This article has been accepted for publication in a future issue of Computing Journal (Springer)

Trust-driven Reinforcement Selection Strategy for Federated Learning on IoT Devices

Gaith Rjoub[1], Omar Abdel Wahab[2], Jamal Bentahar[1], and Ahmed Bataineh[1]

[1] Concordia University, Montreal, QC, Canada
{g_rjoub,bentahar,ah_batai}@encs.concordia.ca
[2] Université du Québec en Outaouais, Gatineau, QC, Canada
omar.abdulwahab@uqo.ca

Abstract. Federated learning is a distributed machine learning approach that enables a large number of edge/end devices to perform on-device training for a single machine learning model, without having to share their own raw data. We consider in this paper a federated learning scenario wherein the local training is carried out on IoT devices and the global aggregation is done at the level of an edge server. One essential challenge in this emerging approach is IoT devices selection (also called scheduling), i.e., how to select the IoT devices to participate in the distributed training process. The existing approaches suggest to base the scheduling decision on the resource characteristics of the devices to guarantee that the selected devices would have enough resources to carry out the training. In this work, we argue that trust should be an integral part of the decision-making process and therefore design a trust establishment mechanism between the edge server and IoT devices. The trust mechanism aims to detect those IoT devices that over-utilize or under-utilize their resources during the local training. Thereafter, we introduce DDQN-Trust, a double deep Q learning-based selection algorithm that takes into account the trust scores and energy levels of the IoT devices to make appropriate scheduling decisions. Finally, we integrate our solution into four federated learning aggregation approaches, namely, *FedAvg*, *FedProx*, *FedShare* and *FedSGD*. Experiments conducted using a real-world dataset show that our DDQN-Trust solution always achieves better performance compared to two main benchmarks: the DQN and random scheduling algorithms. The results also reveal that *FedProx* outperforms the competitor aggregation models in terms of accuracy when integrated into our DDQN-Trust solution.

Keywords: Federated Learning · Edge Computing · Internet of Things (IoT) · Double Deep Q-Network (DDQN) · Trust · IoT Device Selection.

1 Introduction

With the increasing reliance on Internet of Things (IoT) applications and social media platforms, the volume of data that need to be stored and processed is becoming enormous. Cloud computing has long been a great solution to deal with this challenge, owing to the wide array of benefits it has proved to offer for data providers [4, 9, 30, 34, 35]. These benefits include multi-tenancy, elasticity, virtualization and reduced storage and processing costs. Consequently, instead of acquiring and continuously maintaining expensive hardware equipment to store and analyze big data, companies can now migrate these duties to the cloud to be done in a more efficacious and cost-efficient manner. The increasing data privacy and network communication concerns play against the adoption of a centralized cloud-based data storage and analytics approach. First, data owners often feel reluctant to share their data with the cloud platform [3]. This is because these owners will no longer have any control on their own data and hence are not sure which other (possible unauthorized) parties will have access to their sensitive data. Federated learning has been proposed to let IoT devices in the cloud computing environment train machine learning models collaboratively without moving data from any IoT device to one central location [37]. In particular, IoT devices train local models using their data, and then upload the local models, instead of raw data, to a centralized parameter server.

Moreover, the cloud data centers are mostly located in geographical areas that are quite far from the IoT devices. This entails high communication cost and delay to transmit data to the cloud for processing and receive back the insights from the cloud for decision-making. These factors have pushed the research community to design distributed data analytics approaches that are executed either by the end devices or at the edge of the network [41]. Edge computing fetches two great refinements to the existing cloud computing. On one hand, edge computing allows preprocessing large amounts of data at edge nodes before transferring them to the central servers in the cloud. On the other hand, it enables the edge nodes with computing ability to optimize the cloud resources.

1.1 Problem Statement

Inspired by this idea, the concept of *Federated Learning* (*FL*) has recently been proposed to allow end devices to collaboratively train a single machine learning model without having to share their raw data. FL consists of two main phases, i.e., local training and global computation. In the local training phase, a parameter server (e.g., edge node) initializes the machine learning model and then shares initial parameters with the end devices. These devices then use the shared parameters to train the model on their own data. Finally, they share the updated parameters obtained through training the model on their data with the parameter server. In the global computation phase, the parameter server aggregates all the received updates to reconstruct a global machine learning model. This process repeats until a certain accuracy level is attained.

One substantial challenge in federated learning is how to select the end devices that will participate in the collaborative training. Several approaches have been proposed to tackle this challenge [26, 38, 40]. Most existing approaches rely on the devices' resource characteristics when taking their decisions. Despite the importance of the resource factor, we argue in this work that the reliability of the devices cannot be overlooked. In fact, the presence of unreliable devices in the federated training might lead to performance degradation and even security hazards. Unreliable devices might, for example, use bogus data to do their local training. To address this problem, we propose in this paper a device selection solution, also called scheduling, for federated learning that takes into account both the resources availability (in terms of energy level) and trustworthiness of the IoT devices [5, 11, 12]. The considered scenario consists of an edge server which plays the role of the parameter server that is responsible for the global computation phase and IoT devices that are responsible for the local training phase. A fundamental challenge in federated learning is the uncertainty that the edge server faces regarding the resource and trust levels of the IoT devices. Specifically, with the large numbers of deployed IoT devices in edge-based systems, the chances of encountering untrusted or poorly-performing IoT devices is fairly high. Such IoT devices would not only cause the process and execution time to be high, but will also increase the amounts of wasted resources in case of malicious or compromised IoT devices. We address this challenge by proposing DDQN-Trust, a trust-driven double deep Q-network reinforcement learning-based algorithm [31]. Compared to the traditional reinforcement learning approaches, DDQN-Trust has the advantage of reducing the overestimation of Q values and thus helps us achieve a faster training and have a more stable learning. Moreover, as argued in [18, 21, 37], DDQN reveals better performance compared to classic optimization methods such as Monte-Carlo search, swarm intelligence, genetic algorithms and Bayesian methods.

Another major limitation of most existing federated learning scheduling approaches stems from the fact that they base their solutions, by default, on the federated averaging aggregation approach (i.e., *FedAvg*). According to this approach, clients perform several batch updates on their local data and then transmit their updated weights to the server. The server then takes the average of the weights to derive the global model for the next training iteration. Unfortunately, *FedAvg* suffers from two main limitations from both the statistical and system perspectives [36]. From the statistical perspective, the performance of *FedAvg* begins to diverge in settings where the data is non-identically distributed across devices, which is the case in our scenario. From the system's perspective, *FedAvg* does not allow participating devices to perform variable amounts of local work based on their underlying systems constraints, making it less resilient to dynamic federated learning settings. To address this problem, we investigate three other aggregation approaches and integrate them, in addition to *FedAvg*, into our solution to study its applicability to a wider set of scenarios.

1.2 Application Scenario

As an application example, we adopt an image and video object prediction model over an autonomous vehicles (AVs) environment. In this scenario, we use real-time street images and videos, including dashcam videos, camera car images, pedestrian images and videos in public transportation. Those images and videos are the core of smart city applications that rely on IoT systems [32]. The following is the scenario that we consider in our simulation. Data coming to devices contain unique, overlapping, or heterogeneous inputs. In addition, the labeled data on an edge server are available for devices to access at any time, and the initial dataset on the AVs will act as a template on different iterations. In order to ensure consistency in the testing, we reuse this public dataset on the edge server as the test set. For training, each vehicle private data are used to train the CNN learning model. The publicly available dataset on the edge server is used to test the CNN model. The reason behind using this dataset is because it contains varying labels instead of testing the model on only the local limited data.

