16                 if $m_{i,k} = 1$, $\varphi_{i,k} = 0$, $u^{l_0}_{i,k} = 0$,
                     $S^l[i] + S^0_{i,k} \le \mathcal{S}^l$, and $\tau_{i,k} \le T^{\max}_{i,k}$ then
17                     $u^{l_0}_{i,k} = 1$, $S^l[i] = S^l[i] + S^0_{i,k}$
18                 else
19                     $u^{l_0}_{i,k} = 0$
20                 Update $u^{l_1}_{i,k}, u^{e_0}_{i,k}, u^{e_1}_{i,k}$;
21                 Execute the inference task;
22                 • Step 3: Compute completion delay $d_{i,k}$
23                 if $m_{i,k} = 1$, $\varphi_{i,k} = 0$, $u^{l_0}_{i,k} = 0$,
                     $S^l[i] + S^0_{i,k} \le \mathcal{S}^l$, and $d_{i,k} \le T^{\max}_{i,k}$ then
24                     $u^{l_0}_{i,k} = 1$, $S^l[i] = S^l[i] + S^0_{i,k}$
25                 else
26                     $u^{l_0}_{i,k} = 0$
27                 Update $u^{l_1}_{i,k}, u^{e_0}_{i,k}, u^{e_1}_{i,k}$;
28                 • Step 4: Perform caching actions
                     $\{\omega^{l_n}_{i,k}, \omega^{e_n}_{i,k}\}$
29                 if $u^{l_0}_{i,k} = 1$ and $\omega^{l_0}_{i,k} = 1$ then
30                     $u^{l_0}_{i,k} = 1$
31                 else
32                     $u^{l_0}_{i,k} = 0$
33                 Update $u^{l_1}_{i,k}, u^{e_0}_{i,k}, u^{e_1}_{i,k}$;
34         ▷ Learning algorithm iterating
35         Obtain the reward $r_t$ and observe next state $s_{t+1}$;
36         Store the experience $\{s_t, a_t, r_t, s_{t+1}\}$ into the $\mathcal{D}_r$,
             and replace the oldest ones if the buffer is full;
37         Randomly sample a minibatch of $D_b$ experiences
             $\{s_q, a_q, r_q, s_{q+1}\}$ from $\mathcal{D}_r$;
38         Set $y_q = r_q + \gamma \min_{j=1,2} Q_{\boldsymbol{\omega}'_j}(s_{q+1}, \pi_{\boldsymbol{\theta}'}(s_{q+1}) + \epsilon)$,
             $\epsilon \sim \text{clip}(\mathcal{N}(0,\rho), -c, c)$;
39         Update $\boldsymbol{\omega}_1, \boldsymbol{\omega}_2$ by minimizing the loss in Eq. (16);
40         if $t$ mod $d$ then
41             Update $\boldsymbol{\theta}$ via the policy gradient in Eq. (19);
42             Update target networks by Eq. (21).
```

where D_b is the size of the sampled minibatch experiences from D_r. Then, parameter $\boldsymbol{\omega}_j$ can be updated by minimizing the loss function, i.e.,

$$\nabla L(\boldsymbol{\omega}_j) = -\frac{1}{D_b}\sum_{q\in\mathcal{D}_b}\left(y_q - \min_{j=1,2} Q(s_q, a_q; \boldsymbol{\omega}_j)\right)\cdot \nabla_{\boldsymbol{\omega}_j} Q(s_q, a_q; \boldsymbol{\omega}_j). \tag{17}$$

Moreover, the objective function of the evaluation actor can be expressed by

$$J(\boldsymbol{\theta}) = \frac{1}{D_b}\sum_{q\in\mathcal{D}_b} Q(s_q, a_q; \boldsymbol{\omega}_1). \tag{18}$$

Although two critics are deployed in the proposed scheme, only one critic, either Critic1 or Critic2, needs to be used to perform policy update. Here, the policy update depends on the value estimations of Critic1. Thus, the parameter $\boldsymbol{\theta}$ of the evaluation actor can be updated via policy gradient theorem, i.e.,

$$\nabla J(\boldsymbol{\theta}) = \frac{1}{D_b}\sum_{q\in\mathcal{D}_b} \nabla_{\pi(s_q)} Q(s_q, \pi(s_q); \boldsymbol{\omega}_1)\nabla_{\boldsymbol{\theta}}\pi(s_q). \tag{19}$$

Hence, the parameters $\boldsymbol{\omega}_j$ and $\boldsymbol{\theta}$ of the evaluation critics and actor can be updated by the gradient descent method, i.e.,

$$\begin{aligned} \boldsymbol{\omega}_j &= \boldsymbol{\omega}_j - \epsilon_c \nabla L(\boldsymbol{\omega}_j), \\ \boldsymbol{\theta} &= \boldsymbol{\theta} - \epsilon_a \nabla J(\boldsymbol{\theta}), \end{aligned} \tag{20}$$

where ϵ_a and ϵ_c denote the learning rate of the actor and critics, respectively. In addition, the soft-updating technique is used for the parameter updating of target networks, i.e.,

$$\begin{aligned} \boldsymbol{\omega}'_j &= \tau\boldsymbol{\omega}_j + (1-\tau)\boldsymbol{\omega}'_j, \\ \boldsymbol{\theta}' &= \tau\boldsymbol{\theta} + (1-\tau)\boldsymbol{\theta}', \end{aligned} \tag{21}$$

where τ denotes the update factor. Notably, the updates of policy and target networks are only occurred after d times updates to the evaluation critics, and thus effectively limits the likelihood of repeating updates with an unchanged critic. The proposed TD3-based algorithm for solving the formulated TADP problem in real time can be summarized in Algorithm 1.

