Annals of Telecommunications
https://doi.org/10.1007/s12243-023-00977-4

Federated deep Q-learning networks for service-based anomaly detection and classification in edge-to-cloud ecosystems

Mays AL-Naday[1] · Vlad Dobre[2] · Martin Reed[1] · Salman Toor[3] · Bruno Volckaert[4] · Filip De Turck[4]

Received: 15 June 2023 / Accepted: 9 July 2023

Abstract
The diversity of services and infrastructure in metropolitan edge-to-cloud network(s) is rising to unprecedented levels. This is causing a rising threat of a wider range of cyber attacks coupled with a growing integration of a constrained range of infrastructure, particularly seen at the network edge. Deep reinforcement-based learning is an attractive approach to detecting attacks, as it allows less dependency on labeled data with better ability to classify different attacks. However, current approaches to learning are known to be computationally expensive (cost), and the learning experience can be negatively impacted by the presence of outliers and noise (quality). This work tackles both the cost and quality challenges with a novel service-based federated deep reinforcement learning solution, enabling anomaly detection and attack classification at a reduced data cost and with better quality. The federated settings in the proposed approach enable multiple edge units to create clusters that follow a bottom-up learning approach. The proposed solution adapts a deep Q-learning network (DQN) for service-tunable flow classification and introduces a novel federated DQN (FDQN) for federated learning. Through such targeted training and validation, variation in data patterns and noise is reduced. This leads to improved performance per service with lower training cost. Performance and cost of the solution, along with sensitivity to exploration parameters, are evaluated using examples of publicly available datasets (UNSW-NB15 and CIC-IDS2018). Evaluation results show the proposed solution to maintain detection accuracy in the range of ≈75–85% with lower data supply while improving the classification rate by a factor of ≈2.

Keywords Cyber security · Federated deep reinforcement learning · Deep Q-learning · Anomaly detection · Cloud-to-edge continuum · Fog computing

1 Introduction

Digital transformation increases the diversity of actors and distribution of data, services, and resources in metropolitan edge-to-cloud (i.e., fog) computing ecosystems. This intuitively raises the threat of a wider mixture of cyber attacks to new levels [1, 2]. Added to that, the shift towards service-based paradigms attracts finer granularity attacks, targeting application components rather than hosts. Consequently, there is a mounting need to develop efficient service-based anomaly detection solutions able to cope with this change, at the granularity of hosts and services. The prospect of deploying such solutions in the resource-constrained edge of the fog motivates the requirement for efficient use of resources and incentivizes the interconnection of such solutions for collaborative knowledge building.

Machine learning is an attractive approach to anomaly detection, supporting intelligence-based rapid and autonomous reaction. Existing learning methods vary in their capabilities of detecting cyber attacks [3–5]. Supervised learning is found to be highly effective in identifying known attacks while being relatively easy to implement; but the strong dependency on labeled data during training renders the approach less effective against zero-day attacks. Unsupervised learning does not require labeled data; however,

✉ Mays AL-Naday
mfhaln@essex.ac.uk

[1] Computer Science and Electronic Engineering, University of Essex, Colchester, UK

[2] School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Stockholm, Sweden

[3] Department of Information Technology, Division of Scientific Computing, Uppsala University, Uppsala, Sweden

[4] IDLab, Department of Information Technology (intec), Ghent University-imec, Ghent, Belgium

Published online: 31 August 2023

the complexity of the approach is considerably higher than that of the supervised counterpart, making it computationally expensive. The complexity and cost are even higher for deep learning variants. Added to that, while both approaches perform well when the training and testing sets are drawn from similar types of data, their performance plummets when the training and testing datasets differ [6]. Reinforcement learning presents a trade-off in having weak dependency on labeled data while offering the ability to associate patterns. This approach is attractive for anomaly detection as it allows detecting known and unknown attacks with less computation complexity than unsupervised counterparts and better ability to adapt to differing datasets.

So far, research efforts have proposed few solutions for intelligent anomaly detection in service-based systems [7–9]. Most of them follow a centralized approach in which data feeds are collated in a large data warehouse and utilized to train global models, irrespective of individual services. Fewer solutions, such as [10, 11], propose a federated learning approach where models are trained in parts of the network and interconnected to shared knowledge. These solutions are better aligned with the fog computing paradigm, but they remain service-agnostic. While this has proven to be successful, two main drawbacks are emerging from using a dataset of all services: first, such a set has a maximum pattern-diversity and all the outliers. Consequently, training would require longer cycles, which adds to the computation cost, and may suffer from an increased likelihood of raising false alarms. Second, using such a set obscures the individuality of each service, hindering the ability to adapt the learning task to the service. Moreover, with prominent adoption of a cloud-to-edge continuum, there is a stronger need for service-based federated learning. In particular, there is a need to improve the detection and classification accuracy despite the lower local supply of data, while reducing the demand for computation resources.

This article extends the novel solution, presented earlier in [12], of service-based federated deep reinforcement learning for anomaly detection in fog ecosystems. The proposed solution defines learning clusters that follow a bottom-up approach; each is a system of agents and an aggregation point. Each agent, likely to run at the edge, trains and scores a service-tunable deep Q-learning network (DQN), then sends it to its aggregation point. The latter incorporates a novel federated DQN (FDQN) that calculates an aggregate of local scores and uses it to decide whether or not to generate and disseminate a cluster model.

This is to achieve faster training without compromising the detection accuracy, hence achieving sophisticated learning over constrained resources. We evaluate the performance and cost of the proposed solution over publicly available example datasets, UNSW-NB15 [13] and CIC-IDS2018 [14, 15].

Evaluation results illustrate the superiority of the proposed service-based solution (both central and federated) over service-agnostic counterparts. This is reflected by improvements in detection accuracy and attack classification, at a lower supply of training data. Furthermore, we analyze sensitivity to the exploration parameters and show the impact of different settings on performance quality.

The contributions of this work are summarized as follows:

- A service-based federated deep reinforcement learning solution for anomaly detection and classification in fog ecosystems. This includes a FQDN algorithm that provides score-based generation of aggregate models.
- Utilization of DQN in combination with a class-based reward policy to enable service-tunable anomaly detection and classification.
- Inter-cluster, same-service, model exchange to allow for knowledge transfer across clusters.

To the best of our knowledge, the initial work of [12] and its extension here are the first to propose such a service-based solution. Finally, we evaluate the performance, cost, and sensitivity of the proposed solution using example datasets and compare the results against state-of-the-art service-agnostic baseline.

The rest of this paper is structured as follows: Section 2 provides an overview of state-of-the-art work on anomaly detection using machine learning. Section 3 describes the proposed federated learning system, while Section 4 presents the performance and cost evaluation along with the results of the sensitivity analysis. Finally, Section 5 draws our conclusions.

2 Related work

Recent years have witnessed a rapid adoption of machine learning for detecting anomalies in network traffic [4, 5, 16] or application domains such as [17]. Multiple solutions for service-agnostic detection based on centralized deep learning have been proposed, such as [7, 18]. Nedelkoski et al. [7] propose a solution that combines a variational autoencoder and are current neural network (RNN) to learn from datasets of IT operations in large distributed systems, in order to identify anomalous behavior of services. Although their work is focused on service anomalies, the learning process is agnostic to services. Sethi et al. [18] propose a centralized deep Q-learning solution for intrusion detection in the cloud that is able to identify zero-day attacks with low false-positive rate. These solutions have high computation cost that comes with training over all the captured data, making them computationally-heavy for the resource-constrained edge. In contrast, by focusing a learning task on a service,

using only the service's data, both the volume of needed data for training and number of training cycles can be tailored to a service's needs. This allows flexibility in tuning resource allocation based on needs and capacities.

The propositions of [8, 9] focus on service-based anomaly detection. Zuo et al. [8] propose microservice-based anomaly detection system using a spatio-temporal deep learning for temporal service logs and spatial queries. Zoppi et al. [9] propose a self-adaptive online learning method combining *Historical Checker* and *Statistical Predictor and Safety Margin* algorithms for anomaly detection in dynamic service-oriented systems. While their system shows improvement over evaluated alternatives, it is unclear how well the system performs in detecting anomalies of individual services, nor how the system parameters are tuned towards individual services. Furthermore, these models remain central without incorporating knowledge from different operational sites.