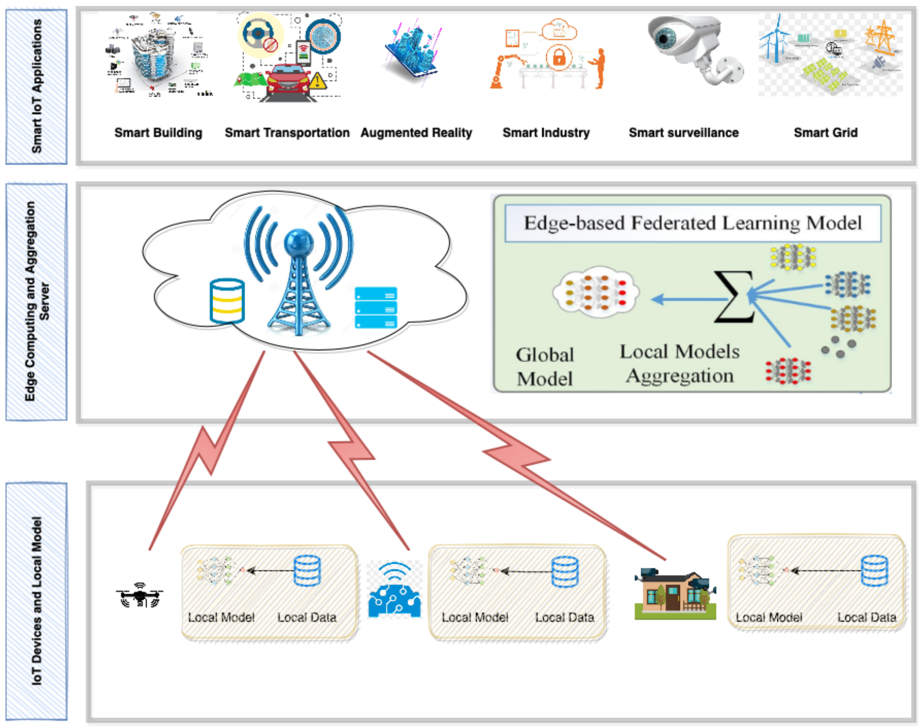


Fig. 1: Architectural overview of federated learning for IoT networks.

As illustrated in Fig. 1, the FL-based architecture for IoT data consists of three layers: the IoT devices and local model layer, the edge computing and aggregation server layer, and the smart IoT application layer. Each layer affects the model's reliability, and it also influences the quality of service on the network. Based on our scenario, the AVs environment has edge server, including wireless base stations and roadside units, that are situated along the roadside. The edge computing server is used to aggregate local models from vehicles into a global model based on their local data. The vehicles only share their local models, rather than their local data, in order to protect their privacy. However, FL is the underlying principle of the system, which involves AVs completing learning tasks in a distributed manner.

1.3 Contributions

In a preliminary version of this work [31], we proposed an intelligent scheduling approach for federated learning that helps the server select the subset of IoT devices that minimizes the resources utilization and maximizes the energy efficiency. This paper builds on and extends our previous work by integrating our scheduling approach with different federated learning aggregation models, namely *FedProx*, *FedShare*, *FedSGD*, and *FedAvg* to select the aggregation model that best suits our solution.

To provide a solution to the problems mentioned above, we use in our paper the autonomous vehicular edge computing (AVEC) scenario as shown in Fig. 2 to illustrate how the computing, data storage, and communication resources can take advantage of vehicular networks at the edge computing environment. In order to facilitate on-road tasks while respecting the real-time and reliability requirements, edge servers such as roadside units and base stations can be used. The edge server aggregates the local models of vehicles in a certain area, allowing the new vehicle to download the edge network model, monitor the road conditions, and then enhance driver assistance features such as real-time navigation, lane change, collision warnings, and recognize traffic light.

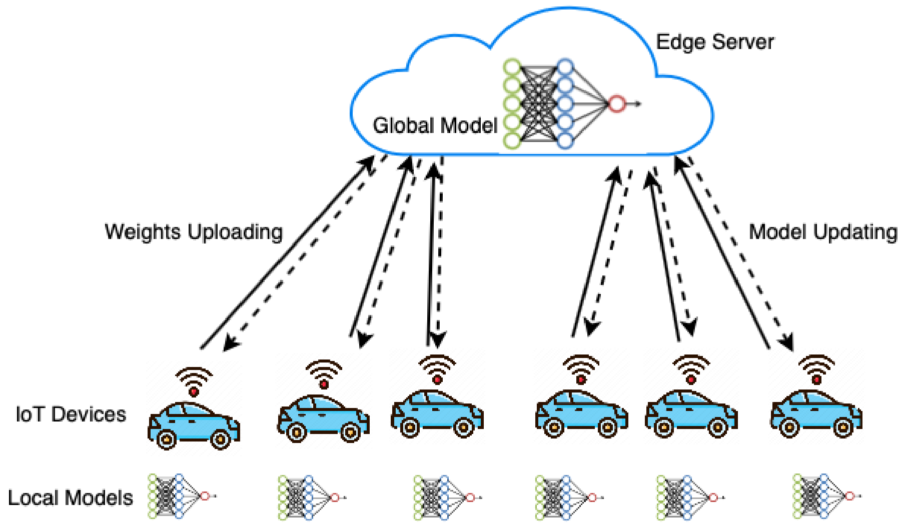


Fig. 2: System process of the proposed edge federated learning model.

Executing FL models over IoT environments has been mainly categorized into two directions [17]: computation resources and communication. From the computation perspective, the idea is to reduce the overhead and time in the local training phase over IoT devices to make the model learning efficient. The main objective of our scheduling solution is to attain better resource exploitation, to guarantee the security of the user's data, and to ensure that sufficient energy is available for the training process requirements. To this end, our solution focuses on the computing aspect to select the trusted IoT that will execute the federated training. Therefor, we take into account the CPU, memory, and energy of the IoT devices to predict their ability to perform the federated learning tasks. Particularly, at each local training round, we select the most trusted IoT devices in terms of computation to improve the training performance. The main contributions of the paper can be summarized as follows:

- We propose a trust establishment technique for the IoT devices. The algorithm monitors the CPU and memory consumption of the IoT devices and employs a modified Z-score statistical method to identify the IoT devices that exhibit any abnormal behavior in terms of over-consumption or under-consumption. This is of prime importance to detect those devices that do not dedicate enough resources to serve the federated learning tasks as well as those that carry out additional computations to achieve some malicious objectives. The modified Z-score is more robust than the standard Z-score technique since it relies on the median (instead of the mean) for calculating the Z-score. It is thus less influenced by the outliers. Moreover, compared to classification techniques such as Support Vector Machine (SVM) and decision tree, the modified Z-score technique needs less training time and can hence be executed with less overhead.
- We introduce DDQN-Trust, an algorithm which enables the edge server to find the optimal scheduling decisions in terms of energy efficiency and the trustworthiness. In particular, we first formulate a stochastic optimization problem that seeks to derive a set of IoT devices that, by sending the federated learning tasks to them, the edge server can maximize the trust and minimize the energy cost. The algorithm is designed to solve the optimization problem while modeling the uncertainty that the server faces regarding the resource and trust levels of the IoT devices.
- Unlike most existing scheduling approaches in FL which base their solutions on the *FedAvg* aggregation method by default, we integrate four aggregation approaches, namely *FedAvg*, *FedProx*, *FedShare* and *FedSGD* into our DDQN-Trust solution. This is important to broaden the applicability of our solution to a wider set of federated learning scenarios.
- We analyse the proposed solution experimentally in an image recognition scenario using a Convolutional Neural Network (CNN). The experimental results reveal that our solution shows a better performance compared to the DQN and random scheduling algorithms.

1.4 Organization

The rest of the paper is organized as follows. In Section 2, we conduct a literature review on the existing task scheduling approaches in cloud and edge computing settings, and in the context of federated learning. We also survey the main deep and reinforcement learning-based resource management approaches. In Section 3, we describe the details of the proposed solution. In Section 4, we explain the experimental environment, evaluate the performance of our scheduling solution, and present empirical analysis of our results compared to other benchmarks. Section 5 concludes the paper.

2 Related Work

In this section, we survey the main task scheduling approaches in federated learning as well as in edge computing environments.

2.1 Task Scheduling with Federated Learning

In [6], the authors study the problem of training FL algorithms over a realistic wireless network. To do so, the authors formulate an optimization problem that jointly considers user selection and resource allocation to minimize the value of the loss function. To solve this problem, a closed-form expression of the expected convergence rate of the FL algorithm that considers the wireless factors is derived. In [2], the authors adopt a DQN algorithm that allows the server to learn and find optimal decisions without any a priori knowledge of the network dynamics. The authors employ Mobile Crowd-Machine Learning (MCML) to address data privacy issues of traditional machine learning. In [14], the investigators propose a segment-level decentralized federated learning to improve the efficiency of network capacity utilization among client nodes. In particular, the authors introduce a segmented gossip approach, which not only makes full utilization of node-to-node bandwidth, but also achieves good training convergence. In [27], the authors present *FedCS*, a protocol that aims to improve the efficiency of FL in a mobile edge computing environment with heterogeneous clients. FedCS proposes to solve a client selection problem with resource constraints, which allows the server to aggregate as many client updates as possible and to accelerate the training convergence rate. In [26], the authors present a DQN algorithm for resource allocation in a mobility-aware FL network. The authors propose to employ DQN to enable the model owner to find the optimal decisions in terms of energy and channels without priori knowledge about the network. The authors formulate the energy and channel selection decision of the model owner as a stochastic optimization problem. The optimization problem aims to maximize the number of successful transmissions of the model owner while minimizing the energy and channel costs.

