FortisEDoS: A Deep Transfer Learning-empowered Economical Denial of Sustainability Detection Framework for Cloud-Native Network Slicing

Chafika Benzaïd*, Tarik Taleb*, Ashkan Sami†,‡ and Othmane Hireche§
* University of Oulu, Oulu, Finland
† Edinburgh Napier University, Edinburgh, UK
‡ Shiraz University, Shiraz, Iran
§ University of Sciences and Technology Houari Boumediene, Algiers, Algeria
Emails: chafika.benzaid@oulu.fi, tarik.taleb@oulu.fi, sami@shirazu.ac.ir, othmane.hireche@nanoform.com

***Abstract*—Network slicing is envisaged as the key to unlocking revenue growth in 5G and beyond (B5G) networks. However, the dynamic nature of network slicing and the growing sophistication of DDoS attacks rises the menace of reshaping a stealthy DDoS into an Economical Denial of Sustainability (EDoS) attack. EDoS aims at incurring economic damages to service provider due to the increased elastic use of resources. Motivated by the limitations of existing defense solutions, we propose FortisEDoS, a novel framework that aims at enabling elastic B5G services that are impervious to EDoS attacks. FortisEDoS integrates a new deep learning-powered DDoS anomaly detection model, dubbed CG-GRU, that capitalizes on the capabilities of emerging graph and recurrent neural networks in capturing spatio-temporal correlations to accurately discriminate malicious behavior. Furthermore, FortisEDoS leverages transfer learning to effectively defeat EDoS attacks in newly deployed slices by exploiting the knowledge learned in a previously deployed slice. The experimental results demonstrate the superiority of CG-GRU in achieving higher detection performance of more than 92% with lower computation complexity. They show also that transfer learning can yield an attack detection sensitivity of above 91%, while accelerating the training process by at least 61%. Further analysis shows that FortisEDoS exhibits intuitive explainability of its decisions, fostering trust in deep learning-assisted systems.**

***Index Terms*—AI Explainability, Anomaly Detection, Application-layer DDoS, Deep Transfer Learning, Economical Denial of Sustainability (EDoS), Network Slicing, 5G and Beyond Networks (B5G).**

I. Introduction

Network virtualization and softwarization are considered key technological enablers for empowering highly dynamic operation and management of 5G and beyond (B5G) networks. Their joint use is vital for allowing next-generation mobile networks support diversified and flexible deployment scenarios, whereby multiple services/verticals can share the same physical substrate [1]. The concept is commonly referred to as network slicing, which enables multiple virtual networks (i.e., slices) to be created on top of a shared physical infrastructure. Each slice is devised with customized network capabilities to fulfill the performance needs of a specific service type, such as Enhanced Mobile Broadband (eMBB), Ultra Reliable Low Latency Communications (URLLC), Massive Internet of Things (mIoT), Vehicle-to-Everything (V2X), and High-Performance Machine-Type Communications (HMTC) [2].

A network slice instance comprises a set of network functions which are chained and can span across multiple network domains, including the radio access network (RAN), transport network, core network (CN), and edge network. The evolution towards a cloud-native telco architecture is promoted by standardization bodies as a crucial facilitator for supporting network slicing, owing to its inherent advantages of network scalability, elasticity, flexibility, and automation. 3GPP has mandated cloud-native principles in the design of CN network functions [2]. Open RAN (O-RAN) Alliance (https://www.o-ran.org/) is building upon and expanding the 3GPP's functional split of the New Generation RAN (NR-RAN) to enable cloudification of RAN functions. Furthermore, initiatives like SD-Fabric (https://opennetworking.org/sd-fabric/) and TeraFlowSDN (https://tfs.etsi.org/) are paving the way for cloudified control plane functions in transport networks. By embracing cloud-native principles, network functions are designed as loosely coupled micro-services and deployed as scalable workloads on cloud infrastructure.

A key life cycle management operation of network slices is auto-scaling, which consists in dynamically expanding or contracting the capacity of a network slice instance to adapt resources to slice workload to meet the performance desire. Network slicing allows for flexible and efficient utilization of resources with greater cost reduction, thanks to the infrastructure sharing, the dynamic resource provisioning and the auto-scaling feature enabled by network function virtualization, software defined networking and cloud computing [3]. Nevertheless, the auto-scaling capability is a double-edged sword when a (Distributed) Denial of Service – (D)DoS – attack is underway. In fact, the auto-scaling capability can reshape an undetected (D)DoS attack into an Economical Denial of Sustainability (EDoS) attack, which engenders economic damages to service provider due to the increased elastic use of resources as well as performance degradation as a result of shared resource starvation [4]. In network slicing, the undesirable economic and performance impact of EDoS is a critical concern as it may spread beyond the slice under attack, affecting the other slices co-hosted on the same infrastructure [5]. Such adverse impacts of EDoS attack are posing a serious threat to accelerate the B5G-powered

digital transformation of the industry verticals such as media & entertainment [6], automotive [7], industry [8] and energy [9], to name just a few. Thus, providing reliable dynamic resource provisioning that is EDoS attack aware is paramount to reap the benefits of network slicing in enabling profitable beyond 5G services.

Achieving the aforementioned goal is challenging as the recent years have seen a growing trend towards more sophisticated and stealthier DDoS attacks that are targeting the application layer rather than the network layer. According to Cloudflare's DDoS attack trends 2022 reports, the application-layer DDoS attacks have massively spiked in the first quarter and their amount has risen by 72% in the second quarter compared to the same period last year. The reports reveal also that application-layer DDoS attacks targeting HTTP protocol increased by 164% compared to 2021. This trend is also reflected in NETSCOUT's latest DDoS Threat Intelligence Report released in early April 2023, revealing that HTTP(S) application-layer DDoS attacks have spiked by 487% since 2019, with the most significant surge in the second half of 2022. Such trend is owing to the ability of application-layer DDoS attacks to imitate legitimate behavior with low network bandwidth usage, which allows them to bypass typical traffic-based intrusion detection systems [10]. Moreover, NETSCOUT's report highlights a significant increase of 79% in DDoS attacks on the wireless telecommunications industry, primarily propelled by the growing adoption of 5G wireless for residential use. This trend is only going to escalate in the future as 5G penetrates globally. Therefore, without advanced protection measures, the ever-evolving stealthiness of application-layer DDoS attacks coupled with the cloud-native transformation of B5G networks is a serious danger that will foster the prevalence of EDoS attacks.

Although extensive work has been engaged and several solutions have been proposed to counter DDoS attacks, addressing the stealthy application-layer DDoS issue is far from being completely resolved, and even less in 5G and beyond network slicing environment. Existing solutions suffer from a number of limitations, which impedes their efficiency and effectiveness. The complete isolation among slices advocated by resource isolation based approaches (e.g., [11], [12]) may lead to resource usage inefficiency or may not be possible to realize due to lack of strong hardware isolation in the emerging cloud-native platforms [13]. The ability of application-layer DDoS attacks to mimic legitimate traffic endows them with the capacity to elude detection by network traffic analysis based solutions (e.g., [10], [14], [15]). Leveraging resource scaling capability as a mitigation strategy by resource allocation-based methods (e.g., [16]) rises the issue of reshaping an undetected application-layer DDoS attack into an EDoS attack [17]. The emerging anomaly detection approaches (e.g., [18]–[20]) that exploit the potential of Deep Learning (DL) to identify abnormal behavior based on anomalies detected in resource usage and/or service performance metrics are a promising direction to deal with EDoS attack. However, existing anomaly detection approaches only consider temporal dependencies between metrics and/or assume that sufficient amount of historical data is available for training the DL models to recognize normal behavior patterns.

Motivated by the above-discussed limitations of DL-based anomaly detection approaches, the serious economic impacts of EDoS attacks on cloud-native B5G networks, and the limited research in addressing the EDoS issue in 5G network slicing environment, we propose in this paper a novel AI-powered framework that aims to proactively mitigate EDoS attacks against network slicing. The proposed framework, coined FortisEDoS, enables elastic cloud-native network slices providing 5G services on the edge while intelligently safeguarding from malicious resource scaling requests caused by stealthy application-layer DDoS attacks. It is worth noting that the creation and deployment of network slices is beyond the scope of this work. Additionally, and without loss of generality, we consider that at least CN and service-level network functions of a network slice are cloudified. FortisEDoS exploits both temporal and spatial correlations among resource usage and service performance metrics and adopt a dynamic thresholding strategy to foster accurate discrimination of anomalous status of a slice's VNF under application-layer DDoS attack, allowing effective deterring of malicious requests for scaling VNF's resources. FortisEDoS capitalizes on the promising capabilities of emerging DL techniques, particularly Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN) and Graph Neural Networks (GNN), in uncovering complex patterns to capture the spatio-temporal dependencies. Furthermore, FortisEDoS leverages the concept of transfer learning to facilitate the transfer of knowledge regarding the normal VNF's behavior acquired in a previously deployed slice to a newly deployed slice, empowering effective identification of anomalous VNF's status caused by application-layer DDoS attacks even when representative historical data of normal behavior are scare. To the best of our knowledge, this is the first contribution of deep transfer learning in tackling EDoS attacks against network slicing.

The key contributions of this paper can be summarized as follows:

- We propose FortisEDoS, a novel framework that integrates a deep transfer learning model to empower highly elastic and resilient B5G services, deployed as slices, that can deliver superior quality of experience (QoE) while being impervious to EDoS attacks.
- We build a new DL-powered forecast-based DDoS anomaly detection model, coined CG-GRU, that allows to discriminate anomalous VNF's status caused by an application-layer DDoS attack in order to prevent its reshaping into EDoS attack. The model can effectively identify anomalous resource usage and performance metrics of VNFs by measuring and comparing against a dynamic threshold the error between the observed metric's values and the predicted ones. CG-GRU combines the advantages of different deep neural network algorithms to provide both feature extraction and forecasting capabilities. Specifically, CNN, Graph Attention (GAT) and Gated Recurrent Unit (GRU) algorithms are used to extract relevant local features, spatial correlations and time dependencies among VNF's metrics, respectively, and Multi-Layer Perceptron (MLP) algorithm is utilized

for enabling forecasting capability.

- We demonstrate how forecasting error heatmaps, created using predictions generated by CG-GRU, can be used to enable the explainability of decisions made by FortisEDoS about the legitimacy or maliciousness of the observed metrics of a slice's VNF, allowing to foster trustworthiness in its decisions.
- We introduce the transfer learning concept into CG-GRU to quickly and effectively defeat EDoS attacks in newly deployed slices. This is done by embedding the knowledge about normal VNF's behavior learned by a CG-GRU model associated to a VNF from a previously deployed slice into the model of the new slice's VNF.
- We develop an experimental testbed based on the cloud-native platform Kubernetes to evaluate the effectiveness of FortisEDoS in preventing malicious resource scaling operations induced by application-layer DDoS attacks launched against a virtualized Content Delivery Network (vCDN) service. The experimental results demonstrate the superiority of CG-GRU model in achieving high overall attack detection performance with low computation/storage complexity, compared to baseline methods. They show also the attack detection effectiveness and the computational efficiency of the transfer learning-powered CG-GRU model.

The remainder of the paper is organized as follows. Section II discusses previous relevant works. Section III introduces the proposed FortisEDoS framework, delineating its architecture and the design of the deep transfer learning-based DDoS anomaly detection model. Section IV describes the experimental setup, detailing the implemented dataset generation and model's hyper-parameter tuning processes, and provides a comprehensive analysis of the performance results. Finally, Section V concludes the paper and highlights future research directions.

For ease of reference, Table I summarizes the most important abbreviations (upper part) and notations (lower part) used in this paper.

TABLE I: Main nomenclature and notations used in the paper.

