Collaborative AI-enabled Intelligent Partial Service Provisioning in Green Industrial Fog Networks

Abhishek Hazra, Mainak Adhikari, Tarachand Amgoth, and Satish Narayana Srirama

***Abstract*—With the evolutionary development of the latency-sensitive Industrial Internet of Things (IIoT) applications, delay restriction becomes a critical challenge, which can be resolved by distributing IIoT applications on nearby fog devices. Besides that, efficient service provisioning and energy optimization are confronting serious challenges with the ongoing expansion of large-scale IIoT applications. However, due to insufficient resource availability, a single fog device cannot execute large-scale applications completely. In such a scenario, a partial service provisioning strategy provides a promising outcome to enable the services on multiple fog devices or collaboration with cloud servers. By motivating this scenario, in this paper, we introduce a new Deep Reinforcement Learning (DRL)-enabled partial service provisioning strategy in the green industrial fog networks. With this strategy, multiple fog devices share the excessive workload of an application among themselves. To reflect this, a task partitioning policy is introduced to partition the requested applications into a set of independent or interdependent tasks. Further, we develop an intelligent partial service provisioning strategy to utilize maximum fog resources in the network. Experimental results express the significance of the proposed strategy over the traditional baseline algorithms in terms of energy consumption and latency up to 25% and 16%, respectively.**

***Index Terms*—IIoT application, industrial fog networks, partial service provisioning, DRL, energy optimization, latency.**

I. Introduction

RECENTLY, the industrial revolution has caught the attention of researchers and developers toward intelligent service provisioning of the large-scale Internet of Things (IoT) applications by integrating Artificial Intelligence (AI) technology in fog networks. Besides that, integrating intelligent sensors with modern industrial applications such as smart coal mining, connected vehicles, smart cities, and a sustainable grid can generate enormous data, attaining several challenges while transmitting and processing in the networks. With the advancement of fog computing, the industrial sector is revolutionizing to a large extent for processing the delay-sensitive applications at the edge of the network by reducing delay and energy usage [1]. At present, AI-enabled technology comes into play for processing large-scale industrial applications at the fog networks by distributing the applications efficiently with an intelligent service provisioning strategy. As a result, computation offloading and intelligent service provisioning aids in shortening execution time and extending battery life in IIoT devices to support the design of green industrial networks.

Thus, to utilize the computing resources efficiently of resource-constrained fog devices and meeting the objectives of the industrial applications, an intelligent service provisioning strategy needs to be designed. Nowadays, researchers prefer to design a smart service provisioning strategy with AI-enabled technology for distributing large-scale industrial applications over the fog networks [2]. Despite such technologies and recent advances, there are several significant challenges in distributing the IIoT applications in the fog networks due to the limited resource capacity of the fog devices. Although resource-rich cloud servers mitigate the storage and processing demand to some extent, latency and energy-associated issues for delay-sensitive industrial applications cannot be neglected [3]. In such a scenario, a partial service provisioning strategy with AI-enabled technology plays a critical role in distributing the industrial applications on local fog devices instead of transmitting them to the centralized cloud servers.

A. Related Work

In recent years, several service provisioning and offloading strategies have been designed in fog networks to efficiently process industrial applications while meeting various quality of service (QoS) constraints. For example, in [4], Hazra *et al.* have designed a service deployment strategy for utilizing fog resources efficiently in the network. Furthermore, in [3], Adhikari *et al.* have developed deadline-aware fair scheduling and computation offloading strategy for delay-sensitive applications. However, binary service provisioning strategies for large-scale industrial applications might not fully handle all the users' requests due to the limited resource capacity of the fog devices [5]. Thus, to satisfy the resource requirements of industrial applications, it is preferable to partition the application data into a set of independent sub-tasks and distribute them among multiple fog devices instead of transferring them to the centralized cloud servers [6]. Existing frameworks, for example, [7] and [8] analyze the partial data transmission of IoT applications but fail to consider the intelligent service provisioning mechanism in fog networks.

DRL is an emerging topic of interest from both the industry and academia due to its high applicability in the dynamic

Manuscript received January XX, 2021; revised May XX, 2021; January XX, 2021; accepted February XX, 2021. Date of publication February XX, 2021; date of current version July XX, 2021. (Corresponding author: Satish Narayana Srirama and Tarachand Amgoth)

A. Hazra and T. Amgoth are with the Indian Institute of Technology(Indian School of Mines) Dhanbad, Jharkhand, India, 826004. (e-mail: abhishekhazra.18DR0018@cse.iitism.ac.in, tarachand@iitism.ac.in).

M. Adhikari is with the Department of Computer Science and Information Technology, IIIT Lucknow, (e-mail: mainak.ism@gmail.com).

S. N. Srirama is with the School of Computer and Information Sciences, University of Hyderabad, India (e-mail: satish.srirama@uohyd.ac.in).

Digital Object Identifier 10.1109/JIOT.2020.3021XXX

industrial environment. Over the years, several attempts have been made to incorporate DRL technology and take advantage of self-learning capability in industrial networks. For example, in [9], Misra *et al.* designed a decentralized computation offloading framework using reinforcement learning for fog-enabled IoT applications. Similarly, in another work [9], Tiwari *et al.* designed an intelligent service provisioning strategy for fog-based IoT applications. It can be seen that the DRL strategy efficiently optimizes the energy and latency constraints of sensitive IIoT applications without the explicit human intervention [10]. However, these advancements can be further enhanced by allowing partial service provisioning into the industrial environment [11]. Specifically, the DRL-enabled intelligent service provisioning strategy further helps to boost up the distribution of excessive workload from one overloaded fog device to another underloaded computing device.

Current research initiatives focus on standalone IIoT-enabled fog/cloud service distribution frameworks, which are not beneficial for large-scale industrial data processing. However, digital industrial applications mostly rely on 4K or 8K video streaming, big data, and multimedia data streaming, etc. Therefore, a satisfactory industrial fog network with an effective service provisioning strategy should support excessive service requests of the large-scale IIoT applications [12]. Thus, the two key challenges of partial service provisioning in the industrial fog networks are a) *how to design a task partitioning strategy to decompose a large-scale industrial application into a set of independent sub-tasks?*, and b) *how to develop an intelligent partial service provisioning strategy that can optimize the energy-latency trade-off by distributing the partitioned tasks on the nearby fog devices?*

B. Motivation

In standard industrial networks, IIoT devices transfer data to the nearby fog devices through a wired/wireless channel for optimizing energy-latency constraints. Existing fog frameworks partially help to process IIoT applications by offering cloud functionality near the edge devices. However, there is still a high demand for faster processing of the large-scale industrial applications among the local fog devices [13]. In such a scenario, the use of an intelligent partial service provisioning strategy can be beneficial, which can partition the large-scale IIoT applications data into some independent set of sub-tasks and assign them to the distributed fog devices in the network while optimizing latency-energy burden and handle higher data traffic in the industrial network [14]. In addition, DRL is an essential self-learning tool for understanding complex network behaviors and taking perspective action by controlling the network parameters. DRL enabled systems efficiently maximize network run-time, predict the network downtime, self-correct the network events and reduce network congestion. Therefore, the critical research gap in the fog networks is to *design a DRL-enabled partial service provisioning strategy with an efficient task partitioning policy for large-scale industrial applications.*

C. Contribution

By motivating the above-mentioned challenges, in this work, we aim to jointly optimize the energy usage and latency for the large-scale industrial applications in the fog networks with an efficient task partitioning and intelligent service provisioning strategy. The foremost contributions of this work are summarized as follows.

- We develop an AI-enabled framework for supporting large-scale IIoT applications with an intelligent partial service provisioning strategy, where a set of distributed fog devices intelligently process the partitioned tasks at the edge of the network with minimum energy usage.
- We formulate the proposed service provisioning strategy as a nonlinear integer programming problem while the objective is to minimize the service delay and energy usage of industrial applications. Furthermore, a heuristic task partitioning strategy is designed to deal with higher processing delay by considering both CPU frequency and the data transmission power of fog devices.
- Considering high congestion in the industrial environment, we devise an intelligent partial service provisioning strategy with a DRL-based mode to distribute the sub-tasks intelligently and optimize the available fog resources within minimum energy-delay consumption on the local fog devices.
- Finally, the simulation results demonstrate the efficiency of the proposed service provisioning strategy over the existing baseline algorithms in terms of various performance matrices.

