Multi-Model based Federated Learning Against Model Poisoning Attack: A Deep Learning Based Model Selection for MEC Systems

Somayeh Kianpisheh[1], Chafika Benzaïd[1], and Tarik Taleb[2]

[1]*Centre for Wireless Communications, University of Oulu, Finland,* [2] *Ruhr University Bochum, Germany*
Emails: somayeh.kianpisheh@oulu.fi, chafika.benzaid@oulu.fi, tarik.taleb@rub.de

***Abstract*—Federated Learning (FL) enables training of a global model from distributed data. However, the singular-model based operation of FL is open with uploading poisoned models compatible with the global model structure and can be exploited as a vulnerability to conduct model poisoning attacks. This paper proposes a multi-model based FL as a proactive mechanism to enhance the opportunity of model poisoning attack mitigation. A master model is trained by a set of slave models. To enhance the opportunity of attack mitigation, the structure of client models dynamically change and the supporter FL protocol is provided. For a MEC system, the model selection problem is modeled as an optimization to minimize loss and recognition time, while meeting a robustness confidence. A deep reinforcement learning based model selection is proposed. For a DDoS attack detection scenario, results illustrate a competitive accuracy gain under poisoning attack with the scenario that the system is without attack, and also a potential of recognition time improvement.**

Index Terms- federated learning, poisoning attack, MEC, deep reinforcement learning.

I. INTRODUCTION

Federated Learning (FL) provides collaboration opportunities for devices in training a global model from locally distributed data while preserving the data privacy by sharing model parameters instead of raw data [1]. The learning is performed through some epochs. At each epoch, locally trained models of devices are transmitted to an aggregator server which aggregates the models to shape a global model for the next epoch of training. Utilizing a centralized server is inefficient due to single point of failure and intolerable latency in realtime scenarios [2], [3]. FL in edge computing is an alternative where Multi-Access Edge Computing (MEC), collocated at base stations either aggregate the parameters from the devices in their district and return the results for a fast response, or partially aggregate the parameters and further transmit to a server for global aggregation [2], [3]. Most of the FL studies in edge computing assume a secure FL protocol.

A model poisoning attack exploits system vulnerabilities to control the global model's aggregation by generating poisoned local model updates to inject into the system so that the global model becomes either useless or less accurate. To make the FL robust, defense mechanisms mainly focus on mitigating model poisoning attack through outlier-detection mechanisms [4], [5], [6], [7] or behavioural-based analysis [8], [9]: The studies in [4], [5] utilize robust aggregation rules in FL, that apply outlier detection methods to remove malicious models. The study in [6] provides a cost-efficient outlier detection method to isolate the attackers. The authors of [10] emphasize on performance of learning when applying an outlier detection. The authors of [7] make FL robust against Byzantine clients while preserving the privacy of individual users. The studies in [8], [9] analyze the behaviours of the clients to determine unreliable clients and remove them in the aggregation phase. Either outlier-detection based mechanisms or mechanisms using behavioural-based analysis might not be efficient in recognition of new emerging model poisoning attack patterns, or attacks with complex behaviours. Also, the performance of outlier-detection methods reduces in heterogeneous environment where benign model might be considered as outlier [10], [11]. In contrast to the reactive robustness mechanisms in the literature, this paper advocates a proactive approach to make the opportunity for poisoning lower, which can foster defense lines in addition to outlier-detection approaches, particularly for the attacks which can deceive the attack detection by intelligent model crafting.

The poisoning attacks have a knowledge about global model structure so that they emulate a poisoned model compatible with global model. For example, *A Little Is Enough* attack [11], according to a Gaussian-based statistics deviates *each dimension* of the model parameter from the *mean* with a fraction of the *standard deviation*. Model poisoning attack has been mathematically modeled in [8]. The attacker tries to deviate from global model with an arbitrary malicious model. Poisoned models compatible with global model structure, are uploaded. *The poisoned model is calculated by updating global model with a learning rate based on the difference between the malicious target model and the global model.*

The poisoning attacks use the singular-model based operation of FL as a vulnerability. They assume that all clients use the same model as the global model. Then, through applying an algorithm that follows their objectives in global model deviation, they emulate poisoned models with the same structure of the global model and inject them in learning process e.g., [11], [8], [4], [6]. This paper, proposes a multi-model based FL at which a master model or global model is trained by a set of slave models, letting dynamic change of client models within the learning epochs.