Notation	Description
B5G	5G and Beyond
CNF	Cloud-native Network Function
CNN	Convolutional Neural Network
Conv1D	one-dimensional Convolutional
DDoS	Distributed Denial of Service
DL	Deep Learning
EDoS	Economical Denial of Sustainability
GAN	Generative Adversarial Network
GAT	Graph Attention Network
GNN	Graph Neural Network
GRU	Gated Recurrent Unit
LSTM	Long Short-Term Memory
MLP	Multi-Layer Perceptron
RNN	Recurrent Neural Network
SLA	Service Level Agreement
vCDN	virtual Content Delivery Network
VNF	Virtual Network Function
$\mathcal{S}$	The set of network slices
$\mathcal{V}_i$	The set of VNFs in slice S_i
f_j^i	The VNF j in slice S_i
$\mathcal{X}$	A multivariate time series
d	The number of metrics (features) of a VNF
$\boldsymbol{x}^{(t)}$	A d-dimensional vector representing the metrics data observed on a VNF at time step t
$x_i^{(t)}, \hat{x}_i^{(t)}$	The actual and forecast value of the i-th VNF's metric at time step t
$\bar{x}_i^{(t)}$	The normalized value of a metric $x_i^{(t)}$
w	The size of the look-back sliding window
$\mathfrak{h}$	The length of the forecast horizon
τ	The step length
$\mathcal{D}_{train}$	The training dataset
$\mathcal{D}_{val}$	The validation dataset
$e_i^{(t)}$	The forecast error of the i-th VNF's metric at time t
$e^{(t)}$	The global forecast error of a VNF at time t
$e_s^{(t)}$	The smoothed global forecast error of a VNF at time t
ξ_s	The smoothed global forecast errors of a VNF at time t for the previous w time steps
ε	The dynamic anomaly detection threshold
$\mu(.)$, $\delta(.)$	Mean and standar deviation functions

II. Related work

Despite the research efforts devoted to deal with the DDoS attacks in general and stealthy application-layer DDoS in particular [21]–[24], very few contributions have focused on tackling the issue in 5G network slicing environment. In what follows, we review the main defense approaches proposed in the literature to handle application-layer DDoS attacks, considering either approaches that are specifically devised or that may apply for 5G systems. Table II summarizes the investigated defense solutions, highlighting the adopted methodology, ML techniques used, and key limitations that are either specific to each solution or common to all solutions adhering to the same methodology.

A. Isolation based Solutions

Kotulski *et al.* [11] explored the use of host resource isolation and network communication isolation as measures to mitigate DDoS attack in 5G network slicing. Similarly, the work in [12] attempts to proactively mitigate DDoS attacks in 5G core network slicing using inter- and intra-slice isolation. The work demonstrates that complete isolation among slices achieved by inter-slice isolation enables DDoS attack mitigation. However, complete isolation may result in inefficient resource usage. Furthermore, the recent trend to evolve VNFs into Cloud-native Network Functions (CNFs), where the network functions are running on containers, makes the complete isolation hard to realize owing to the lack of strong hardware isolation.

B. Network Traffic Analysis based Solutions

Thantharate *et al.* [14] and Kuadey *et al.* [25] applied, respectively, Convolutional Neural Network (CNN) and Long Short Term Memory (LSTM) deep learning techniques to detect DDoS attacks in 5G network slicing. Nevertheless, the two contributions do not consider low-rate DDoS attacks, focusing only on high-rate DDoS attacks. In [26], multiple machine learning (ML) and deep learning (DL) techniques have been used to detect low-rate DDoS attacks in SDN-based setting. The authors observed the superiority of the DL

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2023.3318606

TABLE II: Classification and gap analysis of existing application-layer DDoS and EDoS mitigation techniques.

Ref.	Methodology	Description	ML Model	Limitations	
				Specific	Common
[11]	Isolation	- Mitigate DDoS attack in 5G network slicing using host resource and network communication isolation			- Inefficient resource usage - CNFs makes the complete isolation hard to realize
[12]		- Mitigate DDoS attack in 5G network slicing using inter- and intra-slice isolation			
[14]	Network Traffic Analysis	- Detect high-rate DDoS attack in 5G network slicing	CNN	- Do not consider low-rate DDoS attacks	- Can be escaped by application-layer DDoS attacks that can imitate legitimate traffic
[25]			LSTM		
[26]		- Detect low-rate DDoS attacks in a SDN-based setting	MLP		
[10]		- Detect both low-rate and high-rate DDoS attacks in a SDN-based setting	MLP		
[15]			GAN		
[27]	Resource Allocation	- Mitigate low-rate DDoS attacks by sacrificing the victim resources in favor of the mitigation service		- Performance degradation	- Risk of reshaping a DDoS attack into an EDoS attack - Resource starvation - Indirect EDoS to co-hosted slices
[16]		- Mitigate low-rate DDoS attacks in container -based cloud environment using a dynamic resource allocation strategy			
[28]	Resource Usage / Performance Analysis	- Detect EDoS attack in NFV-based SONs using an entropy-based approach		- Entropy-based detection approaches are vulnerable to spoofing	- Assume that sufficient amount of historical data is available
[29]		- Detect EDoS attack on a Kubernetes cluster using a supervised learning approach	XGBoost	- Labeled data may be scarce, expensive, or unfeasible at scale	
[18]		- Detect EDoS in an SDN-based cloud environment using a reconstruction-based anomaly detection approach	VAE + GRU	- Only temporal dependencies are considered - Ineffective in detecting low-rate DDoS attacks - POT method fails when there are many extreme values	
[19]		- Detect EDoS in an SDN-based cloud environment using a reconstruction-based anomaly detection approach and nonparametric dynamic thresholding	GAN + LSTM	- Only temporal dependencies are considered	
[20]		- Detect EDoS in an SDN-based cloud environment using a forecasting-based anomaly detection approach and nonparametric dynamic thresholding	LSTM		

model (i.e., Multi-Layer Perceptron(MLP)) compared to the other ML techniques used in the paper. In the same vein, the contributions in [10] and [15] demonstrated, respectively, the potential of MLP and Generative Adversarial Network (GAN) DL techniques in detecting both high-rate and low-rate application-layer DDoS attacks based on analysis of network traffic flows. However, identifying DDoS attacks by only analyzing the characteristics collected from network flows may not always be possible, particularly with the emergence of stealthy application-layer DDoS attacks which focus on depleting the server's resources (e.g., CPU, memory, I/O) while generating a traffic flow that imitates the legitimate one.

C. Resource Allocation based Solutions

Somani *et al.* [27] devised a DDoS mitigation approach that ensures resource availability to mitigation service during the attack by sacrificing victim service resources. This may result in performance degradation, preventing legitimate users from accessing the service. Li *et al.* [16] proposed a mitigation strategy based on dynamic resource allocation to thwart low-rate DDoS attacks in container-based cloud environment. The mitigation strategy dynamically regulates the number of container instances serving for different users and coordinates the resource allocation between instances to maximize the quality of service. However, the use of resource scaling approach to mitigate DDoS attacks may result in resource starvation and/or undesirable costs under DDoS attack. Indeed, the auto-scaling capability can reshape a DDoS attack into Economical Denial of Sustainability (EDoS) attack, which incurs economic damages to service provider due to the increased elastic use of resources as well as performance degradation [4]. Furthermore, the interdependence between network slices due to virtual network functions and infrastructure resources sharing rises the risk of indirect EDoS [30]; that is, the direct DDoS exhausts the resources of one slice, which may affect the resources shared with other slices, impacting the performance and availability of provided services.

D. Resource Usage / Performance Analysis based Solutions

The solutions in this category leverage new sources of information, namely resource usage and/or performance of service under attack, to discriminate malicious behavior caused

by DDoS attacks at application layer to prevent its reshaping into EDoS attacks. The authors in [28] introduced two variants of EDoS attack dedicated to NFV-based Self-Organizing Networks (SON), namely: Workload-based EDoS (W-EDoS) and Instantiation-based EDoS (I-EDoS). An entropy-based EDoS detection approach is proposed where a set of indicators (e.g., resource usage, application response time, number of VNF instances and their productivity) is used to distinguish between malicious and legitimate behaviors. However, entropy-based detection approaches are vulnerable to spoofing, where an attacker can make the entropy fit the expected distribution during the DDoS attack [31]. In [29], XGBoost classifier is used to detect EDoS attack on a Kubernetes cluster. The XGBoost model is trained on labeled data to perform a binary classification based on the statistical information (e.g., mean, minimum, maximum) computed on response time, CPU load, pod count and node count metrics. Nevertheless, in real-world applications, labeled data may be scarce or expensive, and a fully labeled data set on large scale may not be feasible. Su *et al.* [32] devised OmniAnomaly, an unsupervised anomaly detection approach that uses data's stochasticity and temporal dependence characteristics to learn the patterns of normal behavior. The stochasticity and temporal dependence features are extracted from multivariate time series using, respectively, variational autoencoder (VAE) and gated recurrent unit (GRU) techniques. The anomaly detection approach proposed in [32] has been leveraged by the work in [18] to detect EDoS in an SDN-based cloud environment. The detection approach allows to define a dynamic threshold following the peaks over threshold (POT) approach [33] to discriminate malicious traffic. It is worth noting that the proposed solution has difficulties in detecting low-rate DDoS attacks. Furthermore, POT method may not work when there are many extreme values (or outliers) which do not satisfy the generalized Patorley distribution (GPD) [34]. The authors in [19] adopted MAD-GAN [35], a GAN-based multivariate time series anomaly detection model, to identify EDoS attack in an SDN-based cloud. The MAD-GAN model uses LSTM algorithm to capture the temporal correlation of time series distributions. The nonparametric dynamic thresholding technique [36] is used to compute the anomaly threshold for discriminating EDoS attack. Unlike [18], [19], which adopt a reconstruction-based approach, the work in [20] follows a forecasting-based approach to counteract EDoS risk in an SDN-based cloud environment. Specifically, the forecasting-based anomaly detection approach proposed in [36] is leveraged, which uses LSTM and nonparametric dynamic thresholding to identify anomalies. This approach detects anomalies by measuring the error between the observed metrics' values and the predicted ones. However, the anomaly detection models in [18]–[20] only consider temporal dependencies while not explicitly addressing spatial correlations among features. In fact, the resource usage and performance metrics are very likely to impact each other; for example, the increase in the CPU load will certainly affect the service's response time. This makes the spatial dependency a valuable information that needs to be captured to improve the detection performance. Different from these works, the model we are proposing in this paper exploits both temporal and spatial dependencies within the multivariate time series. We leverage the potential of Graph Neural Networks (GNNs) to model the spatial correlations. Furthermore, rather than using a reconstruction-based approach as in [18], [19] and similar to [20], we adopt a forecasting-based approach. Following this approach, it is possible to build a multi-purpose model that can serve not only for anomaly detection task but also for proactive and dynamic resource allocation tasks, which will inevitably reduce the cost of training and running different models. Finally, the above mentioned solutions assume that sufficient amount of historical data is available for training the deep learning models to recognize normal behavior patterns, which may not be the case for a newly deployed network slice. To address this challenge, we exploit transfer learning paradigm to leverage the knowledge gained by a model at a network slice with enough data to improve the learning in the new slice.

III. FortisEDoS Elastic Mobile vCDN Framework

A. Framework Overview

In the following, we present FortisEDoS, a novel framework that aims to enable elastic 5G network slicing while intelligently safeguarding from malicious resource scaling requests caused by stealthy application DDoS attacks. As illustrated in Fig. 1, we consider a virtualized Content Delivery Network (vCDN) provided as a service over a MEC-enabled 5G network to deliver video content. It is worth mentioning that our solution is generic and not tied to this specific use case. The solely motivation behind considering the vCDN use case is the foreseen growth in mobile video traffic, which is currently estimated to account for 69% of all mobile data traffic and forecast to increase to 79% in 2027, according to the recent Ericsson Mobility Report [37].