The remaining sections are organized as follows. Section II highlights the network model of industrial fog networks followed by the problem formulation of the work. The proposed intelligent service provisioning strategy is discussed in Section III. Then, the empirical analysis of the proposed strategy is elaborated in Section IV. Finally, Section V concludes the work followed by the future research plan.

II. Network Model

As per Fig. 1, the traditional fog framework suffers from a lack of data co-offloading among the distributed fog devices, which causes high processing delay. Partial service provisioning strategy resolves the issue of traditional fog framework by partitioning the real-time data into several independent tasks and distributing among the fog devices in the network with a controlling node, namely master fog device. Here, we consider a fog framework with a set of IIoT devices $\mathcal{M} = \{1, 2, \ldots, M\}$ and a set of fog devices $\mathcal{F} = \{1, 2, \ldots, F\}$, are randomly distributed over the network as presented in Fig. 1. The master fog device belongs to set $\mathcal{F}$ that distributes the partitioned tasks among the nearby fog devices for further processing. Thus the main responsibility of the master fog device is to partition the incoming data and distribute them among the standard fog devices in the network, whereas the main purpose of the standard fog devices is to process the partitioned data independently and send the results back to the master fog device for further decision making.

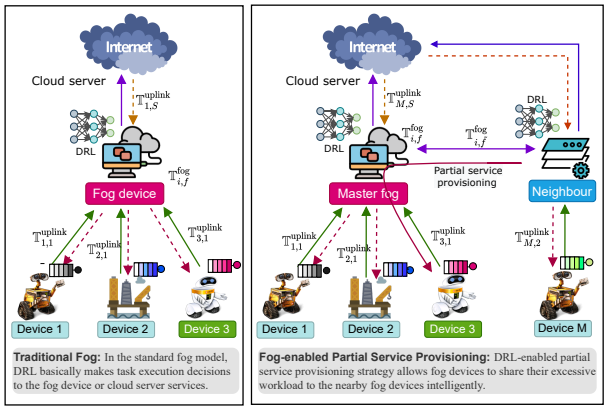

Fig. 1. Differentiation between traditional fog framework and fog-enabled partial service provisioning framework in the industrial networks.

Additionally, a set of $\mathcal{S} = \{1, 2, \ldots, S\}$ cloud servers are deployed to reduce the excessive workload of the local fog devices in the industrial network. Further, consider a quasi-static network, where IIoT devices $\mathcal{M}$ request computing services $\mathcal{I} = \{1, 2, \ldots, I\}$ on various computing devices $\mathcal{CD} = \{\mathcal{F} \cup \mathcal{S}\}$ for further processing. Let $\mathcal{D}_i$ amount of CPU cycle is needed to execute $\mathcal{B}_i$ bits of task on a computing device $\mathcal{CD}$. We consider an IIoT device $m \in \mathcal{M}$, which transmits the data to the master fog device $f \in \mathcal{F}$ through a gateway device and requests services in the fog network with minimum latency and energy usage. We consider $\gamma(i, m) = \{0, 1\}$ as a binary decision variable, where $\gamma(i, m) = 0$ represents that services are executed in IIoT devices itself, otherwise offload to the fog master for further processing [15]. Further, we consider $I \times J$ as the sub-task deployment matrix Γ for master fog device $f \in \mathcal{F}$, where each entry $\Gamma(i, j) \in [0, 1]$, $\forall j \in \{\mathcal{F}, \mathcal{S}\}$ represents the amount of task requested for partial computation. To make the fog network more realistic, we consider IIoT devices $\mathcal{M}$ can request multiple services $\mathcal{I}$ from fog devices $\mathcal{F}$ [16]. Subsequently, fog devices $\mathcal{F}$ can receive requests from multiple IIoT devices $\mathcal{M}$ and other working fog devices $\widetilde{\mathcal{F}}$.

A. Local Execution

In an industrial environment, an IIoT device $m \in \mathcal{M}$ often executes a small amount of service request $i \in \mathcal{I}$ locally. Denote $\mathscr{G}_m^{\text{CPU}}$ be the CPU frequency of the IIoT devices $m \in \mathcal{M}$. Then the processing delay $\mathbb{T}_{i,m}^{\text{IoT}}$ and energy consumption $\mathbb{E}_{i,m}^{\text{IoT}}$ to execute ith service request on the mth IIoT device can be represented as follows.

$$\mathbb{T}_{i,m}^{\text{IoT}} = \frac{\gamma(i,m)\mathcal{B}_i\mathcal{D}_i}{\mathscr{G}_m^{\text{CPU}}} \tag{1}$$

$$\mathbb{E}_{i,m}^{\text{IoT}} = \gamma(i,m)\mathcal{B}_i\mathcal{D}_i k(\mathscr{G}_m^{\text{CPU}})^2 \tag{2}$$

Where $i \in \mathcal{I}$ and $m \in \mathcal{M}$. For IIoT devices, we consider the energy consumption model as $k(\mathscr{G}_m^{\text{CPU}})^2$, presented in [17], where k is the chip coefficient of the IIoT device. Let $\mathcal{B}_{mf}^{\text{trans}}$ and $\mathcal{N}_0$ are the data transmission bandwidth and additive gaussian noise of the fog network, respectively [3]. Thus the uplink transmission rate $\mathbb{R}_{m,f}^{\text{trans}}$ of IIoT device $m \in \mathcal{M}$ to master fog device $f \in \mathcal{F}$ can be defined as $\mathbb{R}_{m,f}^{\text{trans}} = \mathcal{B}_{mf}^{\text{trans}} log_2\left(\frac{1+\mathscr{P}_m^{\text{power}}\mathcal{G}_{mf}}{\mathcal{N}_0}\right)$, where $\mathscr{P}_m^{\text{power}}$ represents the expected transmission power of IIoT device $m \in \mathcal{M}$ in the industrial networks.

B. Service Deployment on Master Fog Device

Initially, IIoT devices $\mathcal{M}$ try to execute the tasks locally, however, limited processing capacity creates execution barrier to IIoT devices. The uplink transmission time $\mathbb{T}_{m,f}^{\text{uplink}}$ for sending a service request $i \in \mathcal{I}$ from IIoT device $m \in \mathcal{M}$ to master fog device $f \in \mathcal{F}$ can simply be represented as $\mathbb{T}_{m,f}^{\text{uplink}} = \big(1 - \gamma(i,m)\big)\mathcal{B}_i/\mathbb{R}_{m,f}^{\text{trans}}$. As the task size $\mathcal{B}_i$ increases, the required transmission energy $\mathbb{E}_{m,f}^{\text{uplink}}$ also increases and is represented as $\mathbb{E}_{m,f}^{\text{uplink}} = \big(1 - \gamma(i,m)\big)\mathcal{B}_i\mathscr{P}_m^{\text{power}}/\mathbb{R}_{m,f}^{\text{trans}}$, where $m \in \mathcal{M}$ and $i \in \mathcal{I}$. Moreover, we can simply define the time required to process ith service requests $\forall i \in \mathcal{I}$ i.e., execution delay $\mathbb{T}_{i,f}^{\text{fog}}$ on master fog device $f \in \mathcal{F}$ as follows.

$$\mathbb{T}_{i,f}^{\text{fog}} = \frac{\big(1 - \gamma(i,m)\big)\mathcal{B}_i\mathcal{D}_i}{\mathscr{G}_f^{\text{CPU}}} \tag{3}$$

Where $\mathscr{G}_f^{\text{CPU}}$ be the achievable CPU frequency of the master fog device f, $\forall f \in \mathcal{F}$. Now, if we consider the power consumption model for the IIoT device as $k(\mathscr{G}_f^{\text{CPU}})^2$, then we can calculate the total energy consumption $\mathbb{E}_{i,f}^{\text{fog}}$ to process ith service request on fth master fog device as follows.

$$\mathbb{E}_{i,f}^{\text{fog}} = (1 - \gamma(i,m))\mathcal{B}_i\mathcal{D}_i k(\mathscr{G}_f^{\text{CPU}})^2 \tag{4}$$

In this model, we exclude delay and energy burden for obtaining the output results from master fog devices $\mathcal{F}$. This is due to the fact that downloading time and energy consumption from fog devices $\mathcal{F}$ are as low as 1/30 and 5.5 times respectively, than the uplink data transmission [17].