Orthogonally, federated learning solutions have been proposed for anomaly detection [10, 11, 19–21]. Preuveneers et al. [10] propose a deep federated learning solution that incorporates the use of blockchain to maintain an immutable record of benchmark copies of the models as a protection mechanism against adversarial poisoning attacks. Shengjie et al. [11] present an AI-enabled anomaly detection mechanism for fog environments. The proposed framework follows a bottom-up approachÂ based on three components: Fog-enabled infrastructure, Fog-enabled AI, and threat intelligence. The framework combines both local and global parameters to build a comprehensive AI framework. However, it requires extensive communication between Â different components. Amangele et al. [19] provide a hierarchical learning solution for anomaly detection in software defined networks (SDN); whereby, a lightweight model is co-located with SDN switches to facilitate lightweight detection of anomalies, which is then pushed to a central model co-located with the SDN controller to provide a more comprehensive analysis of suspicious traffic. Liu et al. propose in [20] an on-device federated learning solution with aim of providing collaborative training on an anomaly detection model as a way of improving the generalization abilities. Hellander et al. propose in [21] Fedn, a federation framework for management and control of federated learning deployments. The aim is to provide a production-level, resilient, federation system.

Although these federated solutions enable collaborative learning between different sites, such as in a fog ecosystem, they are service-agnostic. This results in sub-optimal use of computation resources as a result of the "un-focused" learning approach while missing pattern details of individual services. In comparison with the aforementioned efforts, the proposed solution in this article offers a focused approach to identify anomalies in network traffic. By working at the level of services, resources and learning tasks can be tailored to the service traffic and needs. Aside from the higher flexibility in managing resources, having focused learning reduces the chances of false alarms. In the following section, we describe the service-based federated deep reinforcement learning solution of this work.

3 Proposed solution

We propose a service-based federated anomaly detection solution, using federated deep reinforcement learning (as illustrated in Fig. 1).

3.1 Preliminary on the fog ecosystem

We assume a fog ecosystem that follows the OpenFog reference architecture [22]. The edge of the fog is comprised of a set of nano data centers, defined as *edge nodes*. These have constrained capacity, hence providing limited resource elasticity. Each edge node is co-located with an access gateway, offering a subset of services to a local group of end-devices that belong to an enterprise-level user. Examples of the latter may include (but not limited to) smart building managers, Industry 4.0 and/or a hospital. The set of services may differ, depending on such metrics as local demand and service deployment plans. Consequently, local data observed at each edge node is assumed to affiliate with the services offered by the node. Moving from the edge towards the core, the capacity of a data center (i.e., node) increases while the number

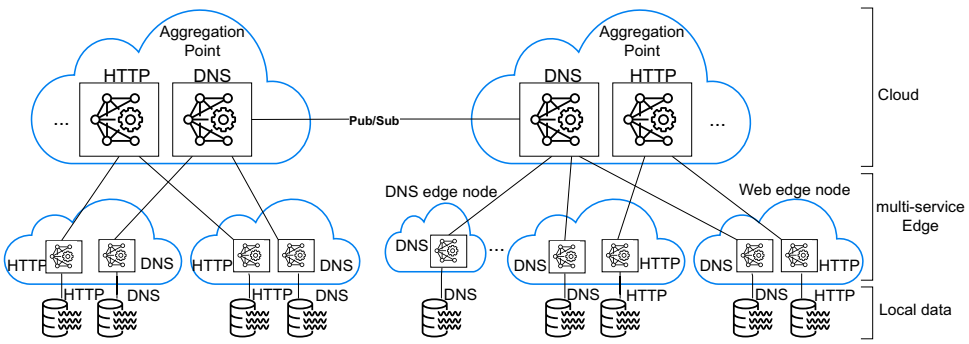


Fig. 1 An overview of the proposed service-based federated deep reinforcement learning system, showing clusters of agents and their aggregation points for DNS and HTTP services. Aggregation points can exchange cluster models following a publish/subscribe approach

of data centers, *nodes*, decreases. Nodes are controlled by one or multiple virtual resource orchestration platforms such as Kubernetes[1] to enable deployment of variety of services, including traffic monitors and anomaly detection applications. The control-plane functions of an application are able to communicate with the resource orchestrator to communicate application requirements.

3.2 Preliminary on deep Q-learning network (DQN)

DQN was introduced by Google in [23, 24] to overcome correlation and overestimation issues in classic Q-learning network. DQN is a system of two deep neural networks (DNN), utilized to select and estimate the value (*reward*) of an action for any given state in a Q-learning network. The system relies on building experience over time and using it to make decisions of higher rewards. For any state, the first DNN selects an action either randomly or following a greedy policy that maximizes rewards, calculated by the network. The selection process is a function of an exploration parameter that decays over training cycles.

Initially, the exploration parameter has a higher weight in the selection function, to allow for exploring the action space and build experience. The second network improves on accumulative knowledge of action rewards through experience replay, selecting random batches from past training records and learning from them. The two networks synchronize periodically, to utilize the Q values learned in the second network in training the first one. DQN lends itself to adaptation for federated learning, as the neural networks aggregation can reflect experience learned by separate, remotely located, instances of DQN. Moreover, the state-action structure of QN makes it suitable for anomaly detection and classification, where a selection of data features can be considered as a set of actions to classify data. In the context of this work, the unique set of labels of benign and malicious flows provides a combination of benign and attack classes that can directly be represented as a set of actions.

3.3 Service-based federated learning clusters

Given the above, we propose an anomaly detection and classification solution that follows a bottom-up learning approach. It consists of a set of federated learning clusters, at least one per service of benign and/or malicious behavior. Each cluster consists of a set of learning agents and an aggregation point. Learning agents are likely to be deployed at edge nodes—though it is not a constraint—in order to limit the propagation of end-user data through the provider's network. Whereas, the aggregation point is likely to be deployed in a node that has reliable connectivity to the majority of nodes where cluster agents are running, to minimize the likelihood of cluster disruption. We assume optimized placement of agents and their aggregation point over fog nodes to be achieved by the fog orchestrator(s), incorporating pre-defined cluster needs. The selection of agents in a cluster and their homogeneity is decided by cluster management entities. For the sake of this work, we assume that cluster's agents are relatively homogeneous in terms of the volume of training and validation data. Furthermore, cluster deployment correlates with the deployment of the service of interest for anomaly detection and classification. The strength of correlation depends on the tightness of the constraint to co-locate an agent with a service runtime. We recognize this challenge but will not further elaborate on it here, as cluster management and orchestration is left for our future work.

Each agent trains a DQN over a flow-based local dataset of a specific service. Each record of the data is labeled either as benign or a type of attack. We assume labeled data to be generated separately, using, for example, unsupervised learning solutions outside the scope of this work. To this end, a *service* can be flexibly defined at the desired granularity of the provider. For instance, a service can be the application-layer protocol, the URL, the IP address, or any other form of identifiers. Here, we define a service by the application layer protocol, represented by the transport-layer port number.

It is worth noting that we recognize the implication of defining HTTP in particular as a single service to be a coarse aggregation of all web-services. A finer-granularity classification of said services is likely to reveal distinctive patterns. However, due to space limitation here and given that our work is the first to introduce service-based anomaly detection, we tolerate this coarse aggregation. Future work will focus on disentangling the behavior of distinct classes of web-services.