The vCDN service provider, which could be a mobile network operator (MNO) or a third party, takes advantages of network slicing and Multi-access Edge Computing (MEC) paradigms to offer vCDN services tailored to specific Service Level Agreements (SLAs) with the vCDN customers (i.e., content providers). A vCDN service is dynamically deployed on-demand as a slice into the MNO's network and could typically be distributed over multiple cloud domains. Each slice is composed of a set of basic VNFs (e.g., streamers, caches, transcoders) chained together to provide a vCDN service instance. The vCDN slices can share 5G core network (CN) functions (e.g., Access and Mobility Management Function (AMF) and (Session Management Function (SMF)) or have their dedicated 5G CN functions that can be deployed in the central cloud (e.g., User Plane Function (UPF)) or moved downward to the edge (e.g., Intermediate UPF (I-UPF)) for the sake of performance. Note that I-UPF and UPF are in charge of steering the user plane traffic towards the targeted CDN service and towards the data network, respectively. It is worth mentioning that the provision of vCDN slices at the edge contribute to CDN's high performance, high throughput, and low latency [38]. The VNFs of a vCDN slice can be deployed over several edge compute nodes and the VNFs of different vCDN slices can be co-located on the same edge compute

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2023.3318606

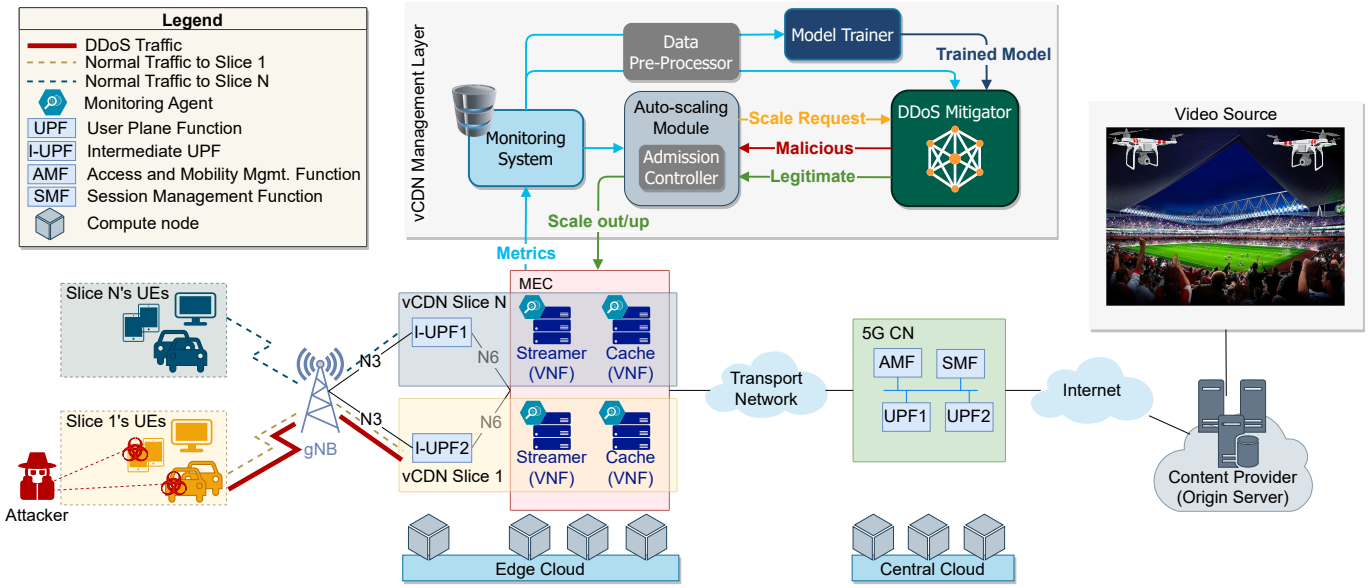


Fig. 1: The overall architecture of FortisEDoS Elastic 5G vCDN Framework.

node. We assume a logical isolation level [39] between vCDN slices instantiated on the edge infrastructure, where the vCDN functions are dedicated to each vCDN slice, but the virtual resources are shared.

During the operation of a vCDN slice, the user's requests for media content are served by the edge cache instance deployed in his/her proximity if the content is available there, otherwise, the content is retrieved from the origin server via the back-haul network. A vCDN slice can be target of stealthy DDoS attacks that can reshape into EDoS attacks due to the auto-scaling capability. The attacker model considered in this study is elaborated in Section III-B.

The FortisEDoS framework includes a *vCDN Management Layer* that encompasses a set of modules providing required functional capabilities to empower highly elastic and resilient vCDN services that can deliver superior Quality of Experience (QoE) while being impervious to EDoS attacks. The vCDN Management Layer is capable of proactively mitigating EDoS attacks by adopting a DL-powered forecasting-based approach for detecting malicious scaling requests caused by application-layer DDoS attacks. The forecasting problem formulation and the methodology followed to achieve this goal are detailed in Section III-C. In particular, the vCDN management layer consists of the following core components:

- *Monitoring System* is constantly tracking relevant monitoring information that can provide the actual state of vCDN services. Data related to resource usage (e.g., CPU, memory, disk and network usage) and performance (e.g., response time) metrics of the different vCDN slice's VNFs and their hosting edge nodes are collected as time series via the deployed monitoring agents. The collected monitoring data are used to drive the resource scaling and anomaly detection decisions made by the auto-scaling module and DDoS Mitigator, respectively.
- *Auto-scaling Module* dynamically expands or contracts the capacity of a vCDN slice instance to adapt resources to slice workload in order to satisfy the committed SLA. The scaling decision occurs at VNF level based on VNF performance, resource usage metrics provided by the Monitoring System and the associated auto-scaling policies. It is worth noting that a VNF can either be scaled horizontally by increasing (scale out) or decreasing (scale in) the number of VNF instances or vertically by increasing (scale up) or decreasing (scale down) the resources (e.g., memory, CPU, storage, network) used by a VNF instance.
- *Admission Controller* is responsible for intercepting the scaling-up/out requests triggered by the Auto-scaling Module in order to delegate the scaling decision to the DDoS Mitigator for validation.
- *DDoS Mitigator* leverages the potential of DL to automatically detect whether the scaling request is due to legitimate load or rather malicious workload caused by application-layer DDoS attacks. It incorporates a DL-based anomaly detection model, coined CG-GRU, which can effectively identify anomalous resource usage and performance metrics of VNFs and their hosting nodes using a data-driven forecasting-based approach. In fact, the anomalies are detected when the predicted metrics' values deviate considerably from the observed ones. If an anomaly is detected, the scaling operation is tagged as malicious and will be refused by the Admission Controller. Details on the proposed CG-GRU model and the selection of the anomaly threshold will be provided in the subsequent sections III-E and III-F.
- *Data Pre-Processor* is responsible for preparing the raw time series data into appropriate format to fit for CG-GRU model during training and inference phases. This includes data cleansing, normalization, and segmentation operations, as detailed in Section III-D. Note that the data used for training the model includes only time series of normal behavior.
- *Model Trainer* is in charge of building the CG-GRU model to integrate in the DDoS Mitigator. This involves

tuning the model's hyper-parameters, training the model on a training dataset, and assessing its performances on unseen data. We elaborate further on the strategies adopted for fine-tuning the model's hyper-parameters in Section IV-C. In addition to supporting training from scratch, a key feature of the Model Trainer is its ability to train the CG-GRU model using transfer learning. This feature is particularly crucial when there is a scarcity of representative historical data for normal behavior, as is often the case for newly deployed slices. We elaborate further on the proposed transfer learning-based CG-GRU model in Section III-G.

B. Attacker Model

In the context of this study, we assume that the attacker has control over a subset of user equipments (UEs) that can legitimately use a 5G vCDN service (e.g., live network streaming service) delivered via HTTP-based technologies. The attacker aims at exhausting the vCDN slice's resources (e.g., CPU, memory, disk I/O, network I/O) to prevent legitimate users from accessing the vCDN service or at the very least increase the service response time, leading to SLA violation. To this end, we assume that the attacker has the capability to carry out application-layer DDoS attacks against the vCDN's VNFs exposed to end user for content delivery, such as the video streamer VNF. Particularly, the attacker is able to launch both high-rate and low-rate HTTP-based DDoS attacks. In high-rate mode, the attacker mimics a flash-crowd event by flooding the exposed service with a large number of legitimately formed HTTP requests in a very short period of time. In the low-rate mode, however, the attacker sets up multiple HTTP connections with the exposed service by sending partial HTTP requests at a very slow rate, which results in exhausting the connection queue space.

We further assume that the attacker possesses the ability to generate stealthier patterns of the application-layer DDoS traffic that can fly under the radar of protection mechanisms that detect DDoS attacks based solely on characteristics collected from network flows [10]. Thus, the malicious traffic will reach the exposed VNF and leads to request for provisioning additional resources through auto-scaling capability, which can result in undesirable costs and/or resource starvation, impacting not only the vCDN slice under attack, but also the other slices co-hosted on the same computing infrastructure. By exploiting the auto-scaling capability, the attacker can reshape the application-layer DDoS attack into an EDoS attack to incur economic damages to vCDN service provider due to the increased elastic use of resources as well as performance degradation due to shared resource starvation.

C. Problem Formulation and Methodology

We consider a set of n slices $\mathcal{S} = \{S_1, S_2, \cdots, S_n\}$. Each slice S_i is composed of a set of m VNFs $\mathcal{V}_i = \{f_1^i, f_2^i, \cdots, f_m^i\}$. As depicted in Fig. 1, the VNFs of a slice can be deployed through several nodes and the VNFs of different slices can be co-hosted on the same node.

Each VNF $f_j^i \in \mathcal{V}_i$ is characterized by a set of features $\boldsymbol{x} \in \mathbb{R}^{\boldsymbol{d}}$ representing the resource usage (e.g., CPU utilization, memory utilization, system load) and performance (e.g., response time) metrics of the VNF. $\boldsymbol{d}$ refers to the dimension of the features set (number of features) for the VNFs.

The VNF's metrics recorded at regular intervals over a period of time can be formulated as a multivariate time series $\mathcal{X} = \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(T)}\} \in \mathbb{R}^{T \times d}$, where T and d are the length of the time series and the number of metrics, respectively. Each step $\boldsymbol{x}^{(t)} \in \mathbb{R}^d$ in the time series is a d-dimensional vector $\{x_1^{(t)}, x_2^{(t)}, \cdots, x_d^{(t)}\}$ representing the metrics data observed on VNF at time t.

We aim to detect the application-layer DDoS attack by identifying anomalies in the resource usage and performance metrics of VNFs and their hosting nodes using a forecasting-based approach, where an anomalous VNF's status is detected when the expected metrics values deviate greatly from the measured ones. As each metric may not only depend on its own historical values, but also on other metrics' past, we adopt a multivariate time series forecasting approach in order to improve the metrics forecasts, and consequently the anomaly detection accuracy. Given the observed metrics values of previous w time steps $\boldsymbol{x}^{(t-w+1)}, \cdots, \boldsymbol{x}^{(t)}$, the multivariate time series forecasting task aims to learn a model $F : \mathbb{R}^{w \times d} \mapsto \mathbb{R}^{\mathfrak{h} \times d}$ for predicting the future metrics values for the next $\mathfrak{h}$ time steps, denoted by $\hat{\boldsymbol{x}}^{(t+1)}, \cdots, \hat{\boldsymbol{x}}^{(t+\mathfrak{h})}$. It can be formally written as

$$[\hat{\boldsymbol{x}}^{(t+\mathfrak{h})}, \cdots, \hat{\boldsymbol{x}}^{(t+1)}] = F(\boldsymbol{x}^{(t)}, \cdots, \boldsymbol{x}^{(t-w+1)}) \qquad (1)$$

Note that the forecasting model is trained to successfully predict future metrics values from normal values by minimizing the prediction error. Hence, during the inference, the prediction error is expected to rise in the presence of abnormal metrics values due to DDoS attack. Relying on this hypothesis, we use the prediction error to measure the anomaly score, which represents the deviation of true metrics values from the predicted ones. The derived anomaly score is compared against a detection threshold to determine whether the VNF status at a given time step is anomalous or not; if the anomaly score is above the detection threshold, the VNF status is flagged as anomalous.

In the following, we provide details on how the forecasting model is built and how the anomaly scores and detection threshold are calculated. For the reader's convenience, we have summarized the key notations used in this paper in the lower part of Table I.

D. Data Preprocessing

The data pre-processing module aims at preparing the raw time series data into the appropriate format to fit for the forecasting model during training and inference phases. Note that the train dataset includes only time series data of normal behavior.

Firstly, the raw time series data are cleaned by imputing missing/infinity values. In this study, we leverage the Last Observation Carried Forward (LOCF) method to impute the

time series missing values with their corresponding last observed value. The rationale behind using LOCF method is its simplicity and the fact that very few missing data were found in our dataset. However, more advanced imputation methods can be adopted, such as those relying on Generative Adversarial Networks (GANs) [40], to deal with high missing-rate situation.

The raw time series data are then normalized using the Min-Max scaling technique, which scales the values in each time series to be in the range $[0, 1]$. The data normalization helps in alleviating the impact of different scaling among collected metrics, which improves the model stability and speed up the training process. The normalized value $\bar{x}_i^{(t)}$ of a metric $x_i^{(t)}$ observed at time step t is calculated as follows:

$$\bar{x}_i^{(t)} = \frac{x_i^{(t)} - min(\mathcal{X}_{train})}{max(\mathcal{X}_{train}) - min(\mathcal{X}_{train})} \tag{2}$$

$min(\mathcal{X}_{train})$ and $max(\mathcal{X}_{train})$ are, respectively, the minimum value and the maximum value of the training set.