C. Service Deployment on Nearby Fog Device

Ideally requesting a computation service $i \in \mathcal{I}$ and transfer a portion of generated tasks $\Gamma(i, \widetilde{f})\mathcal{B}_i$ to the nearby active fog device $\widetilde{f} \in \widetilde{\mathcal{F}}$ can be a suitable option for reducing overall delay. Similarly, we can derive the data uploading time and energy consumption to nearby fog device as $\mathbb{T}_{m,\widetilde{f}}^{\text{uplink}} = \Gamma(i, \widetilde{f})\mathcal{B}_i/\mathbb{R}_{m,\widetilde{f}}^{\text{trans}}$ and $\mathbb{E}_{m,\widetilde{f}}^{\text{uplink}} = \mathscr{P}_f^{\text{power}}\Gamma(i, \widetilde{f})\mathcal{B}_i/\mathbb{R}_{m,\widetilde{f}}^{\text{trans}}$, respectively. Let $\mathscr{G}_{\widetilde{f}}^{\text{CPU}}$ be the maximum CPU frequency (in cycles/second) of the neighbour fog devices $\widetilde{\mathcal{F}}$, thus the execution delay $\mathbb{T}_{i,\widetilde{f}}^{\text{fog}}$ to satisfy service requests $i \in \mathcal{I}$ on the fog device $\widetilde{f} \in \widetilde{\mathcal{F}}$ can be represented as follows.

$$\mathbb{T}_{i,\widetilde{f}}^{\text{fog}} = \frac{\Gamma(i, \widetilde{f})\mathcal{B}_i\mathcal{D}_i}{\mathscr{G}_{\widetilde{f}}^{\text{CPU}}} \tag{5}$$

Thus the energy consumption rate $\mathbb{E}_{i,\widetilde{f}}^{\text{fog}}$ to process ith service request with κ chip coefficient on a nearby active fog device $\widetilde{f} \in \widetilde{\mathcal{F}}$ can be expressed as follows.

$$\mathbb{E}_{i,\widetilde{f}}^{\text{fog}} = \Gamma(i, \widetilde{f})\mathcal{B}_i\mathcal{D}_i k(\mathscr{G}_{\widetilde{f}}^{\text{CPU}})^2 \tag{6}$$

D. Service Deployment on Cloud Server

Let $\mathbb{R}^{\text{trans}}_{m,s}$ be the expected transmission rate of fog devices $\mathcal{F}$, then the uplink transmission delay $\mathbb{T}^{\text{uplink}}_{m,s}$ and energy consumption $\mathbb{E}^{\text{uplink}}_{m,s}$ for deploying a service request $i \in \mathcal{I}$ from master fog device $f \in \mathcal{F}$ to cloud server $s \in \mathcal{S}$ can be represented as $\mathbb{T}^{\text{uplink}}_{m,s} = \Gamma(i,s)\mathcal{B}_i/\mathbb{R}^{\text{trans}}_{m,s}$ and $\mathbb{E}^{\text{uplink}}_{m,s} = \mathscr{P}^{\text{power}}_f \Gamma(i,s)\mathcal{B}_i/\mathbb{R}^{\text{trans}}_{m,s}$, respectively, where $\mathscr{P}^{\text{power}}_f$ be the transmission power of master fog device $f \in \mathcal{F}$. Similarly, we can derive processing delay $\mathbb{T}^{\text{cloud}}_{i,s}$ and energy consumption $\mathbb{E}^{\text{cloud}}_{i,s}$ on the cloud server $s \in \mathcal{S}$, defined as follows.

$$\mathbb{T}^{\text{cloud}}_{i,s} = \frac{\Gamma(i,s)\mathcal{B}_i\mathcal{D}_i}{\mathscr{G}^{\text{CPU}}_s} \tag{7}$$

$$\mathbb{E}^{\text{cloud}}_{i,s} = \Gamma(i,s)\mathcal{B}_i\mathcal{D}_i k(\mathscr{G}^{\text{CPU}}_s)^2 \tag{8}$$

As the cloud servers $\mathcal{S}$ are capable of faster data transmission, we simply omit the result fetching delay from cloud servers [11]. In summary, when an IIoT device $m \in \mathcal{M}$ requests a computation service $i \in \mathcal{I}$ from remote processing devices j, $\forall j \in \{\mathcal{F}, \widetilde{\mathcal{F}}, \mathcal{S}\}$, it may include transmission and execution time and energy on various computing devices i.e., $\mathbb{T}^{\text{remote}}_{i,j} = \mathbb{T}^{\text{uplink}}_{m,f} + \mathbb{T}^{\text{fog}}_{i,f} + \mathbb{T}^{\text{uplink}}_{m,\widetilde{f}} + \mathbb{T}^{\text{fog}}_{i,\widetilde{f}} + \mathbb{T}^{\text{uplink}}_{m,s} + \mathbb{T}^{\text{cloud}}_{i,s}$ and $\mathbb{E}^{\text{remote}}_{i,j} = \mathbb{E}^{\text{uplink}}_{m,f} + \mathbb{E}^{\text{fog}}_{i,f} + \mathbb{E}^{\text{uplink}}_{m,\widetilde{f}} + \mathbb{E}^{\text{fog}}_{i,\widetilde{f}} + \mathbb{E}^{\text{uplink}}_{m,s} + \mathbb{E}^{\text{cloud}}_{i,s}$.

E. Problem Formulation

In this section, we formulate the proposed energy-optimized partial service provisioning strategy in terms of minimizing processing delay ($\mathbb{T}^{\text{total}}_i$) and energy consumption ($\mathbb{E}^{\text{total}}_i$) *i.e.,* $\mathcal{J} = \sum_{i\in I}\left(\mathbb{E}^{\text{total}}_i + \mathbb{T}^{\text{total}}_i\right)$ under the condition of local execution power $g_m = (\mathscr{G}^{\text{CPU}}_m)_{\forall m}$, remote execution speed $g_j = (\mathscr{G}^{\text{CPU}}_j)_{\forall j}$, uplink transmission power $p = (\mathscr{P}^{\text{power}}_m)_{\forall m}$ and offloading ratio $\varsigma = (\Gamma(i,j))_{\forall(i,j)}$, $\forall i \in \mathcal{I}, \forall j \in \{\mathcal{F}, \widetilde{\mathcal{F}}, \mathcal{S}\}$. Where $\mathbb{T}^{\text{total}}_i = \mathbb{T}^{\text{IoT}}_{i,m} + \mathbb{T}^{\text{remote}}_{i,j}$ and $\mathbb{E}^{\text{total}}_i = \mathbb{E}^{\text{IoT}}_{i,m} + \mathbb{E}^{\text{remote}}_{i,j}$. Thus without the loss of generality, the optimization problem is expressed as follows.

$$\underset{\varsigma, p, g_m, g_j}{\text{minimize}} \quad w_i\mathcal{J}\left(\Gamma(i,j), \mathscr{G}^{\text{CPU}}_m, \mathscr{G}^{\text{CPU}}_j, \mathscr{P}^{\text{power}}_m\right) \tag{9a}$$

$$\text{subject to} \quad \sum_{i\in I}\sum_{j\in\mathcal{CD}} \Gamma(i,j)\mathcal{B}_i\mathcal{D}_i \le \mathbb{C}^{\max}, \tag{9b}$$

$$\sum_{i\in I} \mathbb{E}^{\text{local}}_i \le \mathbb{E}^{\max}, \quad \forall i \in \mathcal{I}, \tag{9c}$$

$$\sum_{i\in I} \mathbb{T}^{\text{local}}_i \le \mathbb{T}^{\max}, \quad \forall i \in \mathcal{I}, \tag{9d}$$

The above optimization problem represents the joint optimization of energy and delay with an associated weight w_i, where $w_i \in [0,1]$. Constraints (9b), (9c) and (9d) represent that the total processing capacity, energy consumption and delay, which must be bounded by threshold $\mathbb{C}^{\max}$, $\mathbb{E}^{\max}$ and $\mathbb{T}^{\max}$ respectively.