C. Main Techniques of the Proposed TD3-Based Solution

As described in Algorithm 1, a DNN caching pool updating (DCPU) subroutine is proposed to achieve efficient caching resource utilization. In each time slot, once the DNN models requested by inference tasks are sent to the nodes, a certain caching resource needs to be allocated to cache the DNN models in the corresponding caching pools. To improve the flexibility of the system for DNN model placement, we propose a DCPU subroutine to support flexible DNN model placement among end, edge, and cloud, thereby ensure DNN inference can be completed within strict delay requirements. The designed DCPU subroutine consists of the following steps.

Step 1 (Lines 12-14): At the beginning of each time slot, the edge controller first checks the state of DNN caching pools, and allocates caching resources for the DNN models cached in the previous time slot. Thus, the remaining caching spaces of network nodes can be calculated and then can be allocated to the requested DNN models that are transmitted from the ECaS or CCaS.

Step 2 (Lines 15-21): According to the amounts of spectrum and computing resources allocated to the inference tasks, the DNN transmission delay $\tau_{i,k}^{c_n}$, $\tau_{i,k}^{e_n}$, and $\tau_{i,k}^{l_n}$ can be calculated based on the sending location of the requested DNN models. If $\tau_{i,k}^{c_n}$, $\tau_{i,k}^{e_n}$, or $\tau_{i,k}^{l_n}$ is smaller than $T_{i,k}^{\max}$, which means the requested DNNs can be placed to the corresponding caching pools within the delay requirement, thus the DNN models can be cached in this time slot. In such cases, although the inference tasks cannot be guaranteed to be completed within $T_{i,k}^{\max}$, the DNN models requested by current tasks can provide instant inference services for the tasks of following time slots.

Step 3 (Lines 22-27): Similarly, the task completion delay $d_{i,k}$ can be calculated. If $d_{i,k}$ is smaller than the delay requirement of inference task $T_{i,k}^{\max}$, the requested DNN model can be cached in the caching pools to be provided for executing the inference tasks.

Step 4 (Lines 28-33): According to the decision variables $\omega_{i,k}^{l_n}$ and $\omega_{i,k}^{e_n}$, the states of caching pools can be updated, where $\omega_{i,k}^{l_n} = 1$ or $\omega_{i,k}^{e_n} = 1$ indicates that the DNN models requested by inference tasks are continuously cached in the IGWs or ECaS, otherwise indicates that the requested DNN models are deleted from the corresponding nodes.

In addition, as shown in Fig. 3, the proposed learning algorithm is supported by three main techniques:

1) The experience replay technique is adopted for accelerating the convergence efficiency in an off-policy way, in which the experience of the agent at each time slot $e_t = \{s_t, a_t, r_t, s_{t+1}\}$ is stored in a replay buffer $\mathcal{D}_r = \{e_1, e_2, \ldots, e_r\}$ with a certain capacity. When the training process is started, the first D_r experiences are stored in the replay buffer. During the time slots of the learning stage, the agent randomly samples a minibatch of experiences from the replay buffer, thus the correlation among the samples can be effectively reduced.

2) The "double-network" (i.e., evaluation and target networks) mechanism is utilized to stabilize the learning process, and the actor-critic framework is adopted in each network to handle the high-dimensional and continuous action space. Moreover, the clipped double Q-learning for actor-critic is adopted to reduce the overestimation bias. Hence, the agent is equipped with six function approximators, namely, evaluation actor/Critic1/Critic2 and target actor/Critic1/Critic2. In each time slot, the evaluation actor firstly makes resource allocation decisions according to the observed environment state and the policy π. The evaluation critics (i.e., Critic1 and Critic2) evaluate the decisions by generating a pair of Q-values, and the minimum between the two estimates is selected to update the pair of critics. The target network is responsible for calculating a pair of target Q-values and the minimum between the target Q-values is provided to the evaluation network for parameter iterating, such that mitigating overestimation bias and improve the algorithm performance. And then, the target network periodically updates the latest network parameters from the evaluation network. Notably, we deploy multi-layer fully-connected neural networks

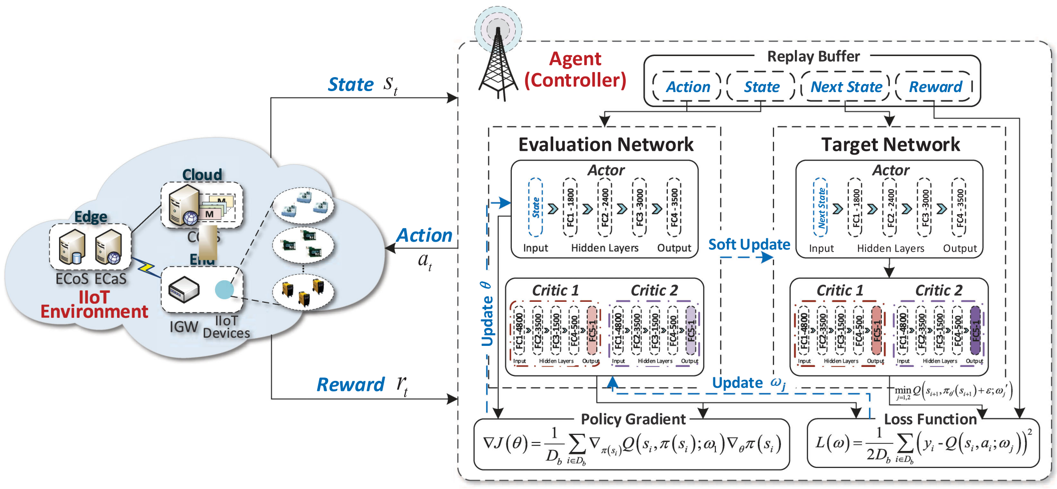

Fig. 3. The framework of the proposed TD3-based algorithm.