Finally, the raw time series data are segmented into a series of sub-sequences by applying a sliding window technique. As shown in Fig. 2, the training dataset is constructed as a supervised dataset, where the inputs are the observed metrics values of previous w time steps and the outputs are the future values to forecast for the next $\mathfrak{h}$ ($= 1$ in Fig 2) time steps. Given a look-back sliding window of size w and step length τ, the number of sub-sequences in the training dataset can be calculated as:

$$|\mathcal{D}_{train}| = \frac{|\mathcal{X}_{train}| - (w + \mathfrak{h})}{\tau} + 1 \tag{3}$$

E. Forecasting Model Architecture

Fig. 2 illustrates the overall architecture of the proposed CG-GRU forecasting model. It is an hybrid model that combines the advantages of different deep neural network algorithms, specifically CNN, GNN, RNN and MLP, to provide both feature extraction and forecasting capabilities. Indeed, deep learning techniques have proved their capability in unveiling hidden patterns from a large-amount of time-varying multi-dimensional data and achieving accurate decisions [41].

The feature extraction stage consists in capturing both temporal and spatial dependencies within the multivariate time series. Leveraging the high ability of CNN in extracting high-level representations from data, the local useful features are extracted from the pre-processed multivariate time series using a one-dimensional Convolutional (Conv1D) layer. The resulting features are then fed into a Graph Attention (GAT) layer to derive the spatial inter-dependencies between the VNF's metrics. Thanks to the attention mechanism of GAT, different weights (i.e., attention coefficients) are assigned to each pair of features, allowing to measure the degree of influence of VNF's metrics on each other. The features extracted by the GAT layer are processed by multiple GRU layers to characterize the temporal dynamics of the VNF's metrics. The use of GRU is motivated by their demonstrated effectiveness and efficiency in modeling long-term temporal sequences owing to their ability to remember relevant past observations while inducing reduced computation costs and complexity.

The forecasting stage takes the spatio-temporal representations learned by the feature extraction block as inputs for predicting the future VNF's metrics values. It relies on a fully-connected network comprising multiple fully-connected layers.

1) Conv1D Layer: A Conv1D layer is employed for the purpose of automatically extracting relevant local features of the raw VNF's metrics data within a sliding window. The local features are obtained by first convolving the input data with a learned convolution kernel and then applying a non-linear activation function. The process of the Conv1D layer can be formalized as:

$$h_k = f(\sum_{i=1}^{d} W_{ik} * x_i + b_k) \tag{4}$$

where $f(.)$ is the non-linear activation function, x_i is the time series of the i-th VNF's metric, and W_{ik} is the k-th convolution kernel corresponding to the i-th time series. b_k and h_k denote, respectively, the bias and the learned feature map of the k-th convolution kernel.

2) Graph Attention Layer: The values of the VNF's metrics are very likely to influence each other; for instance, the increase in the number of service requests will certainly increases the VNF's CPU usage to handle the received requests. Hence, capturing the spatial dependency among the VNF's metrics will help in achieving more accurate forecasts. To this end, we leverage the effectiveness of graphs to model relationships between entities and the potential of emerging Graph Attention Networks (GATs) [42] to learn complex relationships in graph-structured data while assigning varying levels of importance to each relationship through attention mechanism. Compared to previous graph-based methods, and thanks to the attention mechanism, GATs have the advantage of capturing the importance levels, being storage and computationally efficient, not requiring prior knowledge of the global graph structure, allowing inputs of variable sizes, and providing the interpretability of the model. It is worth noting that interpretability is a key property to foster trustworthy ML-based systems by ensuring accountability, reliability and transparency [43].

We incorporate a graph attention layer to model and capture the causal relationships between the VNF's metrics using a graph structure where nodes represent the different VNF's metrics and an edge between two nodes denotes the relationship between the corresponding VNF's metrics. The i-th node is characterized by a feature vector v_i containing the values of the corresponding VNF's metric across all w time steps. The relationship between nodes is weighted according to learned attention coefficients, which measure the degree of influence of VNF's metrics on each other through attention mechanism. The output produced by the graph attention layer for each node is calculated as follows:

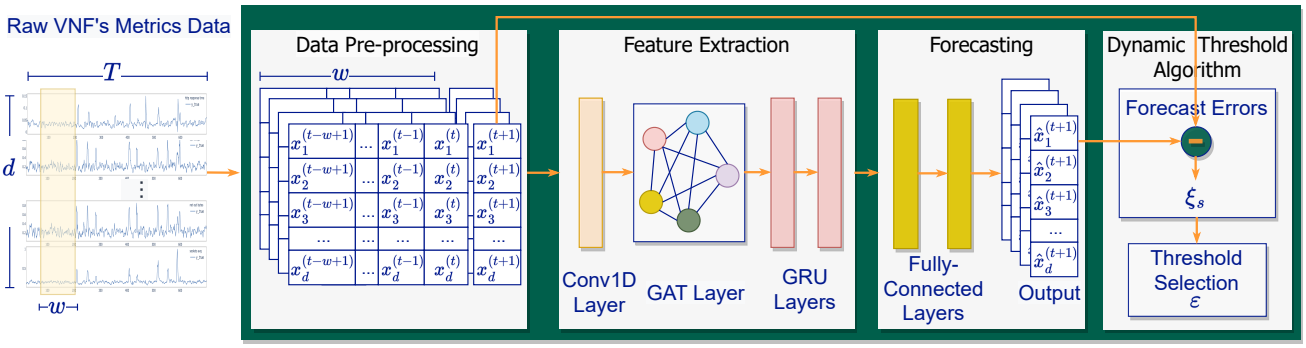


Fig. 2: The overall architecture of training CG-GRU model and selecting the dynamic anomaly threshold.

$$h_i = \sigma\Big(\sum_{j=1}^{d} \alpha_{ij} v_j\Big) \tag{5}$$

$$\alpha_{ij} = \text{softmax}(o_{ij}) = \frac{\exp(o_{ij})}{\sum_{k=1}^{d} \exp(o_{ik})} \tag{6}$$

$$o_{ij} = \text{LeakyReLU}(A^{\top}.(v_i || v_j)) \tag{7}$$

where h_i is the output of node v_i with the same dimension, α_{ij} is the normalized attention coefficient between nodes v_i and v_j, and $A \in \mathbb{R}^{2w}$ is a column vector of learnable parameters. σ and LeakyReLU represent non-linear activation functions. The symbols $.^{\top}$ and $||$ denote the vector transpose and the concatenation operator, respectively.

3) GRU & Dense Layers: The forecasting model is based on a multi-layer stacked GRU architecture, which consists of multiple GRU layers followed by multiple dense (i.e., fully connected) layers with non-linear activation functions. It is worth mentioning that the stacked nature of the GRU and dense layers coupled with the non-linear activation functions facilitate capturing complex spatio-temporal relationships existing among the VNF's resource usage and performance metrics, which positively influence the prediction accuracy. Moreover, the use of GRU not only prevents the exploding and vanishing gradient problems of RNNs, but also reduces the complexity of the recurrent unit structure while achieving comparable performance [44].

Each GRU layer contains several hidden units, each of which consists of two gates, called reset gate (r_t) and update gate (z_t), to update the hidden state. The reset gate forgets irrelevant past information, while the update gate aims at retaining relevant information from the previous time step. Formally, the reset and update gates for time step t are computed by:

$$z_t = \sigma(L_z x_t + U_z s_{t-1} + b_z) \tag{8}$$

$$r_t = \sigma(L_r x_t + U_r s_{t-1} + b_r) \tag{9}$$

where $\sigma(.)$ denotes the sigmoid activation function, which restricts the value of all element in reset gate and update gate between 0 and 1 to capture short and long-term temporal dependencies, respectively. L_z and L_r are the weight matrices connecting the current time step input x_t to the update gate and reset gate, respectively. Meanwhile, U_z and U_r represent the weight matrices connecting the previous hidden state s_{t-1} to the update gate and reset gate, respectively. b_z and b_r are the bias vectors.

The hidden state s_t of a GRU unit can be computed based on the previous hidden state s_{t-1} and the candidate hidden state $\tilde{s}_t$ as

$$\tilde{s}_t = tanh(L_s x_t + U_s(r_t \odot s_{t-1})) \tag{10}$$

$$s_t = (1 - z_t)s_{t-1} + z_t \tilde{s}_t \tag{11}$$

where $\odot$ denotes the element-wise multiplication. L_s and U_s are the weight matrices related to the current time step input x_t the previous hidden state s_{t-1}, respectively. b_s is the bias vector. Note that L_*, U_* and b_* are the learnable parameters.

F. Forecast-based DDoS Anomaly Detection

To set an appropriate threshold for detecting anomalous VNF scaling requests due to application-layer DDoS attacks, we adopt a dynamic thresholding methodology [34]. This method allows to calculate an anomaly detection threshold that is automatically adjusted according to the past smoothed forecasting errors. It is worth noting that the key advantage of the dynamic thresholding method is its reliance on a non-parametric probability distribution estimation approach, which avoids the limitations of traditional Gaussian assumptions on the past smoothed forecasting error distribution.

The forecasting error of the i-th VNF's metric at time step t, $e_i^{(t)}$, is calculated by

$$e_i^{(t)} = |x_i^{(t+1)} - \hat{x}_i^{(t+1)}| \tag{12}$$

where $x_i^{(t+1)}$ and $\hat{x}_i^{(t+1)}$ are, respectively, the actual value and forecast value of the i-th VNF's metric at time step t.

Utilizing the metric-specific forecasting errors, the global forecasting error of the VNF at time step t, $e^{(t)}$, is computed as

$$e^{(t)} = \frac{1}{d} \sum_{i=1}^{d} e_i^{(t)} \tag{13}$$

The dynamic threshold is derived using the smoothed global forecasting errors at time step t, $\xi_s = [e_s^{(t-w)}, \cdots, e_s^{(t-1)}, e_s^{(t)}]$, where w is the historical observing length. The exponentially weighted moving average

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2023.3318606

(EWMA) algorithm [45] is applied to smooth the global forecasting errors, allowing to reduce the false positives. The threshold ε is selected from the set:

$$\boldsymbol{\epsilon} = \mu(\xi_s) + \beta * \delta(\xi_s) \tag{14}$$

such that

$$\varepsilon = \text{argmax}(\boldsymbol{\epsilon}) = \frac{\Delta\mu\left(\xi_s\right)/\mu\left(\xi_s\right) + \Delta\delta\left(\xi_s\right)/\delta\left(\xi_s\right)}{|\boldsymbol{\xi}_a| + |\mathbf{E}_{seq}|^2} \tag{15}$$

where

$$\begin{aligned}
\Delta\mu\left(\xi_s\right) =& \mu\left(\xi_s\right) - \mu\left(\{e_s \in \xi_s \mid e_s < \varepsilon\}\right) \\
\Delta\delta\left(\xi_s\right) =& \delta\left(\xi_s\right) - \delta\left(\{e_s \in \xi_s \mid e_s < \varepsilon\}\right) \\
\boldsymbol{\xi}_a =& \{e_s \in \xi_s \mid e_s > \varepsilon\} \\
\mathbf{E}_{seq} =& \text{continuous sequences of } \xi_a \in \boldsymbol{\xi}_a
\end{aligned}$$

Note that $\Delta\mu\left(\xi_s\right)$ and $\Delta\delta\left(\xi_s\right)$ refer to the decrease in the mean and the standard deviation of the global forecasting errors, respectively. $\boldsymbol{\xi}_a$ represents all the global forecasting errors that are above the dynamic threshold. β in Eq. (14) is selected from an ordered set B of positive values representing the standard deviations above $\mu\left(\xi_s\right)$. The process of training CG-GRU model and selecting the dynamic anomaly threshold ε is summarized in Algorithm 1.

Algorithm 1 CG-GRU Training and Anomaly Threshold Selection.

Input:
$\mathcal{X}_{train}$: The train multivariate time series
d: The number of metrics
w: The size of the look-back sliding window
$\mathfrak{h}$: The length of the forecast horizon
$CGGRU()$: The forecasting model architecture
val_{split}: The ratio of training dataset used for validation
Output:
$\mathcal{M}_t$: The trained model; ε: The anomaly threshold

▷ ***Data pre-processing phase***
$\mathcal{D}_{train} \leftarrow SlidingWindow(Normalize(\mathcal{X}_{train}), w, \mathfrak{h})$
▷ ***Model training phase***
$\mathcal{M}_t \leftarrow Train(CGGRU(), \mathcal{D}_{train}, val_{split})$
▷ ***Threshold selection phase***
$Y \leftarrow [\,]$
for x, y in $\mathcal{D}_{train}$ **do** ▷ $x \in \mathbb{R}^{w \times d}$ and $y \in \mathbb{R}^{\mathfrak{h} \times d}$
  $\hat{y} \leftarrow \mathcal{M}_t(x)$ ▷ $\hat{y}$ is the forecast values of the actual values y
  $Y.append(\hat{y})$
end for
$T \leftarrow |\mathcal{D}_{train}|$
▷ Calculate forecast errors per metric using Eq. (12)
$\{\{e_i^{(t)}\}_{i=1}^{i=d}\}_{t=1}^{t=T} \leftarrow Pred\_Err(Y, \hat{Y})$
▷ Calculate the global forecast error using Eq. (13)
$\{e^{(t)}\}_{t=1}^{T} \leftarrow Global\_Err(\{\{e_i^{(t)}\}_{i=1}^{i=d}\}_{t=1}^{t=T})$
▷ Select threshold using Eq. (14) & Eq. (15)
$\varepsilon \leftarrow Find\_Epsilon(ewma(\{e^{(t)}\}_{t=1}^{T}))$
Return $\mathcal{M}_t$, ε

The values of the VNF's metrics at a time step t are flagged as anomalous if the corresponding smoothed global forecasting error $e_s^{(t)}$ exceeds the calculated threshold. Algorithm 2 summarizes the anomaly detection process using CG-GRU model.