3.4 Cluster model

Here, we describe the mathematical model of a learning cluster and the variables utilized in the remainder of the paper. For a service i, a learning cluster is defined as c. It consists of a set of learning agents $\mathcal{L}$ that run a service-tunable DQN, and an aggregation point that runs FDQN. Each flow of service i is defined as $s \in \mathcal{S}_l^i$, with $\mathcal{S}_l^i$, $|\mathcal{S}_l^i| = S_l^i$ being the set of flows (states) analyzed by agent $l \in \mathcal{L}$. The action set of i is defined as $\mathcal{A}^i$, $|\mathcal{A}^i| = A^i$. Notice that the action set is unified across all agents of a cluster. This homogeneity is required in order to enable accurate aggregation of local DQN models into a cluster counterpart. The replay buffer is defined as *athcalD*, and the reward matrix per service $\mathcal{R}^i$ as:

[1] https://kubernetes.io/

$$\mathcal{R}^i = \begin{cases} -\rho & \text{if } (a'_s > 0 \wedge a_s < 0) \vee (a'_s < 0 \wedge a_s > 0) \\ +\rho & \text{if } (a'_s > 0 \wedge a_s > 0) \vee (a'_s = 0 \wedge a_s = 0) \\ +\sigma, \sigma > \rho & \text{if } (a'_s > 0 \wedge a_s > 0) \wedge (a'_s = a_s) \end{cases} \tag{1}$$

where ρ and σ are predefined reward values. Additional hyper parameters of DQN are defined as: $\mu > 0$ the replay buffer size and $0 < \beta < \mu$ the mini batch size.

The Epoch size in number of states per learning cycle is defined as m^i. Hence, the number of Epochs (i.e., rounds for federated learning) is defined as $M^i = \min(\{S^i_l | l \in \mathcal{L}\})/m^i$, where $\min(\{S^i_l | l \in \mathcal{L}\})$ is the smallest dataset in the cluster.

3.5 An agent: anomaly detection and classification with DQN

Each agent runs a DQN system of two fully-connected deep neural networks (DNN). The number of layers and the number of neurons per layer are tunable per service, with some services requiring shallower and/or smaller networks than others. In this work, each DNN is set to be of 6 layers, with 30 neurons at each of the hidden layers. This is to ensure sufficient depth and breadth for flow analysis, as well as a leveled-field for comparison across services. An action set $\mathcal{A}^i$ that includes unique labels of all flows in the service dataset is provided to DQN. The label set includes 0 for benign flows and $a \in 1, 2, ... \in \mathcal{A}^i$ for attack classes with the value of a representing the attack class. Recall that services may have different classes of attacks and hence the action set is defined on a per-service basis. For each state $s \in \mathcal{S}^i_l$ (i.e., data flow), DQN selects an action $a'_s \in \mathcal{A}^i$ (i.e., flow label) with a predictive reward, calculated as described earlier in Section 3.2 for a given exploration parameter ϵ of a decay rate α.

The reward is chosen from the matrix defined by (1), determined by comparing the selected label a'_s with the actual counterpart a_s. If the selected label $a'_s > 0$ but the actual one $a_s = 0$ or vice versa (i.e., false-positive or false-negative), the reward is of negative value. In contrast, if both $a'_s > 0$ and $a_s > 0$ or $a'_s, a_s = 0$ (i.e., true-positive or true-negative), the reward is of positive value. Moreover, if the true-positive is classified by a known class of attacks (i.e., $a'_s, a_s > 0, a'_s = a_s$), an even higher positive reward is given.

The DQN of an agent is illustrated in Algorithm 5. By the end of each training cycle $t \leq M^i$, every agent would have accumulated a total reward, $\mathcal{R}^i_{l,t}$ for all actions taken by their DQN over the set of states used in the cycle. This reward is counted as a *score* of the model quality at the cycle. The model checkpoint at the end of the cycle, $DQN_{l,t}$, is then sent to the aggregation point to contribute in generating an update of the cluster model, $DQN_{c,t}$.

Algorithm 1 Agent: DQN

Given: service i, agent l, dataset $\mathcal{S}^i_l$, $S^i_l = |\mathcal{S}^i_l|$, actions $\mathcal{A}^i, m^i, \beta, \mu, \mathcal{R}^i$
Initialize: $\mathcal{R}^i_{l,t} \leftarrow 0$
Send mS^i_l to aggregation point
Receive from the aggregation point: $(t, \epsilon_t, DQN_{c,t})$
if $t = 0$ **then**
  Initialize: $DQN_{l,0}$ with random weights θ, $\mathcal{D} \leftarrow \{\emptyset\}, \mathcal{S}^i_{l,0} \leftarrow \mathcal{S}^i_l$
else
  $\chi_t \leftarrow DQN_{c,t} \neq \emptyset$
  $DQN_{l,t} \leftarrow (\chi_t DQN_{c,t}) \vee (\neg\chi_t DQN_{l,t-1})$
  $\mathcal{S}^i_{l,t} \leftarrow Random.Sample(\mathcal{S}^i_l, m^i)$
  $\mathcal{S}^i_l \leftarrow \mathcal{S}^i_l \setminus \mathcal{S}^i_{l,t}$
end if
/* Learning phase
for $s \in S^i_{l,t}$ **do**
  if $\epsilon > RND(\mathbb{N}^+)$ **then**
    $a'_s \leftarrow Random(\mathcal{A}^i, 1)$
  else
    $a'_s \leftarrow \max_a Q(s, a, \theta)$
  end if
  $r_s \leftarrow Compare(a'_s, a_s, \mathcal{R}^i)$
  $\mathcal{R}^i_{l,t} \leftarrow \mathcal{R}^i_{l,t} + r_s$
  Store transition: $(s, a'_s, r_s, s_{next}) \rightarrow \mathcal{D}$
  if $|\mathcal{D}| \geq \mu$ **then**
    $Experience.Replay(\mathcal{D}, \mu)$
  end if
  Send $(DQN_{l,t+1}, \mathcal{R}^i_{l,t+1})$ to the aggregation point
end for

3.6 An aggregation point: federated DQN (FDQN)

Agents share their trained models and score with the aggregation point, which utilize them to calculate a cluster-wide score of the cycle, $\mathcal{R}^i_{c,t}$. The function to calculate the cluster score may differ for different clusters; here, we apply a simple average function defined as:

$$\text{ClusterScore} = \frac{\sum_{l \in \mathcal{L}} \mathcal{R}^i_{l,t}}{|\mathcal{L}|} \tag{2}$$

If the new score is higher than the current best score, the aggregation point applies an algorithm such as FedAvg [25] to aggregate local models into a cluster counterpart. The selection of model aggregation function depends on the service and the cluster; FedAvg is applied here merely for its simplicity. It is realized by calculating the average of every layer in each of the two neural networks of all models. Consequently, each of the two neural networks of the cluster model is the result of averaging layers of the same network in the local models. The cluster model is disseminated back to the cluster's agents for use in the next cycle. Alternatively, if the new cluster score is lower or the same as the last best one, agents are simply notified to continue training using their current models (i.e., no cluster model is disseminated). The FDQN of the aggregation point is summarized in Algorithm 2.

Algorithm 2 Aggregation Point: FDQN

Given: service i, set of learning agents $\mathcal{L}$, actions $\mathcal{A}^i$
Initialize: $t \leftarrow 0, \mathcal{R}^i_{c,0} \leftarrow \emptyset, \epsilon_0, \alpha, \beta, \mu, m^i, \mathcal{R}^i$
$M^i \leftarrow \min(\{S^i_l \mid l \in \mathcal{L}\})/m^i$
Send $(t, \epsilon_0, \emptyset)$ to each agent in $\mathcal{L}$
while $t \leq M^i$ **do**
  $t \leftarrow t + 1$
  Wait until all local models and scores $\{(DQN_{l,t}, \mathcal{R}^i_{l,t}) \mid l \in \mathcal{L}\}$ are received
  $\mathcal{R}^i_{c,t} \leftarrow ClusterScore(\{\mathcal{R}^i_{l,t} \mid l \in \mathcal{L}\})$
  if $\mathcal{R}^i_{c,t} > \mathcal{R}^i_{c,t-1}$ **then**
    $DQN_{c,t} \leftarrow Aggregate(\{DQN_{l,t} \mid l \in \mathcal{L}\})$
  else
    $DQN_{c,t} \leftarrow \emptyset$
  end if
  $\epsilon_t \leftarrow \epsilon_{t-1} * (1 - \alpha)$
  Send $(t, \epsilon_t, DQN_{c,t})$ to all $l \in \mathcal{L}$
end while

3.7 Inter-cluster model exchange

An aggregation point may publish updates of its cluster model to other clusters. This is foreseen to enable clusters facing shortage in data supply to compensate with models trained elsewhere in the ecosystem. A published model may be associated with metadata specifying such information as: the service of focus, the model score, and the hyper parameters used in training. An aggregation point may further be interested in external cluster models of specific metadata, such as the service of focus and model score. If a published model of matching specifications exists, the aggregation point subscribes to it to receive model updates. The latter can be treated differently; here, we assume an external model will merely be considered as an added "local" model with the publishing cluster perceived as an "agent" that supplies model updates but not necessarily receive cluster updates.

