Collaborative Federated Learning for 6G With a Deep Reinforcement Learning-Based Controlling Mechanism: A DDoS Attack Detection Scenario

Somayeh Kianpisheh and Tarik Taleb, *Senior Member, IEEE*

***Abstract*—Offering intelligent services with ultra low latency and high reliability is one of the main objectives of 6G networks. Federated Learning (FL) is a solution to enhance the security of data and the accuracy, in comparison with local training of data in devices. The transmission cost in conventional FL is high. Performing FL using edge infrastructure is a solution. However, edge servers might not be available at every location or the communication with edge resources may prolong the learning process. This paper proposes a collaborative federated learning approach to provide intelligent services through collaboration of various learning levels including central cloud level, edge cloud level, and device level. Computational capabilities of neighbourhood devices are exploited to provide a fast recognition via 6G D2D communication. The learning is modeled as an optimization that performs trade-off between recognition accuracy and response time of recognition for devices. Considering the dynamicity in communication and computation status of the network/devices, a deep reinforcement learning method is proposed to decide about the collaboration of learning levels, and performing the appropriate trade-off. For a DDoS attack detection scenario, the evaluation results show improvement in the gained rewards, the attack detection accuracy, the response time of recognition,and the accumulation of accuracy and response time.**

***Index Terms*—6G, federated learning, deep reinforcement learning, recurrent neural network, and DDoS attacks.**

I. Introduction

MOBILE communication generations have evolved to provide ubiquitous and seamless communication infrastructure for massive distributed end-devices. Supporting intelligent services, e.g., for autonomous vehicles, immersive services, with ultra-low latency and ultra-high reliability, is one of the essential visions of 6G networks [1], [2]. To realize such intelligent services, the infrastructure capabilities of 6G network should be leveraged to analyze heterogeneous and distributed data collected by massive end-devices, above all in real time. Utilizing local training of data can not end to adequate accuracy due to the small quantity of local data and computational power limitations of end-devices.

Federated Learning (FL) [3], provides capability to train a global model over distributed data. In FL, multiple participants train a shared model through an aggregator server (also called as parameter server). FL has shown to have the potential to provide the intelligent services for participating devices in 6G networks [4], [5]. Furthermore, the sharing of only the learning model parameters, instead of raw data of the participants, shall cope with the privacy concerns of end-users; one of the critical requirements of 6G networks [4].

However, the transmission cost is high in conventional FL, since the parameters of the learning model need to be transmitted to the cloud. To overcome this problem, some studies (e.g., [4], [5], [6], [7]) utilized edge processing capabilities, e.g., base stations, Road Side Units (RSUs), for partial model aggregation. This approach shall reduce the traffic at 6G backhaul network, however the edge-cloud infrastructure may not be accessible at some locations. For example, vehicles may not have access to RSUs at some roads or devices at disaster-affected sites may be disconnected from edge-cloud infrastructure. On the other hand, the capability of distributed cloud computing in 6G offers computation capabilities of intelligent neighbourhood devices [8], which can provide an opportunity of fast recognition. The literature, however, lacks an efficient mechanism for FL that utilizes the full potential of 6G infrastructure, i.e., end-devices, and edge/central-cloud.

This paper proposes a Collaborative Federated Learning (CFL) method that is consistent with the heterogeneous cloud computing based architecture of 6G [9]. The proposed method is capable to provide intelligent services in different learning levels including end-devices, edge-clouds, e.g., MEC servers and central-cloud. A Deep Reinforcement Learning (DRL) based controlling mechanism is proposed that decides about learning levels collaboration. The decision is based on a trade off between recognition accuracy and response time of recognition, depending on some variables indicating the dynamicity of the network. The communication status of the network, the computation capability of the edge/central-cloud resources, and geographical location/computation status of end-devices will be included to optimally collaborate among various learning levels.

To the best of our knowledge, this paper is the first work that adopts a FL version that exploits both the advantages of high accuracy of recognition, available by performing FL

Manuscript received 8 March 2023; revised 14 October 2023 and 8 February 2024; accepted 5 April 2024. Date of publication 12 April 2024; date of current version 21 August 2024. This work was supported in part by the EU's HE research and innovation programme HORIZON-JU-SNS-2022 under the RIGOUROUS project (Grant No. 101095933). The associate editor coordinating the review of this article and approving it for publication was C. Assi. *(Corresponding author: Somayeh Kianpisheh.)*

Somayeh Kianpisheh is with the Centre for Wireless Communications, University of Oulu, 90570 Oulu, Finland (e-mail: somayeh.kianpisheh@oulu.fi).

Tarik Taleb is with the Faculty of Electrical Engineering and Information Technology, Institute of Networked Energy Efficient Systems, Ruhr University Bochum, 44801 Bochum, Germany (e-mail: tarik.taleb@rub.de).

Digital Object Identifier 10.1109/TNSM.2024.3387987

through edge/central-cloud resources; and the fast recognition capabilities of neighbourhood devices, available through 6G D2D communications. Furthermore, to the best of our knowledge, this is the first work that exploits collaboration between various learning levels, and introduces the trade off between the recognition accuracy and the response time of recognition to optimally manage the collaboration.

As security is a major concern in 6G networks, the proposed method is evaluated for a DDoS attack detection scenario. The collaborator devices will take part in FL to learn a model for DDoS attack detection. As attack detection has shown to have the characteristic of a time-series optimization and GRU-based detection approaches have shown promising results in terms of detection accuracy [10], [11], we also adopt a GRU based method to detect the DDoS attacks. The evaluation results illustrates the outperformance of the proposed method in terms of gained rewards and the recognition accuracy in comparison with existing methods in the literature; when FL is performed either based on sole edge infrastructure or sole end-devices computation capabilities. Furthermore, the response time of recognition has been reduced in comparison with widely advocated approach in the literature that performs FL with exploitation of edge infrastructure. Finally, the experimental results illustrate the effectiveness of CFL to perform appropriate trade-off between accuracy and response time. The contributions of this paper are as follows:

1) Optimization framework for Collaborative Federated Learning (CFL) to optimize the accumulated trade-off of recognition accuracy and response time of recognition for all devices in the network.
2) A deep reinforcement learning based method to solve the proposed optimization, while considering the network status variables including the network communication/computation status and geographical location/computation status of end-devices.
3) Extensive experiments to illustrate the outperformance of the CFL in comparison with the baselines, for a DDoS attack detection scenario from the aspects of criteria including the response time of recognition, the attack detection accuracy, and the accumulated response time and accuracy.

The rest of this paper is organized as follows. Section II discusses related works. Section III provides the motivation and the incorporation of the proposed federated learning in 6G. In Section IV, we explain our proposed collaborative federated learning. In Section V, we explain DRL based optimization and the controlling mechanism for CFL. Section VI gives the technique for DDoS attack detection, as well as the FL scenario to learn an attack detection model in a distributed manner. In Section VII, we discuss the experimental results. Finally, conclusions and the future work are given in Section VIII.

II. Related Works

As this paper studies FL in 6G, we investigate the literature of FL in the scope of 6G/5G and the relevant scope of edge computing. The related studies are categorized in two groups: Edge/Cloud-Based FL and End-Devices-Based FL. Edge/Cloud-Based FL methods perform FL in a hierarchical manner with specific parameter aggregator(s) provided by edge/cloud computing nodes. End-Devices-Based FL methods perform FL in a fully distributed manner and rely only on end-devices, with no specific aggregator server at edge/cloud. In the rest, we discuss the related works, their drawbacks and the contributions of this paper.

A. Edge/Cloud-Based FL

To reduce the overhead of transmission cost to a central aggregator in conventional FL, several studies advocate hierarchical FL by utilizing edge processing capabilities. Indeed, these studies exploit edge infrastructure elements, e.g., edge servers, Base Stations (BSs), Road-Side Units (RSUs) for partial aggregation of the model. In the rest, we first review non-blockchain-based FL methods in the general context of edge computing and specific applications. Then, we review blockchain-based FL methods.

FL has been studied in the general context of edge computing in [6], [12], [13], [14], [15]. The authors of [12] study FL for the cases of non-Independent and Identically Distributed (non-IID) data for wireless networks. The study in [6] utilizes evolutionary game theory to dynamically assign participants to head clusters in a FL process for an edge intelligence application. The authors of [13] focus on the problem of joint user association and wireless resource allocation in hierarchical FL, under IID and non-IID cases. In this study, BSs perform partial aggregation and the central cloud performs global aggregation. Joint optimization of mobile devices transmission power and training task scheduling was studied in [14]. Facilitating FL through intra-cluster collaboration as well as inter-cluster collaboration using hierarchical edge servers has been discussed in [15].

FL has been studied for specific applications of 6G digital twin [5], [7], vehicular networks [4], [16], and Unmanned Aerial Vehicles (UAV) networks [17]. Hierarchical FL for digital twin applications in 6G was investigated in [5], [7]. FL is organized through base stations and macro-base stations to accomplish the digital twin processing in [5]. Considering the assignment of digital twin processing to the base stations as well as the bandwidth allocation, the problem is modeled as an optimization that minimizes the time/cost of federated learning. Similarly, the study in [7] provides an application of digital twin processing at the base stations. Applications of hierarchical FL in vehicular communications were investigated in [4], [16], [18]. The study [4] focuses on a CNN based FL for the application of object detection at which road-side units perform partial aggregation, and global aggregation is performed in a central cloud server. The study in [16] customizes parameters including local training for vehicular clients and the update time. It also proposes a strategy for partial client participation to optimize the communication cost in FL. In [17], UAVs and remote radio heads, besides a centralized controlling unit, perform the aggregation in a hierarchical manner. The study focuses on a content placement decision in 6G, which decides based on the content popularity and the UE mobility.

Blockchain technology can provide secure storage capabilities and distributed solutions [19]. It has been integrated with edge/cloud-based FL to enhance the security/reliability of FL from the aspects including verification of local models, e.g., [20], [21], [22], [23]; verification of global models, e.g., [18], [24]; verification of aggregators [25]. Generally, blockchain can enhance the safety/reliability of FL by providing a distributed ledger to securely store the local/global model(s) as transactions data in a transparent and traceable manner, e.g., [18], [20], [21], [26], capabilities of decentralized coordination and consensus based decision, e.g., [21], [25], and smart contracts-based controlling, e.g., [20], [22], [23]. In the rest, we explain more details of the blockchain-based FL works.

In [20], FL has been utilized to facilitate content caching for IoT devices. Hierarchies of edge nodes and a cloud server form the architecture of FL. Blockchain has been incorporated to store the model gradients of the edge nodes as transactions, and accordingly verifying the transactions' validity based on a verification smart contract and a consortium decision. The verified models then, will be obtained by the cloud server to be federated for the purpose of caching decision. The authors of [21] propose a two-layered blockchain architecture for FL over mobile edge network. The blockchain structure include a Local Model Update Chain (LMUC) and a Global Model Update Chain (GMUC). LMUC is used to share the local models update among devices, with the capability of a consensus-based verification mechanism. GMUC is used to store reputation of devices from the aspect of model quality as well as coordinating cross-validation among Multi-access Edge Computing (MEC) nodes to form the aggregate global model. The studies in [22], [23] integrate validation mechanisms through a test-data-set, with smart contracts to validate local models against poisoning attacks. The study in [24], applies FL over 5G infrastructure including devices, BSs, and a macro-BS. The local models are aggregated by BSs and macro-BS to form a global model. The global model is stored in the blockchain, where a cryptographic-based computation is utilized to authenticate and verify it before being downloaded by devices. The authors of [18] utilize FL to facilitate collaborations of vehicles to detect intrusion detection. The local-trained models of vehicles are aggregated by RSUs. RSUs jointly maintain a secure blockchain-based model storage to prevent tampering of the aggregate model. The authors of [25] study a blockchain-based FL for cross-domain UAVs with the authentication capability for UAVs. A consortium blockchain is utilized as a platform for information exchange. Committee nodes, i.e., BSs or RSUs, based on a consensus-based algorithm, authenticate the UAVs, and select a subset of the committee nodes for the aggregation of the local models stored in a model pool. Fusion of blockchain and FL for data verification at the edge layer has been investigated in [27] for 5G/6G network slicing application. The authors of [28] study unintended property leakage problem in blockchain-based FL for intelligent edge computing. The authors of [26] propose a decentralized blockchain based trading system to encourage edge nodes to participate for model training in FL.

B. End-Devices-Based FL

These studies have advocated a fully distributed version of FL at which it relies only on end-devices and no aggregator server from edge/cloud computing (e.g., BSs, MEC servers, RSU, cloud server) is involved in FL. FL has been used in [29] to augment the capability of road object classification based on Lidar data in vehicular networks. Each vehicle communicates with its neighbourhood vehicles to receive the weights of the learning model and perform the aggregation itself, as well as updating the neighbours with the aggregated weights. A consensus based approach with the cooperation of devices to perform FL for massive IoT network was proposed in [30]. The studies in [31], [32], [33], [34] apply blockchain mechanisms in FL. The authors of [31] provide a blockchain-based architecture to implement a fully distributed FL. The devices upload the local model parameters to a subset of devices namely called, the miners. The reliability of the model parameters are verified by miners. A consensus algorithm, e.g., PoW, PoS is run in each run to select a miner as the aggregator. The aggregator device, aggregates the parameters and saves the new global parameters in a block, from which the devices download the global parameters for the next round. The study in [32], propose a fully distributed blockchain-based FL for vehicular network at which the verification of the local models are handled by miners (a subset of vehicles). The verified local models are stored as blocks in the distributed ledger of the blockchain, and the global updates are calculated locally at vehicles. The minimization of system delay and driving the optimal block arrival rate is also considered in FL. The study in [33], enhances the data privacy of participants in a blockchain-enabled FL in beyond 5G, with a Wasserstein Generative Adversarial Network (WGAN)-based mechanism. The authors of [34] implement a distributed FL for mobile devices, at which a requester device triggers the learning task, and other mobile devices take part in FL through local training models. To encourage the participation in FL, an incentive mechanism is utilized and a blockchain based game is designed, according which the devices can adjust their strategies for participation, to maximize the individual payoff.