Algorithm 2 CG-GRU based Anomaly Detection.

Input:
$\mathcal{D}_{test}$: The test dataset
d: The number of metrics
$\mathcal{M}_t$: The trained CG-GRU model
ε: Anomaly threshold
Output:
$Anom$: Anomaly decisions on test dataset

for x, y in $\mathcal{D}_{test}$ **do**
  $\hat{y} \leftarrow \mathcal{M}_t(x)$
  $\{e_i(t)\}_{i=1}^{i=d} \leftarrow Pred\_Err(y, \hat{y})$
  $e^{(t)} \leftarrow Global\_Err(\{e_i^{(t)}\}_{i=1}^{i=d})$
  $e_s^{(t)} \leftarrow ewma(e^{(t)})$ ▷ Calculate the smoothed global forecasting error using EWMA algorithm
  if $e_s^{(t)} \geq \varepsilon$ **then**
    $Anom[t] \leftarrow 1$
  else
    $Anom[t] \leftarrow 0$
  end if
end for
Return $Anom$

G. Transfer Learning empowered DDoS Anomaly Detection

The VNFs of a newly deployed vCDN slice will possibly lack representative training data that capture all variations of their normal behavior, resulting in cold-start problem [46] which may lead to performance degradation of DDoS anomaly detection. Moreover, waiting until getting sufficient data to train the forecasting-based model from scratch for effective representation of normal VNF's behavior is time and resource consuming. Considering the envisioned massive number of slices that could be deployed, it is paramount to reduce the (re)training time and cost to enable timely detection of attacks and ensure service profitability.

To address the aforementioned issues, we leverage the potential of transfer learning to exploit the knowledge gained by a model in previously deployed slice (referred to as *source domain*) for improving and accelerating the learning of a model in a newly instantiated slice (denoted as *target domain*). More specifically, we consider transferring knowledge regarding feature representations, characterizing a normal behavior, learned by the pre-trained model of the source domain to the new model of the target domain. In fact, deep neural networks are characterized by their ability to learn general features (i.e., domain-independent) on the first layers and specific features (i.e., domain-dependent) on the layers closer to the output [47], allowing transferability of general features across domains. Inspired by that, we perform the transfer learning by initializing the feature extraction layers of the new CG-GRU model with the weights inherited from the pre-trained CG-GRU model. The fully-connected layers are replaced with new ones that are fine-tuned (i.e., trained) on the target data to make the model customized for the associated VNF. It is worth noting that the weights of the feature extraction layers are frozen during the fine-tuning phase to preserve the transferred knowledge. An illustration of the proposed transfer learning process is given in Fig. 3.

The capability of transferring previous knowledge and fine-tuning only fully-connected layers results in fast training and

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2023.3318606

improved detection performances of the new CG-GRU model.

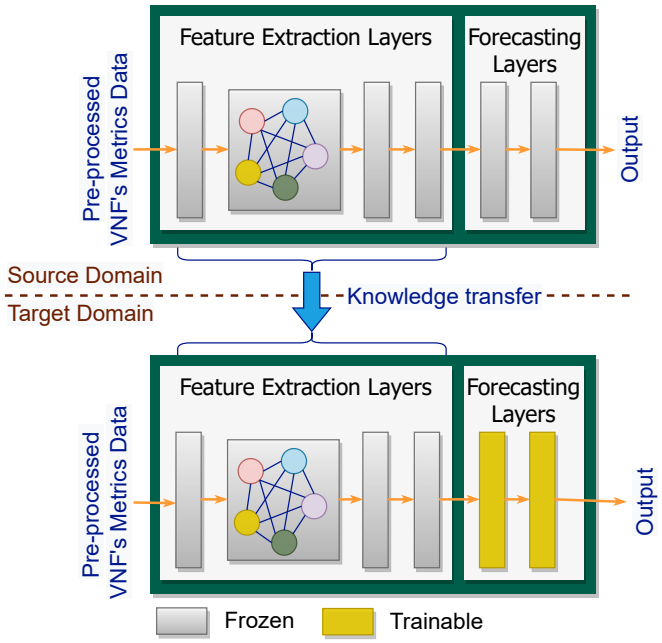


Fig. 3: Illustration of the transfer learning process. The knowledge on normal behavior representation obtained from model trained in source domain is transferred to the new model in the target domain.

IV. Performance Evaluation

A. Experimental Setup

To evaluate the performance of FortisEDoS, we built a testbed based on Kubernetes (K8s)[1] environment. As illustrated in Fig. 4, the testbed is composed of two OpenStack cloud platforms interconnected via a secure communication channel. Four VMs have been deployed on the first OpenStack cloud to setup a K8s cluster with one master node and three worker nodes. The K8s is used to deploy the vCDN slices, where each slice consists of two CNFs, namely a video streamer and a cache, chained together to provide an HTTP-based on-demand video streaming service. The two CNFs are deployed as K8s services running a NGINX web server on a pod, and are spread over two worker nodes. The video streamer service is exposed to the end user for content delivery, while the cache service is only reachable from within the cluster. Each vCDN slice instance has its own namespace to guarantee isolation of API resources between slices.

A fifth VM, deployed on the second OpenStack cloud, is used for running the Monitoring System and DDoS Mitigator; they are instantiated within containers. The VM also serves as a platform for training and testing the CG-GRU model incorporated in the DDoS Mitigator. In this vein, different open-source tools have been deployed on the VM to create the training and testing pipeline, including Pytorch and Python.

The Monitoring System includes a "Metrics Collector" which uses Prometheus API to extract the raw resource usage and performance metrics from the vCDN slices' CNFs and their hosting worker nodes. To this end, Prometheus relies on different monitoring probes, including NGINX-to-Prometheus log file exporter[2], cadvisor[3] and node-exporter[4]. The Monitoring System offers the capabilities to generate on-demand dataset for training and testing the CG-GRU model or continuously scrape the metrics values to be consumed by the trained CG-GRU model integrated in DDoS Mitigator for real-time forecasting-based anomaly detection.

The Auto-scaling module is implemented using the K8s built-in Horizontal Pod Autoscaling (HPA) functionality which we extended with the event-driven scale feature provided by the open-source KEDA[5] tool. We defined different horizontal scaling policies that can trigger a scaling operation to increase the number of pod instances to handle the load on a CNF based on observed per-pod metrics (e.g., CPU, RAM) or external metrics obtained from Prometheus (e.g., number of HTTP requests or response time).

B. Dataset Generation

Due to the lack of real data, we used our testbed to generate realistic datasets to train and test the proposed anomaly detection approach. In fact, we were unable to find public datasets with relevant features and labeled EDoS/DDoS threats to use as ground truth for assessing the solution performances. Existing public datasets are either limited to network traffic characteristics (e.g., CIC-IDS2017 and CSE-CIC-IDS2018 datasets[6]) or lack both application-level and (virtual) machine-level features (e.g., SMD[7], which only includes machine-level metrics).

To generate the dataset, we have developed a normal load generator that models the arrival times of video streaming requests initiated by legitimate users according to Poisson process with fixed hourly rate. Recall that the Poisson process is based on the assumption that the times between successive requests are exponentially distributed and independent [48]. Thus, the inter-arrival times are generated using the inverse Cumulative Distribution Function (CDF) of the Poisson distribution as given by Eq. (16):

$$CDF^{-1} = -\frac{1}{\lambda} * ln(1-u) \tag{16}$$

where λ denotes the expected average number of requests generated per hour and u is a random number sampled between 0 and 1 by a uniform distribution. To characterize the change of request arrival rates over the time that real-life VoD systems exhibit [49], we generated different request patterns by varying λ in various time intervals. Specifically, the value of λ is randomly drawn from the set $\{90, 120, 130, 140\}$.

The normal load generator includes a python script that controls the Selenium WebDriver[8] [50] for automating the loading and playback of the requested videos in a web browser

[1]https://kubernetes.io

[2]https://github.com/martin-helmich/prometheus-nginxlog-exporter
[3]https://github.com/google/cadvisor
[4]https://github.com/prometheus/node_exporter
[5]https://keda.sh
[6]https://www.unb.ca/cic/datasets/index.html
[7]https://github.com/NetManAIOps/OmniAnomaly
[8]https://www.selenium.dev

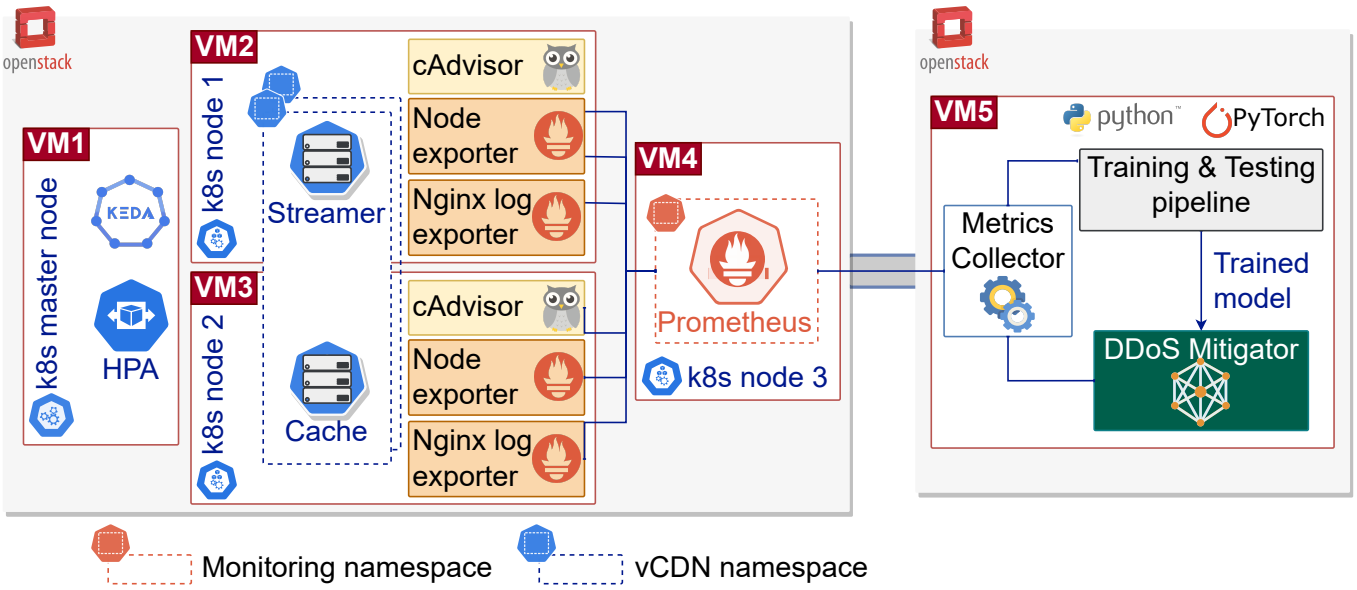


Fig. 4: The experimental testbed for evaluating FortisEDoS Framework.

(e.g., Firefox browser). To mimic a realistic behavior, the generator has also the capability to approximate the "impatient user" behavior where users abandon the video streaming sessions before their end [51]. This has been implemented by randomly varying the duration of the video streaming sessions.

To generate malicious load, we implemented the attack agents using Slowloris[9] tool for low-rate DDoS attacks and Hulk[10] tool for high-rate DDoS attacks. The DDoS attacks are carried out against the exposed video streamer service.