(FCNNs) as the function approximators of the proposed TD3-based learning algorithm.

3) The delayed policy update and target policy smoothing regularization are used to reduce per-update error and avoid overfitting in the value estimation. Generally, deep function approximators require massive gradient iterations to converge. However, policy updates with high-error Q-values often result in divergent behavior, thus the evaluation actor should be updated at a lower frequency than the evaluation critics until the value error is neglectable. In the proposed TD3-based algorithm, the policy and target networks can be updated only after d updates to the evaluation critics. Additionally, when updating the critic networks, the target Q-values are very susceptible to inaccuracies caused by function approximation error, thereby increasing the variance of the target Q-values. Thus, a clipped random noise is added to further improve the accuracy of Q-value estimation and the stability of the learning process.

Remark: The TD3-based algorithm is proposed to achieve efficient resource utilization, which is different from existing learning approaches. The difference is two-fold: 1) The proposed TD3-based resource management scheme is endowed with the DCPU subroutine to achieve more flexible task assignment and DNN placement; and 2) The algorithm is enhanced with three important techniques, i.e., dual critics, delayed policy update and target policy smoothing regularization, which can reduce the overestimation bias and stabilize the learning process effectively, as compared to other learning approaches.

V. Performance Evaluation

A. *Experimental Setup*

In this section, we carry out evaluations to demonstrate the performance of the proposed learning-based algorithm for DNN inference in the end-edge-cloud orchestrated IIoT networks. We consider a smart factory with 10 IGWs deployed in this scenario, where each IGW is equipped with a Raspberry Pi 4B [39] and connected with 35 IIoT devices. We deploy an ECoS and an ECaS on the BS, and deploy a CCaS on the cloud. All these nodes are equipped with a certain amount of resources for supporting the inference task assignment and pre-trained DNN model placement, and the ratio of the noisy data is set to 30%. In the simulation, we set the performance metrics of pre-trained DNN models according to [5] and [40], which are listed in Table III. Here, the parameters of each pre-trained DNN model are sampled uniformly from the corresponding value interval, which follows the sampling rules described in Section III-A.

TABLE III
Parameters of Pre-Trained DNN Models

Parameter	Value	Parameter	Value
$\mathcal{P}_{UL}$	[0.915, 0.938]	$\mathcal{P}_{UH}$	[0.973, 0.992]
$\mathcal{P}_{DL}$	[0.876, 0.897]	$\mathcal{P}_{DH}$	[0.929, 0.954]
$S^0_{i,k}$	[3.5, 5.5] MB	$S^1_{i,k}$	[9, 12] MB
$Q^0_{i,k}$	[15, 20] MHz	$Q^1_{i,k}$	[25, 30] MHz

For the proposed TD3-based algorithm, we deploy a four-layer FCNN with [1800, 2400, 3000, 3500] neurons as the actor, and deploy two five-layer FCNNs with [4800, 3500, 1500, 500, 1] neurons as the critics. Here, the neurons in the output layer of the actor are activated by the $tanh$ function, and other layers' neurons of the actor and critics are activated by the $ReLU$ function. All the network parameters are trained using Adam optimizer [41]. In addition, the variance ρ of the added noise is set to 0.001, and the clip bound c is set to 0.03. Other simulation parameters of the network and learning algorithm are listed in Table IV. To show the efficiency of the proposed algorithm, the following benchmarks are utilized for performance comparison.

- *DDPG:* The edge controller allocates the spectrum, computing, and caching resources with the deterministic policy, where we deploy a four-layer FCNN with [1800, 2400, 3000, 3500] neurons as the actor and a four-layer FCNN with [2000, 1500, 500, 1] neurons as the critic to achieve the optimal performance.
- *Random:* In the random policy, the edge controller randomly assigns the inference tasks to IGWs or ECoS, selects an accuracy level of DNN model for each task, and allocates the multi-dimensional resources for the inference tasks of

TABLE IV
SIMULATION PARAMETERS [31], [38]

Network Parameter	Value	Learning Parameter	Value
$p_{i,k}$, $p^e_{i,k}$	30, 40 dBm	*episode*	300
σ^2	-104 dBm	*step*	50
B, B^e	10, 20 MHz	D_r	10,000
$\mathcal{C}^l$, $\mathcal{C}^e$	1.5, 5 GHz	D_b	64
$\mathcal{S}^l$, $\mathcal{S}^e$	0.3, 2.4 GB	γ	0.95
R^e, R^c	200, 300 Mbps	ϵ_a, ϵ_c	1e-4, 3e-4
$D_{i,k}$	[40, 60] KB	τ	0.01
$T^{max}_{i,k}$	[1.5, 2] s	d	2

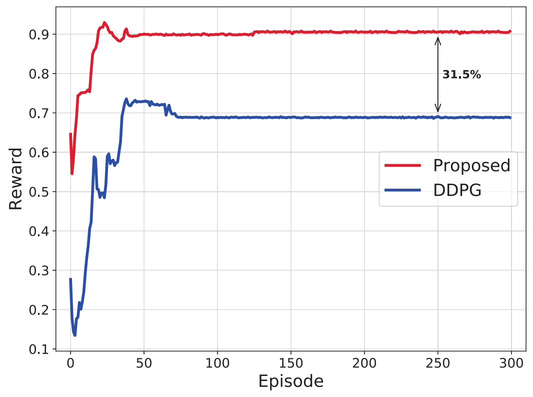

Fig. 4. The convergence performance of the proposed algorithm.

IIoT devices. All available actions are selected with an equal probability.