3.8 Pathway to realization

The proposed system can be realized by exploiting a combination of suitable functionalities of federated learning frameworks such as Fedn [21]; fog orchestration platforms such as Kubernetes; and principles of service resolution from information-centric networking research such as [26]. For instance, using Kubernetes a HTTP-based federated learning cluster X can be defined as a set of connected micro-*services*, one for the aggregation point and $L = |\mathcal{L}|$ for the agents. The microservice used by the aggregation point to publish the cluster model can be in the form, CModel.ap.X.HTTP.com, exposed to all agents of the cluster and potentially to other clusters. Meanwhile, the microservice used by agent *l* to share their local model can be in the form: LModel.l.X.HTTP.com and published in return to the aggregation point. The aggregation point may also subscribe to the cluster model published by remote cluster *Y*, CModel.ap.Y.HTTP.com to achieve the capability described earlier in Section 3.7. Notably, the publish/subscribe logic is not directly supported by existing

platforms; hence, it is part of our future work. To manage and control the operation of a cluster, Fedn can be utilized with suitable modifications.

4 Evaluation

We utilize the UNSW-NB15 [13] and CIC-IDS2018 [14] public datasets to evaluate the performance and cost of the proposed solution and analyze the sensitivity to exploration parameters. Performance is evaluated for two scenarios: with and without inter-cluster model exchange. Notably, the evaluation using CIC-IDS2018 dataset along with solution performance given inter-cluster model exchange are extensions of this work from the earlier baseline of [12]. We use the aforementioned datasets because they provide less noisy data, with sufficient number of flows per-service and per-attack. The UNSW-NB15 dataset is comprised of four sets, providing a total of 2,540,047 records with 12 distinct services each monitored for 9 classes of attacks. The CIC-ID2018 dataset is comprised of ten sets, providing a total of 5,538,479 records with 8 major services each monitored for 13 classes of attacks.

The datasets are summarized in Table 1 for the most commonly targeted services, constituting ≈94.5% of service records in UNSW-NB15 dataset and ≈90% of records in CIC-IDS2018 counterpart. Noteworthy is that attack flows of individual services contain a subset (i.e., not all) of the total attacks. For instance, DNS in UNSW-NB15 dataset has 4 out of 9 attacks with a domination of the "Generic" class. This means that DNS records overall are in binary state of either *benign* or not, with no clear indication of the type of attack. The same binary status is observed for DNS in CIC-IDS2018 dataset, with one attack class recorded as "Infiltration."

Table 1 Summary of service-based flows in the UNSW-NB15 and CIC-IDS2018 datasets

Service	Attacks	Attack flows	Benign flows
UNSW-NB15			
DNS	Generic	209,809	588,669
	Exploits	275	
	DoS	174	
	Reconnaissance	35	
HTTP	Exploits	11,751	206,966
	Reconnaissance	2181	
	DoS	1782	
	Generic	1726	
	Fuzzers	1456	
	Analysis	616	
	Worms	153	
	Backdoors	67	
FTP-data	Exploits	1876	123,893
	DoS	14	
Email (SMTP + POP3)	Exploits	5597	76,660
	DoS	489	
	Generic	419	
	Reconnaissance	7	
	Analysis	6	
FTP	Exploits	2115	46,075
	Fuzzers	804	
	DoS	73	
	Generic	16	
	Backdoor	7	
CIC-IDS2018			
DNS	Infiltration	25,798	1,376,386
HTTP	DDOS HOIC	686,012	800,439
	DoS Hulk	461,912	
	DoS GoldenEye	41,508	
	DoS Slowloris	10,990	
	Infiltration	2976	
	DDOS LOIC-UDP	1730	
	Brute Force-Web	460	
	Brute Force-XSS	225	
	SQL Injection	87	
HTTPS	Infiltration	12,249	647,389

We compare the system performance and cost in service-agnostic and service-based settings, with clusters of 1, 2, and 4 agents. Performance is evaluated with and without the ability to exchange aggregate models across clusters. Notice that a cluster of 1 agent is a special case that resembles central learning. A scenario of a service-agnostic cluster of 1 agent without inter-cluster model exchange is positioned as a baseline representation of state-of-the-art central anomaly detection solutions. Recall that each agent is expected to run in an edge data-center, rather than end-devices; hence, the number of agents does not need to be large. Furthermore, the quality of the solution is not impacted by the number of agents, as the aggregation point only shares a model update when the cluster score improves.

In terms of services, as described earlier, this work defines a service by the application layer protocol. Here, we focus on two examples for which there is a large volume of data in the UNSW and CIC sets, *DNS* and *HTTP*, and compare them with the service-agnostic baseline. The latter involves training and validation over datasets that contain flows of all application layer protocols, as published by the data providers. While, for each of the services, we collate a service-based dataset, where all the flows (benign and malicious) belong to the same application layer protocol.

For performance given inter-cluster model exchange, we assume a baseline scenario: first, each cluster develops its own model through interaction between local agents and their aggregation point, as described in Sections 3.3-3.6. This is grounded by the fact that clusters need to build their models first before sharing with peers. Each cluster then publishes their model and subscribes to an external one, randomly selected from the pool of publications under the same service and given the same set of actions (i.e., attack classes). The similarity is required here to ensure straightforward incorporation of the external model in the aggregation process. Following reception of the external model, the aggregation point will include it as an additional—immutable—input to the FQDN, i.e., the number of input models to the algorithm will increase to $\mathcal{L}+1$. Notably, immutability here means the external model will not be further trained by the subscriber cluster, but merely included in the aggregation phase. This is a deliberate choice, as re-training the external model is associated with a range of open research questions that cannot be adequately addressed here.

We analyze three key performance indicators (KPIs): F1-score, Accuracy and Attack Matching. The first two reflect the detection precision and accuracy, respectively, while the third indicates accurate matching of the anomaly to an attack class. The latter constitutes a classification of the anomaly within known classes of attacks. Notably, for novel attacks, an anomaly might still be identified but not necessarily classified. For performance comparison between scenarios with/without inter-cluster model exchange, we calculate the relative change of each KPI per validation round, t', following the formula:

$$\Delta KPI_{t'} = \frac{KPI^1_{t'} - KPI^0_{t'}}{KPI^0_{t'}} \tag{3}$$

where $KPI^0_{t'}$ is the KPI's value given no inter-cluster model exchange, and $KPI^1_{t'}$ is the value when an external model is incorporated. For sensitivity analysis, we focus on the impact of varying the exploration parameter and its decay rate on model performance. For cost, we analyze the volume of data supply per agent and the number of Epochs per cluster. The first is indicative of data and storage resource demand, while the latter is indicative of compute resource demand. Together, they provide a preliminary evaluation of the cost.

All analyses are shown for the validation phase of trained models. Each dataset is utilized once in validation and another in training. Each experiment involved a unique pair of different sets for training and validation. To this end, we split, curate, and normalize each dataset before using it in training or validation. First, each dataset is split by port number and/or service name into a service-based subset. The latter as well as the original service-agnostic sets are then curated by removing biasing features of: IP addresses, port numbers, transport protocol, and state.

