Information Systems Frontiers
https://doi.org/10.1007/s10796-022-10307-z

Trust-Augmented Deep Reinforcement Learning for Federated Learning Client Selection

Gaith Rjoub[1] · Omar Abdel Wahab[2] · Jamal Bentahar[1] · Robin Cohen[3] · Ahmed Saleh Bataineh[1]

Accepted: 8 June 2022

Abstract
In the context of distributed machine learning, the concept of federated learning (FL) has emerged as a solution to the privacy concerns that users have about sharing their own data with a third-party server. FL allows a group of users (often referred to as *clients*) to locally train a single machine learning model on their devices without sharing their raw data. One of the main challenges in FL is how to select the most appropriate clients to participate in the training of a certain task. In this paper, we address this challenge and propose a trust-based deep reinforcement learning approach to select the most adequate clients in terms of resource consumption and training time. On top of the client selection mechanism, we embed a transfer learning approach to handle the scarcity of data in some regions and compensate potential lack of learning at some servers. We apply our solution in the healthcare domain in a COVID-19 detection scenario over IoT devices. In the considered scenario, edge servers collaborate with IoT devices to train a COVID-19 detection model using FL without having to share any raw confidential data. Experiments conducted on a real-world COVID-19 dataset reveal that our solution achieves a good trade-off between detection accuracy and model execution time compared to existing approaches.

Keywords Federated learning · Deep reinforcement learning · Transfer learning · Internet of things (IoT) · Edge computing · COVID-19 detection

1 Introduction

The Internet of Things (IoT) is increasingly being used by the public and private sectors to process personal and industrial data. This inevitably means that large amounts of data are being generated on a daily basis. These data are often used to train machine learning models, which can then be used to detect, classify, and predict future events. For this purpose, centralized machine learning approaches are usually employed, where data from different IoT devices to transmitted to a central server whose task is training a machine learning model over these data. Unfortunately, this approach has been lately criticized for breaching users' data privacy as these data need to be shared with a third-party server. Federated learning (FL) can provide a solution to this problem through enabling an on-device cooperative training of the machine learning model (Wahab et al., 2021; Zhang et al., 2021; Rjoub et al., 2022; Rjoub et al., 2021b) across many users (called **clients**). We propose in this paper a trust-based client selection mechanism for FL using deep reinforcement learning (DRL) to enable the system to select the most appropriate clients in terms of resource utilization

✉ Jamal Bentahar
bentahar@ciise.concordia.ca

Gaith Rjoub
g_rjoub@encs.concordia.ca

Omar Abdel Wahab
omar.abdulwahab@uqo.ca

Robin Cohen
rcohen@uwaterloo.ca

Ahmed Saleh Bataineh
ah_batai@encs.concordia.ca

[1] Concordia Institute for Information Systems Engineering, Concordia University, 1455 De Maisonneuve Blvd. W.2, Montreal H3G 1M8, Quebec, Canada

[2] Department of Computer Science and Engineering, Université du Québec en Outaouais, 101, Saint-Jean-Bosco, C.P. 1250, succursale Hull, Gatineau J8X 3X7, Quebec, Canada

[3] David R. Cheriton School of Computer Science, University of Waterloo, 200 University Avenue West, Waterloo N2L 3G1, ON, Canada

Published online: 18 July 2022

and training time. As case study, we consider an IoT-enabled healthcare application in which a collection of IoT devices are asked to collaboratively train a COVID-19 detection model using FL without having to share any raw confidential data.

1.1 Motivations of the Work

IoT is one of the most significant advances in the sector of information technology. IoT stays at the intersection of the concepts of edge computing, cloud computing and networking and has been applied in the few past years in many critical domains such as healthcare, traffic planning, and military monitoring tasks. Interestingly, there has lately been many attempts to capitalize on IoT to boost the research efforts on healthcare domain. IoT has been mainly investigated for effective tracing of the patients and the suspicious cases (Singh et al., 2020; Al-Dhaen et al., 2021; Li et al., 2015; Bataineh et al., 2021).

Edge computing is another interesting technology, which has been proposed to complement the concept of cloud computing through enabling the execution of certain data processing tasks at the edge of the network with lower latency. Edge computing is particularly appropriate for boosting the performance of executing deep learning models, as it enables the offloading of some parts of the deep learning layers to the edge servers (ESs) and then the transfer of only the reduced intermediate data to the central cloud server. Recent studies (Rahman et al., 2020; Brunese et al., 2020; Tuli et al., 2020; Otoom et al., 2020; Bataineh et al., 2020; Rjoub et al., 2020a) have tried to employ IoT over cloud and edge computing environments to analyze data stored on ESs for high-accuracy detection results (i.e., healthcare, transport, etc.).

FL is one of the highly effective paradigm shifts in recent years in AI-based computing. It allows us to obtain better results by cooperatively training a single machine learning model instead of forcing each edge machine to share its actual input data. The FL architecture consists of two phases, namely, global computing and local training. In the local training phase, a parameter server, such as an ES, initializes the machine learning algorithm and shares the initial parameters with the end/edge devices (e.g., IoT devices). The shared parameters are thereafter used by these devices to train the model on their own data. Then, the devices share the modified parameters acquired from the training of the model on their data with the parameter server. Global computing allows the whole model to be reconstructed by aggregating all the received parameter updates in coordination with all the IoT devices. This method is repeated until a certain degree of accuracy is achieved. The applications of FL in medical big data are quite promising (Wahab et al., 2021; Brisimi et al., 2018; Kumar et al., 2021).

As a proof of concept, we implemented our trust-based DRL solution for FL client selection on a healthcare scenario (COVID-19 detection) as shown in Fig. 1. Different layers and technologies are involved in this complex scenario. As illustrated in Fig. 1, our architecture consists of three layers: the public environment layer, the physical layer, which has sensors (cameras) for sensing and gathering information about the environment (COVID-19 detection for the case study), the edge computing and aggregation server layer, and the smart IoT devices layer. The role of the edge computing server is to aggregate local models from IoT devices into a global model based on devices' local data. The IoT devices only share their local models with the edge computing server, rather than their local data, in order to protect the privacy. The purpose is to investigate the effectiveness of a trust-based selection mechanism in improving the performance of FL in this scenario. We also aim to explore how the integration of state-of-the-art concepts such as FL, DRL, transfer learning (TL), trust management, IoT and edge computing could contribute in an improved detection of COVID-19 cases.

1.2 Problem Statement

In the healthcare domain, FL can be extremely useful to analyze huge amounts of heterogeneous data from a multitude of sources (e.g., hospitals, clinics, smartphones, and IoT devices) in a privacy-preserving fashion, while eliminating the need to share the raw data with a third-party platform. This is of great importance to encourage citizens and medical centres to participate in any detection process.

In this paper, we argue that FL, combined with IoT and edge computing, has a lot to offer to advance the research in the area of collaborative computing, in particular collaborative detection methods. Yet, a main challenge toward applying these technologies together is how to select the IoT devices to perform the FL tasks and how to transfer the learning among ESs. In fact, it is of prime importance to select devices that enjoy enough computing resources that enable them to perform local training. Moreover, it is crucial to select trusted IoT devices to avoid having (intentionally or unintentionally) bogus or poor-quality results (Bentahar et al., 2022; Drawel et al., 2021, 2022; Wahab et al., 2020, 2022). For example, since the training is carried out locally at the level of the IoT devices, some malicious devices might optimize for a malicious objective that aims to generate targeted wrong results such as misclassifications. Some other devices might not dedicate enough resources to the local training, which could lead to poor-quality results. To address this challenge, we propose a DRL-based scheduling algorithm that takes into account the resources' availability and trust scores of the IoT devices to schedule the tasks, which we then apply to the COVID-19 case study.

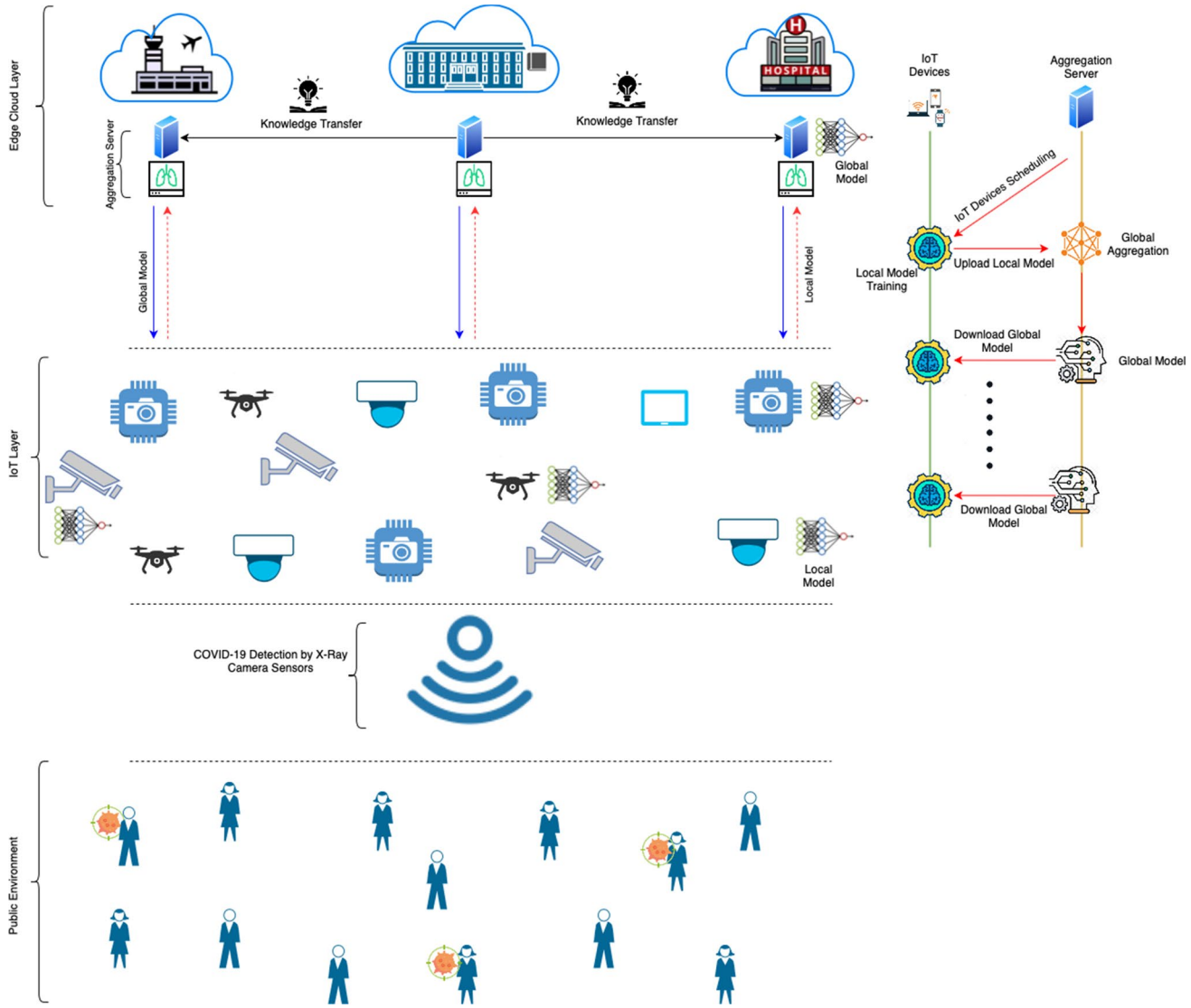


Fig. 1 System architecture and communication process of federated transfer learning in edge cloud

Furthermore, due to the variability in the availability and volumes of data from one region to another and the overhead of constantly training from scratch, it is important to come up with an effective learning sharing approach. Therefore, we complement our solution with a TL approach that is integrated into the FL paradigm to allow the ESs to transfer pre-trained models from one server to another, thus reducing the training time and handling the scarcity of data in some areas.