III. AI-enabled Service Provisioning Strategy

This section presents a brief overview of our intelligent service provisioning strategy while minimizing energy and latency for each service request. In order to introduce the proposed scheme more concisely, we further divide the partial offloading strategy into two steps, namely task partitioning and intelligent service provisioning. The task partitioning strategy divides the large-scale industrial data into a sequence of sub-tasks, whereas the intelligent service provisioning strategy uses the DRL technique for efficient workload distribution among the computing devices, which are discussed below.

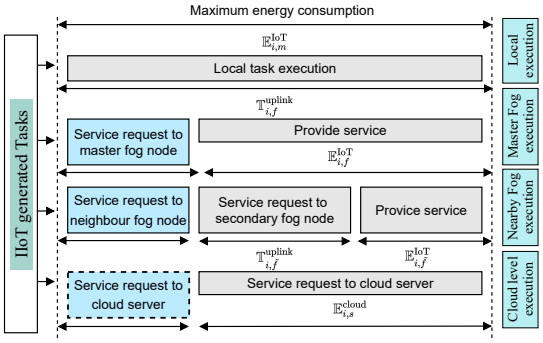

Fig. 2. Partial service deployment among the computing devices.

A. Task Partitioning

An essential factor for partial service provisioning is to ascertain the suitable fraction of service request i distributed by a master fog device, as it concerns both time consumption for fog execution $\mathbb{T}^{\text{fog}}_{i,f}$, data transmission $\mathbb{T}^{\text{uplink}}_{i,j}$, nearby fog processing $\mathbb{T}^{\text{fog}}_{i,\widetilde{f}}$, and energy usage for task execution $\mathbb{E}^{\text{fog}}_{m,f}$ and partial service deployment time $\mathbb{T}^{\text{uplink}}_{i,j}$. The task partitioning strategy mainly depends on maximum energy consumption $\mathbb{E}^{\max}$ and maximum tolerable delay $\mathbb{T}^{\max}$, as presented in constraint (9c) and (9d), respectively. Therefore, the proposed task partitioning strategy is motivated by two premises (a) the local and remote execution should be parallel to complete the execution of the task within the deadline $\mathbb{T}^{\max}$ [4], and (b) the expected energy utilization for data transmission and remote execution should be bounded by maximum energy consumption $\mathbb{E}^{\max}$ [18], as illustrated in Fig. 2.

It can be recognized that the smallest latency can be achieved if sub-tasks execution completes in parallel on different computing devices. Then the upper bound of total latency on the master fog device $\mathbb{T}^{\text{fog}}_{i,f}, \forall f \in \mathcal{F}$ can be derived from constraint $\mathbb{T}^{\max}$, defined as follows.

$$\frac{\big(1-\gamma(i,m)\big)\mathcal{B}_i\mathcal{D}_i}{\mathscr{G}^{\text{CPU}}_f} \le \mathbb{T}^{\max} \tag{10}$$

Now if a task uploaded to master fog device, then from (10), we can derive the lower bound of $\gamma(i,m)$ as $\big(1-\gamma(i,m)\big)\mathcal{B}_i\mathcal{D}_i/\mathscr{G}^{\text{CPU}}_f = \mathbb{T}^{\max}$, by replacing $\big(1-\gamma(i,m)\big)$ as $\Gamma^{min}_{i,f}$ we can define $\Gamma^{min}_{i,f}\mathcal{B}_i\mathcal{D}_i/\mathscr{G}^{\text{CPU}}_f = \mathbb{T}^{\max}$, which can be simply represented as follows.

$$\Gamma^{min}_{i,f} = \frac{\mathbb{T}^{\max}\ \mathscr{G}^{\text{CPU}}_f}{\mathcal{B}_i\mathcal{D}_i} \tag{11}$$

Similarly, the upper bound of $\mathbb{E}^{\max}$ for partial service provisioning of tasks to the remote computing device is obtained from $\mathbb{E}^{\text{fog}}_{i,f} + \mathbb{E}^{\text{fog}}_{i,\widetilde{f}} + \mathbb{E}^{\text{cloud}}_{i,s}$, where.

$$\mathbb{E}^{\text{fog}}_{i,f} + \mathbb{E}^{\text{fog}}_{i,\widetilde{f}} + \mathbb{E}^{\text{cloud}}_{i,s} \leq \mathbb{E}^{\max} \tag{12}$$

If $\gamma(i,m) = 0$, tasks are transferred to master fog device $f \in \mathcal{F}$ and $\Gamma(i,f)$ amount of task executed. Thus we can rewrite equation (4) by replacing $(1-\gamma(i,m))$ as $\Gamma(i,f)$, defined as follows.

$$\mathbb{E}^{\text{fog}}_{i,f} = \Gamma(i,f)\mathcal{B}_i\mathcal{D}_i k(\mathscr{G}^{\text{CPU}}_f)^2 \tag{13}$$

Moreover from equation (12), we can derive the upper bound of partial service provisioning $\Gamma(i,j)$ as $\Gamma^{max}_{i,j}$, which can be derived separately for different computing devices j, $\forall j \in \{\mathcal{F}, \widetilde{f}, \mathcal{S}\}$, defined as follows.

$$\begin{aligned}\Gamma^{max}_{i,f} = \frac{1}{\mathcal{B}_i\mathcal{D}_i k(\mathscr{G}^{\text{CPU}}_f)^2}\Big(&\mathbb{E}^{\max} - \Gamma(i,\widetilde{f})\mathcal{B}_i\mathcal{D}_i k \\ &\big(\mathscr{G}^{\text{CPU}}_{\widetilde{f}}\big)^2 - \Gamma(i,s)\mathcal{B}_i\mathcal{D}_i k\big(\mathscr{G}^{\text{CPU}}_s\big)^2\Big)\end{aligned} \tag{14}$$

$$\begin{aligned}\Gamma^{max}_{i,\widetilde{f}} = \frac{1}{\mathcal{B}_i\mathcal{D}_i k\big(\mathscr{G}^{\text{CPU}}_{\widetilde{f}}\big)^2}\Big(&\mathbb{E}^{\max} - \Gamma(i,f)\mathcal{B}_i\mathcal{D}_i k \\ &(\mathscr{G}^{\text{CPU}}_f)^2 - \Gamma(i,s)\mathcal{B}_i\mathcal{D}_i k\big(\mathscr{G}^{\text{CPU}}_s\big)^2\Big)\end{aligned} \tag{15}$$

$$\begin{aligned}\Gamma^{max}_{i,s} = \frac{1}{\mathcal{B}_i\mathcal{D}_i k\big(\mathscr{G}^{\text{CPU}}_s\big)^2}\Big(&\mathbb{E}^{\max} - \Gamma(i,f)\mathcal{B}_i D_i k \\ &(\mathscr{G}^{\text{CPU}}_f)^2 - \Gamma(i,\widetilde{f})\mathcal{B}_i\mathcal{D}_i k\big(\mathscr{G}^{\text{CPU}}_{\widetilde{f}}\big)^2\Big)\end{aligned} \tag{16}$$

Theorem 1 : *The optimal amount of service requests $\mathcal{B}^{optimal}_i$, $\forall i \in \mathcal{I}$ to be transferred to a nearby active fog device $\widetilde{f} \in \widetilde{\mathcal{F}}$* is $\mathcal{B}^{\text{optimal}}_i = \big(\mathcal{D}_i\mathcal{B}_i/\mathcal{D}_i + \mathscr{G}^{\text{CPU}}_f\beta_{i\widetilde{f}}\big)$.