2.2 Task Scheduling in Cloud and Edge Computing

In [29], the researchers propose *BigTrustScheduling*, a trust-aware scheduling solution tailored for big data tasks. The solution consists of three stages: virtual machines (VMs) trust level computation to derive a trust value for each VM based on its underlying performance, task priority level determination based on resource requirements and prices, and trust-aware scheduling that minimizes the makespan and cost of task execution. In [20], the authors present a smart manufacturing factory framework based on edge computing and investigate the Job Shop Scheduling (JSP) under such a setting. Moreover, the authors adjust the DQN framework with an edge computing framework to solve the JSP. In [8], the investigators consider the characteristics of autonomous-driving tasks to select more suitable mobile edge computing (MEC) servers for task migration. To improve the earliest deadline first algorithm through the replacement and recombination of tasks, the authors propose a Best Fit Replacement Scheduling (BFRS) technique that enables more tasks to be executed at every stage, while considering the time constraints of tasks, the urgency difference among them and their vulnerability to environmental impacts. In [15], the researchers aim to reach an optimal revenue for edge service providers in the contexts of dynamic task scheduling and resource management in MEC environments. Moreover, the authors argue that their solution achieves a favorable property called total unimodularity. This property further helps design an equivalent linear programming problem which can efficiently and elegantly be solved in polynomial time.

2.3 Federated Learning Aggregation Approaches

In [24], the investigators introduce *FedAvg* in which the server collects the local stochastic gradient descents from the client devices and takes the average to produce the next global model. The authors performed extensive experiments on this algorithm, demonstrating its robustness in unbalanced and non-IID data distributions. For strongly convex and non-convex loss functions, the authors in [7] propose the Federated Learning with Heterogeneous Quantization (FEDHQ) approach that uses different aggregation weights to balance over converging clients in the face of heterogeneous quantization errors. In [22], the authors present two class-weighted aggregation methods for federated learning under non-ID data, i.e., FedCA-TDD and FedCA-VSA. Based on the contribution of every client, both strategies assign weights; yet they determine the contributions of the clients' models differently. Under FedCA-TDD, the classes generated using each client's data are taken into account, while under FedCA-VSA, only the accuracy of the local models is assessed. A multi-criteria client selection approach called FedMCCS is described in [1]. The main focus of this approach is to utilize a bilevel optimization scheme that can effectively select and maximize the number of FL participants at each round, while taking into consideration the lack of communication and computation, and client heterogeneity.

In [19], the authors define an aggregation framework, called *FedProx*. *FedProx* addresses the system and statistical heterogeneity that arises from FL.

The authors argue that *FedProx* can be viewed as a generalization and re-parametrization of *FedAvg*, the current state-of-the-art aggregation method for FL. *FedProx* allows variable amounts of work to be performed locally across devices, and relies on a proximal term to help stabilize the aggregation results. In [39], the researchers focus on the statistical challenge of FL when local data is non-IID. To address this challenge, they propose *FedShare* whose main idea is to share parts of a small public dataset among clients to alleviate the weight divergence across the local data of the clients. In [10], the authors define *FedSGD* and provide a comprehensive study of its convergence. *FedSGD* operates under non-IID data for strongly convex and smooth FL problems. *FedSGD* is characterized by a trade-off between the number of local computation rounds and global communication rounds.

Overall, the existing scheduling approaches in FL, cloud and edge computing focus on the resource characteristics of the participant devices, but overlook the reliability of these devices. In this work, we consider both the resource and trust components to guarantee high-quality and reliable performance of the federated training. Moreover, different from the scheduling approaches that employ traditional deep Q learning approaches, we formulate in this paper the scheduling problem as a double deep Q learning algorithm. This is important to consider the uncertainty that the server faces about the trust and resource characteristics of the IoT devices, while avoiding the problem of overoptimism when choosing the scheduling actions.

3 Trust-Aware IoT Scheduling for Federated Learning

We explain in this section the details of our proposed double deep Q learning-based selection algorithm that takes into account the trust scores and energy levels of the IoT devices to make appropriate scheduling decisions for FL.

3.1 Trust Establishment Mechanism

Several approaches have been proposed to improve the scheduling process in cloud environments. The main idea of these approaches is to reduce the makespan by trying to reduce the waiting time of tasks in the queues and attempting to map tasks to the nearest IoTs to reduce the transfer time (i.e., the data locality concept). However, this does not always guarantee a better performance in a dynamic and open environment like cloud computing. In fact, with the increasing number of deployed autonomous IoT devices, the likelihood of coming across untrusted or compromised ones is reasonably high, which motivates the need for a trust mechanism.

In Algorithm 1, we propose a statistical trust establishment method for IoT devices based on monitoring the CPU and RAM consumption of the devices to identify the ones that exhibit some abnormal resource consumption behavior, and the devices whose consumption goes down the normal minimal habitual consumption (e.g., failed IoTs). This is important to detect those devices that do not dedicate enough resources to serve the FL tasks as well as those that exhibit some overly high consumption which could be an indication of some malicious behavior. For example, some malicious devices might optimize for a malicious objective that aims to generate targeted misclassification. Such devices are expected to spend more resources than the regular devices that only try to optimize for the underlying federated task.

Algorithm 1 is executed by an edge server to monitor the IoT devices that are located within its range. The proposed method capitalizes on the modified Z-score statistical technique. Modified Z-score is a standardized score that measures outlier strength, i.e., how much a particular score differs from the typical score by checking the dependability of a particular score on a certain typical score. This method shows a greater robustness to outliers compared to some other statistical techniques (e.g, traditional Z-Score, Tukey method, etc.) since it capitalizes on the median $\bar{x}$ instead of the mean μ. In our algorithm, this method approximates the difference of a certain score from the median using the median absolute deviation $MAD_j^z(t)$ of a metric z (e.g., CPU, RAM) consumed by a device j during a time window $[t-\delta, t]$ (Algorithm 1 line 6).

More specifically, the modified Z-score $\alpha_j^z(i,t)$ is calculated through dividing the difference between the consumption $x_j^z(i)$ of the device j in terms of the resource metric z at time moment $i \in [t-\delta, t]$ and the median consumption of that device in terms of that metric within the time interval $[t-\delta, t]$ by the median absolute deviation of the metric z (Algorithm 1 line 20). The constant $\varrho = 0.6745$ is needed because $\mathbb{E}(MAD_j^z(t)) = 0.6745\sigma$ for a large number n of samples. Observations will be labeled outliers when $\alpha_j^z(i,t) \geq \varphi$, where $\varphi = 3.5$ as argued in [16]. This limit quantifies the patterns of maximal and minimal habitual utilization of each IoT device within a certain time interval. Thus, any future consumption that exceeds or falls under this limit is deemed to be unusual.

The Algorithm then checks for any future consumption of the IoT to determine whether there exists any consumption that exceeds or falls under the computed abnormal limit (Algorithm 1 - lines $22-23$). If such a case is encountered, this observation is added to a table that registers each IoT's unusual consumption (if any) (Algorithm 1 - line 24). The average unusual consumption for each metric is then computed (Algorithm 1 - line 28). The Algorithm then computes the trust value of each IoT by dividing the sum of the proportional abnormal consumption over all the metrics by the number of metrics that the device has overused/underused (if any) (Algorithm 1 - line 36). If no metric has been overused/underused, the initial trust in the IoT's trustworthiness would be set to 1 (Algorithm 1 - line 34), which represents a full trust in that device.