C. The Drawbacks of the Existing Works and the Contributions of This Paper

Edge-Based FL approaches, e.g., [4], [5], [6], [7], [12], [13], [14], [15], [16], [17] reduce transmission cost by performing model aggregation at the edge of the network, instead of parameter transmission to a central cloud. However, still the latency of parameter transmission to edge infrastructure can be intolerable for ultra-low latency applications. Although the blockchain-based FL approaches make the possibility of parameters uploading/downloading to/from an intermediary data ledger, still the storage is maintained through edge/cloud computing. Furthermore, these approaches, e.g., [18], [20], [24], [25], [28] introduce extra computation overhead and latency, due to storage management, and consensus processing. Indeed, Edge/Cloud-Based FL approaches do not utilize the processing capabilities of the neighbourhood smart devices, accessible via 6G D2D communications. Exploiting the processing capabilities of the neighbourhood smart devices can yield a fast recognition and can be more promising for 6G ultra-low latency applications. Furthermore, as edge infrastructures may not be available at every location, there is a limitation in the application of these approaches.

On the other hand, End-Devices-Based FL approaches, e.g., [29], [30], [31], [32], [33], [34] omit the aggregator server at edge/cloud and rely on computational capabilities of devices for the purpose of model aggregation. These approaches may yield a fast aggregation and accordingly recognition. However, they can easily deplete devices due to their limited power, since computation consumes energy in the devices. Particularly, for the blockchain-based FLs, there is even more resource consumption due to the blockchain-related processing at devices, e.g., consensus processing [31], model verification [32], block management [31], [32], [33], [34]. Furthermore, these approaches may not be as accurate as Edge/Cloud-Based FL approaches which perform model composition for a wide range of devices.

To the best of our knowledge, the CFL method proposed in this paper, is the first work that performs federated learning by a collaboration of various learning levels including both edge/cloud-level FL and devices-level FL. The collaborative approach has the potential of exploiting the advantages of both approaches, i.e., fast recognition of End-Devices-based approaches and high accuracy of Edge/Cloud-based FL approaches. On the other hand, the drawback of End-Devices-based approaches can be avoided by offering the capability of offloading parameter aggregation to edge/cloud servers (e.g., MEC servers) in order to avoid depletion of devices. The drawback of Edge/Cloud-based FL approaches can be avoided by offering the capability of sharing the model parameters with neighbourhood devices when the edge infrastructure becomes unavailable. Proposing a collaborative FL approach with respect to the dynamic status of communication/computation in the network, as well as end-devices geographical location, is the focus of this paper. Section III gives more details about the motivation of our collaborative FL approach.

III. Motivation and Incorporation

There are three facts that motivate adopting a collaborative approach to realize federated learning in 6G networks:

(i) 6G network architecture will be based on heterogeneous cloud computing including central cloud, edge clouds, as well as intelligent devices [9]. 6G is envisioned to offer ubiquitous intelligence in the network. In this regard, a federated learning approach that can provide intelligence in the hierarchies covering from end-devices, to edge clouds and central cloud, is required.

(ii) Provisioning the intelligence based on data sharing capabilities of neighbourhood devices would result in fast recognition. On the other hand, provisioning the intelligence using edge/central cloud can end to more accuracy due to

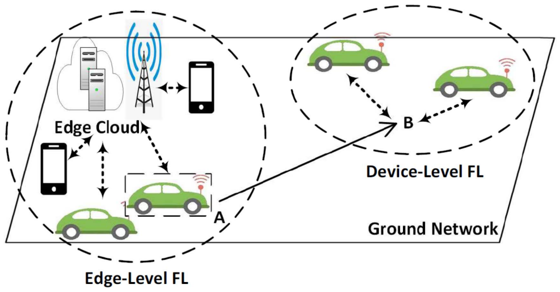

Fig. 1. The motivation behind CFL. Switching from Edge-Level FL to Device-Level FL for the purpose of wide-connectivity/fast recognition.

having access to the learned parameters obtained from wider number of end-devices. Indeed, in comparison with local federation, the aggregated model by edge/central cloud can have better generality, thereby achieving higher recognition capabilities. An appropriate trade-off is required for an optimal decision based on the network condition and the application requirement.

(iii) Wide-connectivity is a critical requirement in 6G architecture [35]. In this regard, an efficient scheme is required to maintain the connectivity of the devices to intelligent services, when the geographical locations of edge-devices change. Our proposed scheme applies automatic controlling for dynamic switching among learning levels of central/edge-cloud-level and device-level to meet the wide-connectivity requirement.

Fig. 1 illustrates the use case of wide-connectivity/fast recognition in a 6G terrestrial network. The vehicle at location A, uses edge-level FL provided by the edge cloud infrastructure at the left side. Indeed, the model parameters of devices in the left-side circle are aggregated and updated by processing capability of the edge cloud. The vehicle moves to the location B which is far from the edge cloud. If at location B, the vehicle continues receiving the intelligent service from the edge cloud, there will be high latency, which can be intolerable for ultra-low latency applications or even be a kind of unavailability of the intelligent service.[1] Note that in the context of this paper, inference based on (partial-) federated model is considered to be the intelligent service for devices. To provide fast recognition capability at location B, the intelligent service can be provided through Device-Level FL with the participants including the neighbors of the vehicle at location B. The vehicle at location B, can receive the neighbors models, aggregate them and update its/their model. Or a neighbour device can perform the aggregation. Collaborative Federated Learning (CFL) provides the capability of switching from Edge-Level FL to Device-Level FL to maintain the connectivity of intelligent service and keep the fast recognition capability, as the vehicle moves. However, after switching to the new learning level, the accuracy of recognition might reduce due to a local aggregation of the models. Indeed, the number of devices that can take part in Device-Level FL is smaller than the number of devices that can take part in

[1]Even in the case that another edge cloud be available at location B, an appropriate decision is required to decide if taking part in Device-Level FL or Edge-Level FL can optimize the accumulation of accuracy and response time of recognition, as we have defined it in Eq. (20)-(22).

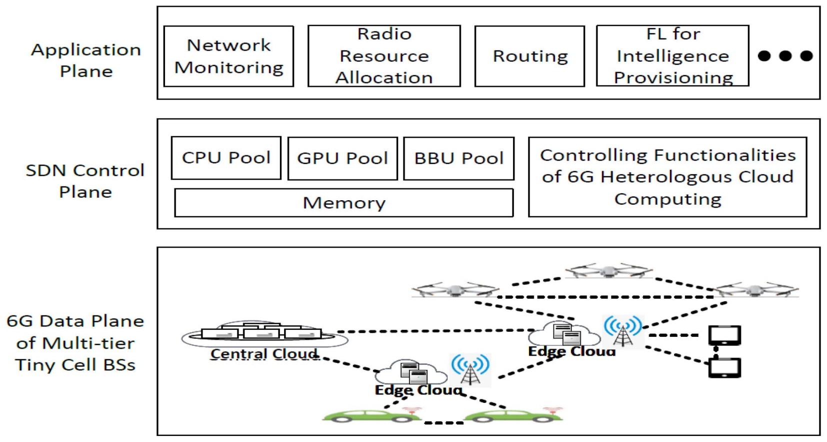
Application Plane
Network Monitoring
Radio Resource Allocation
Routing
FL for Intelligence Provisioning
SDN Control Plane
CPU Pool
GPU Pool
BBU Pool
Memory
Controlling Functionalities of 6G Heterologous Cloud Computing
6G Data Plane of Multi-tier Tiny Cell BSs
Central Cloud
Edge Cloud
Edge Cloud

Fig. 2. The proposed incorporation framework.

Edge-Level FL due to shorter distance limitation for D2D communication, in comparison with wider coverage radius of BSs. Consequently, less data samples can be available in Device-Level FL, and the accuracy of recognition may reduce.[2]

To have an efficient collaboration mechanism, an appropriate trade-off is required depending on the priority between criteria of accuracy and response time of intelligent service. Our proposed scheme of CFL employs a deep reinforcement learning based controlling mechanism to automatically perform the required collaborations among learning levels, as well as the aggregator selection, for devices who take part in the Device-Level learning.

To incorporate CFL into 6G networks, we propose a similar framework to [17]. Fig. 2 illustrates the proposed incorporation framework. A number of BSs equiped with edge clouds as well as a central data center, serve end-devices, e.g., autonomous vehicles, UAVs, pedestrian cellphones, IoT devices. Virtualization technologies based on Virtual Network Functions (VNFs), Virtual Machines (VMs), and network slicing, provide the capabilities by which the network management algorithms can be virtualized as applications running over the 6G network commodity hardware/data plane. Network slicing [36] can share physical resources like Centralized Processing Units (CPUs), Graphics Processing Units (GPUs), Base Band Units (BBUs), and other computing resources, for the purpose of required computations of network management applications in the application plane, e.g., routing, radio resource allocation, network monitoring, and FL for intelligence provisioning. Software Defined Network (SDN) control plane manages the software-defined/virtual resources and the integrated communication links (e.g., facilitating the communication among devices and BSs required for carrying out FL). It also performs controlling functionalities of 6G heterogeneous cloud computing environment. FL as an application for intelligence provisioning can be facilitated by our proposed CFL, to improve an accumulation of response time of recognition and recognition accuracy for the end-devices, as well as keeping the connectivity to the intelligent service. Next section gives details for computation and networking models to perform CFL.

[2]Usually, in DNNs with high number of parameters and non-linear discriminant analysis the accuracy increases when more samples are available for training. However, the accuracy also depends on other factors. e.g., quality of data, the data samples distribution. It might happen that due to unbalanced data or low-quality data, the accuracy decreases even if more samples are available. In the experiments of this paper on CICDDoS 2019 dataset, we found the increment in accuracy when the number of samples increases.

IV. Collaborative Federated Learning (CFL)

A 6G network consists of N end-devices, M BSs equipped with MEC servers, and a (central) cloud. In the rest of this paper, we represent edge clouds with MEC servers collocated with BSs, as it is a common edge computing paradigm to provide computation at the edge of the network. The data of end-device u with size $|D_u|$ is represented with $D_u = \{(x_1^u, y_1^u), \ldots, (x_{|D_u|}^u, y_{|D_u|}^u)\}$, where y is the label for input x. Each end-device u performs learning on its local perceived data to construct a local model M_u, in order to perform the required recognition (e.g., attack detection). Data sharing among the end-devices is possible in order to achieve a more accurate model for the inference. The data sharing can be done either through D2D communication or through MEC/cloud infrastructure. Instead of raw data transmission which has bandwidth consumption overhead, we adopt FL to learn global model M. To cope with the availability of MEC/cloud infrastructure, as well as an appropriate trade-off between the accuracy and the response time of recognition, the decision is made to optimally select the appropriate data sharing mechanism through providing a dynamic context-aware collaboration among the learning levels the device will be involved. The details will be given in the following sections.

A. Cloud Level Federated Learning

The aim of federated learning is to train a global learning model M_c based on the distributed data in the devices that take part in cloud-level learning, i.e., U_c. The parameter w_c for model M_c is found in learning process to minimize the global loss function:

$$\min_{w_c} F_c = \frac{1}{|U_c|}\sum_{u=1}^{|U_c|}\frac{1}{|P(u).D_u|}\sum_{s=1}^{|P(u).D_u|} f(w_c, x_s^u, y_s^u), \quad (1)$$

where *P* is participation vector denoting the training batch size of the participants in the learning. The entry *P*(*u*) indicates the portion of local batch of user *u* that will be involved in the training. Symbol $f(w_c, x_s^u, y_s^u)$ in (1) is the loss value for the sample *s* of data of user *u*.

Considering the computational capabilities of devices in 6G [4], [17] a gradient decent based method is employed at each end-device to minimize the loss function based on its local batch data. Then, the devices that take part in cloud-level learning, will transmit the parameters of the local model to the BSs for the purpose of partial aggregation (which is performed by the MEC server co-located with the BS). Let i^{th} *BS* be represented with BS_i. The partial aggregation at this BS, is performed as a weighted average of received parameters to construct the model $M_{e,i}$, as below:

$$M_{e,i} = \frac{1}{K_i} \sum_{u=1}^{K_i} \mathcal{N}(P(u).|D_u|).\gamma_u.w_u, \tag{2}$$

where K_i is the number of end-devices under the coverage of BS_i, who take part in the cloud-level learning; and $P(u).|D_u|$ is the size of batch data of device *u* involved in the training; γ_u is the context-weighting coefficient which prioritizes the device learned weight according to the context information (e.g., reputation of device in security report); and finally, w_u is the weight of the trained model of device *u*. Symbol $\mathcal{N}(.)$ is the normalization operator.

The BSs send the partial aggregated model parameters to the central cloud to perform the global aggregation. Thus, the global model M_c is constructed at central cloud as average of models of BSs as below:

$$M_c = \frac{1}{M} \sum_{i=1}^{M} M_{e,i}. \tag{3}$$

The global aggregated parameters will be broadcasted to the BSs and accordingly to the devices involved in the cloud level learning. Similarly, the above-mentioned equations can be used for edge level learning. In this regard, the constructed model parameters at a edge BS, i.e., $M_{e,i}$ is transmitted to the devices under the coverage of that BS.

Note: In (2), for the purpose of simplicity in representation, we assumed all devices under the coverage of BS are involved in edge/cloud level FL. However, in CFL through applying a DRL method, at each epoch of federated learning, for each device it is decided if it will take part in edge/cloud level FL or not. Therefore, only the determined devices and their count will be involved in calculations of (2). The variables $E(\tau)$ and $C(\tau)$ in (22) will define the active devices in edge and cloud level federated learning, respectively (at epoch τ).