In a preliminary version of our investigation (Rjoub et al., 2020b), we put forward a scheduling algorithm for FL, which assists the server in selecting the subset of IoT devices to minimize resource consumption while maximizing the overall trust of the process. This paper extends our previous work by (1) adapting and testing our solution in a COVID-19 scenario; and (2) integrating a TL method into our solution to enable inter-server knowledge sharing to handle the problem of data scarcity in some regions.

1.3 Contributions

Our solution consists of a client selection mechanism in FL that relies on a resource and trust-aware DRL strategy. To illustrate the benefits of the solution, we consider the concrete scenario of IoT devices that communicate with ESs to provide detection-aware IoT services. Each set of IoT devices is managed by an ES. The ES initializes a FL model using a publicly available dataset to derive initial model parameters. It then uses our resource and trust-aware DRL scheduling algorithm to select the set of IoT devices to participate in the detection process. The initial set of parameters is hence shared with the selected IoT devices. Upon receiving these

parameters, each IoT device uses them to locally train the machine learning model on its collected data. The different IoT devices then share the set of updated parameters with the ES, which then aggregates the parameters to derive a global model. This process repeats until a desired accuracy level is attained. The knowledge achieved at the ES is then passed to other ESs, resulting in faster data access and reduced training time while addressing the issue of data shortage at some of these devices. This scarcity might be caused by the lack of suitable IoT devices that can participate in the training process or the small crowd in some areas (e.g., airports, shopping malls, etc.). The main contributions of the paper can be summarized by the following points:

- In order to find the most efficient FL selection decision in terms of level of trust and execution time, we put forward a DRL scheduling algorithm. Technically speaking, We begin by formulating a stochastic optimization problem to determine the collection of IoT devices to which the FL tasks will be sent, where the goal of each ES is to minimize the execution time while maximizing the trust of the overall process. The optimization problem is then solved using a DRL algorithm that models the server's uncertainty about the resource and trust levels of IoT devices about which the server has little direct control.
- We propose a multi-faceted approach for a concrete detection-driven scenario, which integrates relevant state-of-the-art approaches and technologies such as federated transfer learning, IoT, edge computing, trust management and DRL. To the best of our knowledge, no existing approach has yet considered the integration and interconnection between all these technologies for a detection method. This makes our approach the most holistic in the literature through considering the different aspects that are necessary for detection such as monitoring and tracking (using IoT and edge devices), trust-aware participant selection and job scheduling (using trust management and DRL), privacy-preserving machine learning (using FL) and training time reduction (using TL).

  We study the performance of the proposed solution experimentally on a COVID-19 dataset. We compare the accuracy of our solution under different combinations, i.e., Trust, Deep Reinforcement learning, Federated and Transfer learning (TDRFT); Trust, Deep Reinforcement learning, and Federated learning (TDRF); and Deep Reinforcement learning, Federated and Transfer learning (DRFT). We also compare these different combinations of our solution with two common scheduling approaches, i.e., Round Robin (RR) and Random Scheduling (RS) while integrating our trust establishment solution into them. In particular, the average accuracy obtained by the TDRFT, TDRF, DRFT, RR and RS approaches are 97.3%,99.4%; 96.4%,98.8%; 94.2%,97.1%; 90%,93.2%; and 87.3%,92.6% respectively.

1.4 Paper Organization

The remainder of this article is organized as follows. In Section 2, we review the literature on current IoT and edge computing scheduling approaches, on existing FL selection approaches, and on COVID-19 detection. In Section 3, we present our solution where the different components are discussed. In Section 4, we explain the experimental setting and comprehensively assess the performance of the proposed approach through experiments and simulations against a benchmark of three selection approaches. Lastly, Section 5 concludes the paper and highlights some possible future extensions.

2 Related Work

In this section, we first survey the main scheduling approaches proposed for edge and cloud computing systems and the main resource management approaches that employ FL to automate the process, and then discuss the COVID-19 detection approaches

2.1 Task Scheduling in IoT and Edge Computing Environments

Since our approach involves a scheduling component over IoT and edge devices, we survey in this section the relevant literature in this field (Arisdakessian et al., 2020). Lei et al. (2019) propose a semi-distributed scheduling algorithm in narrow-band IoT and edge computing systems. The objective is to minimize the long-term average weighted sum of delay and power consumption over all the IoT devices under stochastic traffic arrival. Technically speaking, the stochastic arrival model has been formalized using a dynamic optimization problem. The underlying framework is a continuous-time Markov decision process with infinite-horizon and average reward. The curse-of-dimensionality issue has been addressed using approximate dynamic programming techniques, including linear function approximation and semi-gradient TD learning. Hu and Li (2019) study the request scheduling problem in ultra-dense edge computing (UDEC) networks and provide a non-cooperative game model based on sub-gradients. The considered UDEC network consists of a macro base station, many micro stations, and a large number of mobile users under the 5G architecture. Zhai et al. (2020) introduce a DRL-based approach to deploy mobile edge services in 5G networks. They consider the resource constraints of users along with the request patterns to reduce the total response time by increasing the number

of services executed on the ESs. The problem is solved using the Dueling-Deep Q-network algorithm deployed over a Markov decision process to learn the access patterns on ESs. To improve the management and consumption of resources in cloud-based systems, Song et al. (2018) tackle the host load prediction challenge. The problem in cloud environments is establishing precise forecasts because of the large load variance. To address this problem, a long short-term memory model (LSTM) is being developed to forecast the mean host load in future intervals.

A two-step framework is recommended in Liu et al. (2017) to cover both the distribution of VM resources to servers and the control of power on each individual server. Focusing on DRL to ensure effective allocation of VM resources on servers constitutes the first step. In the second step, LSTM along with weight sharing are used to handle server power management efficiently.

Luo et al. (2019) discuss a multi-fog scheduling approach. The proposed solution considers the transmission energy consumption of the devices and schedules requests in real-time by making use of a dynamic threshold mechanism. The solution guarantees the energy balancing of the devices without increasing the delay. Rjoub and Bentahar (2017) suggest a scheduling method named "Multi Label Classifier Chains Swarm Intelligence (MLCCSI)" with the aim of reducing the job scheduling makespan. Two approaches are introduced. The first approach uses the Ant Colony Optimization (ACO), the Particle Swarm Optimization (PSO), and the Artificial Bee Colony (ABC) to find the best resource allocation solution. In the second approach, the best algorithm to run each appropriate task is predicted using an automatic learning technique based on various variables such as the number of virtual machines (VMs) and the size of the task to reinforce the load balancing and throughput of cloud networks.

FMPSO, a "hybrid task scheduling algorithm" that uses a fuzzy approach combined with the Modified PSO strategy is investigated by Mansouri et al. (2019). To improve the global search capability, FMPSO considers four updated velocity mechanisms along with a selection strategy. Then, to solve any of the PSO's drawbacks, it employs "cross-over and mutation operators" employed in genetic algorithms. Finally, the fitness values are computed using fuzzy inference in the proposed process. A multi-objective optimisation problem is proposed by Gomathi et al. (2018), which aims to optimize three conflicting goals, namely use of resources, cost of execution, and makespan. The "composite discrete artificial bee colony (EDCABC)" centered in the Epsilon-fuzzy domination is then used to extract optimum solutions from Pareto for multi objective task scheduling in the cloud. Zhou et al. (2019) suggest a new algorithm that aims at enhancing the task scheduling process by exploiting a greedy strategy-driven genetic algorithm. Qiu et al. (2016) use deep learning to forecast VM workloads. The Deep Belief Network (DBN) in particular is constructed using regression layers and restricted Boltzmann multi-layered machines. The regression layer is used to forecast potential VM workloads, whereas the DBN layer is executed to extract important features from the VM workload data.

In all these approaches, the main objective is to analyze the historical data obtained from IoT devices to predict potential workload and, as a result, improve IoT allocation processes. However, none of these approaches has yet addressed the issue of dynamic task scheduling automation in complex and large-scale edge computing systems. In fact, no previous study has combined FL and DRL for dynamic scheduling in the way investigated in this paper.

2.2 Client Scheduling in Federated Learning Environments

Hu et al. (2019) propose a decentralised FL at the segment level to enhance the efficiency of network resources usage among client devices. The authors explicitly recommend a segmented gossip strategy, which makes maximum use of node-to-node bandwidth and achieves strong convergence training. Nishio and Yonetani (2019) introduce a protocol that optimizes FL's efficiency with a heterogeneous client in a mobile edge computing setting called *FedCS*. *FedCS* provides a solution for a resource-constrained client selection problem that enables the server to aggregate client updates and speed up the rate of the training convergence.

Chen et al. (2019) address the problem of FL training over wireless realistic networks. They define an optimization problem considering both the user's selection and the allocation of resources to minimize the loss function. The predicted FL algorithm convergence rate, which takes wireless factors into account, is expressed in a closed-form. Nguyen et al. (2019) incorporate Deep Q Network (DQN) into a mobility-aware FL network for resource allocation. The authors suggest using the DQN to allow the model owner to find the optimal decisions that take into account the energy consumption and the quality of the selected channels without having prior knowledge about the network. The authors formulate the model owner's decision as a stochastic optimization problem. The optimization problem's goal is to maximize the model owner's number of successful transmissions while minimizing energy and channel costs. Anh et al. (2019) provide a DQN, which enables the server to learn and find optimal decisions without knowing the network dynamics in the first place. They use Mobile Crowd-Machine Learning (MCML) to tackle the traditional machine learning data privacy.

In general, the existing scheduling methods in edge computing, FL, and IoT concentrate primarily on the resource characteristics of the participant devices. To ensure high-quality and consistent output of the FL, we consider both the

resource and trust aspects in this work. Furthermore, we integrate a transfer learning strategy into our solution to enable servers to exchange knowledge without having to train from scratch. This is crucial to minimize the amount of time spent training per ES and to cope with data shortage issues.