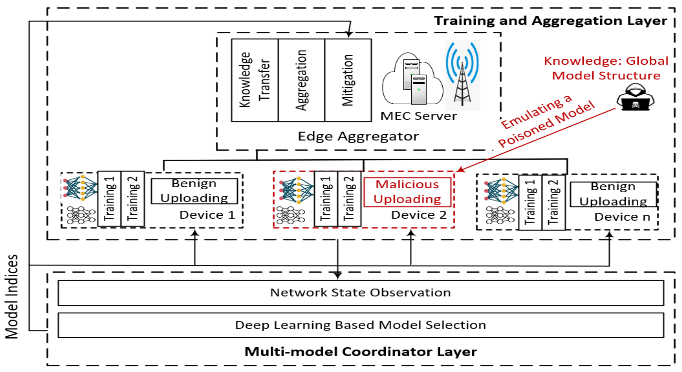
Dynamic change of models that should be trained by clients can be either a barrier for attack operation or reduce the attack triggering chance. For example, *A Little Is Enough* attack [11] *may not be able to drive the mean and standard deviation for the global model dimensions*, when the clients will have heterogeneous models with different structures. Also, poisoning attacks e.g., [11], [8], [4], [6] construct a poisoned model compatible with the structure of the global model. *When the client model is planned to be different from the global model, the mismatch between uploaded poisoned model and planed model, can be detected at aggregation time–providing the opportunity of attack detection and mitigation.* Emulation of client model is impossible if the communication protocol of FL be secure and the client model structure be dynamic and unknown for the attacker. Otherwise, random crafting of client model can be easily detected by detection mechanisms due to high deviation from normal behaviour. Sophisticated crafting of a client model demands designing new advanced attacks and introduces complexities. Advanced knowledge about e.g., dynamicity of the models, models' structures are required and new optimization for crafting should be derived to deviate global model in the desired direction, with crafting of heterogeneous, dynamic and different models from global model at client sides. To the best of our knowledge this sort of attack has not been investigated yet. Thus, multi-model based FL introduces potentials in dealing with poisoning attacks.

This paper introduces **M**ulti-**M**odel based FL (MM-FL) to reduce poisoning opportunity. An optimization is provided for model selection problemand a Deep Reinforcement Learning (DRL) based model selection is proposed. For a Distributed Denial of Service (DDoS) attack detection scenario, results illustrate improvement in accuracy and recognition time.

II. System Model

Network: A network consists of N devices such as IoT and mobile devices, M Base Stations (BSs) equipped with MEC servers, and a (central) cloud. Let m^{th} BS be represented with BS_m with CPU frequency f_m^{cmp}. The available CPU cycle at the central cloud is f_c^{cmp}. We use symbol u for a device, and its data with size $|D_u|$ is represented with $D_u = \{(x_1^u, y_1^u), ...(x_{|D_u|}^u, y_{|D_u|}^u)\}$, where y is the label for input x. CPU frequency of the device u is f_u^{cmp} .
The model sharing can be done through MEC/cloud infrastructure, to construct a global model through FL. The aim of FL is to train a global learning model M_c based on the distributed data in the devices, through some learning epochs. At each epoch of learning, a device performs learning on its local data to train a local model. The local models will evolve global model through FL process, based on which the required recognition is performed (e.g., attack detection).

Adversary Knowledge and Operation: At any epoch of learning, the adversary can exploit vulnerabilities of devices and compromise them to inject poisoned models in learning process. The attacker does not have any control over aggregation process at MEC, nor over the protocol of the benign devices which follow a normal implementation of the protocol.



Training and Aggregation Layer
Knowledge Transfer
Aggregation
Mitigation
MEC Server
Edge Aggregator
Knowledge: Global Model Structure
Emulating a Poisoned Model
Training 1
Training 2
Benign Uploading
Device 1
Malicious Uploading
Device 2
Benign Uploading
Device n
Model Indices
Network State Observation
Deep Learning Based Model Selection
Multi-model Coordinator Layer

Fig. 1. Multi-model FL architecture.