B. Device Level Federated Learning

The aim of federated learning is to train a global learning model M_d based on the data in a device, defined as aggregator, and its neighbour hood N_d. Note that the aim is the globalization within the neighbourhood. The parameter w_d for model M_d is found in learning process to minimize the loss function as below:

$$\min_{w_d} \ F_d = \frac{1}{1 + |N_d|}. \sum_{u \in d \cup N_d} \frac{1}{P(u).|D_u|} \sum_{s=1}^{|P(u).D_u|} f(w_d, x_s^u, y_s^u). \tag{4}$$

A gradient decent based method is employed at each end-device *u* involved in device level learning, to minimize the loss function, i.e., $f(w_u, x_s^u, y_s^u)$ based on its local batch data. Then, a device that has been indicated as aggregator, will receive the parameters of the local models from its neighbourhood to perform the aggregation. The aggregation process is calculating a weighted average of neighbourhood devices parameters as defined in (5). The context-weighting coefficient of device *u*, i.e., γ_u and the size of its batch data, i.e., $P(u).|D_u|$ define its weight in the aggregation. Note that in device level FL, all aggregators in the network, in parallel perform receiving the parameters from their neighbours and the aggregating. Section V-A discusses how the aggregators can be defined within a DRL procedure.

$$M_d = \frac{1}{1 + |N_d|} \sum_{u \in d \cup N_d} \mathcal{N}(P(u).|D_u|).\gamma_u.w_u, \tag{5}$$

Note: In (4) and (5), for the purpose of simplicity in representation, we assumed all neighbors are involved in device level FL. However, in CFL through applying a DRL method, at each epoch of federated learning, for each neighbour of aggregator it is decided if it will take part in device level FL or not. Therefore, only the determined devices and their count will be involved in calculations of (4) and (5). The variables $D(\tau)$ and $G(\tau)$ in (22) will define the active devices in device level federated learning and the aggregators, respectively.

C. Optimization Formulation

We use the 6G wireless communication model in [5], [17] for communication between devices and the base stations. The transmission rate, according which device *u* communicates with BS *i*, i.e., $R_{u,i}$, is estimated as below:

$$R_{u,i} = B_i \ln\left(1 + \frac{Pt_u.g_u}{\eta}\right), \tag{6}$$

$$g_u = C_g.d_{u,i}^{-\alpha}, \tag{7}$$

where B_i, Pt_u, g_u, η are respectively, transmission bandwidth of the base station, transmission power of the device, channel gain of the device, and background noise power. The channel gain can be calculated by (7), as a function of path loss fading coefficient, i.e., C_g, distance between device *u* and base station *i*, i.e., $d_{u,i}$, and path loss exponent, i.e., α.

At every epoch of FL, the response time of recognition for a device can be estimated as summation of the time it takes the device receives the new updates of the parameters of the model and the inference time. Receiving new updates of the parameters, includes the time slots allocated for local training, up/down-link parameter transmission, and the aggregation. These time slots are calculated depending on the learning level, at which the device operates. Indeed, this time ensures inference based on the partial/global aggregated model. Note that in the rest of this section for each levels of learning, we assume only the devices that have been selected to operate on that specific learning level, will be involved in the calculations.

1) Cloud-Level Learning:

- *Aggregation:* The time for partial aggregation at BS i includes: (a) the time it takes the parameters be transmitted from devices (who take part in the cloud level learning) under the coverage area of that base station, i.e., R_i to the base station (the first term in (8)); (b) the aggregation operation time (the second term in (8)):

$$T^i_{ag} = \max_{u \in R_i} \frac{|w_u|}{R_{u,i}} + \frac{|R_i|.|w_g|.f^{cmp}_w}{f^{cmp}_i}, \tag{8}$$

  where $|w_g|$ is the number of global weights (usually the same as the local weights, i.e., $|w_u|$), f^{cmp}_w and f^{cmp}_i are respectively the number of required CPU cycles to aggregate one unit of data, and CPU frequency of base station i.
  The time for global aggregation at central cloud is calculated by (9). It includes the time for partial aggregation at base stations (the first term), as well as the time takes for global aggregation of the parameters collected from M base stations at the central cloud (the second term):

$$T_{ag} = \max_{i=1..M} \left(T^i_{ag} + \frac{|w_g|}{R_{i,c}} \right) + \frac{M.|w_g|.f^{cmp}_w}{f^{cmp}_c}, \tag{9}$$

  where $R_{i,c}$ is the bandwidth of communication between BS i and the central cloud, and f^{cmp}_c is the available CPU cycle at the central cloud. Note that the uplink parameter transmission time has been considered in the aggregation time calculations.
- *Downlink Parameter Transmission:* The required time to download the parameters at device u under the coverage of BS i is calculated based on the size of parameters, the transmission rate between the central cloud and the base station, as well as the transmission rate between the base station and the device. Equation below shows the calculation:

$$T_{down}(u) = \frac{|w_u|}{R_{c,i}} + \frac{|w_u|}{R_{i,u}}, \tag{10}$$

- *Local Training:* Local training time at device u is calculated based on the computing capability of the device and the batch size the device will include in training. Equation below illustrates the calculation:

$$T_{loc}(u) = \frac{P(u).|D_u|.f^{cmp}_s}{f^{cmp}_u}, \tag{11}$$

  where f^{cmp}_s and f^{cmp}_u are respectively number of required CPU cycles to train one sample of data, and CPU frequency of the device.

The response time of recognition in one epoch of FL, for device u who takes part in cloud level learning in that epoch, is calculated as below:

$$T^c_{Int}(u) = \max_u K.T_{loc}(u) + T_{ag} + T_{down}(u) + \frac{f^{inf}_u}{f^{cmp}_u}, \tag{12}$$

where K is the local training iterations before applying the other epoch of learning. In (12), f^{inf}_u is the number of CPU cycles to perform the recognition for a sample of input features. Note that in (12), maximum of local training time of the devices is included in the calculation. The reason is that the parameter aggregation phase can be started, whenever the local training of the devices be completed.

The response time of recognition for device u that take part in edge level learning, is represented with $T^e_{Int}(u)$. It is calculated similar to the cloud level learning calculations that we explained above. However, for the calculation the data flow will be terminated at the base station that cover the device.

Note: At every epoch of FL, only the devices that are selected for cloud/edge level learning collaboration, will be involved in calculations of (8)-(12). For simplicity in representation, we have not indicated this matter in these equations. The variables $E(\tau)$ and $C(\tau)$ in (22) will define the active devices in the edge and cloud level federated learning, respectively (at epoch τ).

2) Device-Level Learning: The 6G infrastructure capabilities for D2D communication, as well as the availability of distributed cloud computing paradigm in 6G enables collaboration of neighbourhood devices for sharing the parameters. In this regard, in an epoch of learning, it is also possible that devices take part in the device level learning. In the rest, we explain the response time of recognition calculation for the aggregator device and the neighbourhood devices.

The response time of recognition for the aggregator device u (in one epoch of FL), is calculated as below:

$$T^d_{Int}(u) = \max_{n \in u \cup N_u} K.T_{loc}(n) + T^u_{ag} + \frac{f^{inf}_u}{f^{cmp}_u}, \tag{13}$$

It includes the time slots required for local training at the device and the neighbourhoods (the first term in (13)), the time slots for parameter transmission from the neighbourhood devices and accordingly the aggregation operation (the second term in (13)), and finally, the inference time (the third term in (13)). In (13), T^u_{ag} is the parameter-download/aggregation time, and is calculated as below:

$$T^u_{ag} = \max_{n \in N_u} \frac{|w_n|}{R_{n,u}} + \frac{N_u.|w_g|.f^{cmp}_w}{f^{cmp}_u}, \tag{14}$$

where $R_{n,u}$ is the transmission rate of device n to communicate with device u.

When device n who takes part in the device level FL is not an aggregator, downloading the parameters of the model after the aggregation (performed by node u) is required for recognition. Thus, the response time of recognition is calculated as below:

$$\widehat{T^d_{Int}(n)} = \max_{u' \in \cup N_u} K.T_{loc}(u') + T^u_{ag} + \frac{|w_g|}{R_{n,u}} + \frac{f^{inf}_n}{f^{cmp}_n}. \tag{15}$$

Note: At every epoch of FL, only the devices that are selected for device level learning collaboration, will be involved in calculations of (13)-(15). For simplicity in representation, we have not indicated this matter in these equations. For example, in (14), only the neighbours that have been selected for the device level learning will be involved in the first term, and their count will be a replace for N_u in the second term. The variables $D(\tau)$ in (22) will define the active devices taking part in the device level federated learning, at epoch of learning τ. Accordingly, the aggregator nodes will be defined by $G(\tau)$ in (22).

3) Objective Function and Constraints: Receiving intelligent service from cloud/MEC servers can end to a higher recognition accuracy, though causing more delay in the service or even the unavailability of the service in the locations far from cloud/MEC infrastructure. On the other hand, participation of a device in the device level federated learning can end to a fast recognition which can be essential for fast react scenario or ultra-low-latency applications. However, some accuracy loss might be experienced due to the issue of more locality in constructing the parameters of the model. Indeed, in comparison with the cloud/edge level learning which offers the possibility of aggregating the parameters of more devices, in device level learning, only the parameters of the neighbourhood devices are aggregated. Thus, appropriate trade-off is required for the optimal collaboration among learning levels.

The dynamicity in geographical location of devices, the quality of channel connection among devices and the base stations/cloud, the dynamicity in D2D channel communication quality, and the variation of available computation capabilities of MEC servers/devices can influence the optimal trading-off decision. Furthermore, controlling on the participation of devices in the training, from the aspect of batch size involvement can avoid prolonging the learning process due to the low available computation of devices. Thus, for constructing an optimization model for collaboration among the learning levels of cloud, edge, and device, a time-series optimization modeling is proposed to perform the required trade-off and the aforementioned controlling, based on the dynamic status of the network/devices. Note that in the rest of this paper, we use the terms *epoch of learning* and *interval* interchangeably.

The optimization variables are defined as $P(\tau)$, $D(\tau)$, $E(\tau)$, $C(\tau)$, $G(\tau)$. Vector variables $P(\tau)$ has N entries which denotes the participation of devices in FL at interval τ. $P(\tau)[u] > 0$ denotes the batch size of device u for training, at interval τ. Vector variables $D(\tau)$, $E(\tau)$, $C(\tau)$ each with N entries, represent FL levels collaboration. $D(\tau)$ denotes the devices which take part in the device level learning at interval τ. The value of 1 for $D(\tau)[u]$ indicates device u would take part in the device level learning at interval τ (the value of 0, otherwise). Similarly, $E(\tau)$, $C(\tau)$ will indicate the collaboration of devices respectively in the edge level and the cloud level learning, at interval τ. For example, when $E(\tau)[u]$ is 1, the device u will take part in the edge level learning at interval τ. As each device participates in either of device, edge or cloud level of learning at every interval, we have:

$$\forall u{:}D(\tau)[u] + E(\tau)[u] + C(\tau)[u] = 1. \tag{16}$$

For the devices that take part in the device level learning, aggregator nodes are required to be defined. Vector variable $G(\tau)$ with N entries, denotes the aggregators at interval τ. In this regard, when device u is aggregator, $G(\tau)[u]$ has the value 1, otherwise it takes the value 0. The required topology for device level learning is defined by the aggregators and their neighbours through constraints (17)-(19).

Constraint (17) ensures an aggregator takes part in the device level learning. Constraint (18) ensures at least one of the neighbours of an aggregator device be involved in the device level learning. Constraint (19) ensures every device involved in the device level learning be associated to an aggregator located in the neighbourhood.

$$\forall u : G(\tau)[u].D(\tau)[u] = G(\tau)[u], \tag{17}$$

$$\forall u : G(\tau)[u]. \sum_{n\in N(u)} D(\tau)[n] = G(\tau)[u], \tag{18}$$

$$\forall u : D(\tau)[u]. \sum_{n\in N(u)} G(\tau)[n] = D(\tau)[u]. \tag{19}$$

Let the set of devices who take part in the device level of learning at time interval τ, be represented with U_d, i.e., $U_d = \{u|D(\tau)[u] = 1\}$. Similarly, we represent the users that take part in the edge level and the cloud level learning with respectively, U_e and U_c. Finally, we represent the set of aggregator devices at interval τ with $AG = \{u|G(\tau)[u] = 1\}$.

The objective function is defined as the mean of the loss function and the response time of recognition for the devices in the 6G network. The accumulation of the aforementioned criteria is defined by performing a trade-off between the gained loss value and response time of recognition. The trade-off for devices involved in the device level learning within interval τ, is calculated by (20), as a weighted accumulation of the loss function (the first term in (20)) and the response time of recognition for aggregator devices (the second term), and the response time for non-aggregator devices (the third term):

$$\begin{aligned} T_d(\tau) = (1-\beta). \sum_{u\in U_d} \mathcal{N}(F_u(\tau)) \\ + \beta. \sum_{u\in AG} \mathcal{N}(T^d_{Int}(\tau)[u])) \\ +\beta. \sum_{u\in U_d \setminus AG} \mathcal{N}(\widehat{T^d_{Int}}(\tau)[u])), \end{aligned} \tag{20}$$

where the coefficient β illustrates the priority of the response time of recognition in comparison with the accuracy. $\mathcal{N}(.)$ is the normalization operator to convert the time and loss in the same scale.

For devices involved in the edge level learning the accumulation of loss function and response time of recognition for interval τ is calculated as below:

$$\begin{aligned} T_e(\tau) = (1-\beta). \sum_{u\in U_e} \mathcal{N}(F_u(\tau)) \\ + \beta. \sum_{u\in U_e} \mathcal{N}(T^e_{Int}(\tau)[u]) \end{aligned} \tag{21}$$

The trade-off at the cloud-level learning, i.e., $T_c(\tau)$ is calculated similar to (21), by replacing U_e with U_c, and T^e_{Int} with T^c_{Int}.

