Accuracy-Guaranteed Collaborative DNN Inference in Industrial IoT via Deep Reinforcement Learning

Wen Wu, *Member, IEEE*, Peng Yang, *Member, IEEE*, Weiting Zhang, *Student Member, IEEE*, Conghao Zhou, *Student Member, IEEE*, and Xuemin Shen, *Fellow, IEEE*

***Abstract*—Collaboration among industrial Internet of Things (IoT) devices and edge networks is essential to support computation-intensive deep neural network (DNN) inference services, which require low delay and high accuracy. Sampling rate adaption, which dynamically configures the sampling rates of industrial IoT devices according to network conditions, is the key in minimizing the service delay. In this article, we investigate the collaborative DNN inference problem in industrial IoT networks. To capture the channel variation and task arrival randomness, we formulate the problem as a constrained Markov decision process (CMDP). Specifically, sampling rate adaption, inference task offloading, and edge computing resource allocation are jointly considered to minimize the average service delay while guaranteeing the long-term accuracy requirements of different inference services. Since CMDP cannot be directly solved by general reinforcement learning (RL) algorithms due to the intractable long-term constraints, we first transform the CMDP into an MDP by leveraging the Lyapunov optimization technique. Then, a deep RL-based algorithm is proposed to solve the MDP. To expedite the training process, an optimization subroutine is embedded in the proposed algorithm to directly obtain the optimal edge computing resource allocation. Extensive simulation results are provided to demonstrate that the proposed RL-based algorithm can significantly reduce the average service delay while preserving long-term inference accuracy with a high probability.**

***Index Terms*—Collaborative deep neural network (DNN) inference, deep reinforcement learning (RL), inference accuracy, sampling rate adaption.**

Manuscript received May 10, 2020; revised July 23, 2020; accepted August 10, 2020. Date of publication August 18, 2020; date of current version April 2, 2021. This work was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada. Paper no. TII-20-2400. *(Corresponding author: Peng Yang.)*

Wen Wu, Conghao Zhou, and Xuemin Shen are with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo ON N2L 3G1, Canada (e-mail: 17111018@bjtu.edu.cn; c89zhou@uwaterloo.ca; sshen@uwaterloo.ca).

Peng Yang is with the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: yangpeng@hust.edu.cn).

Weiting Zhang is with the School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing 100044, China (e-mail: w77wu@uwaterloo.ca).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2020.3017573

I. Introduction

With the development of advanced neural network techniques and ubiquitous industrial Internet of Things (IoT) devices, deep neural network (DNN) is widely applied in extensive industrial IoT applications, such as facility monitoring and fault diagnosis [1]. Industrial IoT devices (e.g., vibration sensors) can sense the industrial operating environment and feed sensing data to a DNN, and then, the DNN processes the sensing data and renders inference results, namely DNN inference. Although DNN inference can achieve high inference accuracy as compared to traditional alternatives (e.g., decision tree), executing DNN inference tasks requires extensive computation resource due to tremendous multiply-and-accumulation operations [2]. A device-only solution that purely executes DNN inference tasks at resource-constrained industrial IoT devices, becomes intractable due to prohibitive energy consumption and a high service delay. For example, processing an image using AlexNet incurs up to 0.45-W energy consumption [3]. An edge-only solution, which purely offloads large-volume sensing data to resource-rich edge nodes, e.g., access point (AP), suffers from an unpredictable service delay due to time-varying wireless channel [4]. Hence, neither a device-only nor an edge-only solution can effectively support low-delay DNN inference services.

Collaborative inference, which coordinates resource-constrained industrial IoT devices and the resource-rich AP, becomes a *de-facto* paradigm to provide low-delay and high-accuracy inference services [5]. Within the collaborative inference, sensing data from industrial IoT devices can be either processed locally or offloaded to the AP. At industrial IoT devices, light-weight compressed DNNs (i.e., neural networks are compressed without significantly decreasing their performance) are deployed due to constrained on-board computing capability, which saves computing resource at the cost of inference accuracy [6], [7]. At the AP, uncompressed DNNs are deployed to provide high-accuracy inference services at the cost of network resources. Through the resource allocation (e.g., task offloading) between industrial IoT devices and the AP, the overall service performance can be enhanced.

However, the sampling rate adaption technique that dynamically configures the sampling rates of industrial IoT devices is seldom considered. Through dynamically adjusting the sampling rates according to channel conditions and AP's workload,

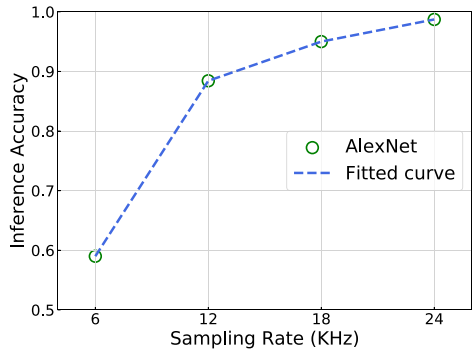


Fig. 1. Inference accuracy with respect to sampling rates on the bearing vibration dataset [8].

sensing data from industrial IoT devices can be compressed, thereby reducing not only the offloaded data volume, but also task computation workload. In our experiments, we implement AlexNet to conduct bearing fault diagnosis based on the collected bearing vibration signal from dataset [8].[1] As shown in Fig. 1, inference accuracy grows sublinearly with the sampling rate. For example, when the sampling rate increases from 18 to 24 kHz, the accuracy increases from 95% to 98.7%. Hence, when the channel condition is poor or edge computation workload is heavy, decreasing the sampling rate can reduce the offloaded data volume and requested computation workload, thereby reducing the service delay at the cost of limited inference accuracy. When channel condition is good and edge computation workload is light, increasing the sampling rate can help deliver a high-accuracy service with an acceptable service delay. Hence, sampling rate adaption can effectively reduce the service delay, which should be incorporated in the collaborative DNN inference.

The sampling rate adaption and resource allocation for collaborative DNN inference are entangled with the following challenges. First, due to time-varying channel conditions and random task arrivals, sampling rate and resource allocation should be dynamically adjusted to achieve the minimum service delay. Minimizing the long-term service delay requires the stochastic information of network dynamics. Second, in addition to minimizing the service delay, the long-term accuracy requirements should be guaranteed for different inference services. The long-term accuracy performance is determined by decisions of sampling rate adaption and resource allocation over time, and hence, the optimal decisions require future network information. To address the aforementioned two challenges, a reinforcement learning (RL) technique is leveraged to interact with the unknown environment to capture the network dynamics, and then, a Lyapunov optimization technique is utilized within the RL framework to guarantee the long-term accuracy requirements without requiring future network information.

In this article, we investigate the collaborative DNN inference problem in industrial IoT networks. First, we formulate the problem as a constrained Markov decision process (CMDP) to account for time-varying channel conditions and random task arrivals. Specifically, sampling rates of industrial IoT devices, task offloading, and edge computation resource allocation are optimized to minimize the average service delay while guaranteeing the long-term accuracy requirements of multiple services. Second, since traditional RL algorithms target at optimizing a long-term reward without considering policy constraints, they cannot be applied to solve CMDP with long-term constraints. To solve the problem, we transform the CDMP into an MDP via the Lyapunov optimization technique. The core idea is to construct accuracy deficit queues to characterize the satisfaction status of the long-term accuracy constraints, thereby guiding the learning agent to meet the long-term accuracy constraints. Third, to solve the MDP, a learning-based algorithm is developed based on the deep deterministic policy gradient (DDPG) algorithm. Within the learning algorithm, to reduce the training complexity, edge computing resource allocation is directly solved via an optimization subroutine based on convex optimization theory, since it only impacts one-shot delay performance according to theoretical analysis. Extensive simulations are conducted to validate the effectiveness of the proposed algorithm in reducing the average service delay while preserving the long-term accuracy requirements.

Our main contributions in this article are summarized as follows.