3.2 DDQN-Trust Scheduling Policy

Reinforcement learning [23, 28] is an active research and application area of machine learning that has been applied to solve uncertainty-driven problems wherein exact models are often infeasible. It aims at guiding a certain agent on how to react to the changes that take place in the environment. The agent performs the appropriate actions that maximize its cumulative reward according

Algorithm 1 IoT Trust

```
    Inputs:
 1: $j$: an IoT being monitored by the edge computing server
 2: $M = \{CPU,\ memory\}$: the set of $IoT$'s metrics to be analyzed by the edge server
 3: $\delta$: size of time window after which the algorithm is to be repeated
    Variables:
 4: $M_j^z(t)$: a table recording $x_j^z(i)$ $(i = t-\delta, t-\delta+1, \ldots, t)$, the amounts of $z \in M$ consumed by
    $j$ during the time interval $[t-\delta, t]$
 5: $\bar{x}_j^z(t)$: the median of $M_j^z(t)$ (median consumption of $z \in M$ by $j$ during the time interval $[t-\delta, t]$)
 6: $MAD_j^z(t)$: the median absolute deviation of $M_j^z(t)$, i.e., $MAD_j^z(t) = median\left\{\left|x_j^z(i) - \bar{x}_j^z(t)\right|\right\}$
    for all $t-\delta \le i \le t$
 7: $\alpha_j^z(i,t)$: the modified Z-score of $x_j^z(i) \in M_j^z(t)$
 8: $AbnormalMetrics_j^z$: sum of unusual consumption of $z \in M$ by $j$
 9: $CountAbnormalMetrics_j^z$: a counter enumerating the occurrence of unusual consumption of
    $z \in M$ by $j$
10: $AvgAbnormalMetrics_j^z$: $j$'s average unusual consumption of $z \in M$
11: $PropAbnormalMetrics_j^z$: $j$'s unusual consumption of $z \in M$ proportionally to the upper and
    lower consumption limits of this $z$
12: $AbnormalMetrics_j$: the number of abnormal usages of all the metrics by $j$.
    Output:
13: $\Gamma_j$: trust value of $j$

14: Initialize $AbnormalMetrics_j$ to 0
15: for each metric $z \in M$ do
16:     Initialize $AbnormalMetrics_j^z$ and $CountAbnormalMetrics_j^z$ to 0
17:     Initialize $AvgAbnormalMetrics_j^z$ and $PropAbnormalMetrics_j^z$ to 0
18:     Compute the median $\bar{x}_j^z(t)$ of $M_j^z(t)$
19:     Compute the $MAD_j^z(t)$ of $M_j^z(t)$
20:     Compute $\alpha_j^z(i,t) = \frac{\varrho(x_j^z(i) - \bar{x}_j^z(t))}{MAD_j^z(t)}$
21:     for each data point $x_j^z(i) \in M_j^z(t)$ do
22:         if $\alpha_j^z(i,t) \ge \varphi$ then
23:             $AbnormalMetrics_j^z = AbnormalMetrics_j^z + x_j^z(i)$
24:             $CountAbnormalMetrics_j^z = CountAbnormalMetrics_j^z + 1$
25:         end if
26:     end for
27:     if $CountAbnormalMetrics_j^z > 0$ then
28:         $AvgAbnormalMetrics_j^z = AbnormalMetrics_j^z / CountAbnormalMetrics_j^z$
29:         $PropAbnormalMetrics_j^z = \frac{\varphi}{AvgAbnormalMetrics_j^z}$
30:         $AbnormalMetrics_j = AbnormalMetrics_j + 1$
31:     end if
32: end for
33: if $AbnormalMetrics_j = 0$ then
34:     $\Gamma_j = 1$
35: else
36:     $\Gamma_j = \frac{\sum_{z \in M} PropAbnormalMetrics_j^z}{AbnormalMetrics_j}$
37: end if
38: return $\Gamma_j$
```

to the current state of the environment. In this work, we propose DDQN-Trust, a trust and energy-aware dynamic double deep Q network scheduling algorithm. The proposed method consists of a multi-layered neural network that, for a given state outputs a vector of actions given a set of parameters of the network. The problem is formulated as a global Markov Decision Process (MDP) where the system global states and global actions are formulated as the combination of IoT devices local states and actions. It is defined by the tuple $\langle S, A, T, R, \gamma \rangle$, where:

- S: the set of global states of the system.
- A: the set of joint actions of all the IoT devices.
- T: the transition probability function defined as: $T(s, a, s') = Pr(s'|s, a)$, where $s, s' \in S$ and $a \in A$.
- $R : S \times A \times S \mapsto \mathbb{R}$: the reward function of the model.
- γ: a discount factor that decreases the impact of the past reward.

Let S_j be the set of local states of the IoT device j and J the set of all the devices. The global state space S is obtained through the Cartesian product of IoT devices local states: $S = \prod\limits_{j \in J} S_j$. Each local state $s_j \in S_j$ is as follows:

$$s_j = (\varGamma_j, \chi_j); \qquad \varGamma_j \in [0, 1], \chi_j \in \{0, 1, \ldots, \chi^{max}\} \tag{1}$$

where $\varGamma_j$ is the trust value of the IoT device j computed in Algorithm 1 and χ_j is the energy state of j. Trust and energy state are dynamic, so they could change from state to state. The global action space of the parameter edge server is the joint action space of each device: $A = \prod\limits_{j \in J} A_j$ where A_j is the set of local actions of j. A local action $a_j \in A_j$ is as follows:

$$a_j = (\sigma_j, l_j^{\chi}, \xi_j); \quad \sigma_j \in \{0, 1\}, l_j^{\chi} \in \{0, 1, \ldots, \chi^{max}\}, \xi_j \in \mathbb{R} \tag{2}$$

where $\sigma_j = 1$ means the parameter server assigns a training task to the IoT device j; $\sigma_j = 0$ otherwise, l_j^{χ} refers to the amount of energy needed by the IoT device j to download, train and upload the model, and ξ_j is the cost of transmitting the model from the parameter server to the device j and running the model. For an action to be feasible from a global state s to s', the following condition should hold:

$$l_j^{\chi}(s') \le \chi_j(s) \quad \forall j \in J \tag{3}$$

where $l_j^{\chi}(s')$ refers to l_j^{χ} in the action leading to s' and $\chi_j(s)$ is χ_j in s. Finally, to define the reward function R, the objective of maximizing the selection of trusted IoT devices having enough energy to receive and perform the training task is considered. The cost ξ_j is also considered proportional to the maximum cost ξ^{max}. The reward ψ_j for the device j is a function of state $s \in S$ and action $a \in A$ as follows:

$$\psi_j(s, a) = \begin{cases} \varGamma_j . \chi_j - \frac{\xi_j}{\xi^{max}}, & \text{if } l_j^{\chi} \le \chi_j. \\ -\frac{\xi_j}{\xi^{max}}, & \text{otherwise.} \end{cases} \tag{4}$$

Thus, along with the trust scores of the IoT devices, the edge server accounts for the available energy level of the devices to make sure that these devices have enough battery capacity to download, train and upload the model.
The global reward of the parameter server is as follows:

$$R(s,a) = \sum_{j \in J} \psi_j(s,a) \tag{5}$$

The parameter edge server determines the optimal policy $\pi^* : S \rightarrow A$ that indicates the actions to be taken at each state to maximize the cumulative reward. The essential goal of the Q-learning (QL) algorithm used to find π^* is to update the Q-value of a state-action pair, $Q(s,a)$, which encodes the expected future discounted reward for taking action a in state s. The optimal action-value function $Q^*(s,a)$ is $Q^*(s,a) = \max_{\pi} Q_{\pi}(s,a)$. This optimal value function can be nested within the Bellman optimality equation as follows:

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s' \in S} Pr(s'|s,a). \max_{a' \in A} Q^*(s',a') \tag{6}$$