The raw resource usage and performance metrics data have been recorded from two vCDN slices by the Monitoring System over a period of 5 days. A time series for each resource usage and performance metric was recorded using a data sampling period of 300s. The training data were collected during the first 4 days of attack-free activity. The 5th day served to create the testing dataset which includes data for normal activity as well as anomalous activity caused by application-layer DDoS attacks executed on different periods of the day. Specifically, three Hulk attacks with different intensities and one Slowloris attack were launched. A training dataset and a testing dataset, containing the raw multivariate time series data, are generated for each vCDN slice's CNF with a total of 5401 and 2701 samples, respectively. During training, 20% of samples in the training dataset are held out for validation. To promote reproducibility and support independent investigations beyond this study, we have made the generated dataset publicly accessible[11].

C. Model Training

It is well known that the architecture of a deep neural network plays a crucial role in improving its performance [43]. Hence, we formulate the problem of finding the optimal model architecture as a search problem (See Eq. (17)) that seeks to maximize the model's performance by minimizing its loss function on the validation dataset.

$$\begin{aligned} \text{argmin}_{A} \quad & = \mathcal{L}(A, \mathcal{D}_{train}, \mathcal{D}_{val}) \\ \text{s.t.} \quad & A \in \mathcal{A} \end{aligned} \tag{17}$$

where $\mathcal{A}$ represents the search space of the possible architectures (i.e., combination of hyper-parameters), and $\mathcal{L}(.)$ measures the forecast loss error of the architecture A on the validation dataset $\mathcal{D}_{val}$ after being trained on the training dataset $\mathcal{D}_{train}$.

The problem formulated in (17) is solved by tuning the model's hyper-parameters leveraging the grid search [52] and ASHA [53] strategies. The ASHA strategy enables to integrate early stopping into the hyper-parameter optimization process, which allows to accelerate the process by terminating bad performing trials early. The search space for finding the best hyper-parameters for the forecast model includes the learning rate of the optimizer, the dropout rate, the number of GRU and fully-connected hidden layers, the number of neurons per layer, the kernel size for CNN layer, the historical window size w, and the batch size. Table III defines the search space used to determine the optimal architecture of the proposed CG-GRU model. Each possible architecture is trained at most 100 epochs using Rectified Linear Unit (ReLU) as the activation function, Adam as the optimizer and Mean Squared Error (MSE) as loss function. The CG-GRU model with the smallest forecast error on the validation set is used to forecast the VNF's resource usage and performance metrics. The best performing model achieved a forecasting loss of 0.0361 (3.61%) on the validation set. The hyper-parameters setting of the best model is reported in Table III.

D. Performance Metrics

To assess the effectiveness of FortisEDoS in detecting and preventing fraudulent resource scaling requests caused by application-layer DDoS attacks, we measure the performances

[9]https://github.com/gkbrk/slowloris
[10]https://github.com/grafov/hulk
[11]https://zenodo.org/record/8111592

TABLE III: The search space for hyper-parameters tuning and the best model configuration.

Hyper-Parameter	**Values**	**Best Configuration**
Learning rate	$(0.01, 0.001, 0.0001)$	0.001
Dropout rate	$(0.1, 0.2, 0.3)$	0.2
Number of hidden layers	$(1, 2, 3)$	GRU $= 2$ Dense $= 2$
Neurons per hidden layer	$(30, 45, 60, 75, 90)$	GRU $= 90$ Dense $= 60$
Kernel size	$(2, 3, 4)$	3
Window size w	$(30, 50, 80, 100)$	80
Batch size	$(60, 90, 120, 150, 180)$	150

of CG-GRU model in terms of forecasting accuracy and attack detection. The forecasting accuracy is evaluated over the training dataset using the Root Mean Square Error (RMSE) and Mean Absolute Error (MAE), which can be calculated by:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x^{(t+1)} - \hat{x}^{(t+1)})^2} \tag{18}$$

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|x^{(t+1)} - \hat{x}^{(t+1)}| \tag{19}$$

where $\hat{x}^{(t+1)}$ and $x^{(t+1)}$ denote the predicted values of the CNF's metrics at time step t and the corresponding ground truth, respectively. N is the number of samples in the dataset. Note that the lower the RMSE and MAE, the higher the forecasting accuracy is achieved.

The attack detection performances are assessed over the testing dataset using the common metrics, namely: Precision, Recall (aka sensitivity) and F1-score (denoted as F1). The *F1* metric is used to characterize the balance between the precision rate and the recall rate. Note that an effective model is the one providing the highest precision, recall and F1 values. The metrics are measured using the formulas in Eq.20.

$$\begin{cases} Precision = \dfrac{TP}{TP + FP}, \\ Recall = \dfrac{TP}{TP + FN}, \\ F1 = 2 \times \dfrac{Recall \times Precision}{Recall + Precision} \end{cases} \tag{20}$$

where TP (True Positive) represents the number of anomalies that are correctly detected, FN (False Negative) denotes the number of anomalies that are falsely detected as normal samples, FP (Flase Positive) is the number of the normal samples that are wrongly flagged as anomalous ones, and TN (True Negative) refers to the number of the normal samples that are correctly detected.

Besides its effectiveness, we evaluate the efficiency of FortisEDoS in terms of average training time, average inference time and size of CG-GRU model.

To validate the effectiveness and efficiency of FortisEDoS, we compare CG-GRU with the state-of-the-art LSTM-based

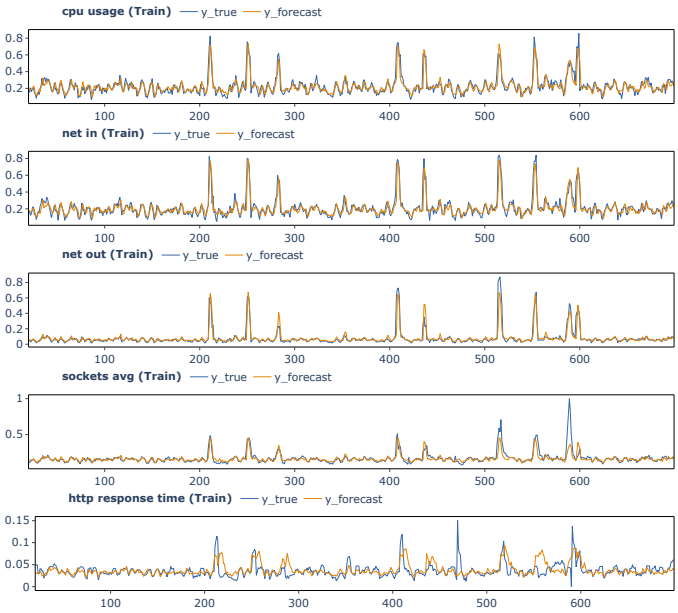


Fig. 5: Forecasting on validation data of video streamer CNF of vCDN slice 1.

multivariate time series anomaly detection model proposed in [36] and other baseline models derived from CG-GRU and model in [36] using ablation and transplantation of 1D Convolutional layer and/or GAT layers, respectively. Recall that the model in [36] has been used in [20] for mitigating EDoS attack in a SDN-based cloud environment.

The experiments are carried out on the fifth VM with 16-cores Intel's Skylake $2.4GHz$ CPU and $64GB$ RAM. To avoid bias from randomness, we report the results of the best-performing model over ten runs.

E. Performance Results

1) Forecasting Accuracy: In this section, we present the evaluation results on the forecasting accuracy of the proposed CG-GRU model. Fig. 5 reports the forecast values of some metrics of the vCDN's video streamer and their corresponding ground truth values in the validation dataset for a forecasting horizon of 5 minutes. We omitted the remaining metrics as they are exhibiting the same results. As can been seen from Fig. 5, CG-GRU is able to effectively predict the actual values of a normal CNF's status with high accuracy. The model recorded a low prediction RMSE of 0.039 and MAE of 0.1189 with a standard deviation of 0.005 on the validation dataset. The high forecasting quality delivered by CG-GRU model could be attributed to its capacity in capturing both time dependencies and spatial correlations among CNF's metrics. Similar observations hold true in Fig. 6 for the testing dataset, where we can see that the video streamer's metrics are accurately predicted in absence of attacks. However, the deviation between the forecast values and the real values are highly pronounced during the application-layer DDoS attack periods, represented by the highlighted red regions in Fig. 6. The results uphold the hypothesis that the prediction error is likely to increase significantly in the presence of abnormal metrics values, making it a strong indicator of an ongoing

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2023.3318606

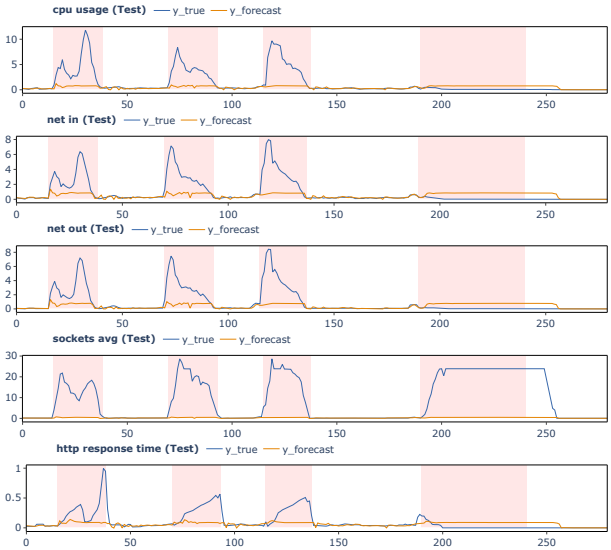


Fig. 6: Forecasting on testing data of the video streamer CNF of vCDN slice 1. The three first highlighted red regions correspond to Hulk attacks, while the last region represents the Slowloris attack.

DDoS attack. Given the high forecasting accuracy of CNF's metrics during normal behavior and the notable increase in prediction error when the CNF is under attack, it can be concluded that CG-GRU model will be capable of effectively detecting application-layer DDoS attacks while reducing the number of false positives, as demonstrated in what follows.

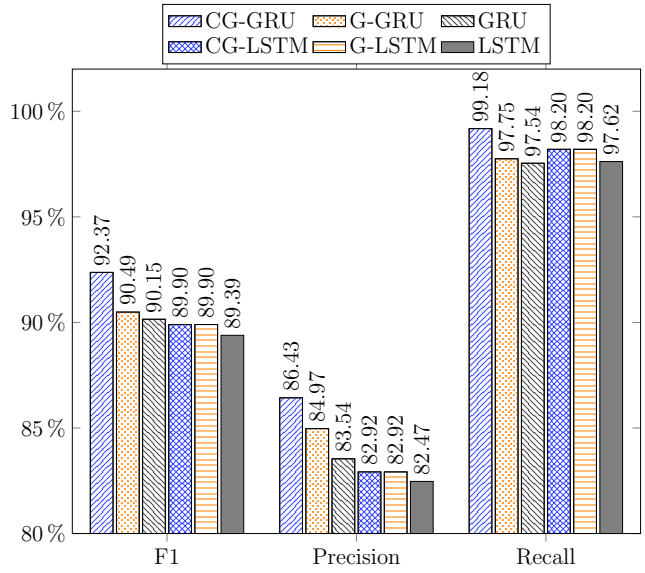


Fig. 7: Attack Detection Performances.

2) Attack Detection Performances: We compare CG-GRU with the LSTM-based multivariate time series anomaly detection model proposed in [36]. Moreover, we conduct a layer ablation study to assess the impact of the features extracted by Conv1D, GAT and GRU layers on the DDoS attack detection performance. To this end, we compare CG-GRU with the following variants: (i) G-GRU, where we remove the Conv1D layer; hence, the inputs to the model are the original data rather than the convolved data; and (ii) GRU, where we remove both 1D convolutional layer and GAT, and only use the temporal information extracted by GRU layers. Similar to G-GRU, GRU model operates on the original data rather than the convolved one. We also transplant the Conv1D layer and/or GAT layers to LSTM model proposed in [36] to evaluate to which extent they can affect the model's performances.

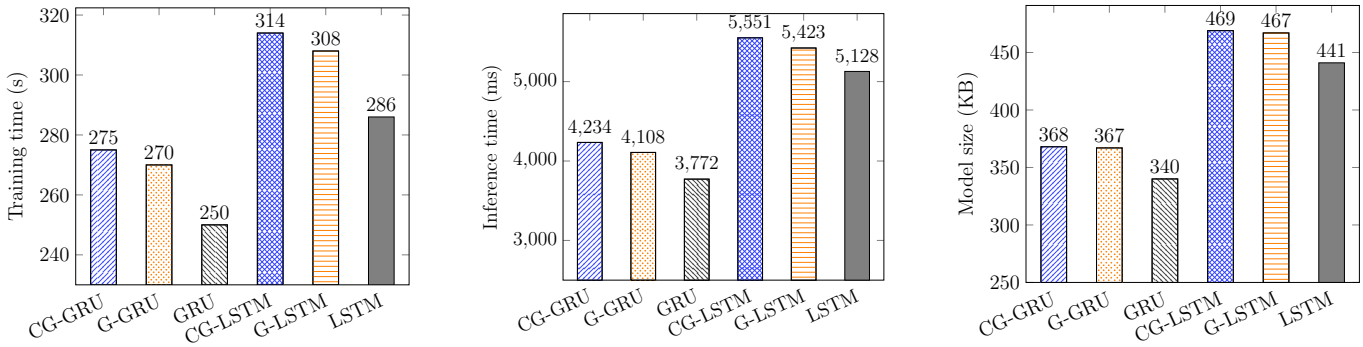
The performances of the different models are evaluated over the testing dataset using Precision, Recall and F1 metrics.