Proof : When a master fog device $f \in \mathcal{F}$ wants to partially provide service request $i \in \mathcal{I}$ to the nearby active fog device $\widetilde{f} \in \widetilde{\mathcal{F}}$, it tries to transfer partial data size of a sub-task $\mathcal{B}^{\text{optimal}}_i$ to minimize delay and increase execution speedup such that the processing time and remote execution time at the active fog device are equal [4], i.e., $\mathbb{T}^{\text{fog}}_{i,f} = \mathbb{T}^{\text{uplink}}_{m,\widetilde{f}} + \mathbb{T}^{\text{fog}}_{i,\widetilde{f}}$. As the uploading delay is an important concern to calculate total delay, we can define the data size of optimal task based on the execution time of tasks in the master fog device as $\mathbb{T}^{\text{fog}}_{i,j} = (1-\gamma(i,m))\mathcal{B}_i\mathcal{D}_i/\mathscr{G}^{\text{CPU}}_f$. Let $\mathcal{B}^{\text{optimal}}_i$ be the optimal amount of task offloaded to the neighbour fog device $\widetilde{f} \in \widetilde{\mathcal{F}}$, where $0 \leq \mathcal{B}^{\text{optimal}}_i \leq \mathcal{B}_i$. Then replacing $\mathcal{B}^{\text{optimal}}_i$ in $\mathbb{T}^{\text{fog}}_{i,f} = \mathbb{T}^{\text{uplink}}_{m,\widetilde{f}} + \mathbb{T}^{\text{fog}}_{i,\widetilde{f}}$, we obtain-

$$\frac{\mathcal{B}_i\mathcal{D}_i}{\mathscr{G}^{\text{CPU}}_f} = \beta_{i\widetilde{f}}\mathcal{B}^{\text{optimal}}_i \tag{17}$$

Where, $\beta_{i\widetilde{f}} = \frac{\Gamma(i,\widetilde{f})}{R^{\text{trans}}_{m,\widetilde{f}}} + \frac{\Gamma(i,\widetilde{f})\mathcal{D}_i}{\mathscr{G}^{\text{CPU}}_{\widetilde{f}}}$. By adding the amount of executable data in IIoT/end-device yields-

$$\mathcal{D}_i(\mathcal{B}_i - \mathcal{B}^{\text{optimal}}_i)/\mathscr{G}^{\text{CPU}}_i = \beta_{i\widetilde{f}}\mathcal{B}^{\text{optimal}}_i \tag{18}$$

By replacing the amount of executable data in master fog device $f \in \mathcal{F}$ and neighbour fog device $\widetilde{f} \in \widetilde{\mathcal{F}}$ with optimal offloaded data yields-

$$\frac{\mathcal{D}_i\mathcal{B}_i}{\mathscr{G}^{\text{CPU}}_f} - \frac{\mathcal{D}_i\mathcal{B}^{\text{optimal}}_i}{\mathscr{G}^{\text{CPU}}_f} = \beta_{i\widetilde{f}}\mathcal{B}^{\text{optimal}}_i \tag{19}$$

Algorithm 1: *Task Partitioning* algorithm

INPUT: $\mathcal{I}$: Set of service requests, $\mathscr{G}^{\text{CPU}}$: Computation frequency, $\mathcal{B}_i$: Size of the service request.
OUTPUT: $\mathcal{I}^*$: Set of new service requests
Initialize $\mathbb{C}^{\max}$, $\mathscr{G}^{\text{CPU}}_f$,$\mathscr{G}^{\text{CPU}}_{\widetilde{f}}$, and $\mathscr{G}^{\text{CPU}}_s$
for i = 1 to I **do**
  if $\sum_{i\in I}\Gamma(i,j)\mathcal{B}_i\mathcal{D}_i \leq \mathscr{G}^{\text{CPU}}_f$ **then**
    Do not split the service requests $\mathcal{I}$
    Execute tasks on master fog device $f \in \mathcal{F}$
  end if
  Observe $\mathscr{G}^{\text{CPU}}_f$, $\mathscr{G}^{\text{CPU}}_{\widetilde{f}}$ and $\mathscr{G}^{\text{CPU}}_s$
  if $\sum_{i\in I}\Gamma(i,j)\mathcal{B}_i\mathcal{D}_i > \mathscr{G}^{\text{CPU}}_f$ **then**
    if $\mathcal{B}_i \geq \mathbb{C}^{\max}$ **then**
      Partition into random number of sub-tasks with $\mathcal{B}_i < \mathscr{G}^{\text{CPU}}_f$, $\mathcal{B}_i < \mathscr{G}^{\text{CPU}}_{\widetilde{f}}$ and $\mathcal{B}_i < \mathscr{G}^{\text{CPU}}_s$
    end if
    else Do not split the service request i, $\forall i \in \mathcal{I}$
  end if
end for
Return set of service requests $\mathcal{I}^* \leftarrow \mathcal{I}$

$$\mathcal{D}_i\mathcal{B}_i = \mathcal{D}_i\mathcal{B}^{\text{optimal}}_i + \beta_{i\widetilde{f}}\mathcal{B}^{\text{optimal}}_i\mathscr{G}^{\text{CPU}}_f \tag{20}$$

Rearranging both the side yields of the above equation-

$$\mathcal{B}^{\text{optimal}}_i = \frac{\mathcal{D}_i\mathcal{B}_i}{\mathcal{D}_i + \mathscr{G}^{\text{CPU}}_f\beta_{i\widetilde{f}}} \tag{21}$$

Thus, by uploading $\mathcal{B}^{\text{optimal}}_i$ amount of service request $i \in \mathcal{I}$ to the neighbour fog device $\widetilde{f} \in \widetilde{\mathcal{F}}$, master fog devices $\mathcal{F}$ efficiently minimize overall end-to-end processing delay in the industrial fog networks. This completes the proof.

At this stage, the master fog device $f \in \mathcal{F}$ checks whether $\mathscr{G}^{\text{CPU}}_f$ is suitable to process all the service requests $\mathcal{I}$. If service requests exceed capacity, then the master fog device f selects a set of service demands and randomly partition them into an appropriate member of independent sub tasks $I_k = \{\mathcal{I}^k_1, \mathcal{I}^k_2, \ldots, \mathcal{I}^k_x\}$, where $k \in \mathcal{I}$ and $x \in \mathcal{CD}$. Then $\mathcal{I}^k_1, \mathcal{I}^k_2, \ldots, \mathcal{I}^k_x$ satisfy the followings:

$$\mathcal{I}^k_1 \cup \mathcal{I}^k_2 \cup \cdots \cup \mathcal{I}^k_x = I_k \tag{22}$$

$$\mathcal{I}^k_i \cap \mathcal{I}^k_j = \phi, \quad (1 \leq i,j \leq x \text{ and } i \neq j) \tag{23}$$

The steps of the task partitioning process are presented in *Algorithm* 1. Next, the set of sub-tasks are put into the DRL-based online learning algorithm for distributing the service requests on various computing devices.

B. Intelligent Service Provisioning

In this section, we have presented our proposed DQN based service provisioning strategy, which aims to select optimal action and maximize reward to achieve our objectives in the industrial fog networks. Initially, we discuss Reinforcement Learning (RL) as presented in [19], then followed by the motivation of the DQN technique, and finally, we briefly explain the proposed intelligent service provisioning strategy.

1) Reinforcement Learning (RL): RL technique regards as the Markov decision-making process, as the underlying RL environment follows the Markovian property [20]. Let s, a, and $r(s_t, a)$ are the state, action, and reward of an environment, where an agent takes action a based on the state s and obtains a reward r from the environment with a policy π. Thus, the total reward, popularly known as the Q function for a set of state-action pairs, can be expressed as follows.

$$Q^{\pi}(s_t, a) = (1-\alpha)\ Q(s_t, a) + \alpha\big(r(s_t, a) + \gamma \max_{a'} Q'(s_{t+1}, a'; \pi')\big) \tag{24}$$

Where α and γ are the learning rate and decaying factor of the Q function, respectively. To represent the proposed strategy, we consider state $s = \{\Gamma(i,j), \mathscr{G}_f^{\text{CPU}}\}$ including fraction of task partition $\Gamma(i,j)$ and computation frequency of various computing devices $\mathscr{G}_j^{\text{CPU}}$. Action a denotes the identical move/change of various states s. It can be noted that the complexity of the Q function increases exponentially with the increase in the dimension of state-action space (s_t, a). Thus, a promising solution is to approximate Q-value with a flexible number of parameters.