2.3 Trust Establishment

In Rjoub et al. (2022), the authors argue that trust should be a component of the decision-making process and therefore design a mechanism to establish trust between the edge server and an IoT device. During local training, the trust mechanism will identify the devices that utilize their resources excessively or inefficiently. After that, they propose DDQN-Trust, a selection algorithm based on double deep Q learning that considers both the trust value and power level of IoT devices in order to plan scheduling accordingly. They finally integrate their solution into four federated learning aggregation approaches, namely *FedAvg*, *FedProx*, *FedShare*, and *FedSGD*. The authors in Drawel et al. (2020) and Drawel et al. (2021), propose a formal framework for reasoning about how well individuals and groups of agents trust one another. Particularly, they propose a branching time temporal logic BT with operators that express the concepts of every-one trust, distributed trust, and propagated trust.

To allow for model checking BT logic at design time, the authors extend their work in Drawel et al. (2022) to develop efficient and scalable reduction algorithms. The researchers analyze satisfiability and model checking problems in this logic. Furthermore, they present in this article BT Transformation Tool (BTT), which is used to automate the verification process. As a solution to the item cold-start problem in recommendation systems, the authors in Wahab et al. (2022) propose a federated learning-based approach. Compared to existing federated learning-based solutions, their originality is derived from (1) using federated learning for cold-start problems; and (2) using a double deep Q learning scheduling algorithm to select the best candidates based on trust and energy levels.

2.4 Covid-19 Detection

Ozturk et al. (2020) tackle the problem of detecting and then classifying COVID-19 cases through the exploitation of X-ray images using a deep learning framework. The framework enjoys an automatic procedure and requires no manual feature extraction. The proposed model provides accurate results in terms of both binary classification (i.e., COVID vs. No-Findings) and multi-class classification (COVID vs. No-Findings vs. Pneumonia). In Hu et al. (2020), the authors design a "weakly supervised deep learning" framework that allows us to classify COVID-19 cases after detecting them using CT images extracted from various scanners and multiple centres. The proposed framework can accurately recognize COVID-19 cases from community-acquired pneumonia and nasopharyngeal, while minimizing the requirements of manual labelling of the CT images. Wang et al. (2020a) introduce a noise-resistant model, which can be trained efficiently by a noisy dataset to segment COVID-19-related pneumonia lesions at various scaling levels. At first, the authors propose a modified U-Net-based COVID-19 lesion segmentation network that generalizes the "Dice loss" used for segmentation and the "Mean Absolute Error (MAE) loss" utilized for robustness against noise. The authors then introduce a new "noise-robust loss function", and a "COVID-19 Pneumonia Lesion segmentation network (COPLE-Net)". The proposed solution enables a better treatment of the scales and appearances of the lesions. Finally, they define a robust noise-driven learning approach using a self-ensembling framework. Toraman et al. (2020) propose a Capsule Neural Network (CapsNet) for the detection of COVID-19 using chest X-ray images. The proposed model provides fast and accurate diagnostics in both binary (i.e., COVID-19, and No-Findings) and multi-class classifications (i.e., COVID-19, No-Findings, and Pneumonia).

Wang et al. (2020b) design a deep learning model based on Convolutional Neural Networks (CNN) for the detection of COVID-19 using chest X-ray (CXR) images. The solution exploits a human-machine collaborative design strategy. At the output layer, the investigators use the Softmax activation function of the initial network to perform a three-way classification task (i.e., normal, non-COVID infection, and COVID infection). Then, they employ a machine-driven strategy to explore the best network design based on the specific design requirements.

Nour et al. (2020) propose an intelligent model that detects positive COVID-19 cases using a CNN architecture. They employ CNN for feature extraction and then capitalize on the extracted features to be used as inputs for several machine learning techniques (i.e., k-nearest neighbor, support vector machine (SVM) and decision tree). Finally, they employ a Bayesian optimization algorithm to tune the hyper-parameters of the machine learning models. In Meng et al. (2020) the investigators introduce a 3D densely connected CNN-based predictive model to recognise high-risk COVID patients. First, the authors use Hounsfield Unit as a threshold-based method to segment lung regions from CT images. Then, the computer automatically gives a set of seed nodes. Finally, the lung volumes are re-sampled.

The key drawback of these detection methods is that they require an offline mode in order to refine the detection precision through a range of parameters. On the other hand, these approaches work with a limited dataset without trying to use multiple data sources from more than one ES and transfer the knowledge among these servers.

In this work, we aim to show how our FL client selection method can be used to further improve the detection of

COVID-19 through proposing a multi-faceted approach which integrates many state-of-the-art approaches and technologies such as federated transfer learning, IoT, edge computing, trust management and DRL. To the best of our knowledge, no existing approach has yet considered the integration and interconnection of all these technologies for COVID-19 detection. This makes our approach the most holistic in the literature through considering the different aspects that are necessary for COVID-19 detection such as health monitoring and tracking (using IoT and edge devices), trust-aware participant selection and job scheduling (using trust management and DRL), privacy-preserving machine learning (using FL) and training time reduction (using TL).

3 Client Selection Solution

3.1 Problem Formulation

Let $M = \{m_1, m_2, \dots, m_x\}$ be a set of x IoT devices (i.e., clients). For the COVID-19 case study, these IoT devices are responsible for COVID-19 detection. In this case, each IoT device is equipped with X-ray scanning and machine learning capabilities, which enable it to train a COVID-19 detection model. $E = \{e_1, e_2, \dots, e_r\}$ is a set of r ESs responsible for aggregating the model updates received from the IoT devices. Let $T_m = \{t_{m1}, t_{m2}, \dots, t_{ml}\}$ be a set of l training models to be analyzed on the IoT device $m \in M$, which forms a tuple. Let α_m be a trust value assigned by an ES from E to the IoT device m, and ζ_m be the cost of transmitting the model from the parameter server (i.e., the ES) to the device m and running the model. An ES $e \in E$ chooses to schedule one or more tasks over the IoT devices at each time step τ.

Let ε_m be the time needed to train a local model on the device m. We introduce two functions: $\Theta_1(\varepsilon_1, \varepsilon_2, \dots \varepsilon_n)$ an aggregation function that computes the execution time of the overall FL process involving the selected n IoT devices participating in the federated training, and $\Theta_2(\alpha_1, \alpha_2, \dots \alpha_n)$ an aggregation function that computes the overall trust of the selected n IoT devices. The sum of ε_i and α_i $(i := 1 \dots n)$, the average, the min and the max are examples of these aggregation functions. The objective of our scheduling solution is to minimize Θ_1 and maximize Θ_2 subject to the constraints Eqs. (1) to (8).

$$K_{mi}^{CPU} \leq CPU_m^{\tau}, \qquad \forall t_{mi} \in T_m \tag{1}$$

$$\sum_{t_{mi} \in T_m} K_{mi}^{CPU} \leq CPU_m \tag{2}$$

$$K_{mi}^{RAM} \leq RAM_m^{\tau}, \qquad \forall t_{mi} \in T_m \tag{3}$$

$$\sum_{t_{mi} \in T_m} K_{mi}^{RAM} \leq RAM_m \tag{4}$$

$$K_{mi}^{BW} \leq BW_m^{\tau}, \qquad \forall t_{mi} \in T_m \tag{5}$$

$$\sum_{t_{mi} \in T_m} K_{mi}^{BW} \leq BW_m \tag{6}$$

$$K_{mi}^{DS} \leq DS_m^{\tau}, \qquad \forall t_{mi} \in T_m \tag{7}$$

$$\sum_{t_{mi} \in T_m} K_{mi}^{DS} \leq DS_m \tag{8}$$

where K_{mi}^{z} represents a requirement of the training model t_{mi} where each z represents a certain resource parameter, i.e., CPU, RAM, bandwidth (BW), and disk storage (DS). $CPU_m^{\tau}, RAM_m^{\tau}, BW_m^{\tau}, and DS_m^{\tau}$ are the current amounts of available CPU, RAM, Bandwidth, and disk storage on IoT device m.

3.2 Trust Management

To model the trust relationships between an ES $e_i \in E$ and the clients in terms of efficiently and honestly executing the required duties, we employ the trust establishment algorithm that we recently proposed in Rjoub et al. (2020b). The algorithm monitors the IoT devices' resource consumption over time and uses a modified Z-score approach to classify the IoT devices that exhibit some suspicious behavior in terms of over-consumption or under-consumption. This is highly important to consider in concrete scenarios such as the COVID case study to identify those devices that (1) do not allocate sufficient resources to perform the COVID-19 detection tasks or (2) perform additional computations to embed some malicious goals (e.g., optimize for a malicious objective that seeks to cause misclassification) into their local training problems. Note that each ES monitors the IoT devices that are located within its range. The fundamental idea behind the trust method is to estimate the difference between a particular score and the median using the median absolute deviation $MAD_m^z(\tau)$ of a metric z (e.g., CPU, RAM, BW, DS) consumed by the IoT device m at the current time τ.

More precisely, the modified Z-score (β) is computed by dividing the difference between the consumption of the device m and the medium consumption of that device in terms of the resource metric z at time moment τ by the median absolute deviation of the metric z as follows:

$$\beta_m^z(\tau) = \frac{\varrho(x_m^z(\tau) - \tilde{x_m^z}(\tau))}{MAD_m^z(\tau)} \tag{9}$$

where the constant $\varrho = 0.6745$ represents the 0.75th quartile of the standard normal distribution, to which the MAD technique converges (Iglewicz and Hoaglin, 1993). Thereafter, the trust method checks to see whether there is any

consumption that surpasses or falls below the calculated abnormal limit for any possible consumption of the IoT device. Afterwards, by dividing the sum of this relative abnormal consumption $Prop\ Abnormal\ Metrics_m^z$ over all the $z \in Z$ metrics by the number of $Abnormal\ Metrics_m$ of metrics that the device has overused/underused, we derive the trust value of each device α_m. If no metric has been overused/underused, the initial trust in the IoT device's trustworthiness will be 100%.

$$\alpha_m = \frac{\sum_{z \in Z} Prop\ Abnormal\ Metrics_m^z}{Abnormal\ Metrics_m} \tag{10}$$

3.3 Deep Reinforcement Learning Selection Policy

The proposed selection strategy is implemented by a multi-layered neural network which capitalizes on the parameters of the network to specify actions, all of which are fed into the network itself, to produce a final action vector for the state (Rjoub et al., 2020b; Rjoub et al., 2021a). As a global Markov Decision Process (MDP), the problem is formulated as a combination of local states and actions of the IoT devices and the system's global actions. The selection policy is defined by the tuple $\langle S, A, \mathcal{T}, R, \gamma \rangle$, where:

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

Let S_m be the set of local states of the IoT device $m \in M$. The global state space S is obtained through the Cartesian product of the IoT devices' local states, i.e., $S = \prod_{m \in M} S_m$. Each local state $s_m \in S_m$ is computed as follows:

$$\begin{aligned} s_m =& (\alpha_m, \vartheta_m, \delta_m); \quad \alpha_m \in \{0, 1, \ldots, \alpha^{max}\}, \\ & \vartheta_m \in \{0, 1, \ldots, \vartheta^{max}\}, \\ & \delta_m \in \{0, 1, \ldots, \delta^{max}\} \end{aligned} \tag{11}$$

where α_m is the trust value of the IoT device m computed in Eq. (10), ϑ_m is the time needed to run the local model on the device m, and δ_m is the normalized amount of resources (i.e., CPU, RAM, bandwidth, and disk storage) on the device m. Trust value and execution time are dynamic, so they could change from state to state. The global action space of the parameter server is the joint action space of each device: $A = \prod_{m \in M} A_m$ where A_m is the set of local actions of m. A local action $a_m \in A_m$ is as follows:

$$\begin{aligned} a_m =& (\sigma_m, \varepsilon_m, K_m, \zeta_m); \\ & \sigma_m \in \{0, 1\}, \varepsilon_m \in \{0, 1, \ldots, \varepsilon^{max}\}, \\ & K_m \in \{0, 1, \ldots, K^{max}\}, \zeta_m \in \mathbb{R} \end{aligned} \tag{12}$$

where $\sigma_m = 1$ means that the parameter server assigns a training task to the IoT device m and $\sigma_m = 0$ means that the server does not assign the task to m. ε_m refers to the time needed by the IoT device m to download, train and upload the model, and K_m is the normalized amount of resources needed to assign the model from the ES to the IoT device m, and ζ_m is the cost of transmitting the model from the parameter server to the device m and running the model. For an action a to be feasible from a state s the following conditions should hold:

$$\varepsilon_m(a) \leq \vartheta_m(s) \qquad \forall m \in M \tag{13}$$

$$K_m(a) \leq \delta_m(s) \qquad \forall m \in M \tag{14}$$

where $\varepsilon_m(a)$ and $K_m(a)$ refer to ε_m and K_m in the action a from s; and $\vartheta_m(s)$ and $\delta_m(s)$ are ϑ_m and δ_m in s respectively. The reward function R is defined in such a way to maximize the selection of trustworthy IoT devices to perform the federated training and to minimize overall training time. The cost ζ_m is also considered proportional to the maximum cost ζ^{max}. The reward Ψ_m for the device m is a function of state $s \in S$ and action $a \in A$ and is computed as follows:

$$\Psi_m(s, a) = \begin{cases} \sigma_m . \varepsilon_m . K_m - \frac{\zeta_m}{\zeta^{max}}, & \text{if } K_m(a) \leq \delta_m(s) \\ & \textit{and} \quad \varepsilon_m(a) \leq \vartheta_m(s). \\ -\frac{\zeta_m}{\zeta^{max}}, & \text{otherwise.} \end{cases} \tag{15}$$

In fact, the reward function considers the trust scores of the IoT devices and 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 given by the following equation:

$$R(s, a) = \sum_{m \in M} \Psi_m(s, a) \tag{16}$$

The parameter ES determines the optimal policy $\pi^* : S \rightarrow A$ that indicates the actions to be taken at each state to maximize the cumulative reward. The Q-learning (QL) algorithm's essential goal 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 certain 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:

$$Q^*(s, a) = R(s, a) + \gamma \sum_{s' \in S} Pr(s' \mid s, a). \max_{a' \in A} Q^*(s', a') \tag{17}$$

The parameter server calculates the optimal action from any state to maximize the cumulative reward, based on the Q-table that emerges from updating the $Q(s,a)$ values. The QL algorithm is only feasible for networks with limited state and action spaces, but the issue of assigning training tasks to IoT devices becomes highly dimensional when the number of network participants increases (as is the case for IoT networks consisting of a large number of devices). To solve the high dimensionality problem, the deep Q network (DQN) algorithm (a combination of QL and deep neural network) comes into play. The DQN takes as input one of the online network's states, and outputs the Q-values $Q(s,a;\theta)$, as well as the weight-matrix θ of the neural network of all eventual actions. In order to obtain approximate values $Q^*(s,a)$, DQN needs to be trained using experience $(s, a, R(s, a), s')$. To define the loss function, we use the mean square error (MSE), and DQN uses the Bellman equation to minimize the following loss function:

$$L(\theta_i) = E[(R(s,a) + \gamma \arg\max_{a' \in A} Q(s', a';\theta_i') - Q(s,a;\theta_i))^2] \tag{18}$$

where θ_i denotes the online network parameters of the i^{th} iteration, θ_i' represents the goal network parameters of the i^{th} iteration, and $E[.]$ represents the expected value. Note that the action a is chosen on the basis of the ϵ-greedy policy (Lopez-Martin et al., 2020). This policy is general as it does not rely on strong assumptions about the underlying domain. It also has the advantage of being simple, not requiring too much implementation effort or per-domain fine tuning. This makes it an appealing alternative despite the fact that it may not be performing better than some of its more complex counterparts (i.e. the ϵ-soft policy and the softmax action selection policy) in all the cases.

3.4 Federated Learning Model

We use the FL model we recently introduced in Rjoub et al. (2020b). Let D_m be a local dataset collected by the IoT device m. For instance, in the COVID-19 scenario, the local dataset includes X-ray images. The dataset is represented as follows: $D_m = \{(x_{1_m}, y_{1_m}), \ldots, (x_{n_m}, y_{n_m})\}$, where x_{i_m} is the i^{th} training sample and y_{i_m} represents the corresponding ground-truth label. We assume that a given learning model is employed on each D_m. For the COVID-19 case study, we employ a general CNN model to perform our analysis on the X-ray data. The ES first trains a global learning model on a publicly available dataset and then sends the initial parameters to the set of IoT devices selected as per our selection solution discussed in Section 3.3. These IoT devices capitalize on the shared parameters to locally train the model on their own set of collected data and hence derive an updated set of the parameters. Upon receiving the updated parameters from the IoT devices, the server aggregates (using Eq. (19)) these parameters to derive a global aggregate model:

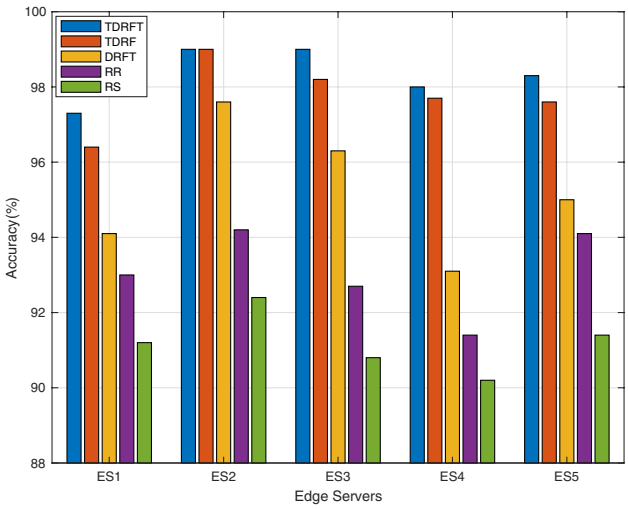


Fig. 2 Comparison of accuracy of final global model at five different ESs

$$g[\nu] = \frac{1}{\sum_{m \in M} |\lambda_m|} \sum_{m \in M} |\lambda_m| \, g_m[\nu] \tag{19}$$

where $\lambda_m \subseteq D_m$ is a subset of local data collected by IoT device m for a training period ν and $g_m[\nu]$ is the local gradient which is computed as per Eq. (20):

$$g_m[\nu] = \nabla_{w_m} L_m(w_m, \lambda_m) \tag{20}$$

where w_m is the set of local parameters of the learning model, L_m is the local loss function (in terms of training error) to be minimized on IoT device m and $\nabla_{w_m} L_m(.)$ is the gradient of the loss function L_m with respect to w_m.

Algorithm 1 DRL-based federated learning algorithm.

1: Initialize the global parameter set of the CNN model on a publicly available X-ray image data
2: **for** each round $\varphi = 1, 2, \ldots$ **do**
3: **Use** our selection solution described in Section 3.3 to select a subset $\mathcal{E} \subseteq M$ of IoT devices to participate in the training
4: **Send** W_φ to each selected IoT
5: **for** each IoT device $m \in \mathcal{E}$ **do**
% $\mathcal{E} = \{1, 2, \ldots, S\}$
6: Execute **IoTLocalUpdate**(m, W_φ)
% See Algorithm 2
7: **end for**
8: $W_\varphi = \frac{1}{n} \sum_{m=1}^{S} n_m w_m$
9: **end for**

We explain in Algorithms 1 and 2 the FL process we propose. In Algorithm 1, n_m is the volume of the data that are available on IoT device m, n is the volume of the overall data

Table 1 Comparisons on COVID-19 detection performance

Methods	Precision	Recall	F1 score
TDRFT	0.915	0.967	0.958
TDRF	0.901	0.946	0.922
DRFT	0.842	0.918	0.878
RR	0.798	0.882	0.838
RS	0.764	0.821	0.791

across all IoT devices, S is the total number of selected IoT devices to participate in the FL process, φ is an index that represents a training communication round and W_{φ} is the set of global parameters at training round φ. In Algorithm 1, at each round φ we use our DRL selection solution (Section 3) to select a subset of IoT devices $\mathcal{E}$ and then send the set of global parameters at training round (W_{φ}) to each selected IoT (line 4). The local update is computed for each IoT device in the selected subset (the call to Algorithm 2). Finally, Algorithm 1 calculates the global parameters for the next training round (line 8). In Algorithm 2, after the global parameters are received from the ESs, the local update is calculated for each IoT device at each local iteration i (Line 4). Finally, the calculated local update is returned to the ES. The Stochastic Gradient Descent (SGD) algorithm is run by each IoT device based on the obtained global gradient. The local loss function $L_m(w_m)$, which has to be minimized on each device m, is calculated as shown in Eq. (21):