The results depicted in Fig. 7 demonstrate the superiority of CG-GRU model in achieving the highest performance scores compared to all other models. In fact, CG-GRU model exhibits a high sensitivity in identifying anomalous CNF's status while yielding an acceptable Precision of 86.43% and a reasonable F1 score of 92.37%. It is worth mentioning that in our case a high Recall is preferred over a high Precision, as the unsuccessful detection of anomalous CNF's status may lead to economical losses due to accepting resource scaling operations caused by DDoS attacks. Compared with the LSTM-based model proposed in [36], we observe that CG-GRU improves the Precision, Recall and F1 scores by at least 3.96%, 1.56% and 2.98%, respectively. This improvement is attributed to the quality of the spatio-temporal features learned by the feature extraction block, which allows better estimation of the anomaly threshold for discriminating anomalous CNF's status. This statement is corroborated by the results of the ablation study, which not only demonstrate the importance of capturing both spatial and temporal dependencies within the multivariate time series, but also reveal that the local features extracted by Conv1D layer are beneficial to boost further the model performances. Indeed, one-dimensional convolution operation helps to reshape the original input data into a more relevant representation format that is robust to possible noise in the data. The results reported in Fig. 7 show that adding Conv1D layer and GAT layers allows CG-GRU model to outperforms the baseline GRU model, increasing the Precision, Recall and F1 scores by at least 2.89%, 1.64% and 2.22%, respectively.

3) Model Computation & Storage Costs: In this section, we explore the computation and storage overhead induced by CG-GRU model and its counterparts. To this end, we measure the average training time, the average inference speed, and the model size. Fig. 8 reports the comparative results. We can observe that GRU-based models are faster to train and make inference, and require much less storage space than LSTM-based models. This can be attributed to the fewer number of parameters and gates used by GRU cells as compared to LSTM cells to achieve the same task. The results in Fig. 8 show that CG-GRU model brings up to 12.42%, 23.73% and 21.54% reduction in training time, inference time and model size, respectively, compared to the best-performing LSTM-based model (i.e., CG-LSTM).

We can also notice that the higher attack detection performances exhibited by CG-GRU model come at the expense of increased computation and storage complexity compared to the other GRU-based variants, owing to the added 1D Convolutional layer and the GAT layers used for extracting local features and spatial inter-dependencies. Nevertheless, the additional cost remains within an acceptable range of no higher

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2023.3318606



(a) Training time (b) Inference time (c) Model size

Fig. 8: Computation and storage costs of the models.

than 10% , 12.25% and 8.24% in terms of training time, inference time, and model size, respectively.

Hence, considering both attack detection performances and computation/storage cost, CG-GRU model provides the best performance-cost balance. This makes CG-GRU model an adequate solution to achieve accurate and real-time detection of fraudulent resource scaling requests associated to DDoS attacks in a cost-effective way, which translate to better economic sustainability and higher profitability.

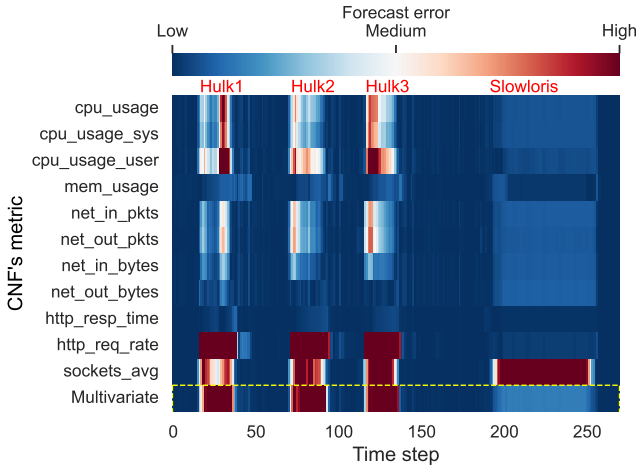


Fig. 9: A heat map visualization of forecasting errors per CNF's metric for interpretability.

4) Attack Interpretability & Root Cause: The ability to explain the decision made by a AI-powered system is instrumental for domain experts to interpret its output and understand the cause-and-effect relationship between the input data and the generated output, allowing to foster trustworthiness in its decisions [43]. In this vein, we use the forecasting error heatmap to elucidate the judgment made by the DDoS Mitigator module about the legitimacy or maliciousness of the observed resource usage and performance metrics of a vCDN's CNF. Using the predictions generated by CG-GRU model and the corresponding actual values of the CNF's metrics, we create a heatmap of the forecasting error per metric (see Eq. 12) and the global forecasting error (see Eq. 13) over the time.

Fig. 9 depicts the forecasting error heatmap of slice 1's video streamer for each time step of the testing dataset, with the x-axis representing the time steps and the y-axis indicating the CNF's metrics. The bottom row in the heat map corresponds to the global forecasting error on the multivariate time series. The forecasting error intensity is color-encoded, with zero error as dark blue and becoming dark red as it gets larger. As shown in Fig. 9, the visualization of the forecasting error heatmap provides tangible and intuitive explanations, enabling the domain experts to ascertain the DDoS Mitigator's decision and to quickly identify the attack patterns and the CNF's metrics that have most contributed to the global forecasting error. We can observe that the largest forecasting errors are recorded during the attacks' periods, with clear segregation between Hulk and Slowloris patterns. The attack patterns in the heatmap can serve as a visual signature of the attack, allowing to determine the CNF's metrics that are impacted by the attack. Those metrics are considered the most important for recognizing the attack. As visualized in Fig 10, we found that the top three CNF's metrics contributing to the global forecasting error when a Hulk attack is underway are (1) HTTP request rate, (2) average number of open sockets, and (3) average CPU usage at the user level, while the top three CNF's metrics impacted by Slowloris attack are (1) average number of open sockets, (2) average number of bytes sent by the CNF, and (3) number of packets received by the CNF. The obtained results are in compliance with the nature of Hulk and Slowloris attacks. In fact, Hulk is a high-rate application-layer DDoS attack that generates a high volume of unique and obfuscated HTTP GET requests, hence the high HTTP request rate, number of open sockets and CPU utilization to process those requests. Meanwhile, Slowloris is a low-rate application-layer DDoS attack that aims to make the service inaccessible by holding multiple connections open a long time, which explains the high effect on the number of open sockets.

The difference between Hulk and Slowloris behaviors is further corroborated by the global forecasting error experienced during the attack period. As shown in Fig. 9 and Fig. 10,

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2023.3318606

the global forecasting error is more pronounced during Hulk attack. In our experiments, an average global forecasting error of 18.95 and 1.06 is caused by Hulk and Slowloris attack, respectively. It is worth mentioning that despite the stealthiness of Slowloris attack, our solution was able to recognize it.

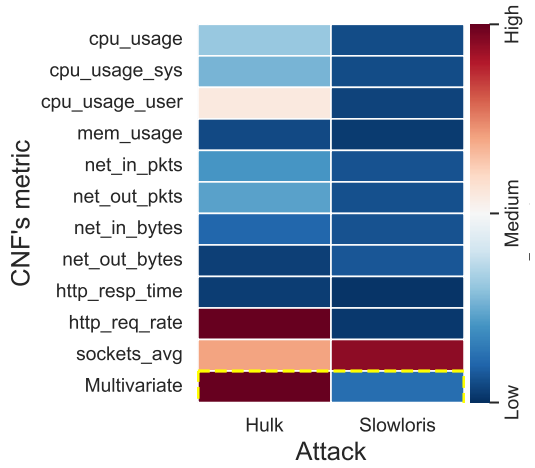


Fig. 10: The visual signature of Hulk and Slowloris attacks using the forecast errors heat map.

5) Effectiveness of Transfer Learning: To test the effectiveness of applying transfer learning in terms of both attack detection performances and training time, we transfer the CG-GRU model trained on data collected from the video streamer CNF of vCDN slice 1 to a newly deployed video streamer CNF of vCDN slice 2. For brevity, let *vStreamer1* and *vStreamer2* denote the video streamer CNF of vCDN slice 1 and vCDN slice 2, respectively. Unlike vStreamer1, only few interactions have been performed between the simulated legitimate users (through our normal load generator) and vStreamer2 and with access to the same video file. Hence, the training dataset collected from vStreamer2 is not representative of a normal behavior.

To build the CG-GRU model for vStreamer2 using transfer learning, denoted as TL-CG-GRU, we freeze the weights in 1D Convolutional, GAT and GRU layers and only fine-tune the fully-connected layers on vStreamer2's training dataset. For comparison, we train another CG-GRU model for vStreamer2 from scratch using its training dataset (hereafter denoted as CG-GRU-vS2). Furthermore, the performances of TL-CG-GRU are compared against those of using directly the vStreamer1's model without fine-tuning (hereafter referred to as CG-GRU-vS1). This allows to assess the importance of model adaptation to data in the target domain (vStreamer2's training data in our case).

Fig. 11 reports the attack detection performance indicators (i.e., F1, Precision and Recall) over Streamer2's testing dataset as well as the training time required to build the model. The results demonstrate the superiority of the transferred model TL-CG-GRU in boosting the detection performance while considerably reducing the training overhead.

We observe that while both CG-GRU-vS1 and CG-GRU-vS2 can identify anomalous vStreamer2's status with an acceptable sensitivity of above 91%, they generate a large number of false alarms. This low precision is expected as

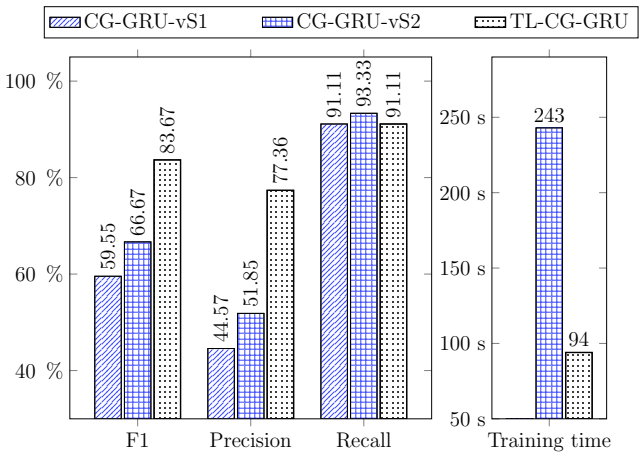


Fig. 11: The performances and training time of CG-GRU model with and without transfer learning.

CG-GRU-vS1 has not seen the training data of vStreamer2 and due to the fact that CG-GRU-vS2 has been trained from scratch with vStreamer2's data which lack representativity of a video streamer CNF's normal behavior. By embedding the prior knowledge (features) on normal behavior learned from vStreamer1's model and updating the weights of the forecasting layers to adapt to the specific vStreamer2's data, TL-CG-GRU has considerably lowered the number of false positives, achieving a gain of 32.79% and 25.51% in precision compared to CG-GRU-vS1 and CG-GRU-vS1, respectively. This yields a significantly increased overall attack detection performance greater than 83%, resulting in up to 24.12% and 17% improvement compared to CG-GRU-vS1 and CG-GRU-vS2, respectively. The obtained results support our idea that the spatio-temporal features derived by the feature extraction layers are more generic and therefore can be transferred among CNFs of different slices, while the weights of the forecasting layers are more specific to the source CNF's behavior, and have to be retrained after transfer to adapt to the target CNF's behavior.

In addition to its effectiveness in discriminating anomalous CNF's status caused by application-layer DDoS attacks, TL-CG-GRU substantially speeds up the training process, decreasing the time required to train the CG-GRU-vS2 model from scratch by at least 61%. The computational efficiency of TL-CG-GRU is attributed to reuse of knowledge regarding feature representations, which allows to reduce the number of model's parameters (weights of forecasting layers in our case) to update during the fine-tuning phase.