2) Introduction to DQN: Q-learning is a model-free learning algorithm that teaches an agent to learn the quality of behavior under unknown environmental conditions. In Q-learning, state-action pairs (s_t, a) are stored in the Q-table. One significant limitation of the Q-learning algorithm is that the conventional Q-learning algorithm becomes computationally expensive when the entries of state-action pairs (s_t, a) in the Q-table becomes large. Although, this challenge can be handled by predicting the Q-values using replay memory as illustrated in Fig 3, which leads to the key essence of the DQN. A batch of previous experience/records is typically used for training the Q-networks, and replay memory contains each record (s_t, a_t, r_t, s_{t+1}) in every step while learning DQN. One can feed a large number of training samples for predicting state-action pairs (s_t, a) accurately, which leads to $Q^{old}(s_t, a, w) \approx Q^*(s_t, a)$ [20]. In DQN, an analysis network is used for prediction, and the resultant network is used for labeling the network, which is defined as follows.

$$Q^{lab}(s_t, a) = r(s_t, a) + \gamma \max_{a'} Q^{out}(s_{t+1}, a', w') \tag{25}$$

The gradient-based algorithm is used to minimize the loss $\mathcal{L}(Q(w))$ so that weight w is modified to reduce the difference between the expected value and the objective values.

$$\mathcal{L}(Q(w)) = \big(Q^{lab}(s_t, a) - Q^{old}(s_t, a, w)\big)^2 \tag{26}$$

3) Service Provisioning Strategy: The proposed service provisioning strategy mainly considers a set of computing devices $\mathcal{CD}$, where fog devices $\mathcal{F}$ have the highest responsibility to satisfy delay-deadline constraints as compared with cloud servers $\mathcal{S}$. In this section, we firstly transform our objective function for the convenience of partial service provisioning, followed by the benefit of using the DRL technique for intelligent service provisioning.

Maximum CPU constraint $\mathbb{C}^{\max}$ on the computing devices $\mathcal{CD}$ considered as the strict deadline constraints. However, in a complex industrial application scenario, there might not be any possible set of variables that takes decisions while satisfying both $\mathbb{C}^{\max}$ and $\mathbb{T}^{\max}$ constraints. Thus, we consider the delay constraint $\mathbb{T}^{\max}$ as a soft deadline constraint and introduce a penalty function $P(\mathbb{T})$ to handle the deadline constraint.

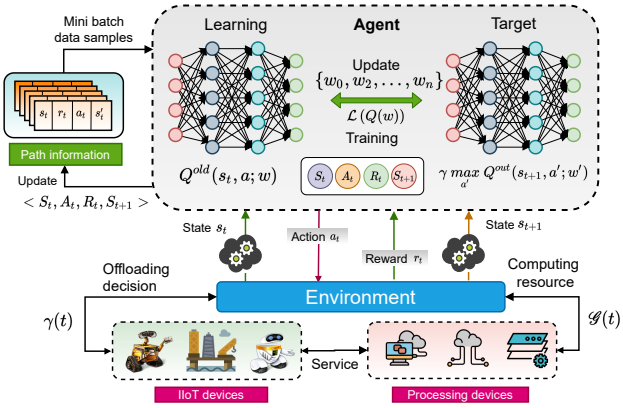

Fig. 3. Illustration of DRL-based service provisioning strategy.

$$P(\mathbb{T}) = \sum_{i \in \mathcal{I}} \max\Big[\mathbb{T}_i^{\text{total}} - \mathbb{T}^{\max}, 0\Big] \tag{27}$$

Penalty functions $P(.)$ intend to transform constrained optimization problems into unconstrained problems by adding a substitute penalty function for breaking the constraint. Thus, the objective function $\mathcal{J}(.)$ for the master fog device f can be transformed as follows.

$$\underset{\gamma, g_j}{\text{minimize}} \quad w_i \mathcal{J}\big(\Gamma(i,j), \mathscr{G}_j^{\text{CPU}}\big) + w^{\mathbb{T}} P(\mathbb{T}) \tag{28a}$$

$$\text{subject to} \quad \sum_{i \in I} \mathbb{E}_i^{\text{local}} \le \mathbb{E}^{\max},\ \forall i \in \mathcal{I}, \tag{28b}$$

$$\sum_{i \in I} \Gamma(i,j) \mathcal{B}_i \mathcal{D}_i \le \mathbb{C}^{\max}, \tag{28c}$$

$$\gamma(i,j) \in \{0\} \cup \mathcal{CD},\ \forall i \in \mathcal{I}, j \in \mathcal{CD}, \tag{28d}$$

Where $w^{\mathbb{T}}$ is the weight used for controlling the penalty function. Based on the above formulation, we can define the system state s_t, action a and reward r as follows.

1) State: To interpret the objective function, we define the system states s_t at the arbitrary index of t, which is defined as follows.

$$\begin{aligned} s_t &= \big\{\gamma(t),\ \mathscr{G}(t)\big\} \\ &= \big\{\gamma_1(t), \gamma_2(t), \ldots, \gamma_j(t), \mathscr{G}_1(t), \mathscr{G}_2(t), \ldots, \mathscr{G}_j(t)\big\} \end{aligned} \tag{29}$$

In particular, the system state is a $1 \times 2N$ dimensional vector, which includes all the partial offloading decision $\gamma(i,j)(t) \in \{0\} \cup \mathcal{CD}$ and computing resource $\mathscr{G}_j(t) \in [0, \mathscr{G}]$ for all computing devices.

2) Action: The action a is used to signify how an agent move between two distinct system states s_t and s_{t+1}. Precisely, we indicate $\gamma'(i,j)(t) = \gamma(i,j)(t+1) \in \{0\} \cup \mathcal{CD}$ as the new service provisioning decision for a master fog device $f \in \mathcal{F}$ and $\Delta\mathscr{G}_n(t) = \mathscr{G}_n(t+1) - \mathscr{G}_n(t)$ as the variation in the allotted service resources by the remote computing devices j. The system actions are defined as follows.

$$a_t = \big\{\gamma'_1(t), \ldots, \gamma'_j(t), \Delta\mathscr{G}_1(t), \ldots, \Delta\mathscr{G}_j(t)\big\} \tag{30}$$

Algorithm 2: *Service Provisioning* algorithm

INPUT: $\mathcal{I}^*$: Service requests, $\mathcal{D}_i$: Input data size,
OUTPUT: $s^*(t)$: Optimal decision,
Initialize state-action pair $Q(s,a)$
Initialize Q network and replay memory
for each episode $e = 1, 2, \ldots, E$ **do**
  if e mod 100 == 0 **then**
    Change initial state
    $s_t =$ argmin $\mathcal{J}_{s(t)}(t)$
  end if
  Observe s_{t+1} and r_t from environment
  Store (s_t, a_t, r_t, s_{t+1}) into replay memory
  Extract mini-batch samples
  Calculate $Q^{lab}(s_t,a) = r(s_t,a) + \gamma \max_{a'} Q^{out}(s_{t+1},a,w)$
  Calculate Loss $= \Big(Q^{lab}(s_t,a) - Q^{old}(s_t,a,w)\Big)^2$
  Update network parameters w
  Update $\mathcal{J}^*_{s(t)}(t) = \mathcal{J}_{s(t)}(t)$
  Take optimal decision $s^*(t)$
end for

3) Reward: For a given state s_t and action a, we can determine next transition s_{t+1} and reward r_{t+1} based on policy π. For the sake of easy understanding, we can rewrite the objective function as follows.

$$\mathcal{J}_{s(t)}(t) = \mathcal{J}\big\{\gamma(t), \mathscr{G}(t)\big\} \tag{31}$$

Where $\gamma(t)$ and $\mathscr{G}(t)$ are given by state s_t at time t. Now based on $\mathcal{J}_{s(t)}(t)$ and $\mathcal{J}_{s(t+1)}(t+1)$, we define reward for the state-action pair (s_t, a) as follows.

$$r(t+1) = \begin{cases} 1, & \text{If } \mathcal{J}_{s(t)}(t) > \mathcal{J}_{s(t+1)}(t+1) \\ 0, & \text{If } \mathcal{J}_{s(t)}(t) = \mathcal{J}_{s(t+1)}(t+1) \\ -1, & \text{If } \mathcal{J}_{s(t)}(t) < \mathcal{J}_{s(t+1)}(t+1) \end{cases} \tag{32}$$

The proposed DRL-enabled partial service provisioning strategy is an efficient workload distribution strategy for handling large data traffic in the industrial networks, as illustrated in Fig. 3. In the first phase, IIoT devices generate a large amount of data, which is difficult to process locally. Thus, IIoT devices $\mathcal{M}$ try to execute a portion of generated tasks locally and request additional services $\mathcal{I}$ from the remote computing device, i.e., connected master fog device $f \in \mathcal{F}$. Initially, the master fog device tries to execute the requested service request itself, however, if the available CPU frequency $\mathscr{G}_f^{\text{CPU}}$ of the master fog device $f \in \mathcal{F}$ is less than the requested services i, then the master fog device first partitions the tasks and partially requests additional assistance from the nearby fog devices $\widetilde{f} \in \widetilde{\mathcal{F}}$ or centralized cloud server $s \in \mathcal{S}$. This process continues until all the requests are satisfied. It is important to note that the proposed strategy is suitable for a partial service provisioning mechanism while minimizing the energy usage and delay of industrial applications. The steps of the DRL-based partial service provisioning strategy are shown in *Algorithm* 2.