In consistent with the poisoned attacks in the literature, adversary has the knowledge about global model and emulates poisoned model(s) with the same structure as global model through applying an algorithm. It then uploads the malicious model(s) on behalf of the compromised device(s) when communicating with aggregator at MEC [8]. The attack can be targeted or untargeted depending on the attacker objective, and our approach is applicable for both cases. In the case of targeted attack, the malicious model(s) are crafted so that the aggregated model over learning epochs approaches a targeted model [8]. Otherwise, untargeted deviation strategy is employed in poisoning [8].

Federated Learning Objective: The aim of federated learning is to train a global learning model, namely called master model based on the distributed data in the devices that take part in learning. Master model is trained through a set of slave models with the objective of making the FL robust with a predefined confidence level, while keeping the performance of FL in terms of accuracy and recognition time.

III. Multi-Model Based FL

Fig. 1 shows the architecture of multi-model based FL. Multiple models are used to train the main model, thereby having multiple training scripts at a device. At each learning epoch, network state is observed, according which a DRL-based model selection decides about the model that should be trained at each device. Then, the decision about models are announced to devices and they will train the model, they have been asked. After uploading model parameters, the edge aggregator will perform attack mitigation, aggregation, and knowledge transfer. The model parameters are downloaded for the next round. In the case of attack, the adversary that has knowledge about the global model structure, emulates poisoned models compatible with global model and upload them on behalf of the compromised devices. The attack can be mitigated before aggregation due to mismatching with the model plan which the multi-model coordinator arranged it.

A. Fundamental

Let M_c be the master model with weights w_c and $M_s = \{M_s^1, ...M_s^l\}$ be the slave models with weights $\{w_s^1, ...w_s^l\}$. M_c also can belong to the set of slave models, i.e., $M_c \in M_s$. The main idea is to train master model using slave modes according to an optimal model selection plan that operates as hindrance against Poisoning attack while maintaining the performance of FL.

Let $x_s^{j,u}(\tau)$ be the binary variable indicating the usage of model M_s^j in federation of knowledge of device u at epoch time τ. The value of 1 indicates the usage of the model, while a value of 0 indicates not use of the model. As at every epoch of FL, only one model is selected for a device:

$$\forall \tau, u : \sum_{j=1}^{l} x_s^{j,u}(\tau) = 1. \tag{1}$$

We define the variable $\mathbb{T}_u(\tau)$ as the slave model that is selected to be used for the device u. We assume the associated weight is $\mathbb{W}_u$. The attack which emulates a poisoned global model on behalf of device u, can be detected at edge aggregator, due to a mismatch with the plan, if we have:

$$\mathbb{T}_u(\tau) \neq M_c \tag{2}$$

The aim is to secure the FL with a user defined confidence level, while maintaining the efficiency of FL performance. The diversity of models, will introduce hidden variables and will reduce the success of triggering poisoning attack. However, two issues will rise: First, new FL protocol should be designed to let the multi-model based learning process. Second, an optimization framework should be developed to decide about the models that are trained by devices, considering the dynamic nature of the network.

Algorithm 1: Knowledge Transfer

```
1 for each BS $m$ do
2     for each slave model $M_{e,m}^{s,j}$ do
3         for each instance in $D_{e,m}$ do
4             input ← instance
5             give input to model $M_{e,m}^{s,j}$
6             label ← output neurons of $M_{e,m}^{s,j}$
7         $w_{e,m}(\tau)$ ← GD optimization to minimize the loss
            function with the input for training as labeled $D_{e,m}$
```

B. Multi-Model Based FL Protocol

Multi-model based FL can train master model from slave model set. Master model can be included in the slave model set, since it can be the most efficient model in the context of recognition problem. The parameters of master model are found in learning process to minimize the global loss function in (4), where $f(w_c, x_s^u, y_s^u)$ is the loss value over master model for the sample s of data of user u.

$$\min_{w_c} F = \frac{1}{N} \sum_{u} \frac{1}{|D_u|} \sum_{s=1}^{|D_u|} f(w_c, x_s^u, y_s^u). \tag{3}$$

FL is performed through some epochs, indexed by τ, until convergence or meeting of termination condition. BS m trains a master model $M_{e,m}$, which has the same neural structure as M_c, from a set of slave models $M_{e,m}^s = \{M_{e,m}^{s,1}, ...M_{e,m}^{s,l}\}$ with the same structures of M_s. It also has an unsupervised data set of instances $D_{e,m}$ that is used for knowledge transfer. Each FL epoch, consists of the steps as below:

1) *Local Training:* Each device employs a Gradient Decent (GD) based method using its local batch data to minimize the loss function over the planed model $\mathbb{T}_u(\tau)$. Then it transmits the local model parameters to its associated BS for the purpose of partial aggregation.
2) *Attack Detection and Mitigation:* For each uploaded model from a device u if the the parameters count are different than the planed model parameters count for that device ($\mathbb{W}_u(\tau)$), the model will be detected as malicious and will be excluded from the aggregation.
3) *Partial Aggregation at BSs:* For each slave model, the partial aggregation at BS m, is performed as a weighted average of received parameters of that model:

$$M_{e,m}^{s,j} = \frac{1}{K_{m,j}} = \sum_{u=1}^{K_{m,j}} \mathcal{N}(|D_u|).w_u, \tag{4}$$

where $K_{m,j}$ is the number of devices under the coverage of BS_i, who use slave model $M_{e,m}^{s,j}$; and $|D_u|$ is the size of batch data of device u; and finally, w_u is the weight of the trained model of device u. Symbol $\mathcal{N}(.)$ is the normalization operator.
4) *Knowledge Transfer To Master Model:* Each BS performs knowledge transfer step to transfer the knowledge of each slave model to the master model. First, the unsupervised data set is given to each slave model to be labeled. Then, the labeled data set is used to train and update the weights of the master model $w_{e,m}(\tau)$, based on a GD based optimization. Algorithm 1 is the pseudocode.
5) *Parameters Uploading:* Each BS transmits it's master model $w_{e,m}(\tau)$ to the cloud for the global aggregation.
6) *Cloud Aggregation:* The global model is constructed at central cloud as average of models of BSs and will be broadcasted to the BSs:

$$M_c(\tau) = \frac{1}{M} \sum_{m=1}^{M} M_{e,m}. \tag{5}$$

7) *Broadcasting the Update Results:* Among the slave models in $M_{e,m}^s$, the model with the same structure of $\mathbb{T}_u(\tau)$ will be transmitted to the user u.

FL epochs will be repeated, and finally the M_c parameters will be broadcasted to the users at the final epoch or when the termination condition be met. *Note that at step 6 after broadcasting of global model, the master model at BSs, and accordingly the devices that have uploaded the master model, will automatically be updated. This lets the master model update distribution even through training.*

IV. Model Selection Optimization

The 6G wireless model in [12] is used for communication between devices and the base stations. The transmission rate device u communicates with BS i, is calculated by (6). 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. Eq. (7) is the channel gain calculation. Here, C_g, $d_{u,i}$, α are respectively path loss fading coefficient, distance between device u and base station i, and path loss exponent. *Note that when devices move transmission rates will change due to distance variation.*

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

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

The time cost of one epoch of FL is the time it takes a device receives the new updates of the model parameters. It includes the time slots allocated for local training, up/down-link parameter transmission, knowledge transfer, and the aggregation.

- *Knowledge Transfer:* In BS m, the time for transferring knowledge to master model is calculated by:
$$T_{knw}^{m} = \sum_{j}[\frac{|D_{e,m}|.f_{ds,j}^{cmp}}{f_{m}^{cmp}} + \frac{|D_{e,m}|.f_{ds,c}^{cmp}}{f_{m}^{cmp}}], \tag{8}$$
where $f_{ds,j}^{cmp}$, $f_{ds,c}^{cmp}$ are the number of required CPU cycles to label one unit of data by slave model M_s^j, and train master model $M_{e,m}$ with one unit of data, respectively.
- *Aggregation:* The time for partial aggregation at BS m includes: (a) the time it takes the parameters be transmitted from devices under the coverage area of that base station i.e., R_m to the base station; (b) the aggregation operation time over the slave models:
$$T_{ag}^{m} = \max_{u \in R_m} \frac{|\mathbb{W}_u|}{R_{u,i}} + \sum_{j} \frac{X_s^j.|w_s^j|.f_w^{cmp}}{f_m^{cmp}}, \tag{9}$$
where f_w^{cmp} is the number of required CPU cycles to aggregate one unit of data. X_s^j is the number of devices under coverage of BS, that uploaded slave model M_s^j.
The time for global aggregation at central cloud is calculated by (10). It includes the time for partial aggregation/knowledge transfer, at base stations, as well as the time takes for global aggregation of the parameters collected from base stations at the central cloud:
$$T_{ag} = \max_{m=1..M}(T_{ag}^{m} + T_{knw}^{m} + \frac{|w_c|}{R_{i,c}}) + \frac{M.|w_c|.f_w^{cmp}}{f_c^{cmp}}. \tag{10}$$
- *Downlink Parameter Transmission:* The required time to download the parameters at device u under the coverage of BS m is calculated by (11), as the time required for parameters transmission from cloud to BS (with transmision rate R_c), and from the BS to the device.
$$T_{down}(u) = \frac{|w_c|}{R_c} + \frac{|\mathbb{W}_u(\tau)|}{R_{m,u}}. \tag{11}$$
- *Local Training:* Local training time at device u is calculated based on the computing capability of the device and the batch size. Eq. (12) is the calculation. Here, f_s^{cmp} is number of required CPU cycles to train one sample of data over selected model.
$$T_{loc}(u) = \frac{|D_u|.f_s^{cmp}(\tau)}{f_u^{cmp}}. \tag{12}$$