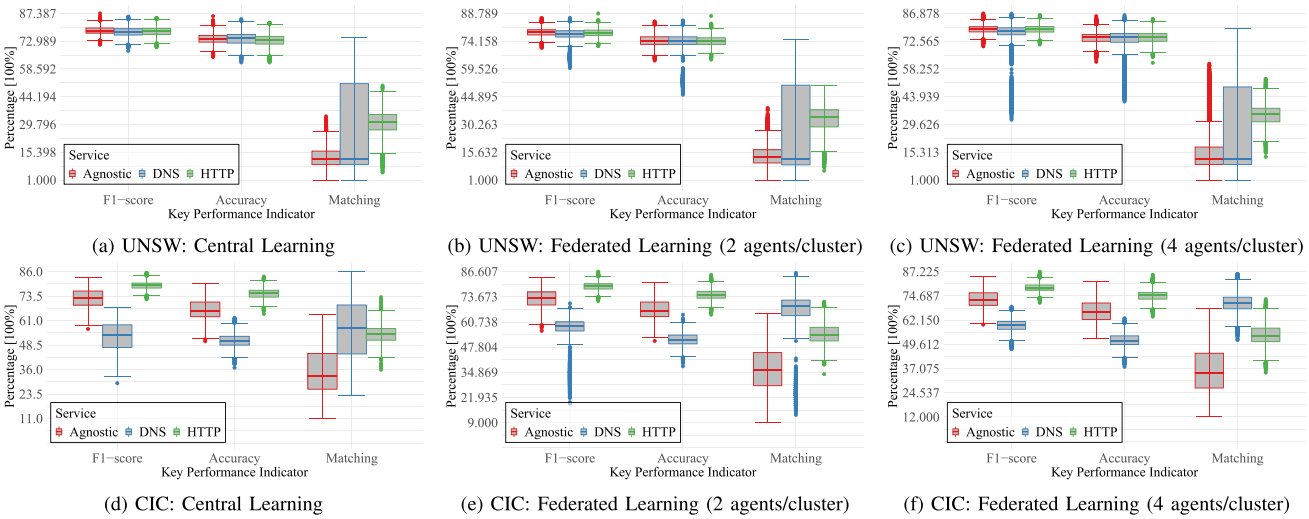
To emulate federated clusters, each dataset is randomly split into subsets equal to the number of agents in the cluster. This means that as the number of agents per cluster grows, the volume of training-validation data per agent shrinks. This assists in illustrating the impact of reduced data supply. A unique subset is provided to each agent for training, while another is provided for validation. Notably, although the email service (i.e., SMTP and POP3) is of interest for analysis, it has been omitted here. Due to space limits and given that the UNSW-NB15 and CIC-IDS2018 datasets have a challenging heterogeneity of attack classes (i.e., actions set). The latter requires more advanced data engineering, beyond a mere random split, that we consider in future work.

The proposed DQN has been implemented using PyTorch and evaluated with hyper-parameters: Epoch size $m^i = 200$ records, replay buffer size $\mu = 10000$ records, and batch size $\beta = 32$ records. The ratio of attack/benign flows is 0.5. The reward values are set as $\rho = 1$ and $\sigma = +2$. The full FDQN system is evaluated over a GPU server of 2 × NVIDIA P100 12 GB cards.

4.1 Performance without inter-cluster model exchange

4.1.1 UNSW-NB15

Figures 2 a–c present the performance results of the proposed solution, over the UNSW-NB15 dataset. From a



(a) UNSW: Central Learning (b) UNSW: Federated Learning (2 agents/cluster) (c) UNSW: Federated Learning (4 agents/cluster)

(d) CIC: Central Learning (e) CIC: Federated Learning (2 agents/cluster) (f) CIC: Federated Learning (4 agents/cluster)

Fig. 2 Validation performance of UNSW and CIC datasets without inter-cluster model exchange for service-agnostic learning and service-based counterpart, for HTTP and DNS services. Learning systems are central (baseline) and federated, with clusters of {2,4} agents per cluster. The exploration parameter $\epsilon = 0.5$ with a decay rate $\alpha = 10^{-.5}$

service perspective, the detection accuracy and precision of service-based learning are comparable to the service-agnostic baseline. HTTP shows a marginal improvement, whereas DNS exhibits higher variation of outliers. This illustrates that service-based learning can achieve similar accuracy per service with smaller datasets. It also reveals the distinct pattern of each service and its effect in agnostic training. Here, DNS has a higher variation than HTTP, yet when considered together in the agnostic baseline, HTTP has a visible impact in masquerading the variation of DNS.

In terms of attack classification, both HTTP and DNS show a significantly higher rate of matching ≈25–35% and ≈10.5–53% for HTTP and DNS, respectively, compared to ≈10.5–25% for the agnostic baseline. Notably, DNS has a wide range between the 1st and 3rd quartiles, compared to HTTP which has a wider range of whiskers. This is caused by the "binary" nature of attack classes in DNS data shown earlier in Table 1, showing the majority of DNS attacks have not been actually classified. Some of these attacks may belong to classes of much fewer records, yet as the dominant class is "Generic," a match is not detected. At the same time, the high value of the 3rd quartile and the maximum indicate that although DNS data classification is not sufficiently granular; the set of attacks is small or has a high similarity.

Moving from central to federated learning with clusters of 2 and 4 agents, average overall accuracy is maintained to similar levels. Albeit, the span of outliers in DNS accuracy is widened by ≈15–17% in clusters of 2 and 4 agents, respectively. The classification rate per service, however, improved when moving to federation clusters of 2 and 4 agents. HTTP exhibits the most significant improvement as the average increases by ≈5% while the minimum and 1st quarterlies increase by ≈15%. This is caused by the federation approach of the FDQN, where a cluster model is only disseminated back if the average score improves. This allows agents to train a cluster model sufficiently such that variation of DQN weights across agents is reduced. This in turn reduces the smoothing impact of weight aggregation in the cluster model.

4.1.2 CIC-IDS2018

Figures 2 d–f show the solution performance over the CIC-IDS2018 dataset. From a service perspective, the results considerably differ from that of the UNSW-NB15 counterpart. HTTP shows the best performance with the highest average F1-score of ≈80% and average accuracy of ≈77%. The attack matching rate is an average of ≈55%, with a negligible variation of ≈1–2% when increasing the number of learning agents. Added to that, the variation between the first and third quartiles is small at ≈2–3%. In contrast, DNS results are the worst with the lowest average F1-score of ≈55% and accuracy of ≈50%. Although, the average rate of attack classificationis high at ≈58%, the variation span between the 1st and 3rd quartiles is large ≈25–27%.

This illustrates the impact of data size and distribution on the quality of detection and classification. In the case of HTTP, the volume of data per attack class for the three most recorded attacks is sufficiently large for DQN to learn the attacks' patterns. Added to that, the difference in volume between benign and malicious data is small, which yields a balance in the selected batch of data for each learning epoch.

In contrast, DNS has a significantly low number of attack records, associated with the Infiltration attack. Added to

that, infiltration records generally have a less distinguished pattern from benign counterparts, as the actual attack traffic is primarily for reconnaissance purpose with the number of packets per flow and average packet size being comparable to that of benign flows. This results in insufficient distribution of attack patterns, compared to a diverse distribution of benign patterns. This in turn leads not only to high false positive but also high false negative. The latter is the key driver of the low F1-score and accuracy. Moving towards a federated learning system with increasing number of agents improves the F1-score (i.e., model precision and recall). Because, splitting the data into smaller subsets—for increasing number of agent—reduces the pattern-diversity in each subset, particularly for benign records. This drives the rate of false negative lower, improving F1-score and the matching rate.

4.1.3 CIC-IDS2018 HTTP attacks

Given the large number of HTTP attack flows in the CIC dataset, here we analyze the effect of segregating them by subsets of attacks and benign traffic. The aim is to draw insights on the impact of focusing the learning experience by reducing the mixture of attacks in a dataset. To achieve the above, we split HTTP dataset into three subsets, corresponding to attacks: *DDoS-HOIC*, *DoS-Hulk* and *Others* (i.e., all the remaining HTTP attacks), and evaluate the relative percentage difference in each of the KPIs. Notably, for DDoS-HOIC and DoS-Hulk, we omit the results of the matching rate as there is only one class of attacks per data subset. The results were collated for clusters of {1,2,4} agents; however, we found that difference in cluster size only results in marginal ≈0.01–0.05% variation in results. Hence, Table 2 presents the five-qauntiles of the relative percentage difference results drawn from clusters of 1 agent as a sample of any of the cluster sizes.