TABLE I
PARAMETERS & RELATED VALUES USED IN SIMULATION

Parameters	Values	Parameters	Values
$\mathcal{M}$	100	$\mathcal{F}$	10
$\mathcal{S}$	3	$w^{\mathbb{T}}$	1
α	0.0001	$\mathbb{P}^{\text{ppower}}$	3.25×10^{-7}
$\mathcal{B}^{\text{trans}}$	130 MHz	Number of epoch	10000
$\mathcal{I}$	4	Training samples	2000

IV. EMPIRICAL EVALUATION

This section emphasizes the performance analysis of the proposed partial service provisioning strategy with existing baseline algorithms, such as *Local Execution* (LE), where IIoT devices process all the service requests locally without considering the remote processing functionality, *Fog Execution* (FE), where IIoT devices assign the complete or incomplete data to the remote fog device for execution, heuristic DPTO [3], and EETO [6] strategies in terms of total energy usage and delay minimization.

A. Simulation Setup

For the simulation setup, we use Python on Intel i7 CPU @ 3.40GHz system with 10GB RAM for implementing our proposed partial offloading strategy. In particular, we use Keras, an open-source library for implementing machine learning or deep learning algorithms. Besides that, we consider total number of IIoT devices $\mathcal{M} = [10, 100]$, fog devices $\mathcal{F} = [5, 10]$, number of service requests from each IIoT device $\mathcal{I} = 4$ and cloud server $\mathcal{S} = 3$. We set the transmission bandwidth is $\mathcal{B}^{\text{trans}} = 130$MHz, task size $\mathcal{B}_i = [5, 20]$ Kb, and processing density is 1900 [cycles/second] for all the IIoT devices. Other simulation parameters are taken from [3] and [4], as depicted in Table I. Moreover, we set the maximum CPU frequency in IIoT devices $\mathscr{G}_a^{\text{CPU}} = 500 \times 10^6$ [cycle/second], fog devices $\mathscr{G}_f^{\text{CPU}} = 50 \times 10^9$ [cycle/second], cloud server $\mathscr{G}_s^{\text{CPU}} = 100 \times 10^6$ [cycle/second], and the transmission energy as 1.42×10^{-7} J/bit.

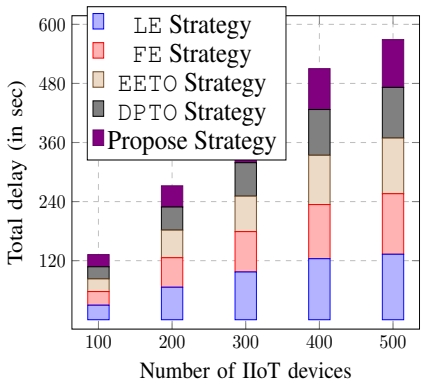

Fig. 4. Performance analysis of execution delay.

B. Execution Delay

The metric delay represents the total processing delay $\mathbb{T}_i^{\text{total}}$ for satisfying one service request $i \in \mathcal{I}$ on different set of computing devices j, $\forall j \in \{\mathcal{M}, \mathcal{F}, \mathcal{S}\}$, which includes processing

delay $\mathbb{T}_m^{\text{IoT}}$, uploading delay $\mathbb{T}_j^{\text{uplink}}$, and remote execution delay $\mathbb{T}_{i,j}^{\text{remote}}$. The average processing delay directly depends on the input data size $\mathcal{B}_i$, and CPU frequency on various computing devices $\mathscr{G}_j^{\text{CPU}}$. Equation $\mathbb{T}_{i,m}^{\text{IoT}} = \gamma(i,m)\mathcal{B}_i\mathcal{D}_i/\mathscr{G}_m^{\text{CPU}}$ represents that the processing delay $\mathbb{T}_{i,m}^{\text{IoT}}$ on the IIoT device m inversely proportional to the computational frequency $\mathscr{G}_m^{\text{CPU}}$. However, the total processing delay $\mathbb{T}_{i,m}^{\text{IoT}}$ increases with the increase of the input data size $\mathcal{B}_i$. From Fig. 4, we can easily observe that the performance of the proposed DRL based service provisioning strategy is much higher than the baseline LE strategy, FE strategy, EETO strategy, and DPTO strategy. The reason is that the proposed strategy uses a penalty function to control the delay deadline restricted service requests and reduces upto 16% processing delay compared with existing baseline algorithms.

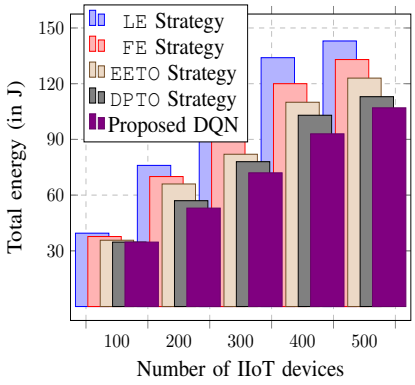

Fig. 5. Performance analysis of energy consumption.

C. Energy Consumption

The energy consumption to process an IIoT generated service request $i \in \mathcal{I}$ can be regarded as the combination of local processing energy $\mathbb{E}_{i,m}^{\text{IoT}}$, data transmission energy to nearby computing device $\mathbb{E}_{i,j}^{\text{uplink}}$ and remote processing energy $\mathbb{E}_{i,j}^{\text{IoT}}$ for any application. The average energy consumption mostly depends on input data size $\mathcal{B}_i$ and the processing frequency $\mathscr{G}_j^{\text{CPU}}$ of the remote computing device j, $\forall j \in \{\mathcal{F}, \mathcal{S}\}$. From $\mathbb{E}_{i,m}^{\text{IoT}} = \gamma(i,m)\mathcal{B}_i\mathcal{D}_i k(\mathscr{G}_m^{\text{CPU}})^2$, it can be observed that the energy consumption on local IIoT device $\mathbb{E}_{i,m}^{\text{IoT}}$ increases with the increase in data size $\mathcal{B}_i$. Similarly from $\mathbb{E}_{i,f}^{\text{fog}} = (1-\gamma(i,m))\mathcal{B}_i\mathcal{D}_i k(\mathscr{G}_f^{\text{CPU}})^2$ and $\mathbb{E}_{i,\widetilde{f}}^{\text{fog}} = \Gamma(i,\widetilde{f})\mathcal{B}_i\mathcal{D}_i k(\mathscr{G}_{\widetilde{f}}^{\text{CPU}})^2$ it can also be analysed that energy consumption rate on the local fog devices can be minimized by reducing the CPU frequency on those devices. From Fig. 5 it can be easy to understand that the energy consumption rate using proposed DRL base strategy is less compared with the baseline LE strategy, FE strategy, EETO strategy and DPTO strategy. The reason is that the proposed strategy partition the requested services and immediately distribute to the underloaded local fog devices instead of transmitting to the centralized cloud servers, which minimizes the energy usage upto 23% to 25%.

D. System Throughput

This parameter describes the absolute number of service requests $\mathcal{I}$ satisfied within the stipulated time period $\mathbb{T}^{\max}$.

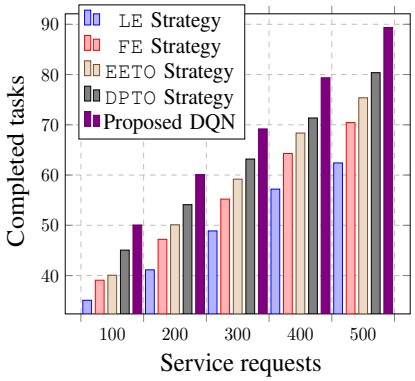

Fig. 6. Performance analysis of system throughput.