B. Evaluation of TD3-Based Algorithm

1) Convergence of the Proposed Algorithm: As shown in Fig. 4, the convergence performance of the proposed TD3-based algorithm is presented with respect to training episodes. It can be seen from the red curve that the reward fluctuates in the first fifty episodes and tends to reach a stable level after parameter iterations for 130 episodes. Additionally, compared with DDPG-based scheme, the proposed algorithm can not only achieve a faster convergence speed, but also obtain a higher converged value which is larger than that of the DDPG benchmark by 31.5%. This is because three improvements of the TD3 algorithm mentioned in Section IV-B effectively mitigate possible overestimation, thereby achieving a faster learning speed and better performance than the DDPG benchmark.

2) Impact of Spectrum, Caching, and Computing Resources at the IGWs: We use the following evaluation metrics to measure the resource management performance of the proposed algorithm: 1) average inference accuracy; and 2) task completion ratio which is defined as the number of inference tasks completed within the delay requirements over the total number of tasks.

As shown in Fig. 5, the achievable average inference accuracy and task completion ratio are presented with respect to the spectrum, caching, and computing resources at the IGWs, respectively. As can be seen from Fig. 5(a-c), with the increasing of the available spectrum/caching/computing resources of IGWs, the average inference accuracy and task completion ratio achieved by the proposed TD3-based algorithm and both benchmarks increase. This is because more spectrum resources can be allocated to the IIoT devices to offload the tasks, more caching capacities can be provided for pre-trained DNN model placement, and more computing resources can be used to execute the inference tasks at IGWs. Particularly, the proposed algorithm can improve the average inference accuracy by 31.4% and 74.5%, respectively, as compared to the DDPG and Random schemes. In addition, the proposed algorithm achieves 1.31$\times$ and 1.71$\times$ task completion ratio improvement compared with DDPG and Random schemes, respectively. When the IGWs are equipped with sufficient spectrum, caching, and computing resources, inference tasks can be offloaded to the IGWs with a low latency, more pre-trained DNN models can be placed at the IGWs from ECaS or CCaS, and adequate computing resources can be utilized to accelerate DNN inference. Thus, urgent inference tasks can be executed at the end side within the strict delay requirements.

3) Impact of Spectrum, Caching, and Computing Resources at the BS: As shown in Fig. 6, we compare the achievable average inference accuracy with benchmark schemes in terms of the spectrum, caching, and computing resources of BS, respectively. Similar to the IGWs, with the increasing of the available spectrum/caching/computing resources, the average inference accuracy achieved by the TD3-based algorithm and the compared schemes also increases. The reason is that more resources can be utilized to satisfy the requirements of inference tasks that are assigned to the BS. In particular, the proposed algorithm can achieve a converged average inference accuracy faster than the benchmark schemes. Specifically, when the amount of spectrum resource is 6 MHz, the proposed algorithm achieves 1.51$\times$ and 2.20$\times$ average inference accuracy improvement compared with DDPG and Random schemes, respectively. When the caching capacity is 0.6 GB, the proposed algorithm achieves 1.48$\times$ and 1.95$\times$ average inference accuracy improvement compared with the benchmarks, respectively. When the computing capability is 2 GHz, the proposed algorithm achieves 1.60$\times$ and 2.14$\times$ average inference accuracy improvement compared with the benchmarks, respectively. The above observations indicate that the proposed algorithm can assign inference tasks and place DNN models among network nodes more reasonably, and can utilize multi-dimensional resources more efficiently, thus faster converged average inference accuracy can be reached with a low resource demand.

4) Impact of Transmission Rates of R^c and R^e: As shown in Fig. 7, the distribution of inference task assignment and pre-trained DNN placement is presented in terms of the transmission rates from cloud to the BS (i.e., R^c) and from BS to the IGWs (i.e., R^e). For the upper part of Fig. 7(a) and Fig. 7(b), the number of DNN models which are placed at the IGWs or ECaS is presented in terms of the R^c and R^e. For the below part of Fig. 7(a) and Fig. 7(b), the number of completed tasks which are assigned to the IGWs or ECoS is presented in terms of the R^c and R^e.

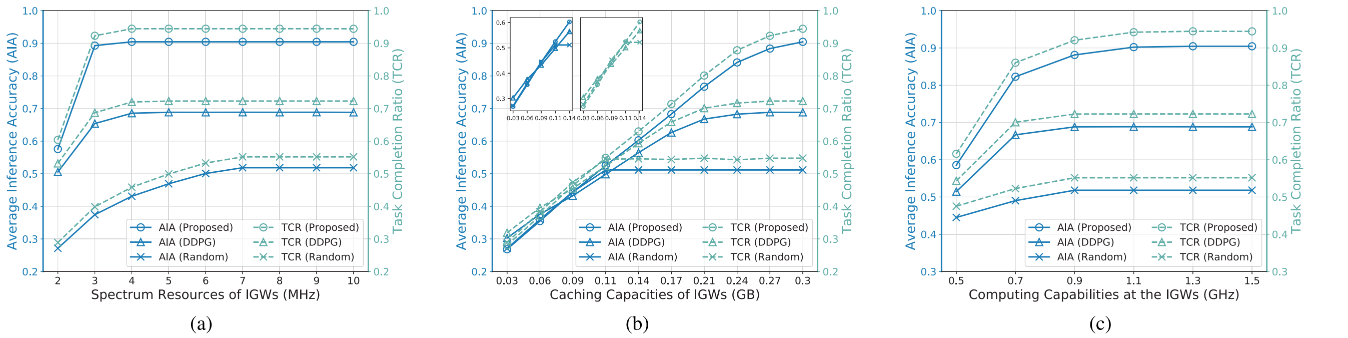

Fig. 5. Average inference accuracy and task completion ratio over inference tasks assigned to the IGWs. (a) Impact of the spectrum resources of IGWs. (b) Impact of the caching capacities of IGWs. (c) Impact of the computing capabilities of IGWs.