Overall, the results indicate a negative effect on performance when segregating these attacks into separate clusters. The value of F1-score drops by a range of ≈2.1–13% across the 5–75% qauntiles of DDoS-HOIC attack, while for DoS-Hulk, the dropage range is ≈2–12.5%. For the subset of remaining attacks, the degradation range of F1-score is ≈0.77–6.8%, for the 5–50% quantiles, while the 75% qauntile shows a positive change of 1.74%. The 95% generally shows a positive change across clusters, which illustrate a benefit for a minor number of validation batches. This is caused by a consistent increase in the number of false negatives in these clusters as opposite to general HTTP clusters. The increase in false negatives is primarily met by a decrease in true positives and true negatives. This in turn causes a comparable degradation in the Accuracy.

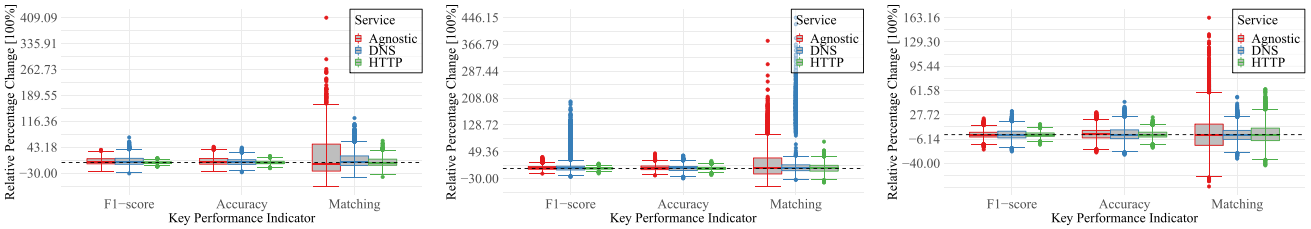
The above illustrates a trade-off between "focus-of-context" and "level of knowledge" learned by a model. By over-narrowing the mixture of "services," a negative overfitting effect may occur over some services and their datasets. Here, maintaining the different classes of DoS/DDoS attacks in one HTTP dataset has a constructive effect in providing pattern diversity, yielding a better differentiation of benign traffic.

4.2 Performance with inter-cluster model exchange

This section evaluates the relative percentage change of KPI value per validation round, when incorporating an external model as calculated by Eq. (3). A negative value indicates a performance degradation for the validation round, while a positive value indicates a performance improvement. Figure 3 shows the relative percentage change for the CIC dataset. Overall, each KPI has positive and negative observations, with the average change remaining at the zero threshold. However, the quartiles and whiskers illustrate a more significant variation, particularly for service-agnostic and DNS clusters. For F1-score and Accuracy, the worst negative value is recorded at ≈30% for DNS clusters of 1 and 4 agents, as opposite to the best positive value of ≈45%, for the same settings. For clusters of 2 agents, DNS has a large number of positive outliers with a maximum of ≈208%. The Matching KPI is the most affected by incorporating an external model. The worst degradation in matching rate is recorded at ≈70% for service-agnostic clusters of 4 agents, while the best improvement is recorded at ≈163% for the same clusters. The highest improvement is ≈446% for DNS clusters of 2 agents. HTTP matching rate varied too in clusters of 4 agents, with the worst degradation recorded at ≈40% and the best improvement at ≈62%.

Table 2 Relative percentage difference per KPI per validation round for all validation rounds of CIC HTTP validation sets

Attack	KPI	5%	25%	50%	75%	95%
DDoS-HOIC	F1-score	−12.90	−8.32	−5.33	−2.11	2.34
DDoS-HOIC	Accuracy	−8.75	−3.43	0	3.55	9.72
DoS-Hulk	F1-score	−12.56	−8.19	−5.17	−1.92	2.74
DoS-Hulk	Accuracy	−8.23	−3.29	0.62	4.05	9.79
Others	F1-score	−6.83	−3.17	−0.77	1.74	4.99
Others	Accuracy	−9.15	−3.97	−0.66	2.77	7.85
Others	Matching	−19.35	−6.89	1.72	10.20	24.55



(a) CIC: Central Learning (b) CIC: Federated Learning (2 agents/cluster) (c) CIC: Federated Learning (4 agents/cluster)

Fig. 3 Percentage change in validation performance of CIC dataset for learning systems with inter-cluster model exchange, relative to no inter-cluster model exchange for service-agnostic learning and service-based counterpart, for HTTP and DNS services. Learning systems are central (baseline) and federated, with clusters of {2,4} agents per cluster. The exploration parameter $\epsilon = 0.5$ with a decay rate $\alpha = 10^{-5}$

Table 3 Percentage difference in attack matching per validation round for all rounds, for learning systems with inter-cluster model exchange relative to systems without inter-cluster model exchange of UNSW validation sets

Agents/cluster	Service	5%	25%	50%	75%	95%
1	Agnostic	−64.00	−26.67	0	46.67	136.36
1	HTTP	−27.50	−7.69	7.40	28.17	90.56
1	DNS	−87.27	−77.16	−8.33	231.77	689.44
2	Agnostic	−61.29	−26.14	0	42.86	153.85
2	HTTP	−28.20	−11.43	2.86	22.69	146.03
2	DNS	−86.54	−58.82	0	42.86	244.44
4	Agnostic	−78.95	−28.57	0	50.00	212.50
4	HTTP	−30.77	−13.54	0	15.15	44.69
4	DNS	−46.15	−14.29	5.77	36.96	470.00

Overall, this shows a positive effect of the external model, particularly for such services as DNS that are lacking sufficient data supply and do not have a large action set. There, the external model brings further knowledge of the data patterns to the cluster counterpart leading to improved performance. Oppositely, the negative results are partially caused by the immutability of the external model, during the training phase. By including the external model in the aggregation process without training it over cluster data, a constant smoothing effect is repeatedly added to the weights of the cluster model. This introduces a randomness factor with an increasing impact, relative to the decay in the exploration parameter ϵ.

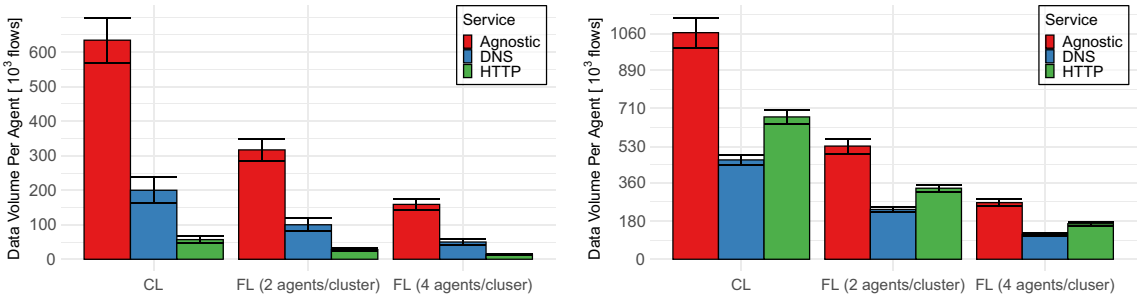
For the UNSW dataset, the relative percentage change of F1-score and Accuracy has been marginal ≈0.1–2%. A more significant change is witnessed in the matching rate, with the results of the five quantiles shown in Table 3.Overall, the results exhibit similar pattern to that seen with the CIC dataset. The 25% quantile shows negative change in observations in the range of 7.69–58.82%, while the 75% counterpart shows a positive change in the range of 15.15–231.77%. Noticeably, in HTTP clusters of 1 and 2 agents, the average change is positive 7.4% and 2.86%, respectively. Meanwhile, the lowest degradation in the 5% quantile is 87.27%, exhibited by DNS clusters of 1 agent. The same set of clusters show the highest improvement of 689.44%, in the 95% qauntile.