The objective function is defined as the accumulated loss function and response time of recognition experienced by the devices during interval of times, as shown in (22). Here, τ_M is the number of FL epochs. Relying on [5], an upper bound is

$o(log(1/\theta_L)/1-\theta_G)$, at which θ_L is the local accuracy gained in the end-devices, and θ_G is the global accuracy gained at the central cloud.

$$\phi = \min_{D(\tau),E(\tau),C(\tau),P(\tau),G(\tau)} \sum_{\tau=0}^{\tau_M} \frac{T_d(\tau) + T_e(\tau) + T_c(\tau)}{N} \tag{22}$$

Sbj. to:(1), (2), (3), (4), (5), (6), (7), (8), (9), (10), (11), (12), (13), (14), (15), (16), (17), (18), (19), (20), (21).

V. DRL Based Optimization and Controlling Mechanism

The search space problem defined in (22) is of exponential order with non-linear involvements of elements, e.g., (1), (4), (13), and the gradient based learning procedure which is applied at each device. Due to the dynamic parameters involved in the problem including wireless channel transmission state, location of the end-devices (see distance involvement in (7)), available computational capabilities of MEC servers/devices, heuristic and (meta)heuristics/evolutionary are not efficient to solve the problem. Indeed, the necessity of recalculations in these methods, every time the parameter changes will make these methods inefficient to solve the problem [37]. We utilize Deep Reinforcement Learning (DRL) that automatically captures the dynamic statistics of the involved parameters to solve the optimization in (22), and applies the required controlling and flexibility. The problem in (22) can be formulated in the form of a Markov Decision Process (MDP).

Function in (22) is the accumulated aggregation of loss function and response time of recognition within time intervals. Let $\phi(t)$ be the aggregated value up to interval *t*. We have:

$$\phi(t) = \phi(t-1) + \frac{1}{N}. \Bigg[(1-\beta).\sum_{u\in U_e} \mathcal{N}(F_u(t)) + \beta.\sum_{u\in U_e} \mathcal{N}(T^e_{Int}(t)[u]) + (1-\beta).\sum_{u\in U_c} \mathcal{N}(F_u(t)) + \beta.\sum_{u\in U_c} \mathcal{N}(T^c_{Int}(t)[u]) + (1-\beta).\sum_{u\in U_d} \mathcal{N}(F_u(t)) + \beta.\sum_{u\in AG} \mathcal{N}(T^d_{Int}(t)[u]) + (1-\beta).\sum_{u\in U_d\setminus AG} \mathcal{N}(\widehat{T^d_{Int}}(t)[u])\Bigg] \tag{23}$$

The terms that appear in (23), i.e., T^d_{Int}, $\widehat{T^d_{Int}}$, T^e_{Int}, T^c_{Int}, are calculated based on number of model parameters, communication/channel characteristics, as well as computational capabilities at current time interval *t*. Loss function calculations, i.e., F_d, F_e, and F_c are also calculated based on data collected at time interval *t*. Indeed, ϕ can be calculated by the current values of the parameters, and it does not depend on the values of the parameters in previous time slots. This indicates that ϕ has memory-less property. Based on four evidences, the problem has Markov property and a MDP formulation/reinforcement learning can be designed for the problem: (i) The objective function, i.e., ϕ has memoryless property. (ii) The involved parameters in the current state of the system include the transmission rates, the computational capabilities of devices/base stations, the association of devices to the learning levels, the role of aggregators, and the devices participation in the training. Considering the parameters determining the current state, every action that is performed by the agent, will end to a new state transition, that only depends on the current state. (iii) The objective function is in the form of accumulated rewards. (iv) According to the evidence that reinforcement learning has been extensively used for providing network solutions with dynamic channel/communication status [38], [39], the motivation behind the use of it to solve the problem, is enhanced.

A. *Main Components of RL Formulation*

To solve the optimization in (22), controlling over these variables are required: collaborations of learning levels, i.e., device/edge/cloud levels; participation of devices in terms of the involved batch size in the training; and aggregator selection (for devices involved in the device level learning). Indeed, the controlling algorithm should give the appropriate values to the variables $D(\tau)$, $E(\tau)$, $C(\tau)$, $P(\tau)$, $G(\tau)$. To realize the automatic controlling and flexibility in FL, the problem is formulated by a RL with MDP elements as below:

1) State: The system state at time interval τ is determined with variables indicating the network status, the devices participation status, the association of devices to the learning levels, and the aggregator status. Note that the values of variable states change within the intervals (FL epochs), and RL agent will experience various states within learning. In the rest, we explain the state variables.

Variables Determining Network Status: These variables include: (i) Communication transmission rates among devices and the BSs, i.e., $\mathbb{R}_{d,e}(\tau)$; (ii) Communication transmission rates among devices, i.e., $\mathbb{R}^{d2d}(\tau)$; (iii) Available computing capabilities of devices, i.e., $\mathbb{F}^{cmp}_d(\tau)$; (iv) Available computing capabilities of MEC servers, i.e., $\mathbb{F}^{cmp}_e(\tau)$.

In the state definition, the transmission rates are realization of random variables $R_{i,j}$ at time interval τ, while the computing capabilities are realization of random variables $f^{cmp}_i(\tau)$. Note that according to (6) dynamic variation of factors including available bandwidth, distance of devices from BSs, wireless channel gain, and power transmission can vary the transmission rates. Similarly, dynamicity in factors including available resources, power capabilities, and work load on devices can influence the computation capability of a device. In this regard, we define:

$$\mathbb{R}_{d,e}(\tau) = \begin{pmatrix} \mathcal{R}_{1,1}(\tau) & \mathcal{R}_{1,2}(\tau) & \cdots & \mathcal{R}_{1,M}(\tau) \\ . & . & . & \\ . & . & . & \\ . & . & . & \\ \mathcal{R}_{N,1}(\tau) & \mathcal{R}_{N,2}(\tau) & \cdots & \mathcal{R}_{N,M}(\tau) \end{pmatrix}, \tag{24}$$

$$\mathbb{R}_{d2d}(\tau) = \begin{pmatrix} \mathcal{R}_{1,2}(\tau) & \cdots & \cdots & \cdots & \mathcal{R}_{1,N}(\tau) \\ \cdot & & \cdot & \cdot & \\ \cdot & & \cdot & \cdot & \\ \cdot & & \cdot & \cdot & \\ \mathcal{R}_{N,1}(\tau) & \cdots & \cdots & \cdots & \mathcal{R}_{N,N}(\tau) \end{pmatrix}, \quad (25)$$

$$\mathbb{F}_d^{cmp}(\tau) = \left(\mathcal{F}_1^{cmp}(\tau) \quad \mathcal{F}_2^{cmp}(\tau) \quad \cdots \quad \mathcal{F}_N^{cmp}(\tau)\right), \quad (26)$$

$$\mathbb{F}_e^{cmp}(\tau) = \left(\mathcal{F}_1^{cmp}(\tau) \quad \mathcal{F}_2^{cmp}(\tau) \quad \cdots \quad \mathcal{F}_M^{cmp}(\tau)\right), \quad (27)$$

where $\mathcal{R}_{i,j}(\tau)$, $\mathcal{F}_i^{cmp}(\tau)$ are the values of the random variables $R_{i,j}$ and f_i^{cmp} at time interval τ. Note that considering the possibility of asynchronous in dynamicity of channel characteristics for the upload and download cases, separate values in the state definition can be considered. In this regard, $\mathcal{R}_{i,j}^{UP}(\tau)$, $\mathcal{R}_{i,j}^{DN}(\tau)$, $\mathcal{R}_{d2d}^{UP}(\tau)$, $\mathcal{R}_{d2d}^{DN}(\tau)$, at which *UP* denote rates for the upload case, and *DN* denote rates for the download case, will be involved in the state specification for upload and download cases.

Variables Determining Participation Status: The state at interval τ also includes the participation status of devices $\mathbb{P}(\tau)$, as realization of random variables $P(\tau)$:

$$\mathbb{P}(\tau) = \left(\mathcal{P}(\tau)[1]\ \mathcal{P}(\tau)[2] \cdots \mathcal{P}(\tau)[N]\right), \quad (28)$$

where $\mathcal{P}(\tau)[i]$ is the value of random variable $P(\tau)[i]$ at time interval τ.

Association of devices to the learning levels: The state at interval τ also indicates the association of devices to the learning levels, i.e., $\mathbb{D}(\tau)$, $\mathbb{E}(\tau)$, $\mathbb{C}(\tau)$ as realization of random variables $D(\tau)$, $E(\tau)$, $C(\tau)$ respectively:

$$\mathbb{D}(\tau) = \left(\mathcal{D}(\tau)[1]\ \mathcal{D}(\tau)[2] \cdots \mathcal{D}(\tau)[N]\right), \quad (29)$$

$$\mathbb{E}(\tau) = \left(\mathcal{E}(\tau)[1]\ \mathcal{E}(\tau)[2] \cdots \mathcal{E}(\tau)[N]\right), \quad (30)$$

$$\mathbb{C}(\tau) = \left(\mathcal{C}(\tau)[1]\ \mathcal{C}(\tau)[2] \cdots \mathcal{C}(\tau)[N]\right), \quad (31)$$

where $\mathcal{D}(\tau)[i]$, $\mathcal{E}(\tau)[i]$, $\mathcal{C}(\tau)[i]$ are the value of random variables $D(\tau)[i]$, $E(\tau)[i]$, and $C(\tau)[i]$ at time interval τ respectively.

Variables Determining The Aggregators: The devices that take part in the device level learning and are aggregator at interval τ, are indicated through the state variables. These variables are represented with $\mathbb{G}(\tau)$, which is the realization of random variables $G(\tau)$:

$$\mathbb{G}(\tau) = \left(\mathcal{G}(\tau)[1]\ \mathcal{G}(\tau)[2] \cdots \mathcal{G}(\tau)[N]\right), \quad (32)$$

where $\mathcal{G}_(\tau)[i]$ is the value of random variable $G(\tau)[i]$ at time interval τ.

In this regard, the state of system at time interval τ is represented as:

$$s(\tau) = \begin{pmatrix} \mathbb{R}_{d,e}(\tau) \\ \mathbb{R}_{d2d}(\tau) \\ \mathbb{F}_d^{cmp}(\tau) \\ \mathbb{F}_e^{cmp}(\tau) \\ \mathbb{P}(\tau) \\ \mathbb{D}(\tau) \\ \mathbb{E}(\tau) \\ \mathbb{C}(\tau) \\ \mathbb{G}(\tau) \end{pmatrix}. \quad (33)$$

2) Actions: The action set at interval τ is represented as below:

$$\mathbb{A}(\tau) = \left(a_{\mathbb{P}}(\tau), a_{\mathbb{D}}(\tau), a_{\mathbb{E}}(\tau), a_{\mathbb{C}}(\tau), a_{\mathbb{G}}(\tau)\right). \quad (34)$$

In the rest, we explain the actions:

(i) Action $a_{\mathbb{P}}(\tau)$ is represented with a vector with size of number of devices. In this vector, the entry *i* with value in the range of [0, 1] indicates the batch size participation in training for device *i* at interval τ.

(ii) Action $a_{\mathbb{D}}(\tau)$ is represented with a vector with size of number of devices. The entry i with value 1, will activate device *i* for participating in the device level learning, at interval τ. The value of 0 indicates no participation in device level learning at that interval.

(iii) Similar to $a_{\mathbb{D}}(\tau)$, the actions $a_{\mathbb{E}}(\tau)$ and $a_{\mathbb{C}}(\tau)$ are defined to respectively define the collaborations of devices at the edge and cloud levels of learning for interval τ.

(iv) Action $a_{\mathbb{G}}(\tau)$ is represented with a vector with size of number of devices. The entry *i* of this vector, decides about the role of aggregator for device *i* at interval τ. The value 1 will activate the device to perform the aggregation at interval τ.

3) Reward: The learning agent will receive a reward after performing an action, as a feedback quantifying the short-term affect of an action. The reward is defined as below:

$$R(s(\tau), a(\tau)) = -T_d(\tau) - T_e(\tau) - T_c(\tau), \quad (35)$$

where $T_d(\tau)$, $T_e(\tau)$, and $T_c(\tau)$ are accumulated loss function and response time of recognition (for all devices) gained respectively in the device, edge, and cloud level participants at interval τ. These values are calculated according to the transmission rates and available computational capabilities defined in the current state, i.e., $s(\tau)$, using equations (20), (21). This design of reward function, guides the agent to optimize the targeted objective function in (22), while the agent proceeds to find policies that maximizes the accumulated reward within intervals.