1) We formulate the collaborative DNN inference problem as a CMDP, in which the objective is to minimize the average service delay while guaranteeing the long-term accuracy constraints.
2) We transform the CMDP into an MDP via the Lyapunov optimization technique, constructs the accuracy deficit queue to characterize the satisfaction status of the long-term accuracy constraints.
3) We propose a deep RL-based algorithm to make the optimal sampling rate adaption and resource allocation decisions. To reduce the training complexity, an optimization subroutine is embedded in the proposed algorithm for the optimal edge computing resource allocation.

The remainder of this article is organized as follows. Section II reviews related works. Section III presents the system model and problem formulation. Section IV proposes a learning-based solution. Section V gives the simulation results. Finally, Section VI concludes this article.

II. Related Work

DNN inference for resource-constrained industrial IoT devices has garnered much attention recently. A device-only solution aims to facilitate DNN inference services resorting to on-board computing resources. To reduce the computational complexity, DNN compression techniques are applied, such as weight pruning [6] and knowledge distillation [9]. Considering the widely equipped energy-harvesting functionality in IoT devices, Gobieski *et al.* [2] designed a light-weight DNN inference model, which can dynamically compress the model size in order to balance inference accuracy and energy efficiency. In another

[1]The experiment is conducted on an open-source dataset [8]. This dataset collects the vibration signal of drive end bearings at a sampling rate of 48 kHz, and there are ten types of possible faults.

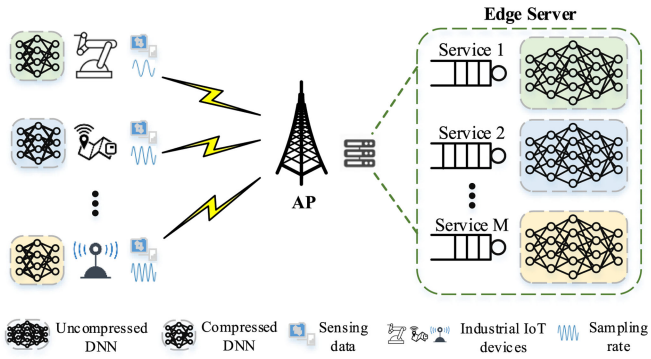


Fig. 2. Collaborative DNN inference framework for industrial IoT devices.

line of research, edge-assisted DNN inference solutions can provide high-accuracy inference services by utilizing powerful edge computing servers. To facilitate low-delay and accurate DNN-based video analytics, Yang *et al.* [10] proposed an online video quality and computing resource allocation strategy to maximize video analytic accuracy. Another inspiring work proposed a novel device-edge collaborative inference scheme, in which the DNN model is partitioned and deployed at both the device and the edge, and intermediate results are transferred via wireless links [5]. The abovementioned works can provide possible resource allocation solutions to enhance DNN inference performance. Different from existing works, our work takes the sampling rate adaption of industrial IoT devices into account, aiming at providing accuracy-guaranteed inference services in dynamic industrial IoT networks.

RL algorithms have been widely applied in allocating network resources in wireless networks, such as service migration in vehicular networks [11], network slicing in cellular networks [12], content caching in edge networks [13], and task scheduling in industrial IoT networks [14]. Hence, RL algorithms are considered as plausible solutions to manage network resources for DNN inference services. However, DNN inference services require minimizing the average delay while satisfying the long-term accuracy constraints. Traditional RL algorithms, e.g., DDPG, can be applied to solve MDPs, in which learning agents seek to optimize a long-term reward without policy constraints while they cannot deal with constrained long-term optimization problems [15], [16]. Our proposed deep RL-based algorithm can address long-term constraints within the RL framework by the modification of reward based on the Lyapunov optimization technique. In addition, an optimization subroutine is embedded in our algorithm to further reduce the training complexity.

III. System Model and Problem Formulation

A. Network Model

As shown in Fig. 2, we consider a wireless network with one AP to serve multiple types of industrial IoT devices. The AP is in charge of collecting network information and resource orchestration within the network. Consider M types of inference services, denoted by a set $\mathcal{M}$, such as facility fault diagnosis and facility monitoring services. Taking the facility fault diagnosis service as an example, vibration sensors installed on industrial IoT devices sense the operating conditions at a sampling rate, and feed the sensed vibration signal into a DNN; then, the DNN diagnoses the facility fault type. The set of industrial IoT devices subscribed to service m is denoted by $\mathcal{N}_m$, and the set of all industrial IoT devices is denoted by $\mathcal{N} = \cup_{m\in\mathcal{M}}\mathcal{N}_m$. In the collaborative inference framework, two types of DNNs are deployed. One is a compressed DNN, which is deployed at industrial IoT devices. The compressed DNN can be implemented via the weight pruning technique, which prunes less-important weights to reduce computational complexity while maintaining similar inference accuracy [6]. The other is an uncompressed DNN, which is deployed at the AP. In this way, M types of uncompressed DNNs share the edge computing resource to serve different inference requests. Important notations are summarized in Table I.

TABLE I
Summary of Notations

Notation	Description
A_m	Achieved instantaneous accuracy of service m
A_m^{th}	Long-term accuracy requirement of service m
B^t	Local computing queue backlog in time slot t
$\mathbf{c}$	Computing resource allocation decision vector
D	Service delay
$L(\cdot)$	Lyapunov function
$\mathbf{o}$	Task offloading decision vector
Q^t	Edge computing queue backlog in time slot t
V	Parameter to balance delay and accuracy requirement
$\mathbf{X}$	Sampling rate selection decision matrix of all devices
Z^t	Accuracy deficit queue backlog in time slot t
ξ_n	Raw task data size of device n
η_m	Task computation intensity of service m
λ_n	Average task arrival rate of device n
$\zeta(\mathbf{x}_n^t)$	Task data size of device n in time slot t
Ψ^t	Amount of dropped tasks in computing queues

The collaborative DNN inference framework operates in a time-slotted manner. Let t denote the time index, where $t \in \mathcal{T} = \{1, 2, ..., T\}$. The detailed procedure is given as follows.

1) Sampling rate selection: Industrial IoT devices first select their sampling rates according to channel conditions and computation workloads. The set of candidate sampling rates is denoted by $\mathcal{K} = \{\theta_1, \theta_2, ..., \theta_K\}$, where θ_K denotes the raw sampling rate. We assume the sampling rate in $\mathcal{K}$ increases linearly with the index, i.e., $\theta_k = k\theta_K/K$. Let $\mathbf{X}^t$ denote the sampling rate decision matrix in time slot t, whose element $x_{n,k}^t = 1$ indicates industrial IoT device $n \in \mathcal{N}$ selects the kth sampling rate.
2) Task processing: The sensing data from industrial IoT devices within a time slot is deemed as a computation task, which can be either offloaded to the AP or executed locally. Let $\mathbf{o}^t \in \mathbb{R}^{|\mathcal{N}|\times 1}$ denote the offloading decision vector in time slot t, whose element $o_n^t = 0$ indicates offloading the computation task from industrial IoT device n. Otherwise, $o_n^t = 1$ indicates executing the computation task locally.

B. Service Delay Model

A computation task can be either processed locally or offloaded to the AP. In what follows, we analyze the service delay in these two cases.