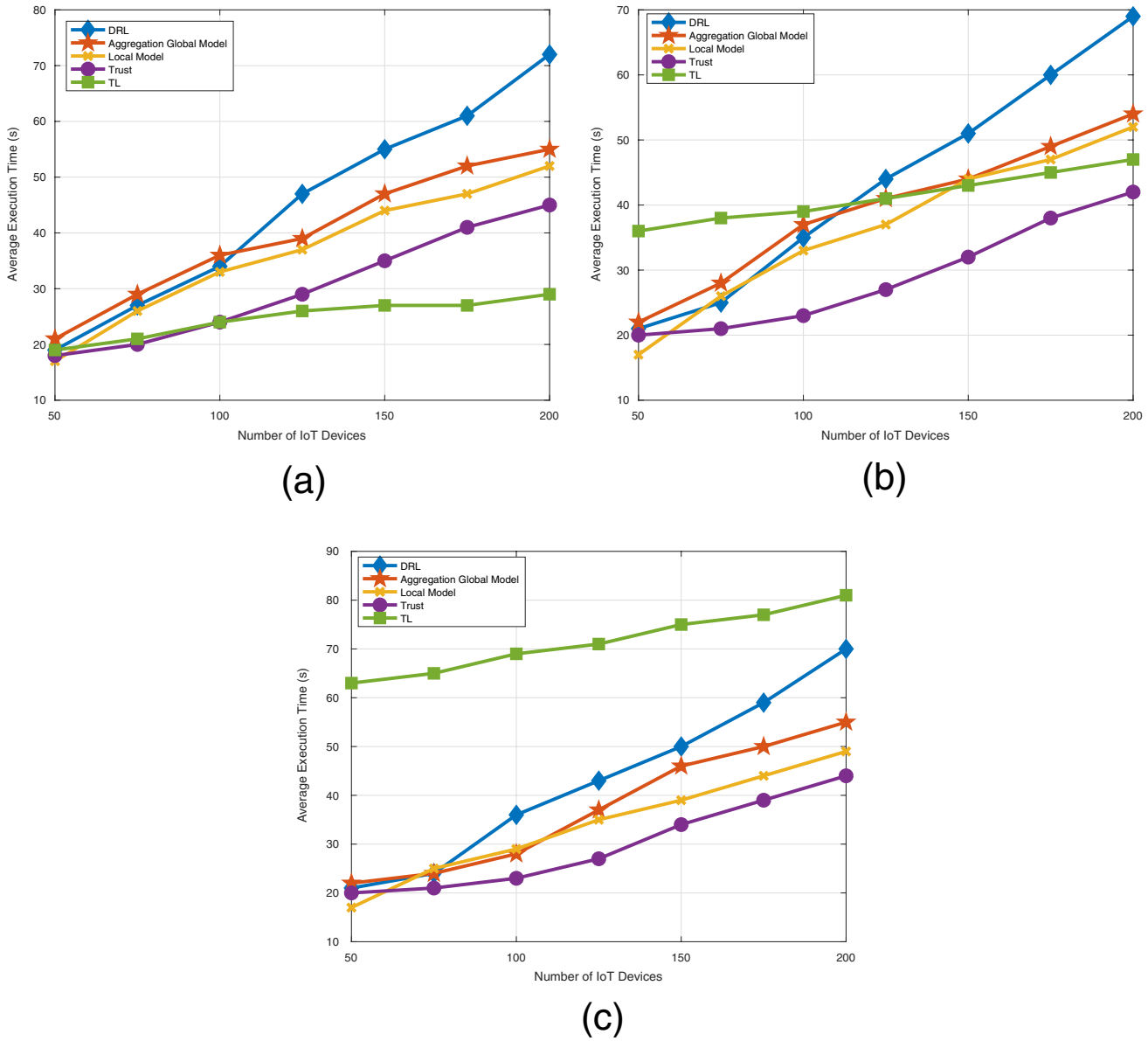


Fig. 3 Average execution time of the proposed model phases. **a** 5 Edge Servers. **b** 25 Edge Servers. **c** 50 Edge Servers

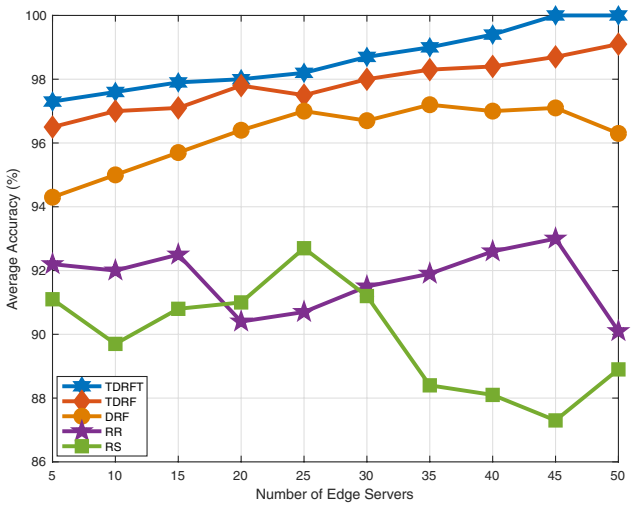


Fig. 4 Comparison of average accuracy of final global model of varying number of ESs

$$L_m(w_m) = \frac{1}{N_m} \sum_{(x,y)\in D_m} \ell(w_m, x, y) \tag{21}$$

where $\ell(w_m, x, y)$ is the sample-level loss function that quantifies the prediction error between the learning output (via the input x and parameter w_m) and ground-truth label y, and N_m is the number of data samples on device m. Each device attempts to minimize this local loss function, thereby reducing the error of the training process. The main objective of the global model at the ES level is to optimize the set of parameters to minimize the global loss function $L(W)$ using the SGD algorithm as shown in Eq. (22):

$$L(W) = \frac{1}{\sum_{m=1}^{S} N_m} \sum_{m=1}^{S} N_m L_m(w_m) \tag{22}$$

Algorithm 2 IoT local training.

1: **IoTLocalUpdate**(m, W_φ)
2: $w_m = W_\varphi$
3: **for** each local iteration $i = 1$ to T **do**
4: $w_m = w_m - \eta \nabla_{w_m} L_m(w_m, \lambda_m)$
% η is the learning rate
5: **end for**
6: return w_m to the ES

3.5 Inter-Edge Transfer Learning

Our solution comprises a TL component that allows ESs' knowledge to be shared without disclosure of their raw data. Doing so is useful in many situations such as:

- One ES is newly deployed, but another server has already explored some knowledge.
- Some ESs do not have enough IoT devices in their vicinity or have IoT devices that do not have enough data to obtain efficient learning.
- Some ESs couldn't obtain enough knowledge for a given task compared to the knowledge obtained by other servers.

Algorithm 3 ESs knowledge transfer.

1: **Input:** $E = \{e_1, e_2, \ldots, e_r\}$
2: **Receive local model updates**(w_m)
3: **for** each round $\varphi = 1, 2, \ldots$ **do**
4: **for** each edge server $e \in E$ **do**
5: Aggregate global parameters W_φ^e
6: Calculate global loss function $L(W_\varphi^e)$
7: **end for**
8: Calculate the optimal Global Model over E:
9: $W^* = \arg\min_{W_\varphi^e, \forall e \in E} (L(W_\varphi^e))$
10: **Send** W^* to each selected IoT
11: **end for**

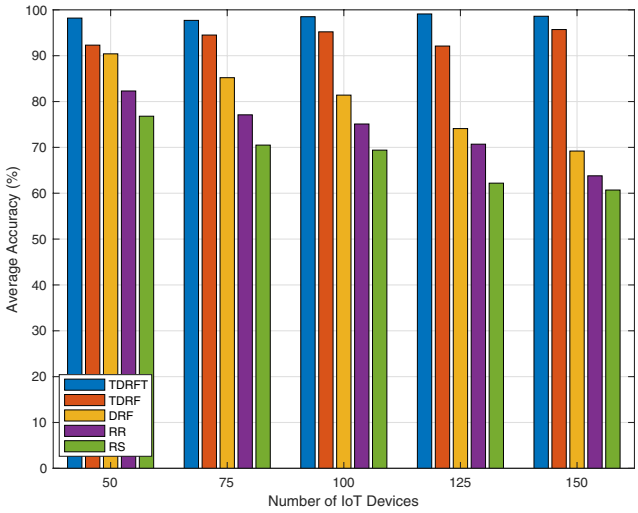
The intuition is that knowledge transferred among the aggregation servers can boost the optimization on the edge. As such, we propose to transfer knowledge bidirectionally. We explain in Algorithm 3 the TL process that is proposed to share the knowledge among ESs. In Algorithm 3, each ES $e \in E$ aggregates the local models received from the selected IoTs (Algorithm 1) and then calculates the global loss function (Eq. 22), where W_φ^e is the global model W_φ on the ES e. The optimal global model W^* is calculated over all the ESs based on the minimum loss function value (line 9) before sending it back to the selected IoT devices.

4 Experiments

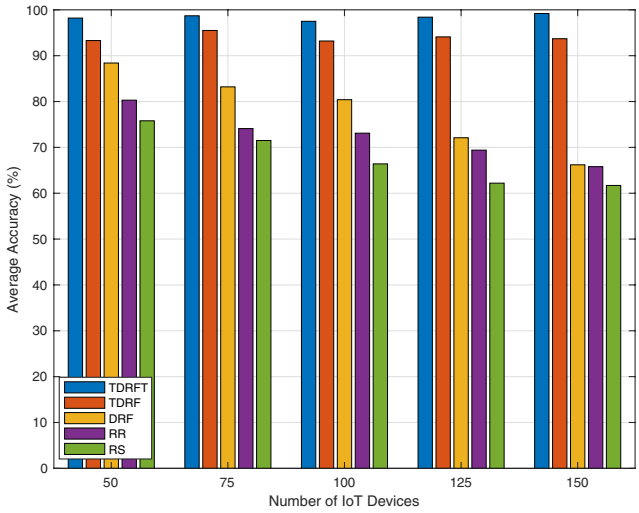
4.1 Experimental Setup

To carry out our experiments, we capitalize on a dataset[1] that consists of chest X-ray images for individuals

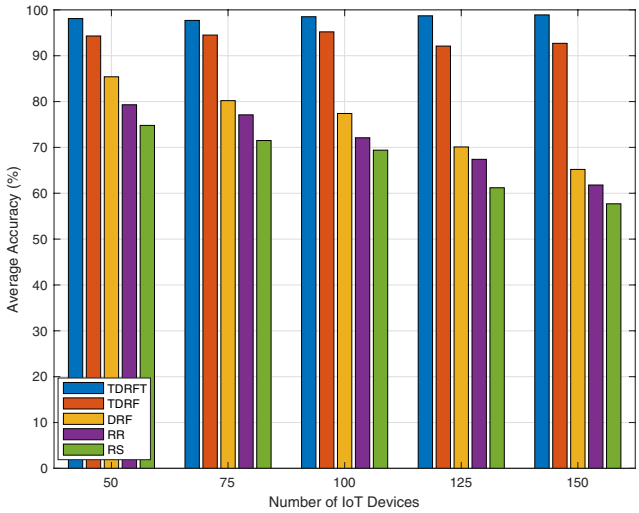
[1] https://www.kaggle.com/tawsifurrahman/covid19-radiography-database