After the download of partially aggregated model, the device can perform the recognition. The recognition time in one epoch of FL, for device u, is calculated as below:
$$T_{Int}(u) = \max_{u} K.T_{loc}(u) + T_{ag} + T_{down}(u) + \frac{f_u^{inf}}{f_u^{cmp}}, \tag{13}$$
where K is the local training iterations before applying the other epoch of learning. f_u^{inf} is the number of CPU cycles to perform the recognition for a sample of input features. The response time varies over epochs of learning.

In a multi-model based FL, model selection for devices impacts the training and communication time, as well as the accuracy (See Eq. (9), (11), (12)). Straggler devices with poor transmission channel can prolong the learning process [13]. Selecting a smaller model than the master model, under poor

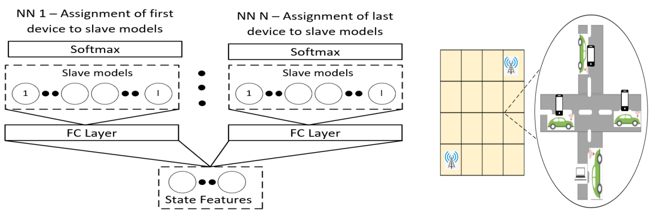

NN 1 – Assignment of first device to slave models
Softmax
Slave models
1
l
FC Layer
NN N – Assignment of last device to slave models
Softmax
Slave models
1
l
FC Layer
State Features

Fig. 2. (a) Policy networks. (b) Simulation grid.

transmission rates can speed up training, and compensate the latency of transmission, as well as reduce the transmission load–all speeding up the learning process. Furthermore, the security is enhanced since a different model than the global model can be utilized for training and poisoning attack can be mitigated at aggregation time (See Fig. 1). However, in the cases that global model be more efficient than slave models, there can be accuracy-reduction. On the other hand, in high quality of communication status, selecting global model in the case that it is more efficient than smaller model can enhance the accuracy, however the robustness reduces. Indeed, appropriate tradeoffs are required and optimization should be performed for the optimal selection.
The slave models are assigned to the users, to minimize the loss function and recognition time over all users, over all epochs of learning. α and β make the loss and time values in the same scale. Besides the already defined constraints, constrain (15) ensures the robustness against poisoning attack. $1.(B)$ is 1 if condition B holds. To reduce the potential for poisoning, it ensures that the ratio of global model selection be less than a threshold value $T_{max} \in [0, 1]$. If $T_{max} = 0$, poisoning attacks that emulate global model can not be triggered. However, the accuracy will be reduced if the master model be the most accurate model.
$$\min_{x_s^{j,u}(\tau)} \sum_{\tau,u} \alpha.F_u(\tau) + \beta.T_{Int}[\tau](u) \tag{14}$$
sbj. (1), (2), (4), (5), (6), (7), (8), (9), (10), (11), (12), (13), (14),
$$\forall \tau : \sum_{u} 1.(\mathbb{T}_u(\tau) = M_c) \leq N.T_{max}. \tag{15}$$