1) Executing locally: Let λ_n^t denote the task arrival rate of the nth industrial IoT device in time slot t, which is assumed to follow a general random distribution. The raw data size of the generated tasks at the nth device is denoted by $\xi_n^t = \lambda_n^t \nu_m, \forall n \in \mathcal{N}_m$, where ν_m denotes the raw data size of a task for service m. After the sampling rate is selected, the data size of the generated task is represented by $\zeta(\mathbf{x}_n^t) = \sum_{k=1}^{K} x_{n,k}^t \xi_n^t k/K$, where $\mathbf{x}_n^t = \{x_{n,k}^t\}_{k\in\mathcal{K}}$ is the sampling rate selection decision vector of the nth device. When the inference task is processed locally by a compressed DNN, the service delay includes the queuing delay in the local computing queue and task processing delay, which is given by

$$d_{n,l}^t = \frac{o_n^t \eta_{m,c} \left(B_n^t + \zeta\left(\mathbf{x}_n^t\right)\right)}{f_n} \quad \forall n \in \mathcal{N}_m \tag{1}$$

where f_n is the CPU frequency of the nth industrial IoT device, and $\eta_{m,c}$ denotes the computation intensity of the compressed DNN for the mth service. Here, B_n^t is the backlogged computation tasks (in bits) in the local computing queue, which is updated via

$$B_n^{t+1} = \min\left\{\left[B_n^t + o_n^t \zeta\left(\mathbf{x}_n^t\right) - \frac{f_n \tau}{\eta_{m,c}}\right]^+, B_n^{\max}\right\} \tag{2}$$

where $[x]^+ = \max\{x, 0\}$, $B_n^{\max}$ is the capacity of the local computing queue, and τ is the duration of a time slot. Tasks will be dropped if the local computing queue is full. Let

$$\Psi_{b,n}^t = \max\left\{B_n^t + o_n^t \zeta\left(\mathbf{x}_n^t\right) - \frac{f_n \tau}{\eta_{m,c}} - B_n^{\max}, 0\right\} \tag{3}$$

denote the amount of the dropped tasks in the local computing queue of device n. Here, $\Psi_{n,b}^t > 0$ indicates that an event of local computing queue overflow occurs at the nth device, and the corresponding penalty will be incurred to avoid queue overflow.

2) Offloading to AP: When a task is offloaded to the AP, it will be processed by an uncompressed DNN. The service delay consists of task offloading delay, queuing delay in the edge computing queue, and task processing delay, which are analyzed, respectively, as follows.

1) Task offloading delay: For the nth industrial IoT device, the offloading delay is given by

$$d_{n,o}^t = \frac{\left(1 - o_n^t\right) \zeta\left(\mathbf{x}_n^t\right)}{R_n^t} \tag{4}$$

where transmission rate between the nth industrial IoT device and the AP, R_n^t, is given by

$$R_n^t = \frac{W}{N} \log_2\left(1 + \frac{P_T G(H_n^t)}{N_f \sigma^2}\right). \tag{5}$$

Here, W, P_T, $G(H_n^t)$, and N_f represent the system bandwidth, transmit power, channel gain, and noise figure, respectively. $\sigma^2 = N_o W/N$ denotes the background noise where N_o is thermal noise spectrum density. Channel gain $G(H_n^t)$ varies in terms of channel state H_n^t. Based on extensive real-time measurements, channel state H_n^t can be modeled with a finite set of channel states $\mathcal{H}$ [17]. The evolution of channel states is characterized by a discrete-time and ergodic Markov chain model, whose transition matrix is $\mathbf{P} \in \mathbb{R}^{|\mathcal{H}|\times|\mathcal{H}|}$.

2) Task processing delay: The tasks from all industrial IoT devices subscribed to the mth service are placed in the edge computing queue for the mth service. The amount of aggregated tasks is given by $\sum_{n\in\mathcal{N}_m}(1 - o_n^t)\zeta(\mathbf{x}_n^t)$. The computing resource is dynamically allocated among multiple services at the AP according to service task arrivals, which can be realized via containerization techniques, such as Dockers and Kubernetes [18]. Let $\mathbf{c}^t \in \mathbb{R}^{M\times 1}$ denote the computing resource allocation decision vector in time slot t. Each element $0 \le c_m^t \le 1$ denotes the portion of computing resource allocated to the mth service. Hence, the processing delay is given by

$$d_{n,p}^t = \frac{\eta_{m,u}\left(1 - o_n^t\right)\zeta\left(\mathbf{x}_n^t\right)}{c_m^t f_b} \quad \forall n \in \mathcal{N}_m \tag{6}$$

where f_b is the CPU frequency of the computing server at the AP, and $\eta_{m,u}$ denotes the computation intensity of processing the mth service task by the uncompressed DNN. Note that $\eta_{m,u} > \eta_{m,c}$, since the uncompressed DNN consumes more computing resource.

3) Queuing delay: The queuing delay consists of two components: first, the time taken to process backlogged tasks in the edge computing queue, which is given by

$$d_{n,q}^t = \frac{Q_m^t \eta_{m,u}}{c_m^t f_b} \quad \forall n \in \mathcal{N}_m. \tag{7}$$

Here, Q_m^t denotes the edge computing queue backlog for the mth service in time slot t, which is updated according to

$$Q_m^{t+1} = \min\left\{\left[Q_m^t + a_m^t - \frac{c_m^t f_b \tau}{\eta_{m,u}}\right]^+, Q_m^{\max}\right\}. \tag{8}$$

Here, $a_m^t = \sum_{n\in\mathcal{N}_m}(1 - o_n^t)\zeta(\mathbf{x}_n^t)$ and $Q_m^{\max}$ denotes the capacity of the mth edge computing queue. Similar to that in local computing queues, tasks will also be dropped if the edge computing queue is full, and the amount of dropped tasks for the mth edge computing queue is given by

$$\Psi_{q,m}^t = \max\left\{Q_m^t + a_m^t - \frac{c_m^t f_b \tau}{\eta_{m,u}} - Q_m^{\max}, 0\right\}. \tag{9}$$

Here, $\Psi_{q,m}^t > 0$ indicates that an event of edge computing queue overflow occurs; and second, average waiting time among all newly arrived tasks until the task of industrial IoT device n is processed, which is given by

$$d_{n,w}^t = \frac{\eta_{m,u}\sum_{i\neq n, i\in\mathcal{N}_m}\left(1 - o_i^t\right)\zeta\left(\mathbf{x}_i^t\right)}{2 c_m^t f_b}. \tag{10}$$

Here, $\sum_{i\neq n, i\in\mathcal{N}_m}(1 - o_i^t)\zeta(\mathbf{x}_n^t)$ denotes the amount of aggregated tasks except the task of industrial IoT device n.

Taking both local execution and offloading into account, the service delay in time slot t is given by

$$D\left(\mathbf{X}^t,\mathbf{o}^t,\mathbf{c}^t\right) = \sum_{n\in\mathcal{N}}\left(d_{n,l}^t + d_{n,o}^t + d_{n,p}^t + d_{n,q}^t + d_{n,w}^t\right) + w_p\left(\sum_{n\in\mathcal{N}}\mathbb{1}_{\{\Psi_{b,n}^t>0\}} + \sum_{m\in\mathcal{M}}\mathbb{1}_{\{\Psi_{q,m}^t>0\}}\right) \tag{11}$$

where $\mathbb{1}_{\{x\}}=1$ and $w_p>0$ are the indicator function and the positive unit penalty cost for queue overflow, respectively. The first term represents the experienced delay to complete all tasks in time slot t. The second term represents the penalty for potential overflow events in local and edge computing queues.

C. Inference Accuracy Model

The inference accuracy depends on the sampling rate of a task and the type of DNN that executes a task. First, we characterize the relationship between the inference accuracy and the sampling rate, which is specified by accuracy function $g(\theta_k),\forall\theta_k\in\mathcal{K}$. Specifically, we implement a DNN inference algorithm, i.e., AlexNet [19], and apply the AlexNet to diagnose facility fault type based on the collected bearing vibration signal from the dataset [8], and, then, measure the accuracy function values with respect to sampling rates, as shown in Fig. 1. Second, the relationship between the inference accuracy and the type of DNN is also characterized via experiments. Here, $h_{m,c}$ and $h_{m,u}$ represent the inference accuracy of the compressed DNN and the uncompressed DNN for the mth service, respectively. Note that, $h_{m,c}<h_{m,u}$, as an uncompressed DNN achieves higher fault diagnosis accuracy.