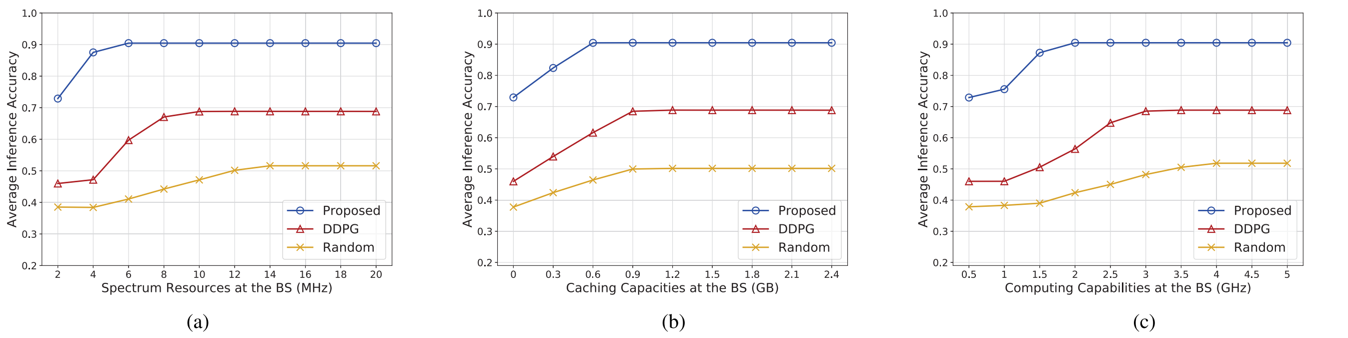

Fig. 6. Average inference accuracy over inference tasks assigned to the BS (i.e., ECaS and ECoS). (a) Impact of the spectrum resources of BS. (b) Impact of the caching capacities of BS. (c) Impact of the computing capabilities of BS.

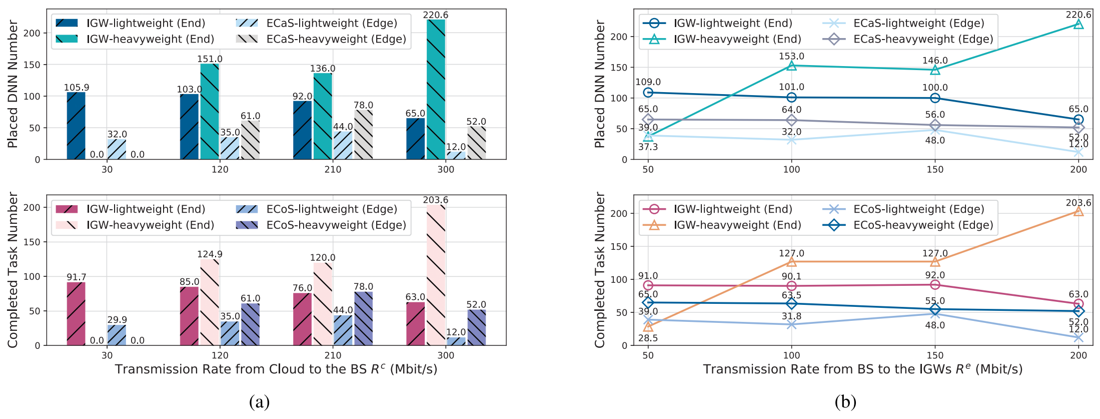

Fig. 7. Numbers of completed inference tasks and placed DNN models at the IGWs and BS with respect to the transmission rates of R^c and R^e. (a) The impact of the transmission rate R^c. (b) The impact of the transmission rate R^e.

Several observations can be captured from Fig. 7. First, with the increasing of R^c and R^e, more pre-trained DNN models can be placed at the IGWs and ECaS, and more inference tasks are executed within the strict delay requirements, thus the task completion number can be improved. Second, if the network is equipped with larger transmission rate R^c, the edge controller is inclined to select heavy-weight DNN models for maximizing the average inference accuracy due to the accuracy-optimal objective. It is interesting to note that, when the R^e increases to 150 Mbit/s, the edge controller tends to execute more light-weight DNNs at the ECoS to improve the average inference accuracy. Third, if R^c and R^e are large enough, the edge controller can place the pre-trained DNN models to the IGWs with low latency, thus more inference tasks can be executed at the

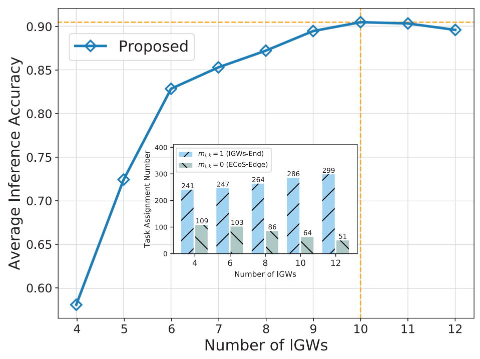

Fig. 8. Average inference accuracy of the proposed TD3-based algorithm with respect to the number of IGWs.

end side, and more heavy-weight DNN models can be selected to maximize the average inference accuracy. For example, when the R^c and R^e are set to 300 and 200 Mbit/s, the edge controller assigns 267 inference tasks to the IGWs, which is larger $4.17\times$ than the assignment number of ECoS. Meanwhile, the edge controller selects 256 heavy-weight DNN models which is larger $3.41\times$ than the selection number of light-weight DNN models. In particular, the utilization of the placed DNN models of IGWs achieves 93.3% for $R^c = 300$ Mbit/s and $R^e = 200$ Mbit/s. The result indicates that the proposed algorithm achieves a high utilization of the placed DNN models, which effectively supports the real-time DNN inference at the end devices or the edge nodes.