V. DRL-based Model Selection

The search space order is exponential with dynamic channel communication. MDP and RL, advocated for solutions adoptable with dynamic network situations, can be used for formulation, since: (i) Function (14) has the memoryless property, as it can be calculated as the sum of loss and recognition time values in devices, in the current interval and the function value in the previous interval; (ii) Considering the parameters determining the current state e.g., transmission rates, association of slave models to devices, every action that is performed by the agent ends to a new state transition, that only depends on the current state. (iii) The function (14) is in the form of accumulated rewards. Through an iterative process of observing the state, choosing an action, and receiving a reward, the state-action Q-values are estimated by Bellman equation [14]. The high dimension of the states and the dynamicity in state transitions makes observing all states and actions in training impossible, thereby inefficiency of conventional RL. To deal with this problem, we adapt DRL [14], that generalizes experienced states/actions to non-observed ones through a neural network-based approximation of Q-values.

MDP Elements: MDP Elements include:

State: The features representing the state of network at time step τ, are as below:

- The transmission rates represented with matrix $\mathbb{R}_{d,e}(\tau)$, at which the entry at row i and column j is the transmission rate between device i and BS j, at epoch τ. Available bandwidth, distance of devices from BSs, channel gain varies the transmission rates.
- The current slave model distribution denoted by vector $\mathbb{X}(\tau)$, at which the entry at row j and column u is the assignment of slave model M_s^j to device u at learning epoch τ (values of $x_s^{j,u}(\tau)$)

Actions: The action is decision about the assignment of devices to slave models (values of $x_s^{j,u}$ for the next epoch of learning).

Reward: In the case of violation of constraints (1), (15), reward is 0. To ensure, optimizing the objective function the reward function is calculated as:

$$\mathbb{R}(s(\tau), a(\tau)) = \sum_u \alpha.F_u(\tau) + \beta.T_{Int}[\tau](u) \tag{16}$$

Training: Using N policy networks, the decision policy is derived by training them. Each Neural Network (NN) represents the assignment of a device to the slave models. The input neurons are the state features. There is a Fully-Connected layer, with Softmax activation function. The neuron i in the output layer of NN u, indicates the probability of assigning the slave model M_s^i to the device u (See Fig. 2.a).

Each episode consists of a run of FL within several epochs of learning. There are variation in network communication status, location of devices, and compromised devices within episodes. Training is done through two steps performed at every epoch of learning: (i) *Exploration:* According to $\epsilon-$greedy policy, with a $1-\epsilon$ probability, a random assignment of slave models to devices is selected. Otherwise, the current state features are given as input to the NNs. For each device, the slave model, with the highest probability at output layer, will be assigned to the device. (ii) *Updating the weights:* After the assignment strategy, reward is calculated by (16), accordingly the NNs' weights are updated by Gradient Descent (GD) method, and using Bellman equation [14].

VI. Experimental Results

A scenario that 10 devices that are moving by vehicles, collaborate in DDoS attack detection, is considered. The advantage of applying FL in DDoS attack detection in terms of enhancing accuracy through sharing attack detection models in comparison with individual learning have been illustrated in recent studies e.g., [3], [15]. The study in [3], extensively discusses the FL benefits for DDoS attack detection. We used CICDDoS 2019 data set [16] comprising realistic traffic to abstract the communications for legitimate and DDoS attack traffic through protocols e.g., HTTP, FTP. We have randomly distributed 9000 instances composed of UDPLag and SYN DDoS attacks, among devices for training, as well as 3000 instances for the purpose of test. The dataset provides 87 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.

We assume a 4×4 bidirectional grid environment with 100 m width for each grid cell, where grid lines are bidirectional roads (Fig. 2.b). Mobility traces of vehicles have been generated by SUMO simulator [17]. Manhattan mobility model in urban areas [17] 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. *The mobility of vehicles will cause dynamicity in transmission rates.* We used GRU as we explained it with details in [3] for attack detection. The feature matrix for the packets in a flow is arranged as rows of patterns. The occurrence probability for each pattern is calculated as a function of previous observations using a GRU. A flow (including 10 packets) is malicious if the ratio of the malicious packets in that flow is larger than a threshold (0.7). TensorFlow and Keras are used to implement the GRUs/DRL.