B. DRL for Optimization and Controlling

The optimization and controlling based on Reinforcement Learning (RL), learns automatic detection of dynamicity of 6G network and applies the required flexibility and collaborations in FL accordingly. The most renowned approach in RL is Q-Learning, which estimates the long-term quality of actions at states, i.e., Q-values through iterations. However, the high dimension of the states and actions involved in the problem, as well as high dynamicity in state transitions due to the dynamic nature of 6G networks, makes the application of classic Q-learning impractical. Indeed, under the circumstances that experiencing all states and actions by the agent is not possible, classic Q-learning would not be efficient [40]. Deep Reinforcement Learning (DRL), also called as deep Q-learning, is a solution which provides generalization of previously experienced states to non-observed states/actions through a neural network based approximation of Q-values. Thereby, eliminating the necessity of visiting all states/actions to calculate Q-values [41]. Q-learning estimates the Q-value

TABLE I
HIGH LEVEL PSEUDOCODE FOR CFL

```
1   Train DRL to optimize (22) through state-action mapping
2   At each epoch of FL
3       Determine the network state using (33)
4       Use the trained DRL to determine the learning levels, participation vector, and aggregators
5       For devices in Device-Level FL
6           Aggregate models using (5) and perform parameters updating
7       End-For
8       For devices in Edge-Level FL
9           Aggregate models using (2) and perform parameters updating
10      End-For
11      For devices in Cloud-Level FL
12          Aggregate models using (3) and perform parameters updating
13      End-For
14  Continue epochs of FL until termination or convergence condition be met
```

for state-action pairs as an accumulated discounted reward in long time, using on an iterative updating of the Q-values based on Bellman equation as below:

$$Q_{\tau+1}(s,a) = Q_\tau(s,a) + \alpha_l.[R + \gamma_l. \max_a Q_\tau(s',a) - Q_\tau(s,a)], \quad (36)$$

where R is the reward achieved from performing action a, s' is the next perceived state, α_l is the learning rate, γ_l is the discounting rate. The optimal policy is gained after experiencing of all state-action pairs in the training phase, which suggests action a^* at state s as below:

$$a^* = arg \max_a Q^*(s,a). \quad (37)$$

To cope with impossibility of visiting every state/action pair in complex problems with high dynamicity, DQN utilizes neural network to estimate Q values. The set of experiences of agent within time intervals, i.e., $< s_\tau, a_\tau, s_{\tau+1}, R_\tau >$ are stored in replay buffer, from which random batches are used to update the weights of a neural network, and accordingly estimating the Q-values. As utilizing one neural network can end to an over-optimistic and unstable approximation of the Q values due to using the values obtained from the same network both to select and evaluate an action, a two neural network based solution has been suggested in [42], that extensively has been used as a DRL method in various network problems. Selection neural network which is with parameter θ_τ and estimations of $Q^S(s,a,\theta_\tau)$, is used for selection of actions. Evaluation neural network which is with parameter θ'_τ and estimations of $Q^E(s,a,\theta'_\tau)$, is used for evaluation of actions. During training, for a batch of experiences in the replay buffer, target Q-value is calculated as below:

$$Y^Q_\tau = R_\tau + \gamma.Q^E(s_{\tau+1}, arg \max_a Q^S(s_{\tau+1}, a, \theta_\tau), \theta'_\tau). \quad (38)$$

Accordingly, after forwarding pass, the error function at interval τ is calculated as below:

$$Er(\theta_\tau) = \frac{1}{2}\Big[Y^Q_\tau - Q^S(s_\tau, a, \theta_\tau)\Big]^2. \quad (39)$$

Next, in back propagation phase, the gradient descent update rule will be applied to the parameters of θ_τ, and periodically

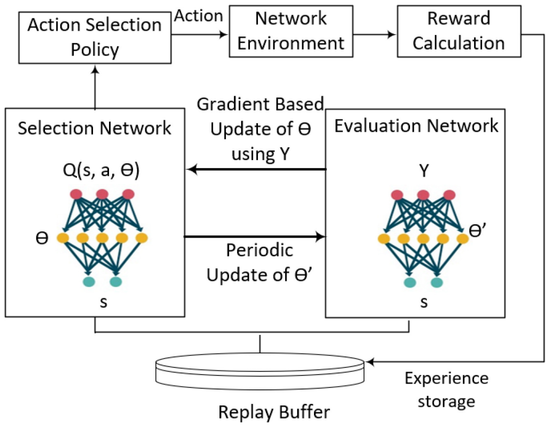

Fig. 3. The training of DRL.

the value of θ_τ will be copied to θ'_τ. After the training phase, the error will be acceptably small and the weights will have the final values. The schematic for training of DRL has shown in Fig. 3. Table I illustrates the high level pseudocode for CFL.

VI. DDoS Attack Detection Scenario

As security is a main concern in 6G, we apply CFL for a case of a Distributed Denial of Service (DDoS) attack detection. In a DDoS attack, IoT devices can be hacked and become networked zombies, leading to notorious botnets. Such zombies are utilized by cybercriminals to attack other devices, thereby paralyzing the functionality of legitimate devices, for the purposes like stealth or expense [11], [43]. Consequently, the DDoS attack has become a public hazard on the Internet [11]. DDoS attack goes beyond server targeting and can target wide-range of smart devices, e.g., vehicles in vehicular networks [44], [45], mobile devices [46], industrial IoT devices [11], smart cities [47], smart home devices [48].

Centralized learning at which the local train sets of devices are transmitted to a server for global training is not efficient in DDoS attack detection due to the latency, bandwidth consumption, and data privacy issues. Performing DDoS attack detection individually, is also not efficient: First, devices are usually application-specific and can not provide sufficient patterns of training samples to infer a precised learning model.

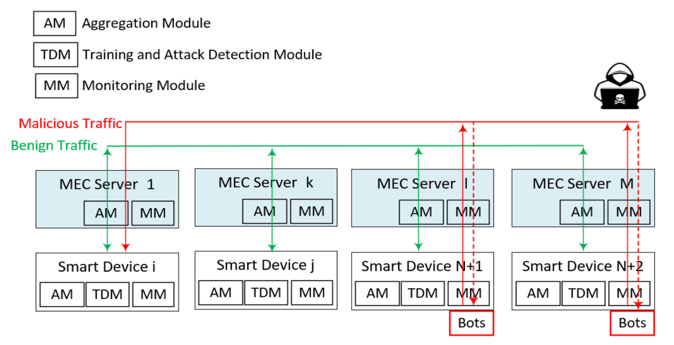

Fig. 4. As a snapshot of network, a DDoS attack has been launched by Devices $N + 1$ and $N + 2$ to target Device i as victim. Device j might be also a victim in future time steps. Individual detection on devices i and j can not prevent potential future attack on device j.

Second, IoT threat landscapes are dynamic due to devices and applications growth, and a static pre-trained model can not cope with threat changes [11]. Finally, threats will remain in the network until the bot be recognized by all individual devices. The latter case is explained by Fig. 4.

Devices $1 \cdots N$ are the legitimate devices. At time *t*, devices $N + 1$ and $N + 2$ launch a DDoS attack with a specific pattern of traffic, against the target device *i*, as victim (e.g., to exhaust it's CPU/memory, or paralyze its functionality [43]). The Monitoring Module (MM) in device *i*, registers the pattern of the traffic (e.g., source, volume, protocol, etc.). At time $t + \mathbb{X}$, $\mathbb{X} \geq 1$, the Training and Detection Module (TDM) at device *i* recognizes the traffic pattern as a DDoS attack. When devices follow an individual detection policy, device *j* can not learn the DDoS attack traffic pattern unless the bots target it in the future which can paralyze its functionality. When devices i and j collaborate and share their models, device j can also recognize the malicious traffic pattern at time $t + \mathbb{X} + 1$. Thus, if devices $N + 1$ and $N + 2$ launch a DDoS attack against device *j*, it will be detected immediately, without causing malfunctioning. Another possible scenario is when device $N + 1$ targets device *i*, and device $N + 2$ targets device *j*. The attack patterns oriented from both of devices $N + 1$ and $N + 2$ can not be detected individually, unless both malicious devices target each of the devices i and j. Collaboration of devices in learning can enhance timely detection of the attack and avoid potential of negative consequences.

Since malicious traffic comes from distributed bots, FL provides the capability of updating the model dynamically and detecting the attacks timely through collaboration of devices. Indeed, FL has been used for DDoS attack detection in recent studies, e.g., [11], [49], [50], [51]. CFL goes a step forward and provides collaboration of various FL levels, i.e., edge level FL and device level FL to overcome the below challenges:

1) *Attack Detection Accuracy Degradation Due to Disconnectivity from Edge Cloud:* Devices will have different data samples to detect the DDoS attack. For example, in Fig. 4, at time $t + \mathbb{X}$, the data samples in device i is different from device j. To have an accurate and timely DDoS attack detection, the mobile devices should be able to take part in FL and share their models. Edge level FL can provide sharing capabilities by Aggregation Module (AM) in MEC servers (see Fig. 4). However, when the device moves, edge cloud might not be available in some areas, thereby disconnecting from FL service and accuracy degrading. Under unavailability of edge infrastructure, CFL provides the opportunity of continuation of FL service through D2D communication (by Aggregation Module (AM) in devices in Fig. 4). Fig. 1 illustrates a schematic form of switching from edge level FL to device level FL. Collaboration of edge- and device-level FLs in CFL can involve more devices in FL, thereby increasing data sharing opportunities and enhancing the accuracy.
2) *Appropriate Trade-off between Attack Detection Accuracy and Response Time:* As we discussed in Section III, and mathematically illustrated in Eq. (20)-(22), when both options of edge level and device level FL are available for a device to take part in FL, optimal decision to perform appropriate trade-off is required. Generally, edge level FL offers higher-sharing opportunity and thus, higher DDoS attack detection accuracy. Device level FL offers lower-sharing opportunity due to shorter distance limitation for D2D communication, however faster response time. CFL considers application-specific trade-off (i.e., β) to optimize the accumulated of DDoS attack detection accuracy and response time as defined in (22).

As DDoS attack detection based on GRU, a kind of recurrent-neural network, has recently been shown to be efficient in attack detection [10], [11] we also use it for the attack detection. Let PK_{fl} be the chain of packets in traffic flow *fl*:

$$PK_{fl} = pk_1, pk_2, \ldots pk_n. \tag{40}$$

Let *FT* be the number of features, based on which the attack is detected. The feature matrix for the packets in flow *fl* is arranged as rows of patterns as below:

$$FM_{fl} = \begin{pmatrix} x_1^1 & x_2^1 & \cdots & x_{FT}^1 \\ x_1^2 & x_2^2 & \cdots & x_{FT}^2 \\ \cdots & \cdots & \cdots & \cdots \\ x_1^n & x_2^n & \cdots & x_{FT}^n \end{pmatrix} = \begin{pmatrix} p_1 \\ p_2 \\ . \\ . \\ . \\ p_n \end{pmatrix} \tag{41}$$

The occurrence probability for each pattern p_i is calculated as a function of previous observations using a GRU. The operation of GRU is based on the utilization of neural-operation based gates which enables the forgetting/retaining of past/current information in prediction. Let w be the size of previous observations that are considered in occurrence prediction of the patterns. Let P_i be the occurrence probability of pattern p_i for packet PK_i:

$$P_i = P(p_i| < p_{i-1}, p_{i-2}, \ldots p_{i-w} >). \tag{42}$$

The benign traffic usually follows a regular distribution of patterns. In contrast, the malicious traffic usually deviates from the regular distribution. Therefore, it is expected that it has smaller probability of occurring. Packet PK_i will be

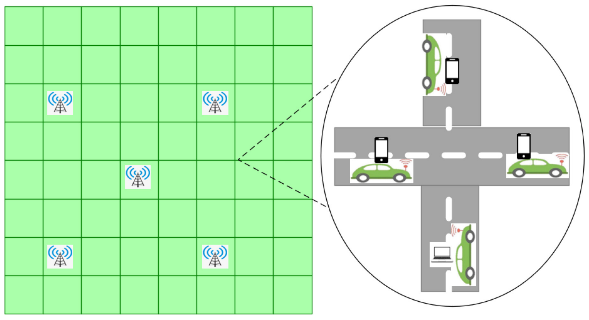

Fig. 5. Simulation environment. The collaborators in DDoS attack detection, are smart devices in 6G network that are moving through some vehicles.

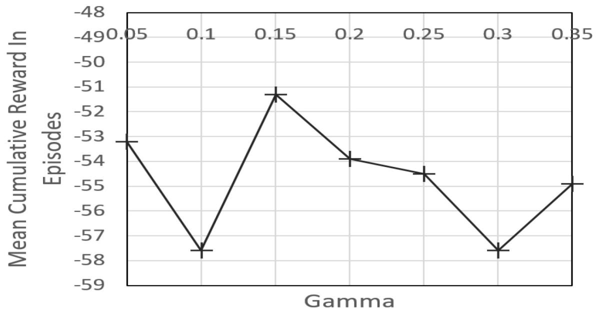

Fig. 6. Effect of DRL discount rate in mean cumulative reward of CFL.

malicious if it's occurrence probability is less than a predefined threshold:

$$P_i < \delta. \tag{43}$$

The flow will be considered as malicious if the ratio of the malicious packets in that flow be larger than a predefined threshold T:

$$\frac{\sum_i 1.(P_i < \delta)}{n} > T, \tag{44}$$

where $1.(B)$ is 1 when B is true. The function of GRU for training is presented as equations below [10]:

$$r_t = \sigma(W_r.[h_{t-1}, x_t] + B_r), \tag{45}$$

$$u_t = \sigma(W_u.[h_{t-1}, x_t] + B_u), \tag{46}$$

$$\widehat{h_t} = \tanh(W_h.[r_t * h_{t-1}, x_t] + B_h), \tag{47}$$

$$h_t = (1 - u_t) * h_{t-1} + u_t * \widehat{h_t}, \tag{48}$$

where x_t is the feature space; h_t is the prediction; r_t, u_t, and $\widehat{h_t}$ control the flow of information through GRU; W_r, W_u, W_h are weight matrices of the neural networks; B_r, B_u, B_h are bias vectors; σ and $\tanh$ are activation functions.

VII. Experimental Results

This section explains the experimental setup we used for evaluation, and the results.

A. Experimental Setup

For the purpose of evaluation, a scenario that 15 devices which are moving by vehicles and will collaborate in DDoS attack detection, is considered. We assume a 8 × 8 bidirectional grid environment with 150 *m* width for each grid cell, where grid lines are bidirectional roads (See Fig. 5). SUMO simulator has been used for mobility trace generation of vehicles [52]. Manhattan mobility model [53] which widely has been used to simulate mobility in urban area with probability of 0.5 for moving straight and 0.25 for moving right/lef at conjunctions, is used for mobility of vehicles. The mean speed of vehicles are 45 km/h. TensorFlow package particularly, Keras deep learning package has been used to implement the DRL and GRUs. Due to the limitation in the number of output neurons with existing tools, without loss of generality, the collaboration for DDoS attack detection is only performed in the level of edge computing resources and the devices. Note that the trend of research in 6G (i.e., most of the studies in the related work), also have utilized an edge computing based simulation.