5) Impact of the Number of IGWs: As shown in Fig. 8, the impact of the number of IGWs on the achievable average inference accuracy of the proposed TD3-based algorithm is presented, where the number of IIoT devices connected to each IGW is the same. It can be seen that the average inference accuracy increases with the number of IGWs when it is less than 10, which is because more resources can be allocated to the inference tasks. In addition, the average inference accuracy improvement introduced by each IGW diminishes with the increasing number of IGWs, and a decline trend appears after the number of IGWs exceeds 10. This is because more caching and computing resources are deployed at the end side with the increasing number of IGWs. In such a case, the edge controller is inclined to make decisions to assign inference tasks to the IGWs. The embedded figure shows the distribution of task assignment decisions with respect to the number of IGWs. However, to perform DNN inference at the end side, more pre-trained DNN models need to be placed to the IGWs from the CCaS or ECaS, thus urgent inference tasks cannot be completed within the strict delay requirements due to the large transmission delay of DNN models.

VI. Conclusion

In this work, we have studied the joint TADP problem for DNN inference in the end-edge-cloud orchestrated IIoT networks with multi-dimensional resource constraints. A learning-based algorithm has been proposed to make resource allocation decisions in real time, such that providing DNN inference services for intelligent IIoT applications. In resource-limited IIoT networks, the proposed algorithm can learn the optimal DNN model placement scheme, and select appropriate DNN models for inference tasks via efficient resource utilization, thereby effectively improving the average accuracy of inference tasks while satisfying the strict delay requirements. For the future work, we will investigate the distributed DNN training problem in IIoT networks.

Acknowledgment

The authors would like to thank Huaqing Wu and Mengjie Liu for valuable comments and suggestions on the paper.

References

[1] W. Zhang *et al.*, "Deep reinforcement learning based resource management for DNN inference in IIoT," in *Proc. IEEE Glob. Commun. Conf.*, Taipei, Taiwan, 2020, pp. 1–6.
[2] N. Kato, B. Mao, F. Tang, Y. Kawamoto, and J. Liu, "Ten challenges in advancing machine learning technologies toward 6 G," *IEEE Wireless Commun.*, vol. 27, no. 3, pp. 96–103, Jun. 2020.
[3] K. Zhang, Y. Zhu, S. Maharjan, and Y. Zhang, "Edge intelligence and blockchain empowered 5 G beyond for the industrial internet of things," *IEEE Netw.*, vol. 33, no. 5, pp. 12–19, Oct. 2019.
[4] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial internet of things: Challenges, opportunities, and directions," *IEEE Trans. Ind. Informat.*, vol. 14, no. 11, pp. 4724–4734, Nov. 2018.
[5] W. Zhang, D. Yang, and H. Wang, "Data-driven methods for predictive maintenance of industrial equipment: A survey," *IEEE Syst. J.*, vol. 13, no. 3, pp. 2213–2227, Sep. 2019.
[6] W. Zhang, D. Yang, Y. Xu, X. Huang, J. Zhang, and M. Gidlund, "Deep-Health: A self-attention based method for instant intelligent predictive maintenance in industrial internet of things," *IEEE Trans. Ind. Inform.*, vol. 17, no. 8, pp. 5461–5473, 2020, doi: 10.1109/TII.2020.3029551.
[7] C. Xu, K. Ma, Y. Xu, Y. Xu, and Y. Fang, "Optimal power scheduling for uplink transmissions in SIC-based industrial wireless networks with guaranteed real-time performance," *IEEE Trans. Veh. Technol.*, vol. 67, no. 4, pp. 3216–3228, Apr. 2018.
[8] V. Sze, Y. Chen, T. Yang, and J. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017.
[9] E. Li, L. Zeng, Z. Zhou, and X. Chen, "Edge AI: On-demand accelerating deep neural network inference via edge computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 447–457, Jan. 2020.
[10] Y. Li, Z. Han, Q. Zhang, Z. Li, and H. Tan, "Automating cloud deployment for deep learning inference of real-time online services," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Las Vegas, USA, 2020, pp. 1668–1677.
[11] N. Zhang, R. Wu, S. Yuan, C. Yuan, and D. Chen, "RAV: Relay aided vectorized secure transmission in physical layer security for internet of things under active attacks," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8496–8506, Oct. 2019.
[12] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5 G network edge cloud architecture and orchestration," *IEEE Commun. Surv. Tut.*, vol. 19, no. 3, pp. 1657–1681, May 2017.
[13] J. Wang, L. Zhao, J. Liu, and N. Kato, "Smart resource allocation for mobile edge computing: A deep reinforcement learning approach," *IEEE Trans. Emerg. Topics Comput.*, to be published, doi: 10.1109/TETC.2019.2902661.
[14] W. Wu *et al.*, "Dynamic RAN slicing for service-oriented vehicular networks via constrained learning," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 2076–2089, Jul. 2021.
[15] Y. Zhou, Z. Fadlullah, B. Mao, and N. Kato, "A deep-learning-based radio resource assignment technique for 5 G ultra dense networks," *IEEE Netw.*, vol. 32, no. 6, pp. 28–34, Dec. 2018.
[16] Y. He, N. Zhao, and H. Yin, "Integrated networking, caching, and computing for connected vehicles: A deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 67, no. 1, pp. 44–55, Jan. 2018.
[17] W. Wu, N. Cheng, N. Zhang, P. Yang, W. Zhuang, and X. Shen, "Fast mmwave beam alignment via correlated bandit learning," *IEEE Trans. Wireless Commun.*, vol. 18, no. 12, pp. 5894–5908, Dec. 2019.