Without loss of generality, the learning collaboration is only performed in the level of edge computing. Two BSs equipped with MEC-servers with respectively CPU frequencies of 3.2 and 2.6 GHz [18] are located in the locations $[50, 50], [350, 350]$ to provide edge coverage for devices. Each BS has 2400 random unsupervised instances from CICDDOS data set for knowledge transfer. The coverage radius of base stations are 300 m, and their transmission bandwidth are respectively, 28 and 30 MHz [3]. The CPU frequency of devices are randomly chosen in the range of 1.9 up to 2.4 GHz. The transmission power of BSs and devices are respectively 34 db and 23 db [18]. Path loss exponent is 5 and back ground noise power is -174 db.m [18].

There are more explorations at early iterations of DRL, while the exploitation gradually increases up to the greedy selection of 98% at the last episode. For GRUs, the learning rates 0.07 for GD optimization operated efficiently. $LearningRateSchedule$ package of Keras reduces the learning rate within training for the purpose of convergence. The out-layer of the GRU is a neuron to predict the occurrence probability of a packet. See [3] for details. Discount rate of 0.1 and ADAM optimization in DRL also operated efficiently.

In the rest, GRU with 28 and 32 neurons in the hidden layer are represented with GRU 28 and GRU 32, respectively. Various scenarios have been considered: Two FL scenarios when GRU 28 and GRU 32 are applied for DDoS attack detection. Each FL is applied in a system without attack (FL-GRU 28, FL-GRU 32), and a system with poisoning attack (FL-GRU 28-With Attack, FL-GRU 32-With Attack). In MM-FL the slave models set include GRU 28 and GRU 32. Two scenarios of MM-FL are considered: MM-FL with master model as GRU 28 and MM-FL with master model as GRU 32. We have MM-FL-DRL which utilizes DRL in model selection, and MM-FL-RND which uses a random selection of slave models. At each episode, the adversary compromises 3 to 5 random devices and emulates Malicious Local Models

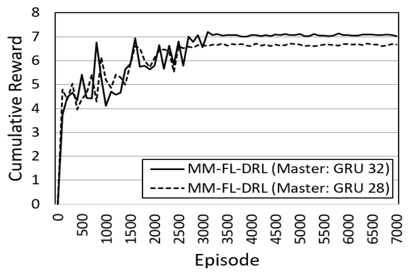


Fig. 3. Cumulative reward.

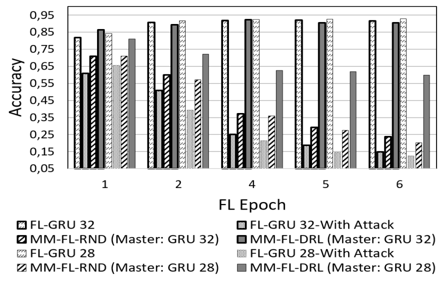


Fig. 4. DDoS attack detection accuracy.

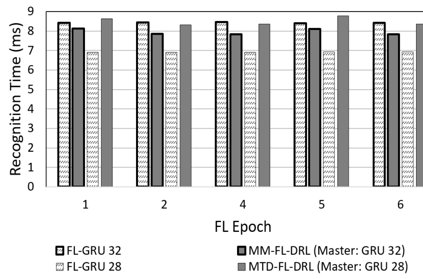


Fig. 5. Recognition time.

(MLMs) with the global model structure, for updating using the method in [8], as explained in Introduction, with random target malicious models and random learning rate in (0.25, 0.35). The results are average over 50 runs of test, $T_{max} = 0.6$.

Fig. 3 illustrates the cumulative reward gain within episodes in MM-FL. The cumulative reward has increased up to range of 6.5 to 7 and become stable around episode 3200, with a dominance in reward with the case that GRU 32 is master model. Since, the scenario that GRU 32 is master has gained better performance in terms of accuracy and recognition time.

Fig. 4 shows the accuracy of DDoS attack detection. The bars with bold boarders are for GRU 32 scenarios, and the rest are for GRU 28 scenarios. For GRU 32 scenarios, the accuracy considerably decreases after poisoning, and this decrease becomes the worse at the late epochs of FL due to more involvement of MLMs in aggregation e.g., drop from 0.92 to 0.15 at epoch 6. MM-FL-RND has increased the accuracy slightly e.g., for 0.09 in epoch 6. MM-FL-DRL has increased the accuracy considerably and in competitive with the scenario of FL without attack, due to optimization of loss function in model selection and selecting atleast 60% of the local models different than the main model, according which MLMs can be detected and mitigated. Similar results has been achieved for GRU 28 scenarios. However, MM-FL-DRL (Master: GPU 32) operates better than MM-FL-DRL (Master: GPU 28), which illustrates the more efficiency of knowledge transfer from GRU 28 to GRU 32, in comparison with the reverse in MM-FL-DRL (Master: GPU 28). This is an application dependent issue.