Depending on the Q-table that results from updating the $Q(s,a)$ values, the parameter server determines the optimal action from any state to maximize the cumulative reward. The QL algorithm is practical for networks with small state and action spaces only, but when the number of network participants increases (which is the case of IoT networks that consist of a large number of devices), the problem of assigning training tasks to the IoT devices becomes high dimensional. The Deep QL (DQL) algorithm (a combination of QL and deep neural network DNN) comes into play to solve the high dimensionality problem. The input of the DNN is one of states of the online network, and the outputs are the Q-values $Q(s,a;\theta)$ of all the possible actions, with θ being the weight matrix of the DNN. The DNN needs to be trained by using experiences $(s, a, R(s,a), s')$ to obtain the approximate values $Q^*(s,a)$. We use the Mean Square Error (MSE) to define the loss function and DNN uses the Bellman equation to minimize this loss function as follows:

$$L(\theta_i) = E[(R(s,a) + \gamma \arg \max_{a' \in A} Q(s',a';\theta_i') - Q(s,a;\theta_i))^2] \tag{7}$$

where θ_i represents the parameters of the online network at the i^{th} iteration, θ_i' represents the parameters of the target network at the i^{th} iteration, and $E[.]$ denotes the expected value. Note that the action a is selected based on the ϵ-greedy policy. By using the max operator (which uses the same Q-values to select and to evaluate an action in standard QL and DQN), we observe that it is more likely that this operator selects overestimated values, resulting in overoptimistic estimates. To prevent such a problem, we should decouple the action selection from the evaluation by employing the double deep Q-network (DDQN) [13]. The main feature of DDQN is the use of two separate DNNs, i.e., an online network with weight set θ and a target network with weight set θ''. The DDQN employs two valuation functions for two autonomous DNNs learned through randomly assigning experiences to update one of the two value functions, resulting in two sets of weights θ for the first DNN and θ'' for the second DNN. At each iteration, the weights of the online network are updated, while those of the target network are kept constant to determine the greedy policy. The target function of our DDQN-Trust error is defined by:

$$T_{DDQN-Trust}(s, a, s') = R(s, a) + \gamma Q(s', \arg\max_{a' \in A} Q(s', a'; \theta); \theta'') \tag{8}$$

To compute the optimal value $Q(s', a'; \theta)$, the weight θ of the online network uses the next state s' to select an action, while the target network θ'' uses the next state s' to evaluate the action. Then, a stochastic gradient descent step is performed to update the weights of the online networks θ based on the loss

3.3 DDQN-Trust-based Federated Learning Model

In this section, we describe how the FL process can be executed after integrating our trust establishment and scheduling mechanisms. A DNN model is distributed over the IoT devices to be collaboratively trained following the FL framework. Let D_j be a local dataset collected by the IoT device j, $D_j = \{(x_{1_j}, y_{1_j}), \ldots, (x_{n_j}, y_{n_j})\}$, where x_{i_j} is the i^{th} training sample and y_{i_j} represents the corresponding ground-truth label. In this work, we take a general Convolutional Neural Network (CNN) model for analysis. The edge server receives the local gradient vectors from the trusted IoT devices and then aggregates (averages) them to obtain the global gradient using Equation (9):

$$g[\nu] = \frac{1}{\sum\limits_{j \in J} |\vartheta_j|} \sum_{j \in J} |\vartheta_j| g_j[\nu] \tag{9}$$

where ϑ_j is a subset of local data collected from the IoT device j for a training period ν, with $\vartheta_j \subseteq D_j$, and $g_j[\nu]$ being the local gradient which is computed as per Equation (10).

$$g_j[\nu] = \nabla_{w_j} L_j(w_j, \vartheta_j) \tag{10}$$

where w_j is the local parameter set of the CNN model, L_j is the local loss function on the IoT device j to measure the training error and $\nabla_{w_j} L_j(.)$ is the gradient of the loss function L_j with respect to w_j.

Algorithm 2 DDQN-Trust-based Federated Learning Algorithm for IoT Selection

```
1: Initialize the global parameter set of the CNN model
2: for each round $\tau = 1, 2, \ldots$ do
3:     Use Algorithm 1 to compute the trust scores of all the IoT devices
4:     Use DDQN-Trust to select a subset $\mathcal{E} \subseteq J$ of IoT devices to participate in the training
5:     Send $W_\tau$ to each selected IoT
6:     for each IoT device $j \in \mathcal{E}$ do      % $\mathcal{E} = \{1, 2, \ldots, E\}$
7:         Execute IoTLocalUpdate($W_\tau$)      % See Algorithm 3
8:     end for
9:         $W_\tau = \frac{1}{n} \sum_{j=1}^{E} n_j w_j$
10: end for
```

In Algorithms 2 and 3, we describe the federated learning process after embedding our proposed trust establishment mechanism and DDQN-Trust scheduling policy to improve the selection of IoT devices. In Algorithm 2, n_j is the data size available on IoT device j, n is the size of the whole data across all devices, E is the total number of selected devices, τ is the training communication round index and W_τ is the global parameter set at round τ.

Algorithm 3 IoT Local Training

```
1: IoTLocalUpdate($W_\tau$)
2:     $w_j = W_\tau$
3:     for each local iteration $t = 1$ to $T$ do
4:             $w_j = w_j - \eta \nabla_{w_j} L_j(w_j, \vartheta_j)$      % $\eta$ is the learning rate
5:     end for
6:     return $w_j$ to the edge server
```

Each IoT device runs the stochastic gradient descent (SGD) algorithm based on the received global gradient. The local loss function on each device j is defined as per Equation (11):

$$L_j(w_j) = \frac{1}{N_j} \sum_{(x,y) \in D_j} \ell(w_j, x, y) \tag{11}$$

where $\ell\ (w_j, x, y)$ is the sample-wise loss function that quantifies the prediction error between the learning output (via input x and parameter w_j) and the ground-truth label y, and N_j is the number of data samples of the device j. Each device seeks to minimize the local loss function defined in Equation (11) to minimize the training error. On a global level, the main target of the training task at the edge server is to optimize the parameters towards minimizing the global loss function $L(W)$ via the SGD algorithm expressed as follows:

$$L(W) = \frac{1}{\sum_{j=1}^{E} N_j} \sum_{j=1}^{E} N_j L_j(w_j) \tag{12}$$

3.4 Federated Learning Aggregation Approaches

We integrate our solution with four existing federated learning approaches, namely, *FedProx*, *FedShare*, *FedSGD*, and *FedAvg*. The objective is to pick the approach that best suits our solution.

3.4.1 FedAvg

The *FedAvg* approach relies on a single-model strategy that leverages an averaged results across many clients. In *FedAvg*, the server chooses a subset of IoT devices in each communication round and sends the global model back to them. Each device will perform a predefined number of gradient descent iterations on its local data, before pushing the model's weight to the server. Finally, the server averages these weights to generate a new global model. Technically speaking, after receiving the local model w_j^τ and gradient g, the server aggregates them using Algorithm 2 (Line 9) and Equation (9) and shares w and g with the IoT devices.

3.4.2 FedSGD

The main idea of *FedSGD* is to let the IoT devices minimize a surrogate function Φ_j^τ [10] after each global round, as follows:

$$\Phi_j^\tau(w) = \Xi_j(w) + \langle \eta \nabla g^{\tau-1} - \nabla g_j(w^{\tau-1}), w \rangle \tag{13}$$

with Ξ_j being L-smooth and β strongly convex, $\forall j$ [25], and Φ inspired by the Distributed Approximate NEewton (DANE) scheme introduced in [33]. Furthermore, we include both local and global gradient estimates weighted by a controllable parameter η. Thereafter, the IoTs send not only the local model w_j, but also local gradient estimates $\nabla g^{\tau-1}$ to speed the convergence up in the experiments.

3.4.3 FedShare

In *FedShare*, a globally accessed dataset G with a uniform distribution is used to mitigate the impact of data heterogeneity across the client devices. Specifically, a sample of this dataset is given to each IoT device in the initialization phase. The shared data from G is combined with the private data of each device to form the training set of the device's local model. The server then aggregates the local models from the IoT devices using *FedAvg* to construct the global model. In *FedShare*, two trade-offs are considered. The first one is the trade-off between the test accuracy and the size of G, which is quantified as $\beta = \dfrac{\|G\|}{\|\prod\limits_{j \in J} D_j\|} \times 100\%$. The second trade-off is between the test accuracy and the random distributed fraction [39].

3.4.4 FedProx

To a certain extent, *FedProx* is similar to *FedAvg* in that devices are chosen at each round, which means that local updates are made to a subset of devices, and these are then averaged to obtain a global model. Therefore, to reiterate, instead of all training devices having the same number of rounds, *FedProx* indirectly trains for various devices having varying rounds. In other words, instead of assuming a uniform number of local rounds Ω for all devices throughout the training process, *FedProx* implicitly accommodates variable Ω's for different devices and at different iterations. We adopt Ω_j^τ-inexact solution [19] to find w_j^τ for each IoT device j at round τ where Ω_j^τ-inexact minimizer minimizes the following objective h_j:

$$w_j^{\tau+1} \approx argmin_w \;\; h_j(w; w^\tau) = L_j(w_j) + \frac{T}{2}\|w - w^\tau\|^2 \tag{14}$$

4 Implementation and Experiments

In this section, we explain the experimental setup we used to implement our double deep Q network scheduling algorithm. We also report and compare the experimental results.