Notably, the analysis above is merely an introductory insight into the impact of inter-cluster model exchange. Further analysis of various model exchange and training approaches is needed to develop an advanced understanding of their impact on performance quality. We preserve such analysis for our future work, due to space limitation and to avoid overloading the work here.

4.3 Cost

Figure 4 shows the volume of data supply per agent per service, utilized to achieve the performance shown above in Sections 4.1 and 4.2. The overall volume of data per service varies considerably, with service-focused datasets being smaller than a service-agnostic alternative. This shows flexibility in utilizing constrained storage resources. Orthogonally, the volume of data supply per local agent is reduced by the same factor used for increasing the number of agents per cluster. This shows that the proposed solution can sustain significant reduction in data supply and still maintain steady-level performance.

Figure 5 shows the maximum number of training epochs (rounds for federated learning), M^i, per cluster given an epoch size of 200 flows. Clusters range from central learning with 1 agent to federated learning of



(a) UNSW (b) CIC

Fig. 4 Data volume per agent in number of flows, used in central learning (CL) of 1 agent and federated learning (FL) of 2 and 4 agents per cluster for UNSW and CIC datasets

{2,4} agents percluster. Notably for time's sake, the actual number of training epochs was capped to 500. Nevertheless, all trained models reached convergence. The number of training epochs directly correlates with the volume of compute resources required to train local DQN models. Having lower volume of noise in service-based datasets allowed for faster convergence, which in turn consumes less compute resources. The saving in resources is realized without compromising performance, as shown above.

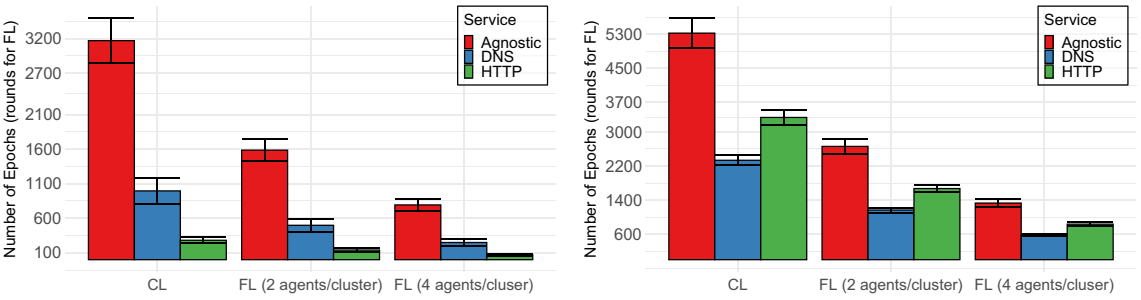
4.4 Sensitivity

Figure 6 shows the impact of varying the exploration parameter and its decay rate on the detection accuracy and matching across services. The results are drawn from federated learning clusters of 4 agents.

4.4.1 UNSW-NB15

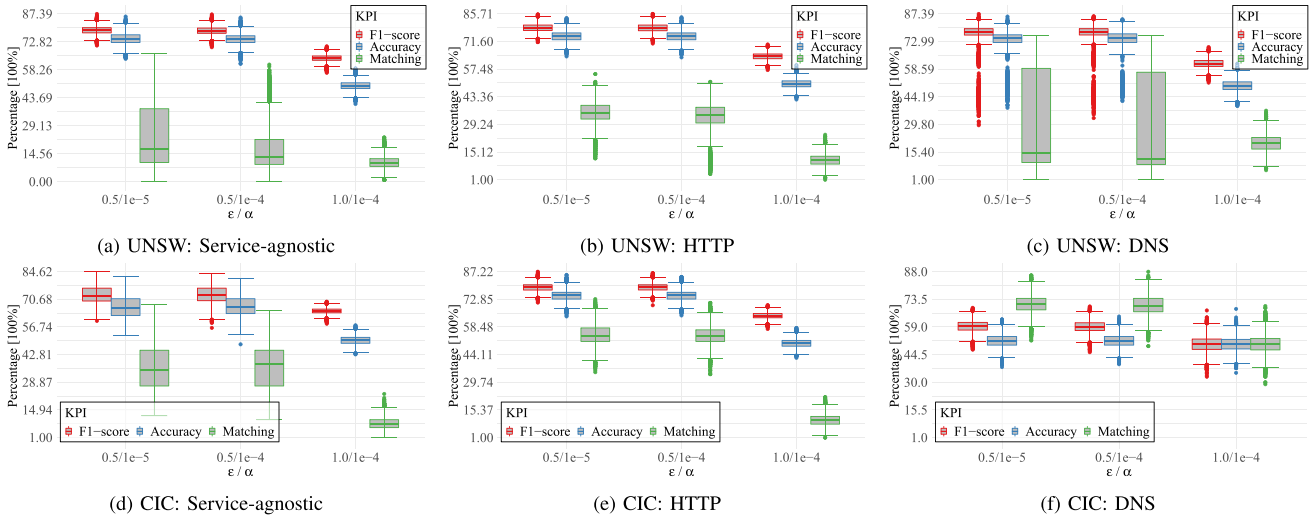
Figures 6 a–c show the results of the UNSW datasets. Overall, a higher value of $\epsilon = 1.0$ yields a performance degradation, despite the fast decay rate. The F1-score drops by ≈15%, while accuracy drops by ≈25%. The matching rate suffers the most, dropping by ≈30–35%. Although, the difference span between matching values reduces here. This is particularly observed for DNS, with a propagative effect in the agnostic baseline. In contrast, the lower end of DNS matching rate improves with the higher ϵ. This shows a non-trivial positive impact of higher randomness, more likely to be deceitful than true (i.e., a match is not really a match). This is due to the binary-like distribution of DNS classes. Higher randomness in action selection means higher likelihood of selecting the "Generic" label and similarly higher chance of matching with the actual label. However, as Generic' does not represent a clear class, the decision is likely not to be an accurate match in reality.

In contrast, a lower $\epsilon = 0.5$ shows improvement of detection accuracy, coupled with better matching rate for HTTP and the upper end of DNS boxes. This is because a lower ϵ allowed for better enforcement of DQN experience during training phase. Consequently, this allowed for taking more informed decisions during validation phase. The change in decay rate α has a lower impact on overall performance, compared to varying ϵ. A slower decay rate $\alpha = 10^{-5}$ has shown to improve the matching rate across board. This is driven by the arising trade-off—during training—between freedom to explore the action space and greediness in



(a) UNSW (b) CIC

Fig. 5 Maximum number of Epochs for central learning (CL) of 1 agent and federated learning (FL) of 2 and 4 agents per cluster for UNSW and CIC datasets



(a) UNSW: Service-agnostic (b) UNSW: HTTP (c) UNSW: DNS

(d) CIC: Service-agnostic (e) CIC: HTTP (f) CIC: DNS

Fig. 6 Performance sensitivity to change of the exploration parameter $\epsilon \in \{0.5,1.0\}$ and its decay rate $\alpha \in \{10^{-4},10^{-5}\}$. This is observed for UNSW and CIC datasets, in service-agnostic and service-based learning, for HTTP and DNS services having federated learning clusters of 4 agents per cluster

selecting maximum-reward decisions. Here, a slower decay of ϵ facilitates abetter exploration of the action space, yielding a better learning. Though, the impact of sensitivity parameters is highly dependent on DQN structure and the datasets. Furthermore, more comprehensive analysis using different datasets and different configurations of DQN are needed to establish a more general view.

4.4.2 CIC-IDS2018

Figures 6 d–f show the sensitivity results of the CIC datasets. For service-agnostic and HTTP clusters, a larger value of the exploration parameter $\epsilon = 1.0$ leads to performance degradation, particularly given the low number of 500 learning rounds (epochs). The average F1-score and Accuracy drop by ≈20%, compared to their values when $\epsilon = 0.5$ and irrespective of the decay rate α. When $\epsilon = 0.5$, slowing the decay rate *lpha* from 10^{-4} to 10^{-5} reduces the average F1-score and Accuracy by ≈5%, yet it increases the min–max matching rate by ≈5%. This shows that a slower decay rate in these cases is allowing the respective DQNs to better learn the attack patterns. DNS KPIs are generally low irrespective of the ϵ and α values, and they only drop by a marginal ≈5% when ϵ varies from 0.5 to 1.0. Here, the sensitivity of a DNS model to the hyper parameters is masked by the more dominant effect of the nature of DNS data in the CIC sets. Recall that the latter have insufficiently low number of DNS infiltration attack flows—compared to benign flows—that do not have distinctive flow patterns from benign small flows. This limits DQN's ability to differentiate between benign and malicious flow, leading to a high number of false negatives. Essentially, DQN of a DNS model acts with a high degree of randomness, where varying ϵ only has a marginal effect on it.