Fig. 5 shows the recognition time. For GRU 32 scenario, the recognition time reduces up to 0.6 ms, after multi-model FL application with GRU 32 as master. Since, in this scenario, MM-FL-DRL uses GPU 28 for atleast 60% of devices which demands less training and transmision time than GRU 32. However, application of MM-FL with GPU 28 as master, increases recognition time in comparison with FL with GRU 28 due to employing larger model of GRU 32 in training.

VII. Conclusion

This paper proposes a multi-model based FL to enhance the opportunity of model poisoning attack detection and mitigation. A protocol at which a master model is trained by a set of slave models, is explained. For federation over a MEC system, the model selection problem is modeled as an optimization and a DRL-based model selection method adaptable with dynamic network conditions is provided. For a DDoS attack detection scenario, results illustrate a competitive accuracy with the scenario without attack. With a smaller model employment in training recognition time reduces.

Acknowledgment

This work was supported in part by the EU's HE programme HORIZON-JU-SNS-2022 under the RIGOUROUS project (Grant No. 101095933). 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] B. McMahan *et al.*, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*, 2017, pp. 1273–1282.

[2] J. Zhou *et al.*, "A differentially private federated learning model against poisoning attacks in edge computing," *IEEE Trans. on Dependable and Secure Computing*, 2022.

[3] S. Kianpisheh and T. Taleb, "Collaborative federated learning for 6G with a deep reinforcement learning based controlling mechanism: A ddos attack detection scenario," *IEEE Trans. on Network and Service Management, DoI: 10.1109/TNSM.2024.3387987*, 2024.

[4] E. M. El Mahdi, R. Guerraoui *et al.*, "The hidden vulnerability of distributed learning in byzantium," in *International Conf. on Machine Learning*, 2018, pp. 3521–3530.

[5] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *International Conf. on Machine Learning*, 2018, pp. 5650–5659.

[6] S.-M. Huang, Y.-W. Chen, and J.-J. Kuo, "Cost-efficient shuffling and regrouping based defense for federated learning," in *IEEE Global Communications Conf.*, 2021, pp. 1–6.

[7] J. So *et al.*, "Byzantine-resilient secure federated learning," *IEEE J. on Selected Areas in Communications*, 2021.

[8] R. Al Mallah *et al.*, "Untargeted poisoning attack detection in federated learning via behavior attestation," *IEEE Access*, 2023.

[9] X. Pan *et al.*, "Justinian's gaavernor: Robust distributed learning with gradient aggregation agent," in *USENIX Security Symp.*, 2020.

[10] L.-Y. Chen, T.-C. Chiu, A.-C. Pang, and L.-C. Cheng, "Fedequal: Defending model poisoning attacks in heterogeneous federated learning," in *IEEE Global Communications Conf.*, 2021, pp. 1–6.

[11] G. Baruch, M. Baruch, and Y. Goldberg, "A little is enough: Circumventing defenses for distributed learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[12] Z. M. Fadlullah and N. Kato, "HCP: Heterogeneous computing platform for federated learning based collaborative content caching towards 6G networks," *IEEE Trans. on Emerging Topics in Computing*, 2020.

[13] R. Schlegel *et al.*, "Codedpaddedfl and codedsecagg: Straggler mitigation and secure aggregation in federated learning," *IEEE Trans. on Communications*, 2023.

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

[15] J. Li *et al.*, "Fleam: A federated learning empowered architecture to mitigate ddos in industrial iot," *IEEE Trans. on Industrial Informatics*, vol. 18, no. 6, pp. 4059–4068, 2021.

[16] I. Sharafaldin *et al.*, "Developing realistic distributed denial of service attack dataset and taxonomy," in *Conf. on Security Technology*, 2019.

[17] "Sumo: https://www.eclipse.org/sumo/."

[18] Y. Lu *et al.*, "Low-latency federated learning and blockchain for edge association in digital twin empowered 6G networks," *IEEE Trans. on Industrial Informatics*, vol. 17, no. 7, pp. 5098–5107, 2020.