4.1 Experimental Setup

To carry out our experiments, we used TensorFlow Federated (TFF), which is an open-source framework for machine learning on decentralized data. TFF supports a variety of distributed learning scenarios executed on a large number of heterogeneous devices having diverse capabilities.

To achieve an effective high level accuracy, the FL model requires a labelled dataset. The majority of the data at the edge computing environment are unlabeled, which cannot be used directly to train the model. Data collected by IoT devices may also have a bias towards the environment within which they operate. For example, the vehicle's dashcam that runs on highways mostly captures traffic signs, whereas a video from a car travelling in the city contains various shop signs. This may lead to a high degree of data diversity across those IoT devices. In other words, the data are nonindependent and identically distributed (non-IID). In contrast, most of the existing analytical or machine learning methods are based on IID data. Therefore, there is a need for an appropriate approach to deal with such a type of real-world datasets. In this article, we used real-world data for object detection, and well-known benchmark dataset for image classification. We used the CIFAR-10 [3] dataset for image classification, which consists of $60,000$ 32×32 colour images in 10 classes, with 6000 images per class. The dataset consists of $50,000$ training images and 10000 test images. We divide the CIFAR-10 non-IID data over 1000 IoT devices into 500 shards of size 100 after sorting the whole data by label index, then assign five shards to each device.

[3] https://www.cs.toronto.edu/ kriz/cifar.html

On the other hand, for the object detection, we used the Udacity Self Driving Car Dataset [4], which contains $97,942$ labels across 11 classes and $15,000$ images. There are $1,720$ images with no labels. The employed CNN model consists of six 3×3 convolution layers as follows: 32, 32, 64, 64, 128, 128. Each layer is activated by a Rectified Linear Unit (ReLU) and batch normalized. Every pair of convolution layers is followed by a 2×2 max pooling layer, followed by three fully-connected layers (where each fully connected layer takes a 2D input of 382 and 192 units) with ReLU activation and another 10 units activated by soft-max. The model is trained on IoT devices using the Stochastic Gradient Descent (SGD) algorithm with a batch size of 128 rows. The training dataset was distributed over a set of 1000 IoT devices (i.e., $|J| = 1000$) of 4 types: type-1 with 1 CPU core and 1.75GB RAM, type-2 with 2 CPU cores and 3.5GB RAM, type-3 with 4 CPU cores and 7GB RAM, and type-4 with 8 CPU cores and 14GB RAM. At each iteration, the edge server selects the top 50 IoT devices returned by the scheduling algorithm (i.e., $E = 50$).

We evaluate the performance of the proposed DDQN-Trust solution against the traditional DQN [26] which has lately been used for client selection in federated learning and with the random scheduling approach, the default approach in federated learning. The proposed DDQN-Trust model consists of two Deep Neural Networks (DNNs), where each DNN has a size of $32 \times 32 \times 32$. The Adam optimizer is used to adjust the learning rate during the training. The learning rate η is initially set to 0.01 to avoid losing the local minima. In general, the deep Q learning approach prefers the long-term reward; therefore, we set the value of the discount factor γ to 0.9. We use the ϵ-greedy policy with $\epsilon = 0.9$ that balances between the exploration and exploitation. During the training phase, ϵ is linearly reduced to zero to move from exploration to exploitation. Our application is written in Python, version 3, and executed in a 64-bit Windows 7 environment on a machine equipped with an Intel Core i7-6700 CPU 3.40 GHz Processor and 16 GB RAM. The implementation code is open source and available on GitHub [5].

4.2 Experimental Results

In Fig. 3, we measure the accuracy of our DDQN-Trust approach against the classic DDQN that does not include our trust algorithm. The considered accuracy is yielded by the *FedProx*, *FedShare*, *FedSGD* and *FedAvg* approaches. We ran the experiments over 1000 iterations to analyse the scalabilty of the different considered solutions. The first observation that can be drawn from the figure is that the trust-based approaches, i.e., *FedProx*, *FedShare*, *FedSGD* and *FedAvg* with DDQN-Trust achieve higher accuracy compared to the approaches that do not consider trust, i.e., *FedProx*, *FedShare*, *FedSGD*, and *FedAvg* with DDQN. In particular, the accuracy levels obtained by the *FedProx*, *FedShare*, *FedSGD*,

[4] https://public.roboflow.com/object-detection/self-driving-car
[5] https://github.com/gaith7/Trust_Fed

and *FedAvg* approaches with DDQN-Trust are 93%, 86%, 83%, and 78% respectively, whereas the accuracy levels obtained by the *FedProx*, *FedShare*, *FedSGD*, and *FedAvg* approaches with DDQN are 77%, 72%, 68%, and 65% respectively. The second observation that can be drawn from Fig. 3 is that the trust-based approaches converge faster to a stable accuracy level compared to the non-trust approaches. The improvements brought by the trust-based approaches mainly stem from their consideration of the trust and energy values when selecting the devices that will participate in the federated training.

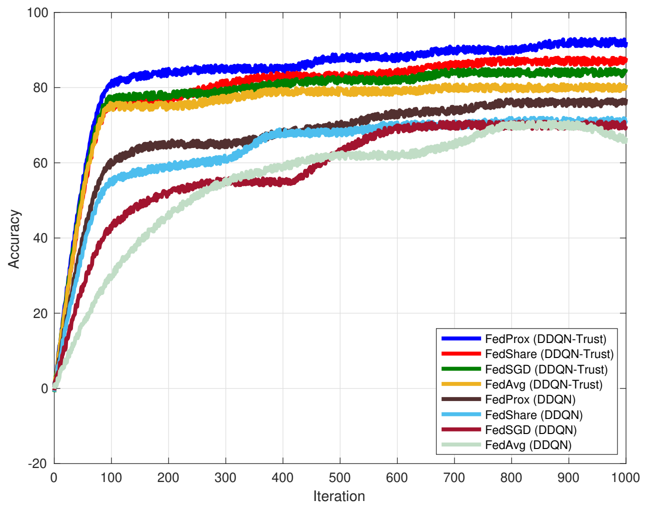
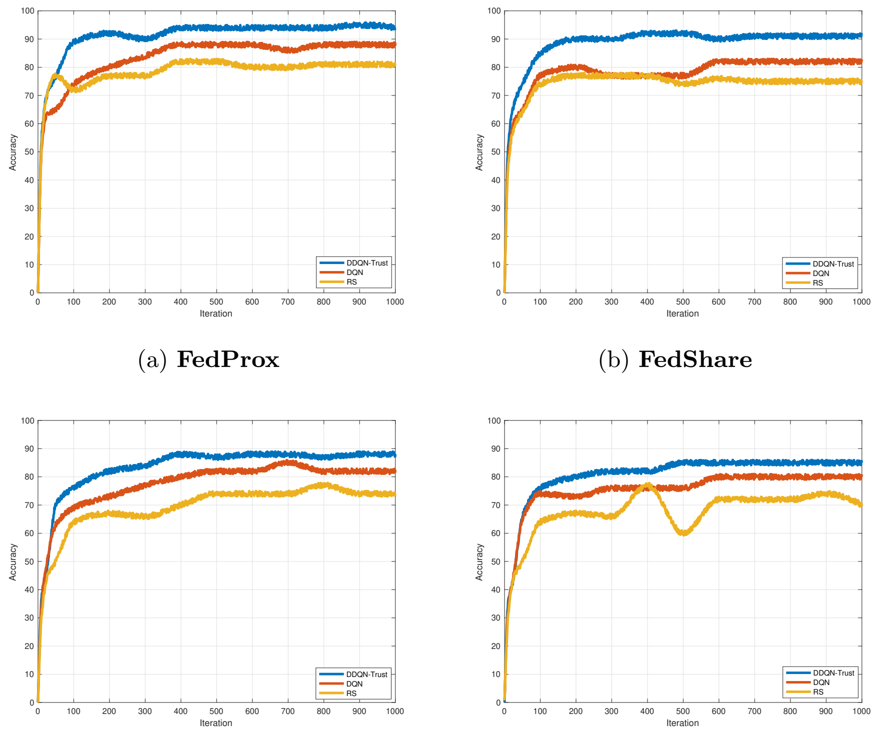