Since the DNN model selection (i.e., task offloading decision) and the sampling rate selection are independent, inference accuracy is the product of the accuracy value with respect to the selected sampling rate and the accuracy value with respect to the selected DNN type, i.e., $g(\sum_{k\in\mathcal{K}}x_{n,k}^t\theta_k)(o_n^t h_{m,c}+(1-o_n^t)h_{m,u})$. Hence, the average inference accuracy for the mth service in time slot t can be given by

$$A_m\left(\mathbf{X}^t,\mathbf{o}^t\right) = \sum_{n\in\mathcal{N}_m}\frac{1}{|\mathcal{N}_m|}g\left(\sum_{k\in\mathcal{K}}x_{n,k}^t\theta_k\right)\cdot \left(o_n^t h_{m,c}+\left(1-o_n^t\right)h_{m,u}\right). \tag{12}$$

Note that the model can be readily extended to cases when other inference methods are adopted, since the accuracy values with respect to sampling rates and DNN types are obtained via practical experiments.

D. Problem Formulation

DNN inference services require not only minimizing service delay, but also guaranteeing their long-term accuracy requirements, which can be modeled via a CMDP. Its action, state, reward, and state transition matrix are defined as follows.

1) Action: The action includes the sampling rate selection, task offloading, and edge computing resource allocation decisions, i.e., $\hat{a}^t=\{\mathbf{X}^t,\mathbf{o}^t,\mathbf{c}^t\}$. Note that the components of the action should satisfy following constraints:
   a) $x_{n,k}^t\in\{0,1\}$ constrains the sampling rate selection decision;
   b) $o_n^t\in\{0,1\}$ requires the binary task offloading decision;
   c) $\sum_{m\in\mathcal{M}}c_m^t\le 1$ and $0\le c_m^t\le 1$ constrain a continuous computing resource allocation decision.
2) State: The state includes local computing queues backlog of industrial IoT devices B_n^t, edge computing queues backlog Q_m^t, channel conditions of industrial IoT devices H_n^t, and the raw data size of the generated tasks at industrial IoT devices ξ_n^t, i.e.,
$$\hat{s}^t=\{\{B_n^t\}_{n\in\mathcal{N}},\{Q_m^t\}_{m\in\mathcal{M}},\{H_n^t\}_{n\in\mathcal{N}},\{\xi_n^t\}_{n\in\mathcal{N}}\}. \tag{13}$$
   The queue backlogs, i.e., $\{B_n^t\}_{n\in\mathcal{N}}$ and $\{Q_m^t\}_{m\in\mathcal{M}}$, adopt a unit in bits, which result in large state space, especially for a large number of industrial IoT devices.
3) Reward: The reward is designed to minimize the service delay in (22) in time slot t, which is defined as $\hat{r}^t=-D(\mathbf{X}^t,\mathbf{o}^t,\mathbf{c}^t)$.
4) State transition probability: State transition probability is given by
$$\begin{aligned}\Pr\left(\hat{s}^{t+1}|\hat{s}^t,\hat{a}^t\right) = &\prod_{n\in\mathcal{N}}\Pr\left(B_n^{t+1}|B_n^t,x_{n,k}^t,o_n^t\right)\cdot\\ &\prod_{m\in\mathcal{M}}\Pr\left(Q_m^{t+1}|Q_m^t,\mathbf{X}^t,\mathbf{o}^t\right)\cdot\\ &\prod_{n\in\mathcal{N}}\Pr\left(H_n^{t+1}|H_n^t\right)\cdot\prod_{n\in\mathcal{N}}\Pr\left(\xi_n^{t+1}|\xi_n^t\right).\end{aligned} \tag{14}$$
   The equality holds due to the independence of different state components. The first two components are governed by the evolution of local computing queues and edge computing queues in (2) and (8), respectively. The third component is evolved according to the discrete-time Markov chain of channel conditions, and the last component is governed by the memoryless task arrival pattern. Note that each of those state components only depends on its previous state components, which means the state transition is Markovian.

Our goal is to find a stationary policy $\pi\in\Pi$ that dynamically configures sampling rates selection $\mathbf{X}^t$, task offloading $\mathbf{o}^t$, and edge computing resource allocation $\mathbf{c}^t$ according to state $\hat{s}^t$, to minimize the service delay while guaranteeing long-term inference accuracy requirements $\{A_m^{th}\}_{m\in\mathcal{M}}$, which is formulated as the following problem:

$$\mathbf{P}_0: \min_{\pi\in\Pi}\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_\pi\left[D\left(\mathbf{X}^t,\mathbf{o}^t,\mathbf{c}^t\right)\right] \tag{15a}$$

$$\text{s.t.}\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}A_m\left(\mathbf{X}^t,\mathbf{o}^t\right)\ge A_m^{th}\quad\forall m\in\mathcal{M}. \tag{15b}$$

Here, $\mathbf{P}_0$ is a CMDP. Directly solving the above CMDP via dynamic programming solutions [15] is challenging due to the

following reasons. First, state transition probability is unknown due to the lack of statistic information on the channel condition variation and task arrival patterns of all industrial IoT devices. Second, even the state transition probability is known, large action space and state space that grow with respect to the number of industrial IoT devices incur an extremely high computational complexity, which makes dynamic programming solutions intractable. Hence, we propose a deep RL-based algorithm to solve the CMDP, which can be applied in large-scale networks without requiring statistic information of network dynamics.

IV. Deep RL-Based Sampling Rate Adaption and Resource Allocation Algorithm

As mentioned before, a CDMP problem cannot be directly solved via traditional RL algorithms. We first leverage the Lyapunov optimization technique to deal with the long-term constraints and transform the problem into an MDP. Then, we develop a deep RL-based algorithm to solve the MDP. To further reduce the training complexity, an optimization subroutine is embedded to directly obtain the optimal edge computation resource allocation.

A. Lyapunov-Based Problem Transformation

The major challenge in solving problem $\mathbf{P}_0$ is to handle the long-term constraints. We leverage the Lyapunov technique [20], [21] to address this challenge. The *core idea* is to construct accuracy deficit queues to characterize the satisfaction status of the long-term accuracy constraints, thereby guiding the learning agent to meet the long-term accuracy constraints. The problem transformation procedure is presented as follows.

First, we construct inference accuracy *deficit queues* for all services, whose dynamics evolves as follows:

$$Z_m^{t+1} = \left[A_m^{th} - A_m\left(\mathbf{X}^t, \mathbf{o}^t\right) + Z_m^t\right]^+ \quad \forall m \in \mathcal{M}. \tag{16}$$

Here, Z_m^t indicates the deviation of the achieved instantaneous accuracy from the long-term accuracy requirement, whose initial state is set to $Z_m^0 = 0$. Then, a Lyapunov function is introduced to characterize the satisfaction status of the long-term accuracy constraint, which is defined as $L(Z_m^t) = (Z_m^t)^2/2$ [20]–[22]. A smaller value of $L(Z_m^t)$ indicates better satisfaction of the long-term accuracy constraint.

Secondly, the Lyapunov function should be consistently pushed to a low value in order to guarantee the long-term accuracy constraints. Hence, we introduce a *one-shot Lyapunov drift* to capture the variation of the Lyapunov function across two subsequent time slots [20]. Given Z_m^t, the one-shot Lyapunov drift is defined as $\Delta(Z_m^t) = L(Z_m^{t+1}) - L(Z_m^t)$, which is upper bounded by

$$\begin{aligned}
&\Delta\left(Z_m^t\right) = \frac{1}{2}\left(\left(Z_m^{t+1}\right)^2 - \left(Z_m^t\right)^2\right) \\
&\le \frac{1}{2}\left(\left(Z_m^t + A_m^{th} - A_m\left(\mathbf{X}^t, \mathbf{o}^t\right)\right)^2 - \left(Z_m^t\right)^2\right) \\
&= \frac{1}{2}\left(A_m^{th} - A_m\left(\mathbf{X}^t, \mathbf{o}^t\right)\right)^2 + Z_m^t\left(A_m^{th} - A_m\left(\mathbf{X}^t, \mathbf{o}^t\right)\right) \\
&\le C_m + Z_m^t\left(A_m^{th} - A_m\left(\mathbf{X}^t, \mathbf{o}^t\right)\right)
\end{aligned} \tag{17}$$

where $C_m = (A_m^{th} - A_m^{\min})^2/2$ is a constant, and $A_m^{\min}$ is the lowest inference accuracy that can be achieved for service m. The first inequality is due to the substitution of (16), and the second inequality is because $A_m(\mathbf{X}^t, \mathbf{o}^t) \ge A_m^{\min}$.