Average Accuracy (%)
Number of IoT Devices
TDRFT
TDRF
DRF
RR
RS

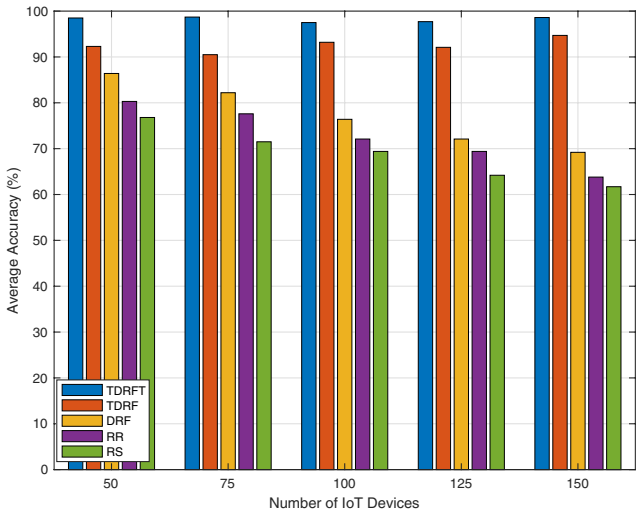
(a)


Average Accuracy (%)
Number of IoT Devices
TDRFT
TDRF
DRF
RR
RS

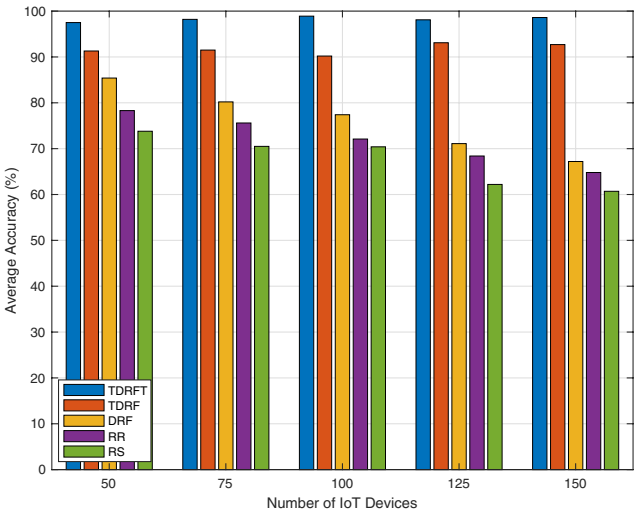
(b)


Average Accuracy (%)
Number of IoT Devices
TDRFT
TDRF
DRF
RR
RS

(c)


Average Accuracy (%)
Number of IoT Devices
TDRFT
TDRF
DRF
RR
RS

(d)


Average Accuracy (%)
Number of IoT Devices
TDRFT
TDRF
DRF
RR
RS

(e)

◂**Fig. 5** Average accuracy values in TDRFT, TDRF, DRF, RR, and RS. **a** 10 Edge Servers. **b** 20 Edge Servers. **c** 30 Edge Servers. **d** 40 Edge Servers. **e** 50 Edge Servers

infected with COVID-19, individuals with viral Pneumonia, as well as normal images. In total, the dataset consists of 219 COVID-19 positive images, 1341 normal images and 1345 viral pneumonia images. The data distribution is non-IID and unbalanced, reflecting the characteristics of real-world FL scenarios. We train a CNN model to determine our algorithm's efficiency and effectiveness. The CNN model used consists of six 3×3 convolution layers as follows: 32, 32, 64, 64, 128, 128. The Rectified Linear Unit (ReLU) activates each layer and normalizes the batch. Every pair of convolution layers is followed by a 2×2 max pooling layer, then 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 the soft-max. We employ the TensorFlow Federated (TFF) platform, which provides an open source framework for decentralized data learning. TFF facilitates a variety of collaborative learning scenarios on a number of heterogeneous devices with different resources. The SGD algorithm is used to train the model on IoT devices with a batch size of 128 rows per IoT device for every training round. We distributed the training data on 1000 IoT devices (i.e., $|M| = 1000$) of four various resource categories:

- Category-1 with 1 CPU core and 1.75GB RAM,
- Category-2 with 2 CPU cores and 3.5GB RAM,
- Category-3 with 4 CPU cores and 7GB RAM, and
- Category-4 with 8 CPU cores and 14GB RAM.

The ES selects the top 50 IoT devices returned by the scheduling algorithm (e.g., $|E| = 50$), at each iteration. Our program is written in Python 3 and executed on a 64-bit Windows 7 threaded environment on an Intel Core i7 3.40 GHz CPU and 16 GB of RAM.

4.2 Experimental Results

In Fig. 2, we compare the accuracy of our solution under different combinations, i.e., Trust, Deep Reinforcement learning, Federated and Transfer learning (TDRFT); Trust, Deep Reinforcement learning, and Federated learning (TDRF); and Deep Reinforcement learning, Federated and Transfer learning (DRFT). We also compare these different combinations of our solution with two common scheduling approaches, i.e., Round Robin (RR) and Random Scheduling (RS) while integrating our trust establishment solution into them. The different approaches are executed at five different ESs to closely inspect the accuracy values. It is important to notice that our model does not depend on the number of edge servers. Each edge server can work independently by playing the role of the parameter server in the federated learning process. This server is mainly responsible for creating the machine learning model and aggregating the updated model weights from the IoT devices. Although our solution aims to help one edge server select the most trusted IoT devices, it can easily be applied in a multi-edge scenario, at the level of each edge server.

We notice from the figure that the accuracy obtained with TDRFT is higher than that obtained with the other combinations and approaches. In particular, the accuracy levels obtained with TDRFT varies between 97.2% and 99.1% across the five ESs. With TDRF, the accuracy level varies between 96.4 and 99.1%. With DRFT, the accuracy level varies between 93% and 97.7%. With RR, the accuracy level varies between 91.6% and 94.2%. Finally, with RS, the accuracy level varies between 90.2% and 91.6%. Thus, we conclude that our solution with all of its components improves the accuracy of detecting COVID-19 cases. The reason is that it employs deep Q-learning to select the IoT devices that achieve the best combinations in terms of resource availability and trust maximization, and includes a TL component to compensate the lack of learning from which some ESs might suffer.

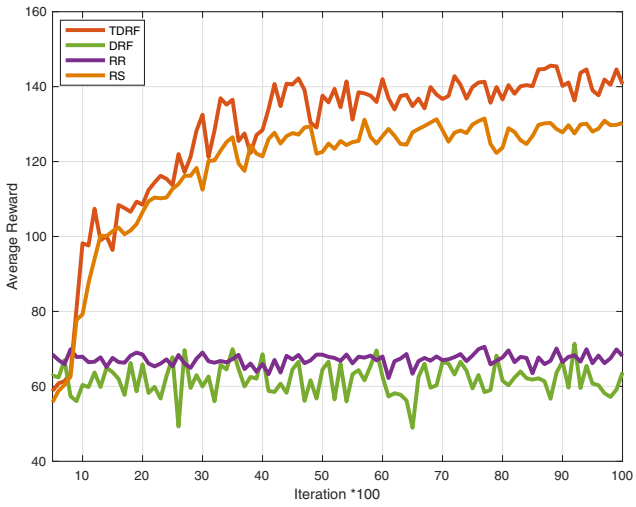
To investigate further the five methods in the COVID-19 detection, we use the following metrics: Precision, Recall, and F1 score. We analyse our solution by omitting the trust first, and then the transfer learning in order to study the impact of such specific components of our solution. This allows us to illustrate the strength of the proposed trust transfer federated learning method in the edge computing scenario. We also compare these different combinations of our solution with RR and RS after augmenting them with our trust establishment solution. The evaluation indicators used in performance comparisons are introduced and defined as follows:

- True positives (TP): The number of times a COVID-19 has been predicted as a correct type.
- True negatives (TN): The number of times a non-COVID-19 has been predicted as a correct type.
- False positives (FP): The number of times a COVID-19 has been predicted as a wrong type.
- False negatives (FN): The number of non-COVID-19 has been predicted as a wrong type.

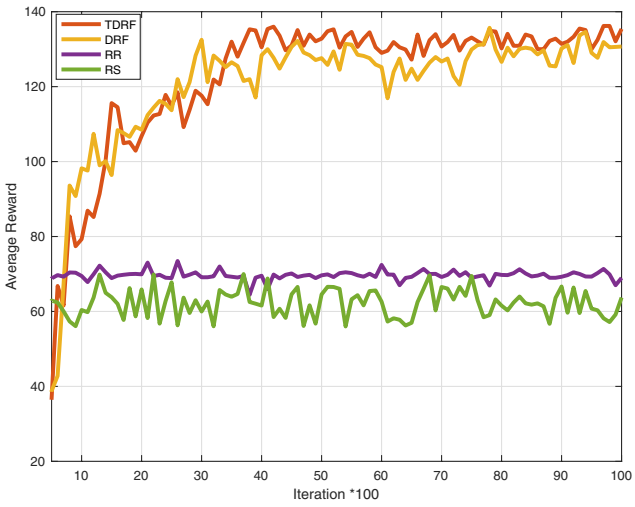
According to the definition above, Precision, Recall, and F1 score, can be calculated as follows.

$$Precision = \frac{TP}{FP + TP} \tag{23}$$

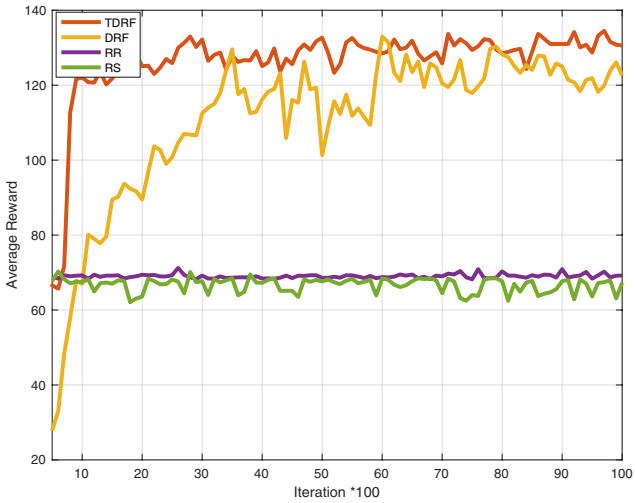
$$Recall = \frac{TP}{FN + TP} \tag{24}$$


TDRF
DRF
RR
RS
Average Reward
Iteration *100

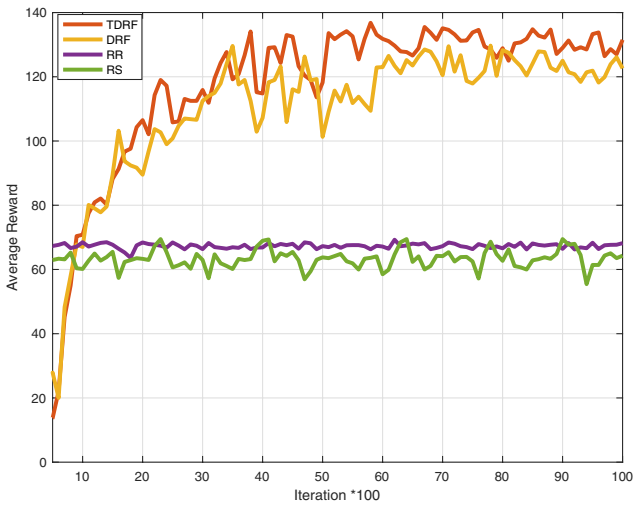
(a)