The matching rate, similar to UNSW, is the most affected KPI. For service-agnostic and HTTP clusters, varying ϵ from 0.5 to 1.0 results in ≈40–50% degradation in the matching rate. This is correlated with degradation in the prediction accuracy, as DQNs tend to have a higher degree of randomness and with the large number of attack flows per distinct classes, the likelihood of random decision being accurate drops significantly. For DNS clusters, varying ϵ results in ≈15–20% degradation in the matching rate. When $\epsilon = 0.5$, the matching rate is higher than HTTP because the number of attack classes in DNS is limited to one. Consequently, an accurate detection of a malicious flow automatically leads to accurate matching.

5 Conclusions

This work proposed a service-based federated deep reinforcement learning system, for efficient anomaly detection and classification in fog ecosystems. The solution includes a federated DQN (FDQN), which incorporates a class-based reward matrix and performs score-based model dissemination. The former enables attack classification per service, while the latter allows to reduce model variation across local agents. We evaluated the performance, cost, and sensitivity of the proposed solution for DNS and HTTP, compared to a service-agnostic baseline. Performance results showed that service-based learning maintains detection accuracy, despite the lower volume of data, while improving attack classification by a factor of ≈2. Similarly, the proposed federation approach allows for maintaining the same accuracy as that achieved by central training, with better attack classification. Furthermore, incorporating additional models from peer clusters can lead to

positive improvements in performance quality. While, over-focusing a service definition for a learning cluster can lead to performance degradation due to overfitting. Sensitivity results of the analyzed scenarios revealed that a larger exploration opportunity is not always needed and can lead to performance degradation. The exploration parameter and decay rate are subject to optimizations that address the trade-off between freedom to explore and greediness in maximizing rewards. Addressing this trade-of is part of our future work.

Acknowledgements This work has been partially supported by the Equal Opportunities Fund, Uppsala University, Sweden.

Declarations

Conflict of interest The authors declare no competing interests.

References

1. Hassija V, Chamola V, Saxena V, Jain D, Goyal P, Sikdar B (2019) A survey on iot security: application areas, security threats, and solution architectures. IEEE Access 7:82 721-82 743
2. Radhakrishnan K, Menon RR, Nath HV (2019) A survey of zero-day malware attacks and its detection methodology, in TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON). pp. 533–539
3. Chkirbene Z, Erbad A, Hamila R, Gouissem A, Mohamed A, Hamdi M (2020) Machine learning based cloud computing anomalies detection. IEEE Network 34(6):178–183
4. Bulusu S, Kailkhura B, Li B, Varshney PK, Song D (2020) Anomalous example detection in deep learning: a survey. IEEE Access 8:132 330-132 347
5. Ruff L, Kauffmann JR, Vandermeulen RA, Montavon G, Samek W, Kloft M, Dietterich TG, Müller KR (2021) A unifying review of deep and shallow anomaly detection. Proc IEEE 109(5):756–795. https://doi.org/10.1109/JPROC.2021.3052449
6. D'hooge L, Wauters T, Volckaert B, De Turck F (2020) Inter-dataset generalization strength of supervised machine learning methods for intrusion detection. J Inf Secur Appl 54:102564
7. Nedelkoski S, Cardoso J, Kao O (2019) Anomaly detection and classification using distributed tracing and deep learning, in 2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), pp 241–250
8. Zuo Y, Wu Y, Min G, Huang C, Pei K (2020) An intelligent anomaly detection scheme for micro-services architectures with temporal and spatial data analysis. IEEE Trans Cogn Commun Netw 6(2):548–561
9. Zoppi T, Ceccarelli A, Bondavalli A (2021) Madness: a multi-layer anomaly detection framework for complex dynamic systems. IEEE Trans Dependable Secure Comput 18(2):796–809
10. Preuveneers D, Rimmer V, Tsingenopoulos I, Spooren J, Joosen W, Ilie-Zudor E (2018) Chained anomaly detection models for federated learning: an intrusion detection case study. Appl Sci 8(12):2663. https://doi.org/10.3390/app8122663
11. Xu S, Qian Y, Hu RQ (2019) Data-driven network intelligence for anomaly detection. IEEE Network 33(3):88–95
12. Al-Naday M, Reed M, Dobre V, Toor S, Volckaert B, De Turck F (2023) Service-based federated deep reinforcement learning for anomaly detection in fog ecosystems, in 2023 26th Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), pp 121–128
13. Moustafa N, Slay J (2015) Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set), in 2015 Military Communications and Information Systems Conference (MilCIS), pp. 1–6
14. Sharafaldin I, Lashkari AH, Ghorbani AA (2018) Toward generating a new intrusion detection dataset and intrusion traffic characterization. in International Conference on Information Systems Security and Privacy
15. A realistic cyber defense dataset (cse-cic-ids2018), was accessed on 15–05–2023. [Online]. Available: https://registry.opendata.aws/cse-cic-ids2018
16. Jayasinghe S, Siriwardhana Y, Porambage P, Liyanage M, Ylianttila M (2022) Federated learning based anomaly detection as an enabler for securing network and service management automation in beyond 5g networks, in 2022 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit), pp 345–350
17. Jithish J, Alangot B, Mahalingam N, Yeo KS (2023) Distributed anomaly detection in smart grids: a federated learning-based approach. IEEE Access 11:7157–7179
18. Sethi K, Kumar R, Prajapati N, Bera P (2020) Deep reinforcement learning based intrusion detection system for cloud infrastructure, in 2020 International Conference on COMmunication Systems NETworkS (COMSNETS), pp 1–6
19. Amangele P, Reed MJ, Al-Naday M, Thomos N, Nowak M (2019) Hierarchical machine learning for iot anomaly detection in sdn, in 2019 International Conference on Information Technologies (InfoTech), pp 1–4
20. Liu Y, Garg S, Nie J, Zhang Y, Xiong Z, Kang J, Hossain MS (2020) Deep anomaly detection for time-series data in industrial iot: a communication-efficient on-device federated learning approach. IEEE Int Things J 8(8):6348–6358. https://doi.org/10.1109/JIOT.2020.3011726
21. Ekmefjord M, Ait-Mlouk A, Alawadi S, Åkesson M, Singh P, Spjuth O, Toor S, Hellander A (2022) Scalable federated machine learning with fedn, in 2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid), pp 555–564
22. OCAW Group (2017) Openfog reference architecture for fog computing, Fremont, CA, USA, Tech. Rep. OPFRA001.020817
23. Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, Riedmiller MA (2013) Playing atari with deep reinforcement learning, CoRR, vol. abs/1312.5602. [Online]. Available: http://arxiv.org/abs/1312.5602. Accessed 01-06-2023
24. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G, Petersen S, Beattie C, Sadik A, Antonoglou I, King H, Kumaran D, Wierstra D, Legg S, Hassabis D (2015) Human-level control through deep reinforcement learning. Nature 518(7540):529–533. https://doi.org/10.1038/nature14236. [Online]
25. Nilsson A, Smith S, Ulm G, Gustavsson E, Jirstrand M (2018) A performance evaluation of federated learning algorithms, in Proceedings of the Second Workshop on Distributed Infrastructures for Deep Learning, ser. DIDL '18.New York, NY, USA: Association for Computing Machinery, p 1–8
26. Al-Naday M, Macaluso I (2021) Flexible semantic-based data networking for iot domains, in 2021 IEEE 22nd International Conference on High Performance Switching and Routing (HPSR), pp 1–6