The 6G network parameter have been set similar to [5], [17], [54]. 5 Base Stations (BSs) equipped with MEC-servers are located in the center and the middle of four quarters of the region in locations [525, 525], [225, 225], [825, 225], [225, 825], [825, 825] (See Fig. 5). The CPU frequencies of MEC servers at the base stations are 3.2, 2.6, 3.6, 1.8, 2.4 GHz [5]. The coverage radius, and the transmission bandwidth of BSs are respectively, 240 m [55] and 30 MHz [5]. The D2D-bandwidths have been chosen randomly in the range of 0.3 THz up to 3 THz [54] and the distance to utilize the D2D communication is at most 60 m [54]. The CPU frequency of devices are randomly chosen in the range of 1.8 up to 2.4 GHz. The range has been selected such that the processing capabilities of devices be in average less than MEC processing capabilities, whilst be competitive with the lowest processing available in MEC. The transmission power of BSs and devices are respectively 34 db [5] and 23 db [17]. The path loss exponent is 5 and the back ground noise power is −174 db.m [5].

We have used CICDDoS 2019 data set [56] that has generated realistic traffic to abstract the behavior of human communications for legitimate and DDoS attack traffic through different protocols such as HTTP, FTP, and SSH. We have randomly distributed 19800 instances composed of UDPLag and SYN DDoS attacks, among devices for training, as well as 9000 instances for the purpose of test. The dataset provides 87 extracted IP flow features, e.g., source/destination IP addresses/ports, protocols, flow packet statistics, flag-related information etc., which we utilize them for the attack detection.

The size of replay-buffer and mini-batch are respectively 600 and 60. At early iterations of DRL, more explorations have been used, while the exploitation gradually increases as the neural networks are trained (up to the greedy selection of 98% at the last episode of learning). Assessing the range of 0.05 to 0.35, the RL-discount rate of 0.15 that has gained the highest mean cumulative reward in episodes, has been selected (See Fig. 6). We found the learning rate of 0.2 appropriate for stochastic gradient descent optimization of neural networks in DRL. We rely on *LearningRateSchedule* package of Keras optimizer to reduce the learning rate within time steps for the purpose of convergence. For GRUs, we found learning rate of

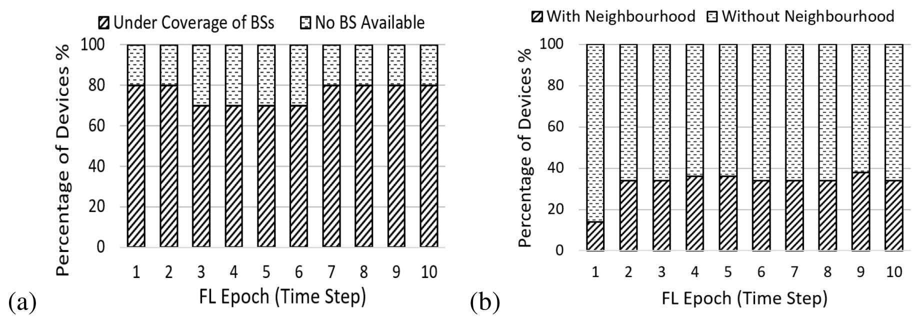

Fig. 7. Mobility-Trace 1 generated by SUMO (a) Coverage status of devices with BSs (b) Neighbourhood Status.

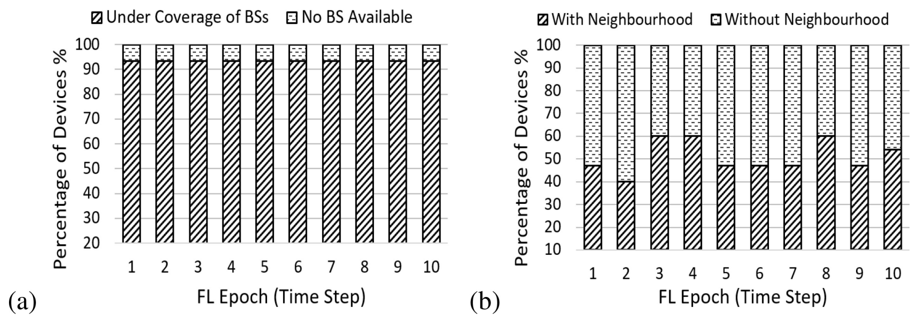

Fig. 8. Mobility-Trace 2 generated by SUMO (a) Coverage status of devices with BSs (b) Neighbourhood Status.

0.07 appropriate for attack detection, and we used the hidden layer with size of 32 neurons. The out-layer of GRU is a neuron to predict the occurrence probability of a packet.

SUMO [52] lets applying random seed to generate different traces with the same mobility pattern and speed features. In the training phase, various random seeds have been utilized within episodes, so that the learning occurs over general exploration of states. For testing, we give the results for two mobility-traces: *Mobility-Trace 1* and *Mobility-Trace 2*. Fig. 7 and Fig. 8 illustrate the BS-coverage and the neighbourhood status under respectively, *Mobility-Trace 1* and *Mobility-Trace 2*. The BS-coverage and the neighbourhood of devices are changing within epochs of FL. For example, in *Mobility-Trace 1* at epoch 5, MEC-servers are not available for 30% of devices and 36% of devices have neighbourhood. Note that two devices are regarded as neighbours if their distance be less than the maximum distance required for D2D communication, i.e., 60 m. In *Mobility-Trace 2*, MEC-server capabilities are available for 93% of devices, while 40% to 60% of devices will have neighbourhood and can exploit D2D-communication capabilities for data sharing.

The baselines for comparison are as below:

(i) No-FL: In this method, no FL is performed and each device exploits the model trained with its local data to detect the DDoS attack.

(ii) Device-Level FL: In this method, at every iteration of FL, the devices that has been determined as aggregator, aggregate the trained models of the neighbourhood devices and broadcast the results back to them for the attack detection. Depending on the dynamic nature of the network, defined by (33), the selection of aggregators will influence the response time of recognition and loss value experienced by devices. To deal with dynamicity of the network, the same DRL approach explained in Section V has been applied for the implementation. However, we have adopted the agent policy exploration, so that at each interval, all entries in action-vector $a_{\mathbb{D}}(\tau)$ have the values of 1 (The values of entries of $a_{\mathbb{E}}(\tau)$ are 0). This implementation will provide a fair comparison, at which Device-Level FL can optimize the accumulated response time of recognition and loss value experienced by devices as calculated by (22).

(iii) Edge-level FL: This method generally has been advocated in [5], [7], [11], [17]. Particularly, it can be considered as an adopted version of the method in [11], at which no mechanisms are applied for attack mitigation and training scheduling. In contrast, we assume all devices will take part in FL. In this method, at every iteration of FL, the base stations aggregate the local trained models of the devices under their coverage, and broadcast the aggregated results to them for the attack detection. To implement this method, the learning agent explained in Section V, will select edge level FL at every interval, irrespective to the observed state, i.e., $\forall s(\tau)$: all entries of action-vector $a_{\mathbb{E}}(\tau)$ have the values of 1 (The values of entries of $a_{\mathbb{D}}(\tau)$ are 0). Also, all entries of action-vector $a_{\mathbb{G}}(\tau)$ have the values of 0.

(iv) DFL [29]: In this method, the FL is performed in a distributed manner. At each round of FL, every device receives the weights (or a subset of weights) of learning models from its neighbourhood devices and aggregates the weights (with the coefficients in proportion with the data size). The aggregated weights will be broadcasted to the neighbour devices, where Adam optimization is performed to update the weights with local data for the next round. To have a fair comparison, the phase of parameter selection for transmission in [29] has been

TABLE II
THE BASELINES FOR COMPARISON

Method	**Reference**	**Attack Dtection Method**	**Learning Level**
No-FL	-	GRU	-
Edge-Level FL	[11]	GRU	Edge
Device-Level FL	-	GRU	Device
DFL	[29]	GRU	Device
TFL-CNN	[4]	CNN	Edge

omitted, and we assume that all the weights of the model are transmitted.

(v) TFL-CNN [4]: In this FL method, the local training at devices is performed using CNN. The devices will send the parameters to the associated BSs and the aggregation of parameters are performed at BSs. The aggregated results will be broadcasted to the devices. The mentioned process will be repeated through the iterations of learning.[3] The study in [4] focuses on object detection problem. To adopt CNN for attack detection, we used the same CNN-architecture as in [57]. The data with dimensions of packet sequences and IP-features of the flow, is given as input to a CNN, to decide about the occurrence probability of the flow. For example, the entry at row i, and column j of the input matrix indicates the value of the j^{th} IP-feature for packet i in the flow. The CNN comprises a convolution layer (Conv1D), a max-pooling layer, a flatten layer, and a dense (fully-connected) layer. The output neuron with sigmoid activation function predicts the occurrence probability of the flow. The input matrix is operated by convolutional layer with filters with size of $kernel\_size \times feature\_size$ (87). Each filter convolves with a step of 1 to extract the attack-related features. As hyper-parameters, like [57], we also found the kernel_size of 3 and the pool_size of 2, appropriate for the attack detection. We found that the most effective parameters in accuracy and response time is the number of filters, accordingly we did the evaluation with 32 and 64 filters (this value also has been shown to be effective in DDoS attack detection in [57]). Table II summarizes the baselines. We use the comparison of CFL with No-FL, Device-Level FL, and Edge-Level FL [11] to illustrate the effect of collaboration of learning levels in CFL. Finally, the performance gain of CFL is investigated with comparison with two existing methods in the literature, i.e., DFL [29] and TFL-CNN [4].

B. Results

1) Training: Fig. 9 illustrates the gained reward within training phase for 8000 episodes; every episode is composed of 12 epochs of learning. Thus, total of 96000 learning epochs, each with collaboration of 15 devices, are involved in the training. Two cases of SDQN [41] and DDQN [42] have been considered. SDQN, is conventional DRL at which the same Neural Network (NN) (i.e., single NN) is utilized for both action selection and evaluation [41], while DDQN [42],

[3]The contextual-related weights in aggregation, as suggested in [4] have been assumed to be equal for all devices since data in devices are equally important in attack detection.

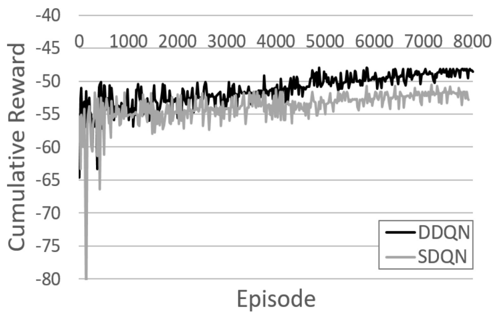

Fig. 9. Cumulative reward in CFL for 8000 episodes of federated learning.

TABLE III
THE COLLABORATION OF DEVICES IN FL IN MOBILITY-TRACE 1 ($\beta = 0.5$). THE RESULTS ARE MEAN VALUES OVER 10 EPOCHS OF LEARNING

Method	**Total Involved Devices in FL**	**Device Level Collaboration Ratio**	**Edge Level Collaboration Ratio**
CFL	85%	25%	75%
Edge-Level FL	76%	0%	100%
Device-Level FL	33%	100%	0%
No FL	0%	0%	0%

exploits two separate NNs as explained in Section V-B. As it can be seen, within the episodes of training, in both methods the agent improves the policy of collaboration among various learning levels of FL, by increasing the reward values. At early iterations, there are larger fluctuations in the reward values due to the higher exploration of actions. However, gradually, the neural network weights and accordingly, the Q-values, will be rather stable and due to the more exploitation of the learnt policy, the fluctuation reduces. DDQN has outperformed and increased the rewards from -65 up to more than -50, while SDQN experiences more fluctuations and has gained less rewards and becomes stable around -52. The performance degradation in SDQN is justified with the over-optimistic and unstable approximation of Q-values due to utilization of the same NN for action selection and evaluation [42].

2) Effect of Collaboration of Learning Levels in CFL: Table III indicates the involvement of devices and collaboration of various learning levels in federated learning over *Mobility-Trace 1*. The results are mean values over 10 epochs of learning and $\beta = 0.5$. Obviously, No-FL can not involve any device in FL. Device-Level FL could involve the lowest fraction of devices, i.e., 33% in FL. The reason is that in average 33% of devices have neighbourhood in *Mobility-Trace 1*[4] (See Fig. 7.b). As BSs can cover 70% till 80% of devices (See Fig. 7.a), Edge-Level FL involves higher portion of devices in FL, i.e., 76%. CFL could involve the highest ratio of devices, i.e., 85% in the FL. Within the epochs of learning, a device takes part in FL through mean device level

[4]As in the first epoch of learning in Mobility-Trace 1, the neighbourhood ratio is low in comparison with other epochs, for device level collaboration statistic, we reported the median value.

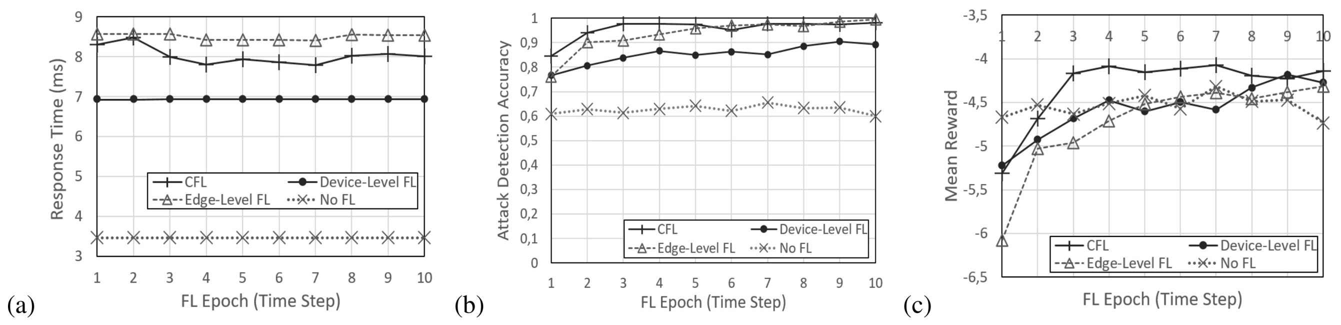

Fig. 10. The behaviour of methods in federated learning epochs for Mobility-Trace 1. (a) response time of recognition. (b) DDoS attack detection accuracy. (c) mean reward ($\beta = 0.5$).

collaboration ratio of 25% and mean edge level collaboration ratio of 75%. Indeed, 20% to 30% of devices that do not have access to any BS can take part in FL through their neighbourhoods, if available. Thus, we see a 9% increment in total involved devices in comparison with edge-level FL. Similarly, the average of 67% of devices that do not have neighbourhood (See Fig. 7.b) can take part in edge-level FL, if edge infrastructure is available. Thus, we see a 52% increment in total involved devices in comparison with Device-Level FL. Finally, CFL is not able to involve all devices in FL, since 15% of devices in *Mobility-Trace 1* are not covered by any BS, and do not have any neighbour.