Fig. 3: Accuracy over iteration rounds of different aggregation methods with the CNN model of our DDQN-Trust and classic DDQN

In Fig. 4, we compare the accuracy of our trust-based approach against the DQN and random scheduling approaches used to select the IoT devices while varying the federation approaches. Specifically, we compare the studied approaches (i.e., *FedProx*, *FedShare*, *FedSGD*, and *FedAvg*) with the DDQN-Trust, DQN, and random scheduling approaches over 1000 iterations. We notice from the sub-figures that the accuracy obtained by the DDQN-Trust scheduling approach is higher than that obtained by the DQN and random approaches under the different aggregation methods. In particular, the accuracy levels obtained by the DDQN-Trust, DQN, and random scheduling are of 94%, 86%, and 77% respectively with the *FedProx* aggregation framework, 91%, 82%, and 74% respectively with the *FedShare* framework, 88%, 81%, and 72% respectively with the *FedSGD* framework, and 83%, 74%, and 69% respectively with the *FedAvg* framework. We also notice from the sub-figures that DDQN-Trust converges to a stable accuracy level faster than the DQN and random approaches. The im-



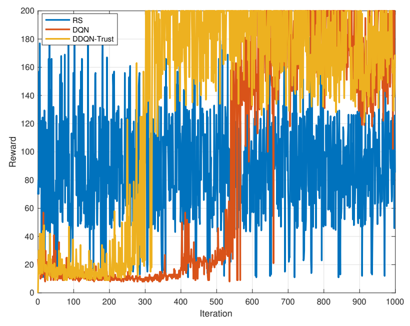
(a) **FedProx** (b) **FedShare**

(c) **FedSGD** (d) **FedAvg**

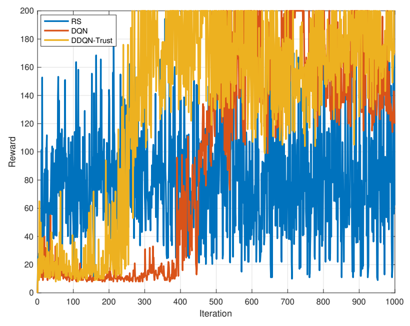
Fig. 4: Performance of the trained CNNs with DDQN-Trust, DQN, and RS scheduling models

provements with regard to the random scheduling approach mainly stem from the fact that DDQN-Trust leverages the trust and energy values of the IoT devices rather than making random selections. Compared to the traditional DQN, DDQN-Trust improves the accuracy since it relies on a double Q learning model that provides a better estimation of the potential actions. This is thanks to its second Q-function approximator, which helps avoid overoptimism. In DQN, on the other hand, the Q values are noisy; thus when we take the maximum over all the actions, there is a considerable risk of obtaining an overestimated value.

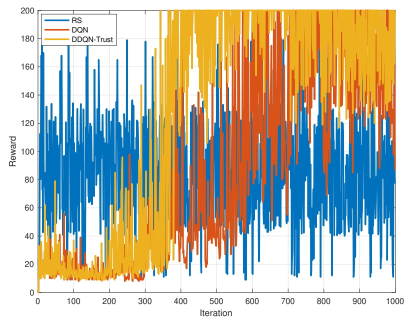
One important observation from Fig. 3 and Fig. 4 is that *FedProx* has the highest accuracy and the fastest convergence compared to the other three aggregation approaches. This is because *FedProx* addresses the system and statistical heterogeneity across the IoT devices, which makes its aggregation results more accurate. On the other hand, the classic *FedAvg* yields the lowest accuracy level and the slowest convergence, may be due to the fact that when the local data distributions across the devices are highly heterogeneous, the local updating schemes may allow local models to move too far away from the ini-



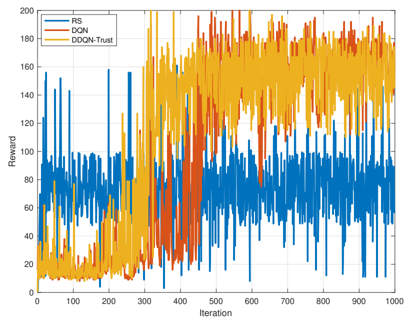
(a) **FedProx**



(b) **FedShare**



(c) **FedSGD**



(d) **FedAvg**

Fig. 5: Reward values in DDQN-Trust, DQN, and Random scheduling policies

tial global model, potentially damaging the convergence. FedShare and *FedSGD* are characterized by an acceptable accuracy level. In fact, *FedShare* depends on a data-sharing strategy that seeks to distribute a small subset of independent global data, characterized by a uniform distribution over classes, to the local devices. This helps the federated learning model be less influenced by the data heterogeneity across the IoT devices. On the other hand, *FedSGD* relies on a local surrogate function that is designed for each IoT device to allow it to solve its local optimization problem approximately up to a local accuracy level.

In Fig. 5, we provide experimental comparisons in terms of cumulative reward. We ran the experiments over 1000 iterations. We notice from the subfigures that the reward obtained by the DDQN-Trust is much higher than those obtained by the DQN and random scheduling approaches. In particular, the average rewards obtained by the DDQN-Trust, DQN, and random approaches are 188, 174, and 107, respectively with the *FedProx* framework (Fig. 5a), 181, 167, and 97 respectively with the *FedShare* framework (Fig. 5b), 177, 162, and 95 respectively with *FedSGD* (Fig. 5c), and 162, 160, and 81 respectively with *FedAvg* (Fig. 5d). This means the proposed DDQN-Trust approach enables the edge server to better learn the scheduling policy that best maximizes the reward. In the random approach, the edge server randomly selects IoT devices, which increases the risk of selecting unreliable devices or devices that have insufficient energy levels and resources. This endangers the whole collaborative training process and makes the performance unstable. Moving to the traditional DQN approach, its overestimation of the future actions leads to a natural reduction in the overall reward that results from the chosen actions.

5 Conclusion

We designed and formulated a trust and energy-aware FL scheduling approach in IoT environments using DDQN while considering four aggregation approaches namely, *FedAvg*, *FedProx*, *FedShare*, and *FedSGD*. Experiments conducted on the CIFAR-10 real-world dataset reveal that our DDQN-Trust solution outperforms, in terms of accuracy and cumulative reward, the most commonly used scheduling approaches in FL, i.e., DQN and random scheduling. Our solution accurately selects the appropriate set of IoT devices whose participation in the federated training improves the machine learning model's accuracy. We studied the accuracy of the three models by implementing a CNN model in a federated fashion on the IoT devices and varying the aggregation approaches. The results revealed that our DDQN-Trust solution, DQN, and random scheduling yield respectively an accuracy of 94%, 86%, and 77% with the *FedProx* aggregation framework, 91%, 82%, and 74% respectively with *FedShare*, 88%, 81%, and 72% with *FedSGD*, and 83%, 74%, and 69% with *FedAvg*. Besides, our DDQN-Trust converges faster to a stable accuracy level. Finally, the results revealed that the reward obtained by our proposed solution is much higher than those obtained by the DQN and random scheduling approaches. In particular, the average rewards obtained by the DDQN-Trust, DQN, and random approaches are 188, 174, and 107, respectively with *FedProx*, 181, 167, and 97 respectively with *FedShare*, 177, 162, and 95 respectively with *FedSGD*, and 162, 160, and 81 respectively with *FedAvg*. In the future, we plan to extend this work by investigating and formulating the scheduling approach using Dueling Double Deep Q Network (DDDQN) and actor-critic, which could help better reduce the overhead and time complexity of the scheduling process by avoiding unnecessary computations.