Third, based on the Lyapunov optimization theory, the original CMDP of minimizing the service delay while guaranteeing the long-term accuracy requirements boils down to minimizing a *drift-plus-cost*, i.e.,

$$\begin{aligned}
&\sum_{m\in\mathcal{M}} \Delta\left(Z_m^t\right) + V \cdot D\left(\mathbf{X}^t, \mathbf{o}^t, \mathbf{c}^t\right) \\
&\le \sum_{m\in\mathcal{M}} C_m + \sum_{m\in\mathcal{M}} Z_m^t\left(A_m^{th} - A_m\left(\mathbf{X}^t, \mathbf{o}^t\right)\right) \\
&\quad + V \cdot D\left(\mathbf{X}^t, \mathbf{o}^t, \mathbf{c}^t\right)
\end{aligned} \tag{18}$$

where the inequality is due to the upper bound in (17). Here V is a positive parameter to adjust the tradeoff between the service delay minimization and the satisfaction status of the long-term accuracy constraints. The underlying rationale is that if the long-term accuracy constraint is violated, i.e., $Z_m^t > 0$, stratifying the long-term constraints by improving the instantaneous inference accuracy becomes more urgent than reducing the service delay.

In this way, the CMDP is transformed into an MDP with the objective of minimizing the drift-plus-cost in each time slot.

B. Equivalent MDP

In the equivalent MDP, the action, state, reward, and state transition matrix are modified as follows due to the incorporation of accuracy deficit queues.

1) Action: The action is the same as that in the CMDP, i.e., $a^t = \hat{a}^t = \{\mathbf{X}^t, \mathbf{o}^t, \mathbf{c}^t\}$.
2) State: Compared with the state of the CMDP, the accuracy deficit queue backlog of services $\{Z_m^t\}_{m\in\mathcal{M}}$ should be incorporated, i.e.,

$$s^t = \{\hat{s}^t, \{Z_m^t\}_{m\in\mathcal{M}}\}. \tag{19}$$

3) Reward: The reward is modified to minimize the drift-plus-cost in (18) in time slot t, i.e.,

$$\begin{aligned}
r^t = &-V \cdot D\left(\mathbf{X}^t, \mathbf{o}^t, \mathbf{c}^t\right) \\
&- \sum_{m\in\mathcal{M}} Z_m^t\left(A_m^{th} - A_m\left(\mathbf{X}^t, \mathbf{o}^t\right)\right).
\end{aligned} \tag{20}$$

Note that the constant term $\sum_{m\in\mathcal{M}} C_m$ in (18) is ignored in the reward for brevity.

4) State transition probability: Since accuracy deficit queue backlogs are incorporated in the state, the state transition probability evolves according to

$$\begin{aligned}
\Pr\left(s^{t+1}|s^t, a^t\right) = &\Pr\left(\hat{s}^{t+1}|\hat{s}^t, \hat{a}^t\right) \cdot \\
&\prod_{m\in\mathcal{M}} \Pr\left(Z_m^{t+1}|Z_m^t, \mathbf{X}^t, \mathbf{o}^t\right).
\end{aligned} \tag{21}$$

where the second term is the evolution of the accuracy deficit queue backlog according to (16). Note that the overall state transition is still Markovian.

Then, problem $\mathbf{P}_0$ is transformed into the following MDP problem:

$$\mathbf{P}_1: \min_{\pi\in\Pi} \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_\pi\left[\sum_{m\in\mathcal{M}} Z_m^t\left(A_m^{th}-A_m\left(\mathbf{X}^t,\mathbf{o}^t\right)\right) + V\cdot D\left(\mathbf{X}^t,\mathbf{o}^t,\mathbf{c}^t\right)\right]. \quad (22)$$

Similar to CMDP, solving an MDP via dynamic programming solutions also suffers from the curse of dimensionality due to large state space. Hence, we propose a deep RL-based algorithm to solve the MDP, which is detailed in Section IV-D.

C. Optimization Subroutine for Edge Computing Resource Allocation

Although $\mathbf{P}_1$ can be directly solved by RL algorithms, an inherent property on edge computing resource allocation can be leveraged, in order to reduce the training complexity of RL algorithms. Through analysis on (22), the edge computing resource allocation is independent of the inference accuracy performance, and hence, it only impacts the one-shot service delay performance. In time slot t, once task offloading and sampling rate selection decisions are made, the optimal computing resource allocation decision can be obtained via solving the following optimization problem:

$$\mathbf{P}_2: \min_{\mathbf{c}^t} D\left(\mathbf{X}^t,\mathbf{o}^t,\mathbf{c}^t\right)$$
$$\text{s.t.} \sum_{m\in\mathcal{M}} c_m^t \le 1 \quad (23a)$$
$$0 \le c_m^t \le 1. \quad (23b)$$

A further analysis of (11) indicates that only the task processing delay and queuing delay at the AP are impacted by the edge computing resource allocation, i.e., $\sum_{n\in\mathcal{N}}(d_{n,p}^t+d_{n,q}^t+d_{n,w}^t)$. In addition, the aggregated delay from the perspective of all devices is equivalent to the aggregated delay from the perspective of all services. Hence, the objective function in $\mathbf{P}_2$ can be rewritten as $\sum_{m\in\mathcal{M}} d_m^t$, where

$$d_m^t = \sum_{n\in\mathcal{N}_m}\left(\frac{\eta_{m,u}\left(1-o_n^t\right)\zeta\left(\mathbf{x}_n^t\right)}{c_m^t f_b} + \frac{Q_m^t\eta_{m,u}}{c_m^t f_b} + \frac{\eta_{m,u}\sum_{i\neq n,i\in\mathcal{N}_m}\left(1-o_i^t\right)\zeta\left(\mathbf{x}_i^t\right)}{2c_m^t f_b}\right) \quad (24)$$

denotes the experienced delay of the mth service. By analyzing the convexity property of the problem, we have the following theorem to obtain the optimal edge computation resource allocation in each time slot.

Theorem 1: The optimal edge computing resource allocation for problem $\mathbf{P}_2$ is given by

$$c_m^{t,\star} = \frac{\sqrt{\Lambda_m^t}}{\sum_{m\in\mathcal{M}}\sqrt{\Lambda_m^t}} \quad \forall m\in\mathcal{M} \quad (25)$$

Algorithm 1: Deep RL-based algorithm for sampling rate adaption and resource allocation

Initialization: Initialize all neural networks and the experience replay memory;
for *each episode* **do**
  Reset the environment and obtain initial state s_0;
  for *time slot* $t\in\mathcal{T}$ **do**
    Determine the sampling rate selection and task offloading actions $\{\mathbf{X}^t,\mathbf{o}^t\}$ by the actor network according to current state s^t;
    Determine edge computing resource allocation action $\mathbf{c}^t$ by (25);
    Send joint action $a^t=\{\mathbf{X}^t,\mathbf{o}^t,\mathbf{c}^t\}$ to all industrial IoT devices by the AP;
    Execute the joint action at industrial IoT devices;
    Observe reward r^t and new state s^{t+1};
    Store transition $\{s^t,a^t,r^t,s^{t+1}\}$ in the epxerience replay memory;
    Sample a random minibatch transitions from the epxerience replay memory;
    Train the critic and actor network by (27) and (28), respectively;
    Update target networks by (29);
  end
end

where

$$\Lambda_m^t = \sum_{n\in\mathcal{N}_m}\left(\eta_{m,u}\left(1-o_n^t\right)\zeta\left(\mathbf{x}_n^t\right) + Q_m^t\eta_{m,u} + \frac{\eta_{m,u}}{2}\sum_{i\neq n,i\in\mathcal{N}_m}\left(1-o_i^t\right)\zeta\left(\mathbf{x}_i^t\right)\right). \quad (26)$$

Proof: Proof is provided in Appendix A.

This optimization subroutine for the edge computing resource allocation is embedded in the following proposed deep RL-based algorithm. In this way, the training complexity can be reduced, because it is no longer necessary to train the neural networks to obtain optimal edge computing resource allocation policy.