Fig. 10 compares CFL with No-FL, Device-Level FL, and Edge-Level FL for *Mobility-Trace 1*. Fig. 10.a shows the mean response time of recognition experienced by devices. No-FL has achieved the lowest response time of recognition stable around 3.5 ms, since it performs only local training. The response time of recognition in Device-Level FL is almost 7 ms. As D2D communication setting is in order THZ, the transmission of parameters takes negligible time, and the dominant factors in response time will be the time for training and aggregation. In Device-Level FL, the response time is rather stable due to the same size of train sets in epochs of learning. The response time of recognition of Edge-Level FL is higher than Device-Level FL, due to the overhead of parameter transmission to the base stations. CFL has reduced response time of recognition up to 7% (0.6 *ms*) in comparison with Edge-Level FL. The reason is that CFL exploits collaborations of both the device and the edge learning levels to minimize the accumulated response time and accuracy. Note that improvement in the scale of 0.6 *ms* in the simulation is acceptable for 6G scenario at which the time is discussed in the scale of *ms*, particularly for ultra-low latency applications.

Fig. 10.b illustrates the mean attack detection accuracy in devices. The lowest accuracy has been achieved by No-FL, since it does not utilize any data sharing. When number of epochs increases, the accuracy increases as well in Device/Edge-Level FL and CFL, due to experiencing more iterations for the model parameters sharing. Edge-Level FL has gained higher accuracy in comparison with Device-Level FL. The reason is that Edge-Level FL provides the capability of model parameter sharing for 70%-80% of devices in comparison with average of 33% in Device-Level FL (See Fig. 7.a and Fig. 7.b). CFL could achieve almost the same accuracy as Edge-Level FL and even higher in the early epochs of learning. There are two reasons: First, CFL still exploits collaboration of Edge-Level learning, to enhance the accuracy. Second, CFL provides the facility for model parameter sharing through the neighbourhood devices, for 20%-30% of devices that do not have access to MEC servers (See Fig. 7.a). These reasons, can end to the involvement of more devices in FL, as we discussed in Table III. Thus, there will be a higher accuracy in comparison with Edge-Level FL, particularly in early epochs of learning, when the model parameters have not been shared enough among devices.

Fig. 10.c illustrates the achieved reward by devices. The highest reward has been gained by CFL. The reason is that CFL learns to optimize the accumulated criteria of the response time of recognition and the attack detection accuracy, through exploiting the Edge-Level FL to increase the accuracy and exploiting the Device-Level FL to decrease the response time of recognition. After epoch 8, the difference between Edge/Device-Level FL and CFL decreases. Because, at the late epochs of learning, the accuracy becomes high in all methods (See Fig. 10.b) and only the outperforming in the response time of recognition becomes the determining factor.

Fig. 11 illustrates the results for *Mobility-Trace 2*. In comparison with *Mobility-Trace 1*, higher attack detection accuracy has been gained by Edge-Level FL, since 93% of devises are under coverage of BSs in *Mobility-Trace 2*, which is higher than 70%-80% in *Mobility-Trace 1*. Similarly, higher accuracy has been achieved by Device-Level FL, since 40%-60% of devices have neighbourhood in *Mobility-Trace 2*, which is higher than 33% in *Mobility-Trace 1* (See Fig. 7.b and Fig. 8.b). In comparison with *Mobility-Trace 1*, from the aspect of accuracy, the outperformance of CFL in comparison with Edge-Level FL has reduced. The reason is that parameter sharing can be done for 93% of devices through edge-Level FL. With $\beta = 0.5$, CFL has gained higher reward in comparison with No-FL, Device-Level FL, and Edge-Level FL, due to high accuracy, as well as acceptable response time.

3) Effect of Trad-Off Coefficient in CFL Operation: Table IV indicates the β effect on the response time of recognition, the accuracy, and the device level collaboration in CFL, for β values of 0, 0.5, and 1. Note that the results are average over all epochs of learning. The case of $\beta = 1$ only considers the response time of recognition (attack detection) in the optimization, whilst the case of $\beta = 0$ only considers the

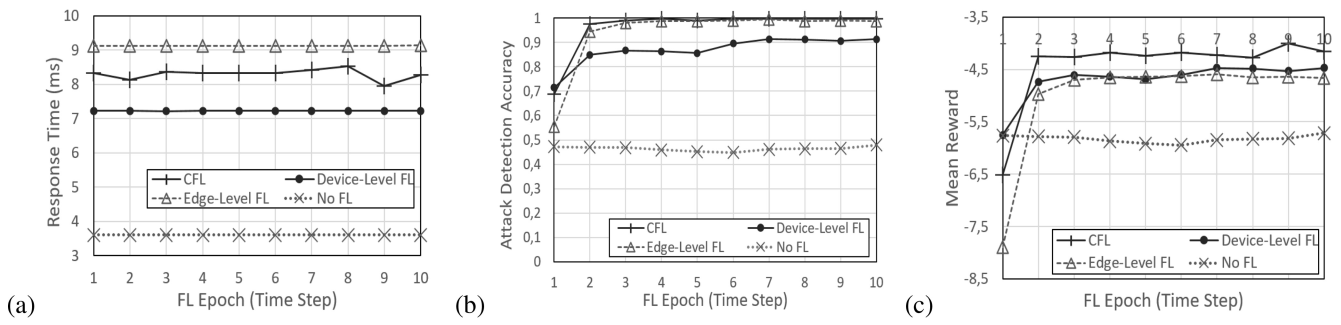

Fig. 11. The behaviour of methods in federated learning epochs for Mobility-Trace 2. (a) response time of recognition. (b) DDoS attack detection accuracy. (c) mean reward ($\beta = 0.5$).

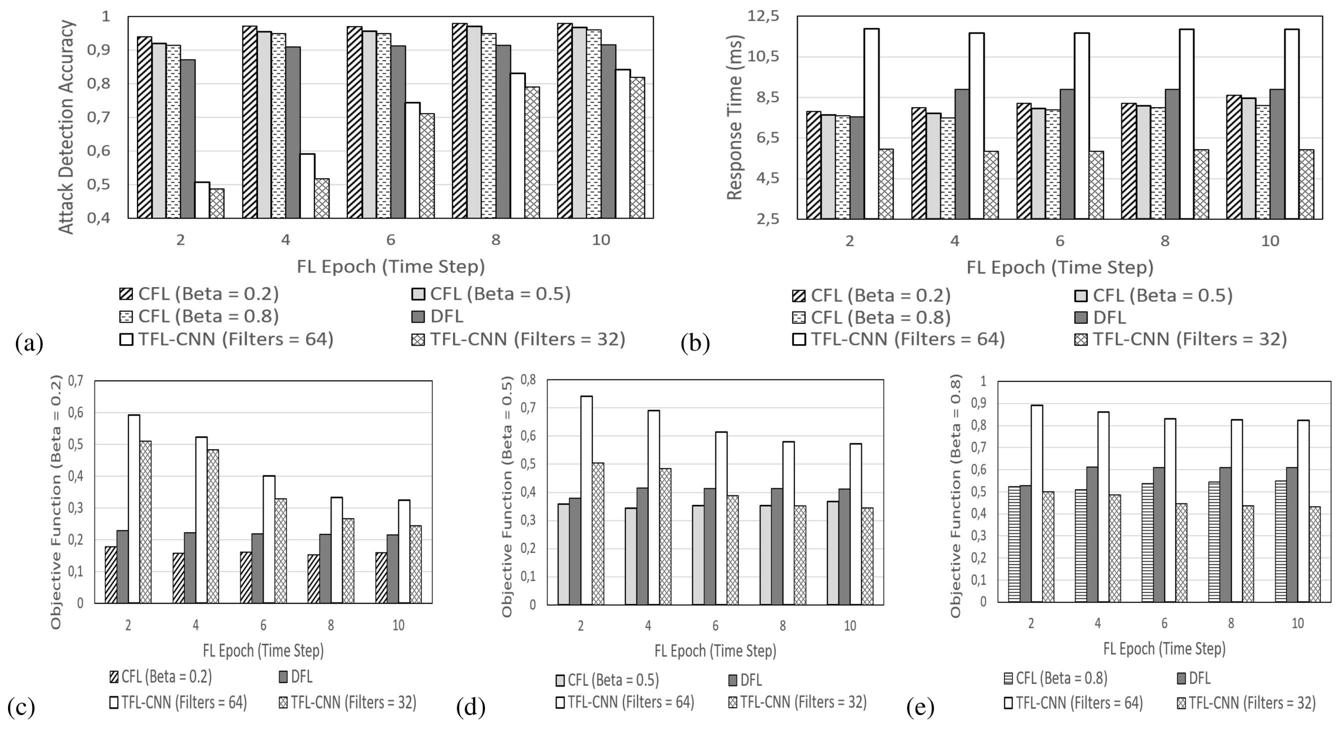

Fig. 12. The performance of CFL vs. other baselines evaluated over Mobility-Trace 1. (a) DDoS attack detection accuracy. (b) response time of recognition. (c) objective function ($\beta = 0.2$) (d) objective function ($\beta = 0.5$) (e) objective function ($\beta = 0.8$).

TABLE IV
THE EFFECT OF β ON RESPONSE TIME OF RECOGNITION, ATTACK DETECTION ACCURACY, AND THE RATIO OF DEVICES COLLABORATE IN DEVICE LEVEL LEARNING IN CFL WITH 7 EPOCHS OF LEARNING

β	Response Time of Recognition (ms)	**Accuracy**	**Device Level Collaboration (Mean)**	**Device Level Collaboration (Maximum)**
0	8	0.97	21%	28%
0.5	7.8	0.94	25%	33%
1	7.6	0.9	27%	33%

accuracy in the optimization. When β increases, the response time of recognition reduces from 8 ms to 7.6 ms, and the accuracy decreases from 97% to 90%. Indeed, to decrease the response time of recognition from 8 ms to 7.6 ms, CFL has increased the mean device level collaboration ratio from 21% to 27%. This will offer the updates on the model parameters for the attack detection, at closer distances to the devices, to speed up the recognition. The table also illustrates the maximum collaboration rates of device level learning in the epochs. The rates increase when the importance of the response time of recognition increases in the optimization, i.e., for higher values of β.

4) Performance of CFL With Various β-Trade-Off Coefficients: Fig. 12 illustrates the performance of CFL with β values of 0.2, 0.5, 0.8 versus DFL [29] and TFL-CNN [4] in *Mobility-Trace 1*. Fig. 12.a shows the attack detection accuracy. TFL-CNN has gained the least accuracy since, CNN is not able to encode the effect of temporal dependency in attack detection. Furthermore, the pattern of traffic encoded in two dimensions with large number of features in one dimension, i.e., 87, makes the (ab)normal pattern recognition more difficult in CNN. TFL-CNN with 64 filters has outperformed TFL-CNN with 32 filters. DFL which relies on GRU has gained higher accuracy. CFL has gained the highest accuracy due to the exploitation of both edge level and device level data sharing; thereby increasing the total number of devices involved in FL (See Table III). In CFL, when β increases, the accuracy decreases due to the more priority to response time optimization. In comparison, with DFL and TFL-CNN, respectively up to 7%, and 46% accuracy enhancement has been achieved by CFL, during epochs of learning.

Fig. 12.b shows the response time. TFL-CNN with 64 filters has gained the highest response time (between 11,6 to 11,9 ms) due to the large number of parameters involved in training and transmission, i.e., 17345 versus 8673 in TFL-CNN with 32 filters, and 11841 in CFL/DFL. The response time in DFL is higher than CFL. The reason is that in DFL, every device receives the parameters and performs the aggregation, which increases the computation and transmission time. In contrast, in CFL only the aggregator nodes perform the aggregation.

Fig. 12.c, Fig. 12.d, and Fig. 12.e show the objective function for β-values of respectively 0.2, 0.5, and 0.8. When the β increases the priority of response time in optimization, increases as well (See Eq. (20)-(22)). When β is 0.2, the objective function mostly reflects the accuracy, and therefore TFL-CNN with the lowest accuracy, has operated the worst. When β is 0.8, the objective function mostly reflects the response time, and therefore TFL-CNN with 64 filters with the highest response time, has operated the worst. When β is either 0.2 (Fig. 12.c) or 0.5 (Fig. 12.d), CFL has gained the minimum of the objective function. The reason is that through DRL, CFL optimizes the trade-off between response time and accuracy as the objective function. However, TFL-CNN and DFL are not able to do the trade-off. When β is 0.8, TFL-CNN with 32 filters has gained the best performance. The reason is that when response time is dominant in the optimization, CFL that operates based on GRU with 11841 parameters consumes more computation and transmission time than TFL-CNN with 8673 parameters. However, this is a model-related issue and as it can be seen in Fig. 12.c and Fig. 12.d, CFL targets the optimization of accumulated accuracy and response time and outperforms the other methods.