TDRF
DRF
RR
RS
Average Reward
Iteration *100

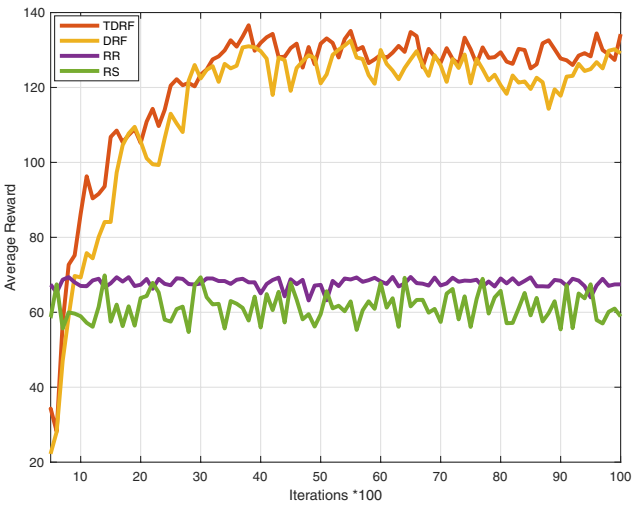
(b)


TDRF
DRF
RR
RS
Average Reward
Iteration *100

(c)


TDRF
DRF
RR
RS
Average Reward
Iteration *100

(d)


TDRF
DRF
RR
RS
Average Reward
Iterations *100

(e)

◂**Fig. 6** Average reward values in TDRF, DRF, RR, and RS. **a** 50 IoT Devices. **b** 75 IoT Devices. **c** 100 IoT Devices. **d** 125 IoT Devices. **e** 150 IoT Devices

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{25}$$

The results are listed and compared in Table 1. The table clearly shows that the proposed TDRFT achieves the best results, with F1 score of 0.958, Recall of 0.967, and Precision of 0.915. This means that the proposed TDRFT approach enables the ES to better learn and recognize COVID-19 by adopting our solution with trust transfer and federated learning.

In Fig. 3, we measure the execution time of each single component of our solution. To do so, we vary the number of IoT devices from 50 to 200, while also varying the number of ESs from 5, to 25, to 50. The key outcome from this figure is that raising the number of IoT devices results in a slight increase in execution time, especially during the DRL process. This can be justified by the fact that having a larger number of IoTs to assign the tasks to increases the search space of the most of the components, except for the TL phase, which is independent from the number of IoT devices. The second observation is that the increase in the number of the ESs leads to increasing the execution time of the TL component, without having any significant impact on the other components. The reason is that having more servers means that more TL processes might need to be performed among these servers.

In Fig. 4, we provide experimental comparisons in terms of average accuracy while varying the number of ESs between 5 and 50. Again, in this scenario, the accuracy obtained with TDRFT is much higher than that obtained than the rest of the compared approaches. In particular, the average accuracy obtained by the TDRFT, TDRF, DRF, RR and RS approaches are $97.3 - 99.4$, $96.4 - 98.8$, $94.2 - 97.1$ $90 - 93.2$ and $87.3 - 92.6$, respectively. This means that the proposed TDRFT approach enables the ES to better learn and recognize COVID-19 by adopting our solution with all of its components.

In Fig. 5, we measure the average accuracy of the CNN that was trained by IoT devices selected by the TDRFT, TDRF, DRF, RR, and RS approaches. We ran the experiments over 1000 iterations (i.e., $T = 1000$) to study the scalability of the different considered solutions, while varying the number of ESs from 10 to 50 and also varying the number of IoT devices from 50 to 150. The main observation that can be drawn from this experiment is that our proposed solution, with all of its components, achieves the highest accuracy level compared to the other approaches and exhibits a better scalability to an increasing number of IoT devices and ESs. In particular, the average accuracy level obtained by our TDRFT varies between 94.2% and 96.4% with 10 edge servers; between 96.3% and 98.1% with 20 edge servers; between 96.6% and 97.9% with 30 edge servers; between 96.3% and 97.6% with 40 edge servers; and between 96.5% and 97.7% with 50 edge servers. A significant hypothesis to be considered is that providing a greater number of IoT devices might increase the likelihood of erroneously assigning global learning to some inappropriate IoT devices. Yet, the TL component that we integrate in our solution, which enables sharing the knowledge among servers, leads to increasing the accuracy at the level of some servers that might have made some poor selections in terms of IoT devices.

In Fig. 6, we provide experimental comparisons in terms of average reward. We ran the experiments over 10000 (i.e., $T = 10000$) iterations. We observe from this figure that the average rewards obtained by TDRF and DRF are much higher than those obtained by the RR and RS approaches. In particular, the average rewards obtained by the TDRF, DRF, RR, and RS approaches are 143, 131, 67, and 61 respectively with 50 IoT devices; 137, 130, 70, and 64 respectively with 75 IoT devices; 134, 122, 66, and 62 respectively with 100 IoT devices; 133, 125, 67, and 63 respectively with 125 IoT devices; and 134, 128, 65, and 60 respectively with 150 IoT devices. This means that TDRF enables the ES to better learn how to schedule the COVID-19 detection tasks in such a way that best maximizes the reward in terms of minimizing the execution time and maximizing the trust.

In Fig. 7, we measure the execution time of the different studied approaches, while varying the number of IoT devices from 50 to 200 and varying the number of servers from 5 to 100. In particular, the execution time obtained by the TDRF, DRF, RR, and RS approaches are 257 ms, 361 ms, 407 ms, and 597 ms respectively. The main observation that can be drawn from this simulation is that increasing the number of IoT devices leads to a modest increase in the execution time in our solution (i.e., TDRF) compared to the other models. This is because our solution employs deep Q-learning to select the IoT devices that achieve the best combinations in terms of resource availability and trust maximization. On the other hand, increasing the number of ESs results in a slight increase in the execution time in all the studied approaches, even in our model as we use TL to exchange knowledge among the servers.

5 Discussion and Conclusion

In this paper, we proposed an approach to select the most appropriate clients for FL in terms of resource consumption and training time based on a trust-augmented DRL. We embedded a transfer learning mechanism on top of the client selection mechanism to address the scarcity of

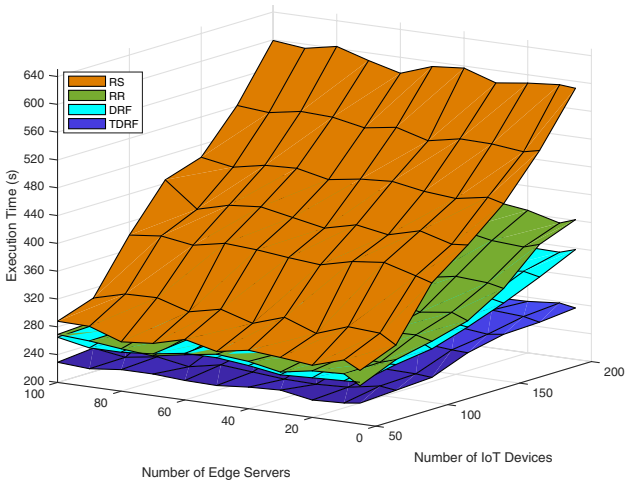


Fig. 7 Execution Time: We study in this figure the impact of varying both the number of IoT devices and number of ESs on the execution time

data in some regions and compensate potential lack of learning at some servers. We applied our solution in the healthcare domain in a COVID-19 detection scenario over IoT devices. To the best of our knowledge, no existing approach has yet considered the integration and interconnection among all these technologies for collaborative tasks such as detection. This makes our approach the most holistic in the literature through considering different aspects that are important for this type of tasks, including trust-aware participant selection and job scheduling (using trust management and DRL), privacy-preserving machine learning (using FL) and training time reduction (using TL). Experiments conducted on a real-world COVID-19 dataset reveal that our solution achieves a good trade-off between detection accuracy and model execution time compared to existing approaches. The results show as well that the components of our solution are important for the success of our approach.

The results presented in this paper are applicable not only for the COVID-19 detection, but also to other domains. This would be possible through tuning the corresponding metrics in the formulation of the problem and solution, as well as in the experimental environment. In fact, the high level architecture of our solution is highly generic and could be tuned to model a variety of applications. Thus, the proposed solution can be effectively extended to other environments including path prediction, traffic planing, military monitoring tasks, etc. For example, to adapt our solution to a traffic planing application, the reward function could be tuned or changed to include domain-specific metrics such as velocity and residual distance.

Although the results obtained by our solution are very promising, the work can be improved in the future in many aspects. The first aspect is the relatively high computation time of our approach which is mainly caused by the fact that in the transfer learning phase, edge servers need to share the knowledge among each other. To address this problem, we plan to include information about the geographic locations of the edge servers into the transfer learning decisions to make sure that no data has to be shared among geographically far servers. The second aspect concerns the cyberattacks that could be launched against many parts of our solution. For example, the federated learning model could be a target to many elaborate attacks such as model poisoning and label flipping. Moreover, the communications among the edge servers could make room for many attacks such as eavesdropping and probing. In the future, we plan to address these security challenges in order to make our model more robust and secure (Wahab et al., 2016; Wahab, 2022).

Declarations

Conflict of Interests The authors declare that the research reported in this paper was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

Al-Dhaen, F., Hou, J., Rana, N.P., & et al. (2021). Advancing the understanding of the role of responsible ai in the continued use of iomt in healthcare. *Information Systems Frontiers*, pp. 1–20.

Anh, T.T., Luong, N.C., Niyato, D., & et al. (2019). Efficient training management for mobile crowd-machine learning: a deep reinforcement learning approach. *IEEE Wireless Communications Letters*, *8*(5), 1345–1348.

Arisdakessian, S., Wahab, O.A., Mourad, A., & et al. (2020). Fogmatch: an intelligent multi-criteria IoT-fog scheduling approach using game theory. *IEEE/ACM Transactions on Networking*, *28*(4), 1779–1789.

Bataineh, A.S., Bentahar, J., Wahab, O.A., & et al. (2020). 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.

Bataineh, A.S., Bentahar, J., Mizouni, R., & et al. (2021). Cloud computing as a platform for monetizing data services: A two-sided game business model. *IEEE Transactions on Network and Service Management*.

Bentahar, J., Drawel, N., & Sadiki, A. (2022). Quantitative group trust: A two-stage verification approach. In *Proceedings of the 21th International Conference on Autonomous Agents and Multiagent Systems, AAMAS, Auckland, New Zealand, May 9-13, 2022. International Foundation for Autonomous Agents and Multiagent Systems, p (in press)*.

Brisimi, T.S., Chen, R., Mela, T., & et al. (2018). Federated learning of predictive models from federated electronic health records. *International Journal of Medical Informatics*, *112*, 59–67.

Brunese, L., Mercaldo, F., Reginelli, A., & et al. (2020). Explainable deep learning for pulmonary disease and coronavirus COVID-19 detection from X-rays. *Computer Methods and Programs in Biomedicine*, *196*, 105,608.