[18] N. Luong *et al.*, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Commun. Surv. Tut.*, vol. 21, no. 4, pp. 3133–3174, May 2019.

[19] K. Li, W. Ni, E. Tovar, and A. Jamalipour, "Deep q-learning based resource management in UAV-assisted wireless powered IoT networks," in *Proc. IEEE Int. Conf. Commun.*, Dublin, Ireland, 2020, pp. 1–6.

[20] N. Cheng *et al.*, "Space/aerial-assisted computing offloading for IoT applications: A learning-based approach," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1117–1129, May 2019.

[21] W. Zhuang, Q. Ye, F. Lyu, N. Cheng, and J. Ren, "SDN/NFV-empowered future IoV with enhanced communication, computing, and caching," *Proc. IEEE*, vol. 108, no. 2, pp. 274–291, Feb. 2020.

[22] Y. You, Z. Zhang, C. Hsieh, J. Demmel, and K. Keutzer, "Fast deep neural network training on distributed systems and cloud TPUs," *IEEE Trans. Parallel Distrib. Sys.*, vol. 30, no. 11, pp. 2449–2462, Nov. 2019.

[23] A. Eshratifar, M. Abrishami, and M. Pedram, "JointDNN: An efficient training and inference engine for intelligent mobile cloud computing services," *IEEE Trans. Mobile Comput.*, vol. 20, no. 2, pp. 565–576, Feb. 2021.

[24] Y. Huang, F. Wang, F. Wang, and J. Liu, "Deepar: A hybrid device-edge-cloud execution framework for mobile deep learning applications," in *Proc. IEEE Conf. Comput. Commun.*, Paris, France, 2019, pp. 892–897.

[25] W. Sun, J. Liu, and Y. Yue, "AI-enhanced offloading in edge computing: When machine learning meets industrial IoT," *IEEE Netw.*, vol. 33, no. 5, pp. 68–74, Oct. 2019.

[26] B. Yang, X. Cao, X. Li, Q. Zhang, and L. Qian, "Mobile edge computing based hierarchical machine learning tasks distribution for IIoT," *IEEE Internet Things J.*, vol. 7, no. 3, pp. 2169–2180, Mar. 2020.

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

[28] N. Zhao, Y. Liang, D. Niyato, Y. Pei, M. Wu, and Y. Jiang, "Deep reinforcement learning for user association and resource allocation in heterogeneous cellular networks," *IEEE Trans. Wireless Commun.*, vol. 18, no. 11, pp. 5141–5152, Nov. 2019.

[29] J. Chen, Z. Wei, S. Li, and B. Cao, "Artificial intelligence aided joint bit rate selection and radio resource allocation for adaptive video streaming over F-RANs," *IEEE Wireless Commun.*, vol. 27, no. 2, pp. 36–43, Apr. 2020.

[30] Z. Zhang, H. Chen, M. Hua, C. Li, Y. Huang, and L. Yang, "Double coded caching in ultra dense networks: Caching and multicast scheduling via deep reinforcement learning," *IEEE Trans. Commun.*, vol. 68, no. 2, pp. 1071–1086, Feb. 2019.

[31] H. Peng and X. Shen, "Deep reinforcement learning based resource management for multi-access edge computing in vehicular networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 4, pp. 2416–2428, Oct. 2020.

[32] W. Wu, N. Zhang, N. Cheng, Y. Tang, K. Aldubaikhy, and X. Shen, "Beef up mmWave dense cellular networks with D2D-assisted cooperative edge caching," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3890–3904, Apr. 2019.

[33] W. Quan, N. Cheng, M. Qin, H. Zhang, H. Chan, and X. Shen, "Adaptive transmission control for software defined vehicular networks," *IEEE Wireless Commun. Lett.*, vol. 8, no. 3, pp. 653–656, Jun. 2018.

[34] D. Yang *et al.*, "Assignment of segmented slots enabling reliable real-time transmission in industrial wireless sensor networks," *IEEE Trans. Ind. Electron.*, vol. 62, no. 6, pp. 3966–3977, Jun. 2015.

[35] X. Shen *et al.*, "AI-assisted network-slicing based next-generation wireless networks," *IEEE Open J. Veh. Technol.*, vol. 1, no. 1, pp. 45–66, Jan. 2020.

[36] W. Wu, P. Yang, W. Zhang, C. Zhou, and X. Shen, "Accuracy-guaranteed collaborative DNN inference in industrial IoT via deep reinforcement learning," *IEEE Trans. Ind. Inform.*, vol. 17, no. 7, pp. 4988–4998, Jul. 2021.

[37] T. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," in *Proc. Int. Conf. Learn. Representations*, San Juan, Puerto Rico, 2016.

[38] S. Fujimoto, H. Van, and D. Meger, "Addressing function approximation error in actor-critic methods," *Proc. Int. Conf. Mach. Learn.*, Stockholm, Sweden, pp. 1587–1596, 2018.

[39] *Raspberry Pi 4 Model B*. [Online]. Available: https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2711/rpi_DATA_2711_1p0_preliminary.pdf, Accessed on: Jun. 21, 2019.

[40] J. Luo, J. Wu, and W. Lin, "Thinet: A filter level pruning method for deep neural network compression," in *Proc. IEEE Int. Conf. Comput. Vis.*, Venice, Italy, 2017, pp. 5058–5066.

[41] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Int. Conf. Learn. Represent.*, San Diego, CA, pp, 1–15, 2015.