VIII. Conclusion

Performing partial aggregation in edge servers has been advocated for 6G networks to reduce the transmission cost overhead of conventional federated learning. To deal with unavailability of access to the edge servers, or higher latency in comparison with processing at the neighbourhood devices, this paper, propose a collaborative federated learning. The collaboration of learning levels including cloud, edge, and device levels are modeled as an optimization problem. In the proposed optimization, the accumulated trade-off of recognition accuracy and response time of recognition for all devices in the network is optimized. Deep reinforcement learning is proposed to solve the optimization, while considering the dynamicity in the communication and computation status of the network/devices. For a DDoS attack detection scenario based on a GRU based detection mechanism, the evaluation results show improvement in the gained rewards, the attack detection accuracy, the response time of recognition, and the accumulation of response time and accuracy, in comparison with the existing methods in the literature. Evaluating the performance of the proposed method for beyond security applications can be regarded as a future work.

Acknowledgment

The paper reflects only the authors' views. The Commission is not responsible for any use that may be made of the information this paper contains.

References

[1] K. B. Letaief, W. Chen, Y. Shi, J. Zhang, and Y.-J. A. Zhang, "The roadmap to 6G: AI empowered wireless networks," *IEEE Commun. Mag.*, vol. 57, no. 8, pp. 84–90, Aug. 2019.

[2] T. Taleb et al., "6G system architecture: A service of services vision," *ITU J. Future Evol. Technol.*, vol. 3, no. 3, pp. 710–743, 2022.

[3] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[4] X. Zhou, W. Liang, J. She, Z. Yan, I. Kevin, and K. Wang, "Two-layer federated learning with heterogeneous model aggregation for 6G supported Internet of Vehicles," *IEEE Trans. Veh. Technol.*, vol. 70, no. 6, pp. 5308–5317, Jun. 2021.

[5] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, "Low-latency federated learning and blockchain for edge association in digital twin empowered 6G networks," *IEEE Trans. Ind. Informat.*, vol. 17, no. 7, pp. 5098–5107, Jul. 2021.

[6] W. Y. B. Lim et al., "Decentralized edge intelligence: A dynamic resource allocation framework for hierarchical federated learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 3, pp. 536–550, Mar. 2022.

[7] S. Prathiba, G. Raja, S. Anbalagan, S. Gurumoorthy, N. Kumar, and M. Guizani, "Cybertwin-driven federated learning based personalized service provision for 6G-v2x," *IEEE Trans. Veh. Technol.*, vol. 71, no. 5, pp. 4632–4641, May 2022.

[8] I. Tomkos, D. Klonidis, E. Pikasis, and S. Theodoridis, "Toward the 6G network era: Opportunities and challenges," *IT Profess.*, vol. 22, no. 1, pp. 34–38, 2020.

[9] V. Ziegler, H. Viswanathan, H. Flinck, M. Hoffmann, V. Räisänen, and K. Hätönen, "6G architecture to connect the worlds," *IEEE Access*, vol. 8, pp. 173508–173520, 2020.

[10] M. V. Assis, L. F. Carvalho, J. Lloret, and M. L. Proença Jr., "A GRU deep learning system against attacks in software defined networks," *J. Netw. Comput. Appl.*, vol. 177, Mar. 2021, Art. no. 102942.

[11] J. Li, L. Lyu, X. Liu, X. Zhang, and X. Lyu, "FLEAM: A federated learning empowered architecture to mitigate DDoS in Industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 18, no. 6, pp. 4059–4068, Jun. 2021.

[12] Z. Zhao et al., "Federated learning with non-IID data in wireless networks," *IEEE Trans. Wireless Commun.*, vol. 21, no. 3, pp. 1927–1942, Mar. 2021.

[13] S. Liu, G. Yu, X. Chen, and M. Bennis, "Joint user association and resource allocation for wireless hierarchical federated learning with IID and non-IID data," *IEEE Trans. Wireless Commun.*, vol. 21, no. 10, pp. 7852–7866, Oct. 2022.

[14] S. Chu et al., "Federated learning over wireless channels: Dynamic resource allocation and task scheduling," *IEEE Trans. Cogn. Commun. Netw.*, vol. 8, no. 4, pp. 1910–1924, Dec. 2022.

[15] Z. Zhang, Z. Gao, Y. Guo, and Y. Gong, "Scalable and low-latency federated learning with cooperative mobile edge networking," *IEEE Trans. Mobile Comput.*, vol. 23, no. 1, pp. 812–822, Jan. 2024.

[16] S. Liu, J. Yu, X. Deng, and S. Wan, "FEDCPF: An efficient-communication federated learning approach for vehicular edge computing in 6G communication networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 2, pp. 1616–1629, Feb. 2022.

[17] Z. M. Fadlullah and N. Kato, "HCP: Heterogeneous computing platform for federated learning based collaborative content caching towards 6G networks," *IEEE Trans. Emerg. Topics Computing*, vol. 10, no. 1, pp. 112–123, Jan.–Mar. 2022.

[18] H. Liu et al., "Blockchain and federated learning for collaborative intrusion detection in vehicular edge computing," *IEEE Trans. Veh. Technol.*, vol. 70, no. 6, pp. 6073–6084, Jun. 2021.

[19] Y. Qu, M. P. Uddin, C. Gan, Y. Xiang, L. Gao, and J. Yearwood, "Blockchain-enabled federated learning: A survey," *ACM Comput. Surv.*, vol. 55, no. 4, pp. 1–35, 2022.

[20] L. Cui et al., "CREAT: Blockchain-assisted compression algorithm of federated learning for content caching in edge computing," *IEEE Internet Things J.*, vol. 9, no. 16, pp. 14151–14161, Aug. 2022.

[21] L. Feng, Z. Yang, S. Guo, X. Qiu, W. Li, and P. Yu, "Two-layered blockchain architecture for federated learning over the mobile edge network," *IEEE Netw.*, vol. 36, no. 1, pp. 45–51, Jan./Feb. 2022.

[22] Y. Liu, J. Peng, J. Kang, A. M. Iliyasu, D. Niyato, and A. A. Abd El-Latif, "A secure federated learning framework for 5G networks," *IEEE Wireless Commun.*, vol. 27, no. 4, pp. 24–31, Aug. 2020.

[23] A. R. Short, H. C. Leligou, M. Papoutsidakis, and E. Theocharis, "Using blockchain technologies to improve security in federated learning systems," in *Proc. 44th Annu. Comput., Softw., Appl. Conf.*, 2020, pp. 1183–1188.

[24] S. K. Singh, L. T. Yang, and J. H. Park, "FusionFedBlock: Fusion of blockchain and federated learning to preserve privacy in industry 5.0," *Inf. Fus.*, vol. 90, pp. 233–240, Feb. 2023.

[25] C. Feng, B. Liu, K. Yu, S. K. Goudos, and S. Wan, "Blockchain-empowered decentralized horizontal federated learning for 5G-enabled UAVs," *IEEE Trans. Ind. Informat.*, vol. 18, no. 5, pp. 3582–3592, May 2022.

[26] S. Fan, H. Zhang, Y. Zeng, and W. Cai, "Hybrid blockchain-based resource trading system for federated learning in edge computing," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2252–2264, Feb. 2021.

[27] E. Bandara, X. Liang, S. Shetty, R. Mukkamala, A. Rahman, and N. W. Keong, "SKUNK—A blockchain and zero trust security enabled federated learning platform for 5G/6G network slicing," in *Proc. 19th Annu. IEEE Conf. Sens., Commun., Netw.*, 2022, pp. 109–117.

[28] M. Shen et al., "Exploiting unintended property leakage in blockchain-assisted federated learning for intelligent edge computing," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2265–2275, Feb. 2021.

[29] L. Barbieri, S. Savazzi, M. Brambilla, and M. Nicoli, "Decentralized federated learning for extended sensing in 6G connected vehicles," *Veh. Commun.*, vol. 33, Jan. 2022, Art. no. 100396.

[30] S. Savazzi, M. Nicoli, and V. Rampa, "Federated learning with cooperating devices: A consensus approach for massive IoT networks," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4641–4654, May 2020.

[31] H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Blockchained on-device federated learning," *IEEE Commun. Lett.*, vol. 24, no. 6, pp. 1279–1283, Jun. 2020.

[32] S. R. Pokhrel and J. Choi, "Federated learning with blockchain for autonomous vehicles: Analysis and design challenges," *IEEE Trans. Commun.*, vol. 68, no. 8, pp. 4734–4746, Aug. 2020.

[33] Y. Wan, Y. Qu, L. Gao, and Y. Xiang, "Privacy-preserving blockchain-enabled federated learning for B5G-driven edge computing," *Comput. Netw.*, vol. 204, Feb. 2022, Art. no. 108671.

[34] S. Fan, H. Zhang, Z. Wang, and W. Cai, "Mobile devices strategies in blockchain-based federated learning: A dynamic game perspective," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 3, pp. 1376–1388, May/Jun. 2023.

[35] H. Yang, A. Alphones, Z. Xiong, D. Niyato, J. Zhao, and K. Wu, "Artificial-intelligence-enabled intelligent 6G networks," *IEEE Netw.*, vol. 34, no. 6, pp. 272–280, Nov./Dec. 2020.

[36] W. Wu et al., "AI-native network slicing for 6G networks," *IEEE Wireless Commun.*, vol. 29, no. 1, pp. 96–103, Feb. 2022.

[37] K. Li, T. Zhang, and R. Wang, "Deep reinforcement learning for multiobjective optimization," *IEEE Trans. Cybern.*, vol. 51, no. 6, pp. 3103–3114, Jun. 2021.

[38] N. C. Luong et al., "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3133–3174, 4th Quart., 2019.

[39] A. Mondal, D. Mishra, G. Prasad, and A. Hossain, "Joint optimization framework for minimization of device energy consumption in transmission rate constrained UAV-assisted IoT network," *IEEE Internet Things J.*, vol. 9, no. 12, pp. 9591–9607, Jun. 2022.

[40] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[41] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[42] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. 13th Conf. Artif. Intell.*, vol. 30, 2016, pp. 2094–2100.

[43] Q. Li et al., "A comprehensive survey on DDoS defense systems: New trends and challenges," *Comput. Netw.*, vol. 233, Sep. 2023, Art. no. 109895.

[44] H. Karthikeyan and G. Usha, "Real-time DDoS flooding attack detection in intelligent transportation systems," *Comput. Electr. Eng.*, vol. 101, Jul. 2022, Art. no. 107995.

[45] A. Verma and R. Saha, "Analysis of BayesNet classifier for DDoS detection in vehicular networks," in *Proc. Conf. Augment. Intell. Sustain. Syst.*, 2022, pp. 980–987.

[46] B. Cusack, R. Lutui, and R. Khaleghparast, "Detecting slow DDoS attacks on mobile devices," in *Proc. 27th Conf. Inf. Syst.*, 2016, pp. 1–12.

[47] A. Elsaeidy, K. S. Munasinghe, D. Sharma, and A. Jamalipour, "A machine learning approach for intrusion detection in smart cities," in *Proc. IEEE Veh. Technol. Conf.*, 2019, pp. 1–5.

[48] B. Tushir, Y. Dalal, B. Dezfouli, and Y. Liu, "A quantitative study of DDoS and E-DDoS attacks on WiFi smart home devices," *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6282–6292, Apr. 2021.

[49] Z. Liu, C. Guo, D. Liu, and X. Yin, "An asynchronous federated learning arbitration model for low-rate DDoS attack detection," *IEEE Access*, vol. 11, pp. 18448–18460, 2023.

[50] J. Li, Z. Zhang, Y. Li, X. Guo, and H. Li, "FIDS: Detecting DDoS through federated learning based method," in *Proc. IEEE Conf. Trust, Security Privacy Comput. Commun.*, 2021, pp. 856–862.

[51] R. Doriguzzi-Corin and D. Siracusa, "FLAD: adaptive federated learning for DDoS attack detection," 2022, *arXiv:2205.06661*.

[52] "Sumo." [Online]. Available: https://www.eclipse.org/sumo/

[53] A. Hanggoro and R. F. Sari, "Performance evaluation of the manhattan mobility model in vehicular ad-hoc networks for high mobility vehicle," in *Proc. Conf. Commun., Netw. Satell.*, 2013, pp. 31–36.

[54] S. Zhang, J. Liu, H. Guo, M. Qi, and N. Kato, "Envisioning device-to-device communications in 6G," *IEEE Netw.*, vol. 34, no. 3, pp. 86–91, May/Jun. 2020.

[55] Q. Wang, X. Zhao, Z. Lv, X. Ma, R. Zhang, and Y. Lin, "Optimizing the ultra-dense 5G base stations in urban outdoor areas: Coupling GIS and heuristic optimization," *Sustain. Cities Soc.*, vol. 63, Dec. 2020, Art. no. 102445.

[56] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy," in *Proc. Conf. Secur. Technol.*, 2019, pp. 1–8.

[57] R. Doriguzzi-Corin, S. Millar, S. Scott-Hayward, J. Martínez-del-Rincòn, and D. Siracusa, "Lucid: A practical, lightweight deep learning solution for DDoS attack detection," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 2, pp. 876–889, Jun. 2020.

Somayeh Kianpisheh received the Ph.D. degree in computer engineering from Tarbiat Modares University, Iran. From 2018 to 2020, for a period of two years, she was a Postdoctoral Researcher with the Concordia Institute for Information Systems Engineering, Concordia University, Canada. She also has a Postdoctoral Research experience with Aalto University, Finland, in 2021. Since 2022, she has been a Postdoctoral Researcher with the Center of Wireless Communications, University of Oulu, Finland. Her research interests include edge/cloud computing, beyond 5G, deep learning, federated learning, and in-network computing.

Tarik Taleb (Senior Member, IEEE) received the B.E. degree in information engineering, and the M.Sc. and Ph.D. degrees in information sciences from Tohoku University. He is currently a Full Professor with Ruhr University Bochum, Germany. Until 2023, he was a Professor with the Center of Wireless Communications, University of Oulu, Oulu, Finland. He is the Founder and the Director of the MOSA!C Lab. From October 2014 to December 2021, he was a Professor with Aalto University. Prior to that, he was a Senior Researcher and 3GPP Standards Expert with NEC Europe Ltd., Germany. He also worked as an Assistant Professor with Tohoku University, Japan. His research interests lie in the field of telco cloud.