D. Deep RL-Based Algorithm

To solve problem $\mathbf{P}_1$, we propose a deep RL-based algorithm, which is extended from the celebrated DDPG algorithm [23]. The main difference between DDPG and the proposed algorithm is that the abovementioned optimization subroutine for computing resource allocation is embedded to reduce the training complexity. The proposed algorithm can be deployed at the AP, which collects the entire network state information and enforces the policy to all connected industrial IoT devices.

In the algorithm, the learning agent has two parts: first, an actor network, which is to determine the action based on the current state; second, a critic network, which is to evaluate

the determined action based on the reward feedback from the environment. Let $\mu(s|\phi^{\mu})$ and $Q(s,a|\phi^{Q})$ denote the actor network and the critic network, respectively, whose neural network weights are ϕ^{μ} and ϕ^{Q}. As shown in Algorithm 1, the deep RL-based algorithm operates in a time-slotted manner, which consists of the following three steps.

The first step is to obtain experience by interacting with the environment. Based on current network state s^t, the actor network generates the sampling rate selection and task offloading actions with an additive policy exploration noise that follows Gaussian distribution $\mathcal{N}(0,\sigma^2)$. The optimization subroutine generates the edge computation resource allocation action. Then, the joint action is executed at all industrial IoT devices. The corresponding reward r^t and the next state s^{t+1} are observed from the environment. The state transition $\{s^t, a^t, r^t, s^{t+1}\}$ is stored in the experience replay memory for actor and critic network training.

The second step is to train the actor and critic network based on the stored experience. To avoid the divergence issue caused by DNN, a minibatch of transitions are randomly sampled from the experience replay memory to break experience correlation. The critic network is trained by minimizing the loss function

$$\text{Loss}\left(\phi^{Q}\right) = \frac{1}{N_b}\sum_{i=1}^{N_b}\left(y_i - Q(s_i,a_i|\phi^{Q})\right)^2 \tag{27}$$

where $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\phi^{\mu'})|\phi^{Q'})$, and N_b is the minibatch size. Here, $\mu'(s|\phi^{\mu'})$ and $Q'(s,a|\phi^{Q'})$ represent actor and critic target networks with weights $\phi^{\mu'}$ and $\phi^{Q'}$. The actor network is trained via the policy gradient

$$\nabla_{\phi^{\mu}} \approx \frac{1}{N_b}\sum_{i=1}^{N_b}\nabla_a Q(s_i,a|\phi^{Q})|_{s=s_i,a=\mu(s_i)}\nabla_{\theta^{\mu}}\mu(s_i|\phi^{\mu})|_{s_i}. \tag{28}$$

The third step is to update target networks. In order to ensure network training stability, the actor and critic target networks are softly updated by

$$\phi^{Q'} = \delta\phi^{Q} + (1-\delta)\phi^{Q'}, \phi^{\mu'} = \delta\phi^{\mu} + (1-\delta)\phi^{\mu'} \tag{29}$$

where $0 < \delta \ll 1$ denotes the target network update ratio.

V. Simulation Results

A. Simulation Setup

We consider a smart factory scenario in our simulation, in which industrial IoT devices, e.g., vibration sensors, are randomly scattered. The industrial IoT devices installed on industrial facilities (e.g., robot arms) sense their operating conditions. The sensing data are processed locally or offloaded to an AP in the smart factory for processing. The transmit power of an industrial IoT device is set to 20 dBm [24]. The channel condition is modeled with three states, i.e., "Good (G)," "Normal (N)," and "Bad (B)," and the corresponding transition matrix is given by [17]

$$\mathbf{P} = \begin{bmatrix} P_{GG} & P_{GN} & 0 \\ P_{NG} & P_{NN} & P_{NB} \\ 0 & P_{NB} & P_{BB} \end{bmatrix} = \begin{bmatrix} 0.3 & 0.7 & 0 \\ 0.25 & 0.5 & 0.25 \\ 0 & 0.7 & 0.3 \end{bmatrix}. \tag{30}$$

TABLE II
Simulation Parameters [24], [25]

Parameter	Value
Thermal noise spectrum density (N_o)	-174 dBm/Hz [25]
Communication bandwidth (W)	{5 - 25} MHz
Transmit power (P_T)	20 dBm [24]
Average task arrival rate (λ)	{0.6-1} request/sec
Noise figure (N_f)	5 dB
Intensity of compressed DNN ($\eta_{1,c}, \eta_{2,c}$)	(80, 160) cycles/bit
Intensity of uncompressed DNN ($\eta_{1,u}, \eta_{2,u}$)	(200, 400) cycles/bit
Device and edge server CPU frequency (f_n, f_b)	(0.1, 2) GHz
Number of Type I/II devices (N_1, N_2)	5, 5
Time slot duration (τ)	1 second
Balance parameter (V)	0.05
Unit penalty for queue overflow (w_p)	1
Accuracy of compressed DNN ($h_{1,c}, h_{2,c}$)	0.8, 0.8
Accuracy of uncompressed DNN ($h_{1,u}, h_{2,u}$)	1, 1
Local/edge queue capacity (B_{max}, Q_{max})	(3.84, 19.2) megabits

Two types of DNN inference services are considered. *Type I service*: A facility fault diagnosis service to identify the fault type based on the collected bearing vibration signal from the dataset [8]. Since the duration of a time slot in the simulation is set to be 1 s, the task data size is the data volume of a 1-s signal, which is a product of the raw sampling rate and the quantization bits of the signal. In the dataset, the bearing vibration signal is collected at 48 kHz sampling rate and 16 b quantization, and hence, the corresponding task data size is 768 kb. The long-term accuracy requirement of the service is set to 0.8. *Type II service*: A service extended from the Type I service to diagnose facility fault based on a low-grade bearing vibration dataset while requiring higher inference accuracy 0.9. The low-grade dataset collects vibration signal at a lower sampling rate of 32 kHz, and hence, the task data size is 512 kb. For both services, the task arrival rate of each industrial IoT device at each time slot follows a uniform distribution $\mathcal{U}(\lambda - 0.5, \lambda + 0.5)$, where λ is the average task arrival rate. We consider four candidate sampling rates for industrial IoT devices, which are 25%, 50%, 75%, and 100% of the raw sampling rate. The corresponding accuracy with respect to the sampling rates are 0.59, 0.884, 0.950, and 0.987, respectively, based on extensive experiments on the bearing vibration dataset [8]. Balance parameter V is set to 0.05 based on extensive simulations. Other important simulation parameters are listed in Table II. The parameters of the proposed algorithm are given in Table III. The proposed algorithm is compared to the following benchmarks.

1) *Delay myopic*: Each industrial IoT device dynamically makes sampling rate selection and task offloading decisions by maximizing the one-step reward in (20) according to the network state.
2) *Static configuration*: Each industrial IoT device takes a static configuration on the sampling rate selection and task offloading decisions, which can guarantee services' accuracy requirements.

TABLE III
PARAMETERS OF THE PROPOSED RL-BASED ALGORITHM

Parameters	Value	Parameters	Value
Actor learning rate	10^{-4}	Critic learning rate	10^{-3}
Actor hidden units	$(64, 32)$	Critic hidden units	$(64, 32)$
Hidden activation	ReLU	Actor output activation	Tanh
Optimizer	Adam	Policy noise (σ)	0.2
Target update (δ)	0.005	Discount factor	0.85
Minibatch size	64	Replay memory size	100,000
Training episodes	$1,000$	Time slots per episode	200

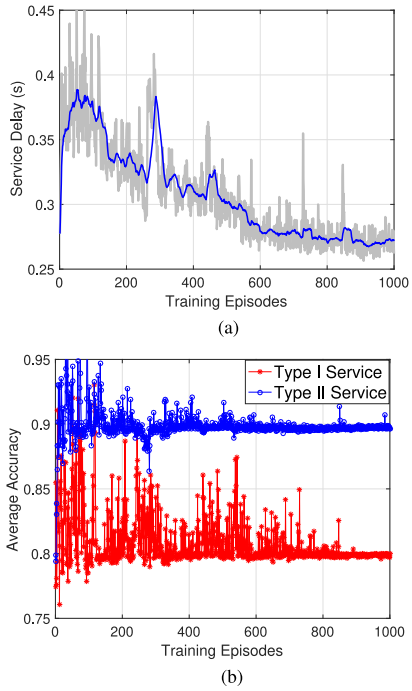


Fig. 3. Performance of the proposed algorithm in the training stage. (a) Average delay performance. (b) Accuracy of two services.