Weiting Zhang (Student Member, IEEE) is currently working toward the Ph.D. degree in communication and information systems with the Beijing Jiaotong University, Beijing, China. From November 2019 to November 2020, he was a Visiting Ph.D. Student with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. His research interests include mobile edge computing, industrial Internet of Things, and machine learning for wireless networks.

Dong Yang (Member, IEEE) received the B.S. degree from Central South University, Hunan, China, in 2003 and the Ph.D. degree in communications and information science from Beijing Jiaotong University, Beijing, China, 2009. From March 2009 to June 2010, he was a Postdoctoral Research Associate with Jonkoping University, Jonkoping, Sweden. In August 2010, he joined the School of Electronic and Information Engineering, Beijing Jiaotong University. Since 2017, he has been a Full Professor of communication engineering with Beijing Jiaotong University. His research interests include network architecture, wireless sensor networks, industrial network, and Internet of Things.

Haixia Peng (Student Member, IEEE) received the M.S. and Ph.D. degrees in electronics and communication engineering and computer science from Northeastern University, Shenyang, China, in 2013 and 2017, respectively. She is currently a Ph.D. Student with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. She has authored or coauthored more than 30 technical papers dealing with network issues. Her current research interests include Internet of vehicles, resource management, multiaccess edge computing, and reinforcement learning. She is or was a Reviewer of the more than 20 prestigious journals, such as the IEEE JOURNALS ON SELECTED AREAS IN COMMUNICATIONS, IEEE TRANSACTIONS ON COMMUNICATIONS, and IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGIES and as a TPC Member in IEEE ICC, Globecom, and VTC conferences.

Wen Wu (Member, IEEE) received the B.E. degree in information engineering from the South China University of Technology, Guangzhou, China, in 2012, the M.E. degree in electrical engineering from the University of Science and Technology of China, Hefei, China, in 2015, and the Ph.D. degree in vv electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2019. Since 2019, he has been a Postdoctoral Fellow with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. His research interests include millimeter-wave networks and AI-empowered wireless networks.

Wei Quan (Member, IEEE) received the Ph.D. degree in communication and information system from the Beijing University of Posts and Telecommunications, Beijing, China, in 2014. He is currently an Associate Professor with the School of Electronic and Information Engineering, BJTU. He has authored or coauthored more than 20 papers in prestigious international journals and conferences, including the *IEEE Communications Magazine*, IEEE WIRELESS COMMUNICATIONS, IEEE NETWORK, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE COMMUNICATIONS LETTERS, IFIP Networking, IEEE ICC, and IEEE GLOBECOM. His research interests include key technologies for network analytics, future Internet, 5G networks, and vehicular networks. He is a TPC Member of IEEE ICC in 2017 and 2018, IEEE INFOCOM (NewIP Workshop) in 2020, ACM MOBIMEDIA in 2015, 2016, and 2017, and IEEE CCIS in 2015 and 2016. He is also a Member of ACM and a Senior Member of the Chinese Association of Artificial Intelligence. He is an Associate Editor for the *Journal of Internet Technology*, *Peer-to-Peer Networking and Applications*, and IEEE ACCESS, and as a Technical Reviewer for many important international journals.

Hongke Zhang (Fellow, IEEE) received the M.S. and Ph.D. degrees in electrical and communication systems from the University of Electronic Science and Technology of China, Chengdu, China, in 1988 and 1992, respectively. From 1992 to 1994, he was a Postdoctoral Researcher with Beijing Jiaotong University, Beijing, China, where he is currently a Professor with the School of Electronic and Information Engineering and the Director of the National Engineering Lab on Next Generation Internet Technologies. He is the author of more than ten books and the holder of more than 70 patents. His research has resulted in many papers, books, patents, systems, and equipment in the areas of communications and computer networks. He is the Chief Scientist of the National Basic Research Program of China (973 Program) and was also on the Editorial Boards of various international journals.

Xuemin (Sherman) Shen (Fellow, IEEE) received the Ph.D. degree in electrical engineering from Rutgers University, New Brunswick, NJ, USA, in 1990. He is currently a University Professor with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. His research interests include network resource management, wireless network security, Internet of Things, 5G and beyond, and vehicular ad hoc and sensor networks. He is a registered Professional Engineer of Ontario, Canada, an Engineering Institute of Canada Fellow, a Canadian Academy of Engineering Fellow, a Royal Society of Canada Fellow, a Chinese Academy of Engineering Foreign Member, and a Distinguished Lecturer of the IEEE Vehicular Technology Society and Communications Society.

He was the recipient of the R.A. Fessenden Award in 2019 from IEEE, Canada, Award of Merit from the Federation of Chinese Canadian Professionals (Ontario) in 2019, the James Evans Avant Garde Award in 2018 from the IEEE Vehicular Technology Society, the Joseph LoCicero Award in 2015 and Education Award in 2017 from the IEEE Communications Society, and the Technical Recognition Award from Wireless Communications Technical Committee (2019) and AHSN Technical Committee (2013). He was also the recipient of the Excellent Graduate Supervision Award in 2006 from the University of Waterloo and the Premier's Research Excellence Award in 2003 from the Province of Ontario, Canada. He was the Technical Program Committee Chair/Co-Chair for IEEE Globecom'16, IEEE Infocom'14, IEEE VTC'10 Fall, IEEE Globecom'07, and the Chair for the IEEE Communications Society Technical Committee on Wireless Communications. He is the President Elect of the IEEE Communications Society. He was the Vice President for Technical $\&amp$; Educational Activities, Vice President for Publications, Member-at-Large on the Board of Governors, Chair of the Distinguished Lecturer Selection Committee, a Member of IEEE Fellow Selection Committee of the ComSoc. He was the Editor-in-Chief of the IEEE INTERNET OF THINGS JOURNAL, IEEE NETWORK, and *IET Communications*.