Chen, M., Yang, Z., Saad, W., & et al. (2019). A joint learning and communications framework for federated learning over wireless networks. arXiv:190907972.

Drawel, N., Bentahar, J., & Qu, H. (2020). Computationally grounded quantitative trust with time. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, AAMAS, Auckland, New Zealand, May 9-13, 2020. International Foundation for Autonomous Agents and Multiagent Systems* (pp. 1837–1839).

Drawel, N., Bentahar, J., Laarej, A., & et al. (2021). Formalizing group and propagated trust in multi-agent systems. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence* (pp. 60–66).

Drawel, N., Bentahar, J., Laarej, A., & et al. (2022). Formal verification of group and propagated trust in multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, *36*(1), 1–31.

Gomathi, B., Krishnasamy, K., & Balaji, B.S. (2018). Epsilon-fuzzy dominance sort-based composite discrete artificial bee colony optimisation for multi-objective cloud task scheduling problem. *International Journal of Business Intelligence and Data Mining*, *13*(1-3), 247–266.

Hu, C., Jiang, J., & Wang, Z. (2019). Decentralized federated learning: A segmented gossip approach. arXiv:190807782.

Hu, S., & Li, G. (2019). Dynamic request scheduling optimization in mobile edge computing for IoT applications. *IEEE Internet of Things Journal*, *7*(2), 1426–1437.

Hu, S., Gao, Y., Niu, Z., & et al. (2020). Weakly supervised deep learning for COVID-19 infection detection and classification from CT images. *IEEE Access*, *8*, 118,869–118,883.

Iglewicz, B., & Hoaglin, D.C. (1993). How to detect and handle outliers, vol 16. Asq Press.

Kumar, P., Dwivedi, Y.K., & Anand, A. (2021). Responsible artificial intelligence (ai) for value formation and market performance in healthcare: the mediating role of patient's cognitive engagement. I*nformation Systems Frontiers*, pp. 1–24.

Lei, L., Xu, H., Xiong, X., & et al. (2019). Joint computation offloading and multiuser scheduling using approximate dynamic programming in NB-iot edge computing system. *IEEE Internet of Things Journal*, *6*(3), 5345–5362.

Li, S., Xu, L.D., & Zhao, S. (2015). The internet of things: a survey. *Information Systems Frontiers*, *17*(2), 243–259.

Liu, N., Li, Z., Xu, J., & et al. (2017). A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)* (pp. 372–382). IEEE.

Lopez-Martin, M., Carro, B., & Sanchez-Esguevillas, A. (2020). Application of deep reinforcement learning to intrusion detection for supervised problems. *Expert Systems with Applications*, *141*, 112,963.

Luo, J., Yin, L., Hu, J., & et al. (2019). Container-based fog computing architecture and energy-balancing scheduling algorithm for energy IoT. *Future Generation Computer Systems*, *97*, 50–60.

Mansouri, N., Zade, B.M.H., & Javidi, M.M. (2019). Hybrid task scheduling strategy for cloud computing by modified particle swarm optimization and fuzzy theory. *Computers & Industrial Engineering*, *130*, 597–633.

Meng, L., Dong, D., Li, L., & et al. (2020). A deep learning prognosis model help alert for covid-19 patients at high-risk of death: a multi-center study. *IEEE Journal of Biomedical and Health Informatics*, *24*(12), 3576–3584.

Nguyen, H.T., Luong, N.C., Zhao, J., & et al. (2019). Resource allocation in mobility-aware federated learning networks: A deep reinforcement learning approach. arXiv:191009172.

Nishio, T., & Yonetani, R. (2019). Client selection for federated learning with heterogeneous resources in mobile edge. In *IEEE International Conference on Communications (ICC)* (pp. 1–7).

Nour, M., Cömert, Z., & Polat, K. (2020). A novel medical diagnosis model for COVID-19 infection detection based on deep features and Bayesian optimization. *Applied Soft Computing*, pp. 106580.

Otoom, M., Otoum, N., Alzubaidi, M.A., & et al. (2020). An IoT-based framework for early identification and monitoring of COVID-19 cases. *Biomedical Signal Processing and Control*, pp. 102149.

Ozturk, T., Talo, M., Yildirim, E.A., & et al. (2020). Automated detection of COVID-19 cases using deep neural networks with X-ray images. *Computers in Biology and Medicine* pp. 103792.

Qiu, F., Zhang, B., & Guo, J. (2016). A deep learning approach for vm workload prediction in the cloud. In *2016 17Th IEEE/ACIS international conference on software engineering, artificial intelligence, networking and parallel/distributed computing (SNPD)* (pp. 319–324). IEEE.

Rahman, M.A., Hossain, M.S., Alrajeh, N.A., & et al. (2020). B5G And explainable deep learning assisted healthcare vertical at the edge: COVID-i9 perspective. *IEEE Network*, *34*(4), 98–105.

Rjoub, G., & Bentahar, J. (2017). Cloud task scheduling based on swarm intelligence and machine learning. In *2017 IEEE 5Th international conference on future internet of things and cloud (FiCloud)* (pp. 272–279). IEEE.

Rjoub, G., Bentahar, J., & Wahab, O.A. (2020a). Bigtrustscheduling: Trust-aware big data task scheduling approach in cloud computing environments. *Future Generation Computer Systems*, *110*, 1079–1097.

Rjoub, G., Wahab, O.A., Bentahar, J., & et al (2020b). 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.

Rjoub, G., Bentahar, J., Abdel Wahab, O., & et al. (2021a). Deep and reinforcement learning for automated task scheduling in large-scale cloud computing systems. *Concurrency and Computation: Practice and Experience*, *33*(23), e5919.

Rjoub, G., Wahab, O.A., Bentahar, J., & et al (2021b). 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.

Rjoub, G., Wahab, O.A., Bentahar, J., & et al. (2022). Trust-driven reinforcement selection strategy for federated learning on iot devices. *Computing*, pp. 1–23.

Singh, R.P., Javaid, M., Haleem, A., & et al. (2020). Internet of things (IoT) applications to fight against COVID-19 pandemic. *Diabetes & Metabolic Syndrome: Clinical Research & Reviews.*

Song, B., Yu, Y., Zhou, Y., & et al. (2018). Host load prediction with long short-term memory in cloud computing. *The Journal of Supercomputing*, *74*(12), 6554–6568.

Toraman, S., Alakus, T.B., & Turkoglu, I. (2020). Convolutional capsnet: A novel artificial neural network approach to detect COVID-19 disease from X-ray images using capsule networks. *Chaos, Solitons & Fractals*, *140*, 110,122.

Tuli, S., Tuli, S., Tuli, R., & et al. (2020). Predicting the growth and trend of COVID-19 pandemic using machine learning and cloud computing. *Internet of Things* pp. 100222.

Wahab, O.A. (2022). Intrusion detection in the IoT under data and concept drifts: Online deep learning approach. *IEEE Internet of Things Journal.*

Wahab, O.A., Bentahar, J., Otrok, H., & et al. (2016). How to distribute the detection load among virtual machines to maximize the detection of distributed attacks in the cloud?. In *In 2016 IEEE International Conference on Services Computing (SCC)* (pp. 316–323). IEEE.

Wahab, O.A., Cohen, R., Bentahar, J., & et al. (2020). An endorsement-based trust bootstrapping approach for newcomer cloud services. *Information Sciences*, *527*, 159–175.

Wahab, O.A., Mourad, A., Otrok, H., & et al. (2021). Federated machine learning: Survey, multi-level classification, desirable criteria and future directions in communication and networking systems. IEEE Communications Surveys & Tutorials.

Wahab, O.A., Rjoub, G., Bentahar, J., & et al. (2022). Federated against the cold: A trust-based federated learning approach to counter the cold start problem in recommendation systems. *Information Sciences.*

Wang, G., Liu, X., Li, C., & et al. (2020a). A noise-robust framework for automatic segmentation of covid-19 pneumonia lesions from ct images. *IEEE Transactions on Medical Imaging*, *39*(8), 2653–2663.

Wang, L., Lin, Z.Q., & Wong, A. (2020b). Covid-net: a tailored deep convolutional neural network design for detection of covid-19 cases from chest x-ray images. *Scientific Reports*, *10*(1), 1–12.

Zhai, Y., Bao, T., Zhu, L., & et al. (2020). Toward reinforcement-learning-based service deployment of 5g mobile edge computing with request-aware scheduling. *IEEE Wireless Communications*, *27*(1), 84–91.

Zhang, L., Shen, B., Barnawi, A., & et al. (2021). Feddpgan: federated differentially private generative adversarial networks framework for the detection of covid-19 pneumonia. *Information Systems Frontiers*, *23*(6), 1403–1415.

Zhou, Z., Li, F., Zhu, H., & et al. (2019). An improved genetic algorithm using greedy strategy toward task scheduling optimization in cloud environments. *Neural Computing and Applications*, pp. 1–11.

Gaith Rjoub is currently a postdoctoral researcher at Concordia University. He received the M.Eng. degree in Quality Systems Engineering from Concordia Institute for Information Systems Engineering (CIISE), Canada in 2014, and Ph.D. in Information and Systems Engineering from the same university in 2022. His research interests include artificial intelligence, cloud computing, big data analytics, and internet of things. Moreover, he is a reviewer of several highly ranked journals.

Omar Abdel Wahab received the M.Sc. degree in computer science from the Lebanese American University, Beirut, Lebanon, in 2013, and the Ph.D. degree in information and systems engineering from Concordia University, Montreal, QC, Canada. He is currently Assistant Professor with the Department of Computer Science and Engineering, Université du Québec en Outaouais, Gatineau, QC, Canada. From 2017 to 2018, he was Postdoctoral Fellow with the École de Technologie Supérieure, Montreal, where he worked on an industrial research project in collaboration with Rogers and Ericsson. His current research activities are in the areas of cybersecurity, Internet of Things, artificial intelligence, and cloud computing.

Jamal Bentahar received his Ph.D. in computer science (artificial intelligence) from Laval University in 2005. He was an NSERC Postdoctoral Fellow at Simon Fraser University in the same year. He is currently a Professor with Concordia Institute for Information Systems Engineering at Concordia University. His research interests include machine learning (deep, reinforcement and federated), multi-agent systems, cloud/edge computing, and computational logics.

Robin Cohen is a Professor in the Cheriton School of Computer Science at the University of Waterloo in Waterloo, Ontario, Canada, conducting research in the subfield of artificial intelligence known as multiagent systems, with a particular interest in the topic of multiagent trust modeling and applications to social media.

Ahmed Saleh Bataineh is Postdoc fellow at QRST lab in Queen's university. He holds a Ph.D. in Information and Systems Engineering from Concordia University, Montreal, Canada. His past research activities include economics, optimization problems and game theory applied to IoT services and big data. His current research interests include artificial intelligence and cyber security.