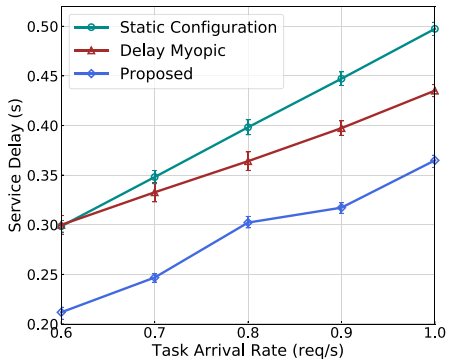


Fig. 4. Service delay performance with respect to task arrival rates.

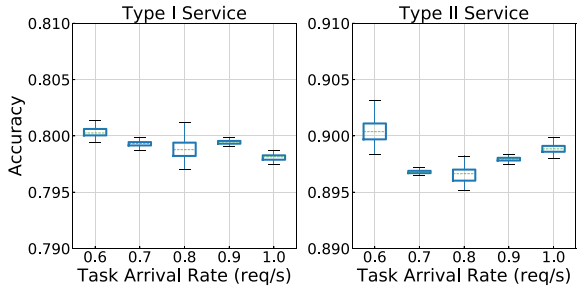


Fig. 5. Inference accuracy performance with respect to task arrival rates.

B. Performance Evaluation

1) Convergence of the Proposed Algorithm: The service delay performance in the training stage is shown in Fig. 3(a). We can clearly see that the average service delay gradually decreases as the increase of training episodes, which validates the convergence of the proposed algorithm. In addition, Fig. 3(b) shows the accuracy performance for both services with respect to training episodes. The accuracy performance is not good at the beginning of the training stage, but after 1000 episodes of training, the accuracy performance converges to the predetermined requirements.

2) Impact of Task Arrival Rate: Once well-trained offline, we evaluate the performance of the proposed algorithm in the online inference. As shown in Fig. 4, we compare the average service delay performance of the proposed algorithm with benchmark schemes in terms of task arrival rates for $W = 20$ MHz. Each simulation point is plotted with a 95% confidence interval. Several observations can be obtained from the figure. First, the service delay increases with the task arrival rate due to constrained communication and computing resources in the network. Second, the proposed algorithm significantly outperforms benchmark schemes. The reason is that the proposed RL-based algorithm can capture network dynamics, such as the task arrival pattern and channel condition variation, via interacting with the environment. The learned knowledge is utilized to make online decisions that target at the long-term performance, whereas benchmark schemes only focus on the short-term performance and do not adapt to network dynamics. Specifically, the proposed algorithm can reduce the average service delay by 19% and 25%, respectively, as compared with delay myopic and static configuration schemes.

As shown in Fig. 5, boxplot accuracy distribution of two services is presented with respect to different task arrival rates. The long-term accuracy requirements for two services are 0.8 and 0.9, respectively. It can be seen that the proposed algorithm guarantees the long-term accuracy requirements of both services with a high probability. Specifically, the maximum error probability is less than 0.5%.

3) Impact of Communication Bandwidth: Fig. 6 shows the impact of communication bandwidth on the average service delay. First, we can see that the average service delay decreases as the growth of bandwidth. The reason is that the transmission delay is reduced when the communication resource becomes sufficient. In addition, the proposed algorithm achieves good performance when the bandwidth is scarce. When system bandwidth is only 5 MHz, the proposed algorithm achieves 1.20× and 1.42× delay reduction compared with delay myopic and static configuration schemes, respectively, which is larger than that when the system bandwidth is 25 MHz (1.15× and 1.31×). The reason is that the proposed algorithm efficiently utilizes

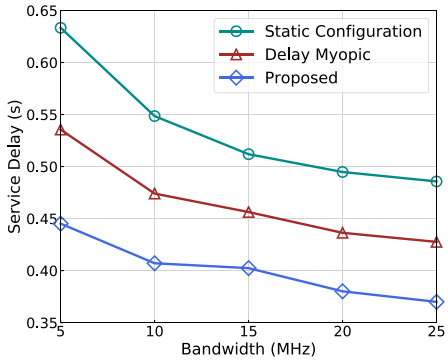


Fig. 6. Service delay performance with respect to communication bandwidth.

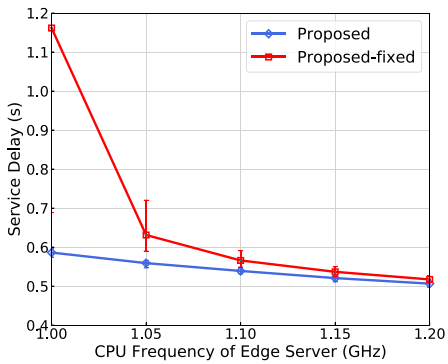


Fig. 7. Service delay in terms of CPU frequency of the edge server.

the on-board computing resources. Simulation results show that the proposed algorithm decides 47.5% computation tasks to be executed locally with 5 MHz bandwidth, whereas the delay myopic benchmark only decides 17%. Due to the efficient resource orchestration among industrial IoT devices and the AP, the proposed algorithm can effectively reduce average service delay for both services.

4) Impact of Optimization Subroutine: As shown in Fig. 7, we evaluate the performance of the proposed algorithm with the fixed computing resource allocation (referred to as proposed-fixed), in which the edge computing resource is allocated based on the average computing demand of two services. Compared with the proposed-fixed solution, the proposed algorithm achieves significant performance gain when the edge computing resource is constrained. Specifically, the performance gain in reducing the service delay decreases from 1.98× at 1 GHz CPU frequency to only 1.02× at 1.2 GHz CPU frequency. The reason is that efficient resource allocation is more important in resource-constrained scenarios, as compared to resource-rich scenarios. The results validate the effectiveness of the optimization subroutine for edge computing resource allocation. In addition to the performance gain, another merit of the optimization subroutine is to reduce the training complexity of RL algorithms.

VI. Conclusion

In this article, we studied the sampling rate adaption and resource allocation problem for collaborative DNN inference in industrial IoT networks. A deep RL-based algorithm was developed to determine the channel variation and the task arrival pattern, which were then exploited to provide accuracy-guaranteed DNN inference services. The proposed algorithm can optimize service delay performance on the fly, without requiring statistic information of network dynamics. The Lyapunov-based transformation technique can be applied to other CMDPs. For the future work, we will investigate the impact of device mobility on the inference performance.

Appendix

A. Proof of Theorem 1

First, the problem is proved to be a convex optimization problem. For brevity of notations, we omit t in the proof. With the definition of Λ_m in (26), the objective function can be rewritten as $\sum_{m\in\mathcal{M}} \Lambda_m/(c_m f_b)$. The second-order derivative of the objective function shows $2\Lambda_m/(f_b c_m^3) > 0$. In addition, the inequality constraint is linear. Hence, the problem is a convex optimization problem.