Fig. 6 illustrates the variation of the completed tasks on various computing devices $\mathcal{CD}$ including master fog device $\mathcal{F}$, nearby fog device $\widetilde{\mathcal{F}}$, and centralized cloud server $\mathcal{S}$ with the number of services requests $\mathcal{I}$. From Fig. 6, it can clearly be witnessed that the proposed partial service provisioning strategy achieves the highest throughput compared with the existing LE strategy, FE strategy, EETO strategy, and DPTO strategy, and this rate increases over time. The intelligence behinds that the proposed strategy initially takes arbitrary action across the network. However, the performance improves after training with an appropriate number of samples. Further to achieve higher throughput, the master fog device divides the long-scale application data into a sequence of independent sub-tasks and distributed them among the nearby fog devices using the DRL technique.

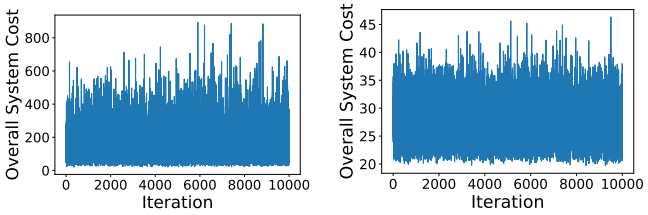

Fig. 7. Overall system performance: (a) using random action strategy, (b) proposed service provisioning strategy.

E. DRL Performance Analysis

To analyze the performance of the proposed DRL-based service provisioning strategy, we consider a mini-batch training method with batch size 32, learning rate 0.0001, number of epochs 10000, and replay memory size 2000. To train both the network parameters and computing resources, we generate initial data samples through a random process and outline a Deep Neural Network (DNN) with one input layer, three hidden layers, and one output layer. We employ the dense layer for the intermediate layers and nonlinear ReLU activation layer to obtain all the network features and understand dynamic network parameters. Besides that, we use descent network optimizers, Boltzmann Q-policy, sequential memory, and the

best set of hyper-parameters from the available parameter set. For the beginning setup, we consider random weights and high target values to train the industrial fog network.

From Fig. 7(a), it is observed that the random action strategy performs in an uncontrolled manner and failed to optimize the overall system performance. The proposed DRL network is also trained with Adamax, SGD, and RMSprop to analyze the variation of reward values over the industrial networks. The performance of system cost for the proposed partial service provisioning strategy with the interpretation of the iterations is illustrated in Fig. 7(b). From the figure, we can easily observe that the proposed DRL-based method optimizes the overall system cost with the comparison of random action strategy. The reason is that the proposed DRL-based approach initially takes random action due to insufficient training samples. However, after training specific iterations and obtaining experiences from the previous mistakes, the DRL agent makes the best decision and achieves desired output. Moreover, we observe the run-time complexity of these optimization algorithms are 116, 117, 117 and 119 seconds, respectively.

V. Conclusion

In this paper, we have introduced a collaborative AI-enabled partial service provisioning strategy for handling large-scale industrial applications in fog networks. The proposed fog framework has considered a two-dimension resource integration policy among distributed fog devices and centralized cloud servers, including horizontal (fog-to-fog) and vertical (fog-to-cloud) resource sharing. At first, we aim to jointly optimize the processing latency and energy consumption rate of all the IIoT applications in the industrial fog networks. To achieve this objective, we introduce a task partitioning strategy for dividing large IIoT tasks into a set of independent tasks. Then a DRL-enabled service provisioning strategy is adopted for distributing the partitioned tasks among the nearby fog devices intelligently. The simulation results demonstrate that the proposed service provisioning strategy outperforms the standard baseline algorithms by minimizing the energy consumption upto 23.3% to 24.5% and delay upto 15.5% to 16.3%, respectively. In the future, we will enhance the proposed service provisioning model for intelligent autonomous transportation systems to improve the performance of the 6G-enabled fog networks.

Acknowledgement

We acknowledge financial support to UoH-IoE by MHRD (F11/9/2019-U3(A)) and in part by DST (SERB), Government of India, under Grant EEQ/2018/000888.

References

[1] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar, and A. Y. Zomaya, "Edge Intelligence: The Confluence of Edge Computing and Artificial Intelligence," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7457–7469, 2020.

[2] N. Magaia, R. Fonseca, K. Muhammad, A. H. F. N. Segundo, A. V. Lira Neto, and V. H. C. de Albuquerque, "Industrial Internet-of-Things Security Enhanced With Deep Learning Approaches for Smart Cities," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6393–6405, 2021.

[3] M. Adhikari, M. Mukherjee, and S. N. Srirama, "DPTO: A Deadline and Priority-Aware Task Offloading in Fog Computing Framework Leveraging Multilevel Feedback Queueing," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 5773–5782, 2020.

[4] A. Hazra, M. Adhikari, T. Amgoth, and S. N. Srirama, "Stackelberg Game for Service Deployment of IoT-Enabled Applications in 6G-aware Fog Networks," *IEEE Internet of Things Journal*, pp. 1–10, 2020.

[5] S. Misra and N. Saha, "Detour: Dynamic Task Offloading in Software-Defined Fog for IoT Applications," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 5, pp. 1159–1166, 2019.

[6] A. Hazra, M. Adhikari, T. Amgoth, and S. Srirama, "Joint Computation Offloading and Scheduling Optimization of IoT Applications in Fog Networks," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 4, pp. 3266–3278, 2020.

[7] Z. Ning, P. Dong, X. Kong, and F. Xia, "A Cooperative Partial Computation Offloading Scheme for Mobile Edge Computing Enabled Internet of Things," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4804–4814, 2019.

[8] Z. Kuang, L. Li, J. Gao, L. Zhao, and A. Liu, "Partial Offloading Scheduling and Power Allocation for Mobile Edge Computing Systems," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6774–6785, 2019.

[9] S. Misra, S. P. Rachuri, P. K. Deb, and A. Mukherjee, "Multi-Armed Bandit-based Decentralized Computation Offloading in Fog-Enabled IoT," *IEEE Internet of Things Journal*, pp. 1–1, 2020.

[10] Z. Cao, P. Zhou, R. Li, S. Huang, and D. Wu, "Multiagent Deep Reinforcement Learning for Joint Multichannel Access and Task Offloading of Mobile-Edge Computing in Industry 4.0," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6201–6213, 2020.

[11] M. Mukherjee, S. Kumar, M. Shojafar, Q. Zhang, and C. X. Mavromoustakis, "Joint Task Offloading and Resource Allocation for Delay-Sensitive Fog Networks," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–7.

[12] Z. Shi, X. Xie, H. Lu, H. Yang, M. Kadoch, and M. Cheriet, "Deep-Reinforcement-Learning-Based Spectrum Resource Management for Industrial Internet of Things," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3476–3489, 2021.

[13] J. Li, A. Wu, S. Chu, T. Liu, and F. Shu, "Mobile Edge Computing for Task Offloading in Small-Cell Networks via Belief Propagation," in *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018, pp. 1–6.

[14] H. Liao, Z. Zhou, X. Zhao, L. Zhang, S. Mumtaz, A. Jolfaei, S. H. Ahmed, and A. K. Bashir, "Learning-Based Context-Aware Resource Allocation for Edge-Computing-Empowered Industrial IoT," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4260–4277, 2020.

[15] R. Saha, S. Misra, and P. K. Deb, "FogFL: Fog-Assisted Federated Learning for Resource-Constrained IoT Devices," *IEEE Internet of Things Journal*, vol. 8, no. 10, pp. 8456–8463, 2021.

[16] P. K. Deb, C. Roy, A. Roy, and S. Misra, "DEFT: Decentralized Multiuser Computation Offloading in a Fog-Enabled IoV Environment," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 15 978–15 987, 2020.

[17] M. Sheng, Y. Wang, X. Wang, and J. Li, "Energy-Efficient Multiuser Partial Computation Offloading With Collaboration of Terminals, Radio Access Network, and Edge Server," *IEEE Transactions on Communications*, vol. 68, no. 3, pp. 1524–1537, 2019.

[18] U. Saleem, Y. Liu, S. Jangsher, X. Tao, and Y. Li, "Latency Minimization for D2D-Enabled Partial Computation Offloading in Mobile Edge Computing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 4472–4486, 2020.

[19] S. Nath, Y. Li, J. Wu, and P. Fan, "Multi-user Multi-channel Computation Offloading and Resource Allocation for Mobile Edge Computing," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6.

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