References

1. AbdulRahman, S., Tout, H., Mourad, A., Talhi, C.: FedMCCS: multicriteria client selection model for optimal IoT federated learning. IEEE Internet of Things Journal **8**(6), 4723–4735 (2020)
2. Anh, T.T., Luong, N.C., Niyato, D., Kim, D.I., Wang, L.C.: Efficient training management for mobile crowd-machine learning: A deep reinforcement learning approach. IEEE Wireless Communications Letters **8**(5), 1345–1348 (2019)
3. Bataineh, A.S., Bentahar, J., Mizouni, R., Abdel Wahab, O., Rjoub, G., El Barachi, M.: Cloud computing as a platform for monetizing data services: A two-sided game

business model. IEEE Transactions on Network and Service Management p. In Press (2021). https://doi.org/10.1109/TNSM.2021.3128160
4. Bataineh, A.S., Bentahar, J., Wahab, O.A., Mizouni, R., Rjoub, G.: A game-based secure trading of big data and IoT services: Blockchain as a two-sided market. In: International Conference on Service-Oriented Computing. pp. 85–100. Springer (2020)
5. Bentahar, J., Drawel, N., Sadiki, A.: Quantitative group trust: A two-stage verification approach. In: Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), P. Faliszewski, V. Mascardi, C. Pelachaud, M.E. Taylor (eds.), May 9–13, 2022,. pp. xx–xx (2022)
6. Chen, M., Yang, Z., Saad, W., Yin, C., Poor, H.V., Cui, S.: A joint learning and communications framework for federated learning over wireless networks. CoRR **abs/1909.07972** (2019)
7. Chen, S., Shen, C., Zhang, L., Tang, Y.: Dynamic aggregation for heterogeneous quantization in federated learning. IEEE Transactions on Wireless Communications (2021)
8. Dai, H., Zeng, X., Yu, Z., Wang, T.: A scheduling algorithm for autonomous driving tasks on mobile edge computing servers. J. Syst. Archit. **94**, 14–23 (2019)
9. Ding, D., Fan, X., Zhao, Y., Kang, K., Yin, Q., Zeng, J.: Q-learning based dynamic task scheduling for energy-efficient cloud computing. Future Gener. Comput. Syst. **108**, 361–371 (2020)
10. Dinh, C.T., Tran, N.H., Nguyen, M.N., Hong, C.S., Bao, W., Zomaya, A.Y., Gramoli, V.: Federated learning over wireless networks: Convergence analysis and resource allocation. IEEE/ACM Transactions on Networking (2020)
11. Drawel, N., Bentahar, J., Laarej, A., Rjoub, G.: Formalizing group and propagated trust in multi-agent systems. In: Bessiere, C. (ed.) Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020. pp. 60–66 (2020). https://doi.org/10.24963/ijcai.2020/9
12. Drawel, N., Bentahar, J., Laarej, A., Rjoub, G.: Formal verification of group and propagated trust in multi-agent systems. Autonomous Agents and Multi-Agent Systems p. In Press (2022). https://doi.org/10.1007/s10458-021-09542-6
13. van Hasselt, H., Guez, A., Silver, D.: Deep reinforcement learning with double Q-learning. In: Schuurmans, D., Wellman, M.P. (eds.) Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA. pp. 2094–2100. AAAI Press (2016)
14. Hu, C., Jiang, J., Wang, Z.: Decentralized federated learning: A segmented gossip approach. CoRR **abs/1908.07782** (2019)
15. Huang, J., Li, S., Chen, Y.: Revenue-optimal task scheduling and resource management for IoT batch jobs in mobile edge computing. Peer Peer Netw. Appl. **13**(5), 1776–1787 (2020)
16. Iglewicz, B., Hoaglin, D.C.: How to detect and handle outliers, vol. 16. Asq Press (1993)
17. Khan, L.U., Saad, W., Han, Z., Hossain, E., Hong, C.S.: Federated learning for internet of things: Recent advances, taxonomy, and open challenges. IEEE Communications Surveys & Tutorials (2021)
18. Lei, L., Tan, Y., Zheng, K., Liu, S., Zhang, K., Shen, X.: Deep reinforcement learning for autonomous internet of things: Model, applications and challenges. IEEE Commun. Surv. Tutorials **22**(3), 1722–1760 (2020)
19. Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A., Smith, V.: Federated optimization in heterogeneous networks. arXiv preprint arXiv:1812.06127 (2018)

20. Lin, C., Deng, D., Chih, Y., Chiu, H.: Smart manufacturing scheduling with edge computing using multiclass deep Q network. IEEE Trans. Ind. Informatics **15**(7), 4276–4284 (2019)
21. Luo, S.: Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning. Appl. Soft Comput. **91**, 106208 (2020)
22. Ma, Z., Zhao, M., Cai, X., Jia, Z.: Fast-convergent federated learning with class-weighted aggregation. Journal of Systems Architecture **117**, 102125 (2021)
23. Mao, H., Alizadeh, M., Menache, I., Kandula, S.: Resource management with deep reinforcement learning. In: Proceedings of the 15th ACM Workshop on Hot Topics in Networks. pp. 50–56 (2016)
24. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial Intelligence and Statistics. pp. 1273–1282. PMLR (2017)
25. Nesterov, Y., et al.: Lectures on convex optimization, vol. 137. Springer (2018)
26. Nguyen, H.T., Luong, N.C., Zhao, J., Yuen, C., Niyato, D.: Resource allocation in mobility-aware federated learning networks: A deep reinforcement learning approach. In: 6th IEEE World Forum on Internet of Things, WF-IoT 2020, New Orleans, LA, USA, June 2-16, 2020. pp. 1–6. IEEE (2020)
27. Nishio, T., Yonetani, R.: Client selection for federated learning with heterogeneous resources in mobile edge. In: 2019 IEEE International Conference on Communications, ICC 2019, Shanghai, China, May 20-24, 2019. pp. 1–7. IEEE (2019)
28. Rjoub, G., Bentahar, J., Abdel Wahab, O., Saleh Bataineh, A.: Deep and reinforcement learning for automated task scheduling in large-scale cloud computing systems. Concurrency and Computation: Practice and Experience (2020)
29. Rjoub, G., Bentahar, J., Wahab, O.A.: Bigtrustscheduling: Trust-aware big data task scheduling approach in cloud computing environments. Future Gener. Comput. Syst. **110**, 1079–1097 (2020)
30. Rjoub, G., Bentahar, J., Wahab, O.A., Bataineh, A.S.: Deep smart scheduling: A deep learning approach for automated big data scheduling over the cloud. In: 7th International Conference on Future Internet of Things and Cloud, FiCloud 2019, Istanbul, Turkey, August 26-28, 2019. pp. 189–196. IEEE (2019)
31. Rjoub, G., Wahab, O.A., Bentahar, J., Bataineh, A.: A trust and energy-aware double deep reinforcement learning scheduling strategy for federated learning on IoT devices. In: International Conference on Service-Oriented Computing. pp. 319–333. Springer (2020)
32. Rjoub, G., Wahab, O.A., Bentahar, J., Bataineh, A.S.: Improving autonomous vehicles safety in snow weather using federated YOLO CNN learning. In: International Conference on Mobile Web and Intelligent Information Systems. pp. 121–134. Springer (2021)
33. Shamir, O., Srebro, N., Zhang, T.: Communication-efficient distributed optimization using an approximate newton-type method. In: International conference on machine learning. pp. 1000–1008. PMLR (2014)
34. Wahab, O.A., Bentahar, J., Otrok, H., Mourad, A.: Resource-aware detection and defense system against multi-type attacks in the cloud: Repeated bayesian stackelberg game. IEEE Transactions on Dependable and Secure Computing (2019)
35. Wahab, O.A., Cohen, R., Bentahar, J., Otrok, H., Mourad, A., Rjoub, G.: An endorsement-based trust bootstrapping approach for newcomer cloud services. Inf. Sci. **527**, 159–175 (2020)
36. Wahab, O.A., Mourad, A., Otrok, H., Taleb, T.: Federated machine learning: Survey, multi-level classification, desirable criteria and future directions in communication and networking systems. IEEE Communications Surveys & Tutorials (2021)

37. Wang, X., Han, Y., Wang, C., Zhao, Q., Chen, X., Chen, M.: In-edge AI: intelligentizing mobile edge computing, caching and communication by federated learning. IEEE Netw. **33**(5), 156–165 (2019)
38. Yang, H.H., Liu, Z., Quek, T.Q.S., Poor, H.V.: Scheduling policies for federated learning in wireless networks. IEEE Trans. Commun. **68**(1), 317–333 (2020)
39. Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., Chandra, V.: Federated learning with non-iid data. arXiv preprint arXiv:1806.00582 (2018)
40. Zhou, Z., Yang, S., Pu, L., Yu, S.: CEFL: online admission control, data scheduling, and accuracy tuning for cost-efficient federated learning across edge nodes. IEEE Internet Things J. **7**(10), 9341–9356 (2020)
41. Zhu, G., Liu, D., Du, Y., You, C., Zhang, J., Huang, K.: Toward an intelligent edge: Wireless communication meets machine learning. IEEE Communications Magazine **58**(1), 19–25 (2020)