Second, a Lagrange function for the problem without considering the inequality constraints is constructed, i.e.,

$$\mathcal{L}(\mathbf{c}, a) = \sum_{m\in\mathcal{M}} \frac{\Lambda_m}{c_m f_b} + a\left(\sum_{m\in\mathcal{M}} c_m - 1\right) \tag{31}$$

where a denotes the Lagrange multiplier. Based on Karush–Kuhn–Tucker conditions [26], we have

$$\frac{\partial L(\mathbf{c}, a)}{\partial c_m} = -\frac{\Lambda_m}{f_b c_m^2} + a = 0 \quad \forall m \in \mathcal{M}. \tag{32}$$

By solving the abovementioned equation, we can obtain $c_m^\star = \sqrt{\Lambda_m/a f_b}, \forall m \in \mathcal{M}$. Substituting the abovementioned result into the complementary slackness condition $\sum_{m\in\mathcal{M}} c_m^\star - 1 = 0$, the optimal value of a is given by $a^\star = (\sum_{m\in\mathcal{M}} \sqrt{\Lambda_m})^2/f_b$. From the abovementioned equation, $a^\star$ takes a positive value, and hence, $\{c_m^\star\}_{m\in\mathcal{M}}$ are positive values, which means constraint (23b), i.e., $c_m^t \geq 0, \forall m \in \mathcal{M}$, is automatically satisfied. Substituting $a^\star$ into the complementary slackness condition proves Theorem 1.

References

[1] H. Hu, B. Tang, X. Gong, W. Wei, and H. Wang, "Intelligent fault diagnosis of the high-speed train with big data based on deep neural networks," *IEEE Trans. Ind. Informat.*, vol. 13, no. 4, pp. 2106–2116, Apr. 2017.

[2] G. Gobieski, B. Lucia, and N. Beckmann, "Intelligence beyond the edge: Inference on intermittent embedded systems," in *Proc. ACM Int. Conf. Architectural Support Program. Lang. Oper. Syst.*, 2019, pp. 199–213.

[3] Y. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Jan. 2017.

[4] D. A. Chekired, L. Khoukhi, and H. T. Mouftah, "Industrial IoT data scheduling based on hierarchical fog computing: A key for enabling smart factory," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4590–4602, Oct. 2018.

[5] 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.

[6] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," 2015, *arXiv:1510.00149*.

[7] S. Teerapittayanon, B. McDanel, and H. Kung, "BranchyNet: Fast inference via early exiting from deep neural networks," in *Proc. IEEE Int. Conf. Pattern Recognit.*, 2016, pp. 2464–2469.

[8] International Data Corporation, "Case Western Reserve University." [Online]. Available: https://csegroups.case.edu/bearingdatacenter/pages/download-data-file

[9] G. Chen, W. Choi, X. Yu, T. Han, and M. Chandraker, "Learning efficient object detection models with knowledge distillation," in *Proc. Neural Inf. Process. Syst.*, 2017, pp. 742–751.

[10] P. Yang, F. Lyu, W. Wu, N. Zhang, L. Yu, and X. Shen, "Edge coordinated query configuration for low-latency and accurate video analytics," *IEEE Trans. Ind. Informat.*, vol. 16, no. 7, pp. 4855–4864, Jul. 2020.

[11] S. Wang, Y. Guo, N. Zhang, P. Yang, A. Zhou, and X. Shen, "Delay-aware microservice coordination in mobile edge computing: A reinforcement learning approach," *IEEE Trans. Mobile Comput.*, to be published, DOI: 10.1109/TMC.2019.2957804.

[12] 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.

[13] P. Yang, N. Zhang, S. Zhang, L. Yu, J. Zhang, and X. Shen, "Content popularity prediction towards location-aware mobile edge caching," *IEEE Trans. Multimedia*, vol. 21, no. 4, pp. 915–929, Apr. 2019.

[14] K. Wang, Y. Zhou, Z. Liu, Z. Shao, X. Luo, and Y. Yang, "Online task scheduling and resource allocation for intelligent NOMA-based industrial Internet of Things," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 5, pp. 803–815, May 2020.

[15] E. Altman, *Constrained Markov Decision Processes*. Boca Raton, FL, USA: CRC Press, 1999, vol. 7.

[16] Q. Liang, F. Que, and E. Modiano, "Accelerated primal-dual policy optimization for safe reinforcement learning," 2018, *arXiv:1802.06480*.

[17] L. Lei, Y. Kuang, X. Shen, K. Yang, J. Qiao, and Z. Zhong, "Optimal reliability in energy harvesting industrial wireless sensor networks," *IEEE Trans. Wireless Commun.*, vol. 15, no. 8, pp. 5399–5413, Aug. 2016.

[18] D. Bernstein, "Containers and cloud: From LXC to Docker to Kubernetes," *IEEE Cloud Comput.*, vol. 1, no. 3, pp. 81–84, Sep. 2014.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[20] M. J. Neely, *Stochastic Network Optimization With Application to Communication and Queueing Systems* (Synthesis Lectures on Communication Networks). San Rafael, CA, USA: Morgan & Claypool, 2010, vol. 3, no. 1, pp. 1–211.

[21] J. Luo, F. Yu, Q. Chen, and L. Tang, "Adaptive video streaming with edge caching and video transcoding over software-defined mobile networks: A deep reinforcement learning approach," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 1577–1592, Mar. 2020.

[22] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *Proc. IEEE Conf. Comput. Commun.*, 2018, pp. 207–215.

[23] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," in *Proc. Int. Conf. Learn. Represent.*, 2016, pp. 1–10.

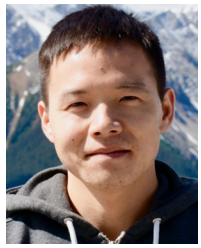
[24] V. Petrov *et al.*, "Vehicle-based relay assistance for opportunistic crowdsensing over narrowband IoT (NB-IoT)," *IEEE Internet Things J.*, vol. 5, no. 5, pp. 3710–3723, Oct. 2018.

[25] 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.

[26] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

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 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. His current research interests include millimeter-wave networks and AI-empowered wireless networks.



Peng Yang (Member, IEEE) received the B.E. degree in communication engineering and the Ph.D. degree in information and communication engineering from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2013 and 2018, respectively.

He was with the Department of Electrical and Computer Engineering, University of Waterloo, Canada, as a Visiting Ph.D. Student from 2015 to 2017, and a Postdoctoral Fellow from 2018 to 2019. Since 2020, he has been a faculty member with the School of Electronic Information and Communications, HUST. His current research interests include mobile edge computing, video streaming, and analytics.

Weiting Zhang (Student Member, IEEE) is currently working toward the Ph.D. degree in communication and information systems with Beijing Jiaotong University, Beijing, China.

His current research interests include mobile edge computing, resource management, and application of artificial intelligence for industrial Internet of Things.

Conghao Zhou (Student Member, IEEE) received the B.S. degree from Northeastern University, Shenyang, China, in 2017, and the M.S. degree from the University of Illinois at Chicago, Chicago, IL, USA, in 2018, both in electrical engineering. He is currently working toward the Ph.D. degree in electrical and computer engineering with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada.

His current research interests include space–air–ground integration networks and machine learning in wireless networks.

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 current research interests include network resource management, wireless network security, Internet of Things, 5G and beyond, and vehicular ad-hoc and sensor networks.

Prof. Shen is a registered Professional Engineer in the State 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, James Evans Avant Garde Award in 2018 from the IEEE Vehicular Technology Society, Joseph LoCicero Award in 2015 and Education Award in 2017 from the IEEE Communications Society, and Technical Recognition Award from Wireless Communications Technical Committee (2019) and AHSN Technical Committee (2013). He is also the recipient of the Excellent Graduate Supervision Award in 2006 from the University of Waterloo and the Premiers Research Excellence Award in 2003 from the Province of Ontario, Canada. He was the Technical Program Committee Chair/Co-Chair for IEEE Global Communications Conference in 2016, IEEE International Conference on Computer Communications in 2014, IEEE Vehicular Technology Conference in 2010, IEEE Global Telecommunications Conference in 2007, and the Chair for the IEEE Communications Society Technical Committee on Wireless Communications. He is the elected IEEE Communications Society Vice-President for Technical & Educational Activities, Vice-President for Publications, Member-at-Large on the Board of Governors, Chair of the Distinguished Lecturer Selection Committee, and Member of IEEE ComSoc Fellow Selection Committee. He was/is the Editor-in-Chief of the IEEE INTERNET OF THINGS JOURNAL, IEEE NETWORK, *IET Communications*, and *Peer-to-Peer Networking and Applications*.