V. Conclusion

In this paper, we proposed FortisEDoS, a novel framework for enabling highly elastic B5G services while being immune to EDoS attacks. FortisEDoS achieves its goal by (i) integrating CG-GRU, a new DL-powered DDoS anomaly detection model which exploits the forecasting errors between the observed VNF's metrics and the predicted ones to determine malicious VNF scaling requests due to stealthy application-layer DDoS attacks; and (ii) adopting the concept of transfer

learning to yield effective detection of EDoS attack in newly deployed slices. The experimental results demonstrated the superior performance of the proposed solution in accurately detecting EDoS attack and confirmed the benefit of transfer learning in boosting both attack detection effectiveness and training speed when representative historical data of normal behavior are scare. The explainability of decisions made about the legitimacy or maliciousness of a VNF's status using forecasting errors heatmaps is another key capability provided by FortisEDoS to foster trust in its decisions. Moreover, adopting a forecasting-based anomaly detection approach makes CG-GRU a multi-purpose model that can serve not only for application-layer DDoS anomaly detection task but also for proactive and dynamic resource allocation tasks, allowing to reduce the cost of training and running several models serving different tasks.

Given the potential occurrence of false alarms, preventing a scaling up operation when the VNF's status is flagged as anomalous may result in a reverse effect, yielding negative impact on service level performances. To deal with this challenge, we intend to extend the capabilities of FortisEDoS to provide a mitigation strategy that intelligently decide when and how much of resources to be provisioned so that the required SLA is guaranteed while minimizing the EDoS attack damage. Another avenue of research is to devise an advanced mechanism for selecting the appropriate VNFs/slices for knowledge transfer. Furthermore, combining transfer learning with federated learning to empower privacy-preserving EDoS mitigation is another interesting research direction.

ACKNOWLEDGMENT

This work was supported in part by the Academy of Finland Project 6Genesis Flagship (Grant No. 346208), the EU's Horizon 2020 research and innovation programme under the INSPIRE-5Gplus project (Grant No. 871808), and the EU's HE research and innovation 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 it contains.

REFERENCES

[1] C. Benzaid, T. Taleb, and M. Z. Farooqi, "Trust in 5G and Beyond Networks," *IEEE Network Magazine*, vol. 35, no. 3, pp. 212 – 222, May 2021.

[2] 3GPP TS 23.501 V17.5.0, "System Architecture for the 5G System (5GS); Stage 2 (Release 17)," June 2022.

[3] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network Slicing and Softwarization: A Survey on Principles, Enabling Technologies, and Solutions," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2429 – 2453, March 2018.

[4] A. Bremler-Barr, E. Brosh, and M. Sides, "DDoS Attack on Cloud Auto-Scaling Mechanisms," in *Proc. of the IEEE Int. Conf. on Computer Communications (INFOCOM)*, May 2017, pp. 1 – 9.

[5] NGMN, "5G Security Recommendations Package #2: Network Slicing," Apr. 2016.

[6] T. Taleb, P. Frangoudis, I. Benkacem, and A. Ksentini, "CDN Slicing over a Multi-Domain Edge Cloud," *IEEE/ACM Trans. Mobile Computing*, vol. 19, no. 9, pp. 2010 – 2027, Sep. 2020.

[7] R. Sedar, F. V.-G. C. Kalalas, L. Alonso, and J. Alonso-Zarate, "A Comprehensive Survey of V2X Cybersecurity Mechanisms and Future Research Paths," *IEEE Open Journal of the Communications Society*, vol. 4, pp. 325 – 391, Jan. 2023.

[8] U. Inayat, S. M. M. F. Zia, H. M. Khalid, and M. Benbouzid, "Learning-based Methods for Cyber Attacks Detection in IoT Systems: A Survey on Methods, Analysis, and Future Prospects," *Electronics*, vol. 11, no. 9, pp. 1 – 20, May 2022.

[9] A. S. Musleh, H. M. Khalid, S. M. Muyeen, and A. Al-Durra, "A Prediction Algorithm to Enhance Grid Resilience Toward Cyber Attacks in WAMCS Applications," *IEEE Systems Journal*, vol. 13, no. 1, pp. 710 – 719, March 2019.

[10] C. Benzaid, M. Boukhalfa, and T. Taleb, "Robust Self-Protection Against Application-Layer (D)DoS Attacks in SDN Environment," in *Proc. of IEEE Wireless Communications and Networking Conf. (WCNC)*, May 2020, pp. 1 – 6.

[11] Z. Kotulski, T. Nowak, M. Sepczuk, M. Tunia, R. Artych, and et al., "On End-to-End Approach for Slice Isolation in 5G Networks. Fundamental Challenges," in *Proc. of the Federated Conf. Comput. Sci. Inf. Syst.*, Sep. 2017, pp. 783 – 792.

[12] D. Sattar and A. Matrawy, "Towards Secure Slicing: Using Slice Isolation to Mitigate DDoS Attacks on 5G Core Network Slices," in *Proc. of the 2019 IEEE Conf. on Communications and Network Security (CNS)*, June 2019, pp. 82 – 90.

[13] C. Benzaid, T. Taleb, and J. Song, "AI-based Autonomic & Scalable Security Management Architecture for Secure Network Slicing in B5G," *IEEE Network Magazine*, vol. 36, no. 6, pp. 165 – 174, Nov./Dec. 2022.

[14] A. Thantharate, R. Paropkari, V. Walunj, C. Beard, and P. Kankariya, "Secure5G: A Deep Learning Framework Towards a Secure Network Slicing in 5G and Beyond," in *Proc. of the 10th Annual Computing and Communication Workshop and Conf. (CCWC)*, Jan. 2020, pp. 0852 – 0857.

[15] M. P. Novaes, L. F. Carvalho, J. Lloret, and M. L. Proença, "Adversarial Deep Learning Approach Detection and Defense against DDoS Attacks in SDN Environments," *Future Generation Computer Systems*, vol. 125, pp. 156 – 167, Dec. 2021.

[16] Z. Li, H. Jin, D. Zou, and B. Yuan, "Exploring New Opportunities to Defeat Low-Rate DDoS Attack in Container-Based Cloud Environment," *IEEE Trans. on Parallel and Distributed Systems*, vol. 31, no. 3, pp. 695 – 706, March 2020.

[17] R. S. Silva, C. C. Meixner, R. S. Guimarães, T. Diallo, B. O. Garcia, L. F. M. de Moraes, and M. Martinello, "REPEL: A Strategic Approach for Defending 5G Control Plane From DDoS Signalling Attacks," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 3231 – 3243, Sept. 2021.

[18] P. T. Dinh and M. Park, "R-EDoS: Robust Economic Denial of Sustainability Detection in an SDN-Based Cloud Through Stochastic Recurrent Neural Network," *IEEE Access*, vol. 9, pp. 35 057 – 35 074, Feb. 2021.

[19] ——, "Economic Denial of Sustainability (EDoS) Detection using GANs in SDN-based Cloud," in *Proc. of the 2020 IEEE Eighth Int. Conf. on Communications and Electronics (ICCE)*, Jan. 2021, pp. 135 – 140.

[20] ——, "Dynamic Economic-Denial-of-Sustainability (EDoS) Detection in SDN-based Cloud," in *In Proc. of the Fifth International Conf. on Fog and Mobile Edge Comput. (FMEC)*, Apr. 2020, pp. 62 – 69.

[21] A. Praseed and P. S. Thilagam, "DDoS Attacks at the Application Layer: Challenges and Research Perspectives for Safeguarding Web Applications," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 661 – 685, Feb. 2019.

[22] N. Agrawal and S. Tapaswi, "Defense Mechanisms Against DDoS Attacks in a Cloud Computing Environment: State-of-the-Art and Research Challenges," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3769 – 3795, Fourthquarter 2019.

[23] W. Zhijun, L. Wenjing, L. Liang, and Y. Meng, "Low-Rate DoS Attacks, Detection, Defense, and Challenges: A Survey," *IEEE Access*, vol. 8, pp. 43 920 – 43 943, Feb. 2020.

[24] N. Tripathi and N. Hubballi, "Application Layer Denial-of-Service Attacks and Defense Mechanisms: A Survey," *ACM Comput. Surv.*, vol. 54, no. 4, pp. 86:1 – 86:33, Apr. 2021.

[25] N. A. E. Kuadey, G. T. Maale, T. Kwantwi, G. Sun, and G. Liu, "DeepSecure: Detection of Distributed Denial of Service Attacks on 5G Network Slicing – Deep Learning Approach," *IEEE Wireless Communications Letters*, vol. 11, no. 3, pp. 488 – 492, March 2022.

[26] J. A. Pérez-Díaz, I. A. Valdovinos, K. K. R. Choo, and D. Zhu, "A Flexible SDN-Based Architecture for Identifying and Mitigating Low-Rate DDoS Attacks Using Machine Learning," *IEEE Access*, vol. 8, pp. 155 859 – 155 872, Aug. 2020.

[27] G. Somani, M. S. Gaur, D. Sanghi, M. Conti, and M. Rajarajan, "Scale Inside-Out: Rapid Mitigation of Cloud DDoS Attacks," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 6, pp. 959 – 973, Nov.-Dec. 2018.

[28] M. A. S. Monge, J. M. Vidal, and G. M. Perez, "Detection of Economic Denial of Sustainability (EDoS) Threats in Self-Organizing Networks," *Computer Communications*, vol. 145, pp. 284 – 308, Sept. 2019.

[29] R. B. David and A. Bremler-Barri, "Kubernetes Autoscaling: YoYo Attack Vulnerability and Mitigation," in *Proc. of the 11th Int. Conf. on Cloud Computing and Services Science (CLOSER)*, Apr. 2021, pp. 34 – 44.

[30] G. Somani, M. S. Gaur, and D. Sanghi, "DDoS/EDoS Attack in Cloud: Affetcting Everyone out There!" in *Proc. of the 8th Int. Conf. on Security of Information and Networks (SIN'15)*, Sept. 2015, pp. 169 – 176.

[31] I. Ozçelik and R. R.Brooks, "Deceiving Entropy based DoS Detection," *Computers & Security*, vol. 48, pp. 234 – 245, Feb. 2015.

[32] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, "Robust Anomaly Detection for Multivariate Time Series through Stochastic Recurrent Neural Network," in *Proc. of the 25th ACM SIGKDD International Conf. on Knowledge Discovery & Data Mining*, July 2019, pp. 2828 – 2837.

[33] M. R. Leadbetter, "On a Basis for 'Peaks over Threshold' Modeling," *Statist. Probab. Lett.*, vol. 12, pp. 357 – 362, Oct. 1991.

[34] J. Yu, Y. Song, D. Tang, D. Han, and J. Dai, "Telemetry Data-Based Spacecraft Anomaly Detection With Spatial–Temporal Generative Adversarial Networks," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1 – 9, Apr. 2021.

[35] D. Li, D. Chen, B. Jin, L. Shi, J. Goh, and S. Ng, "MAD-GAN: Multivariate Anomaly Detection for Time Series Data with Generative Adversarial Networks," in *Proc. of the 28th Int. Conf. on Artificial Neural Networks and Machine Learning – ICANN 2019: Text and Time Series*, Sept. 2019, pp. 703 – 716.

[36] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom, "Detecting Spacecraft Anomalies using LSTMs and Nonparametric Dynamic Thresholding," in *Proc. of the 24th ACM SIFKDD Int. Conf. on Knowledge Discovery & Data Mining*, Jul. 2018, pp. 387 – 395.

[37] Ericsson, "Ericsson Mobility Report," Jun. 2022.

[38] C. Benzaid, T. Taleb, A. Sami, and O. Hireche, "A Deep Transfer Learning-powered EDoS Detection Mechanism for 5G and Beyond Network Slicing," in *IEEE Global Communications Conf. (Under Review)*, Dec. 2023.

[39] GSMA, "NG. 116 - Generic Network Slice Template, Version 5.0," Jun. 2021, p. 68.

[40] Y. Zhang, B. Zhou, X. Cai, W. Guo, X. Ding, and X. Yuan, "Missing Value Imputation in Multivariate Time Series with End-to-End Generative Adversarial Networks," *Information Sciences*, vol. 551, pp. 67 – 82, Apr. 2021.

[41] C. Benzaid and T. Taleb, "AI for Beyond 5G Networks: A Cyber-Security Defense or Offense Enabler?" *IEEE Network Magazine*, vol. 34, no. 6, pp. 140 – 147, Nov./Dec. 2020.

[42] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," in *Proc. of the 6th International Conf. on Learning Representations (ICLR)*, Apr./May 2018, pp. 1 – 12.

[43] C. Benzaid and T. Taleb, "AI-driven Zero Touch Network and Service Management in 5G and Beyond: Challenges and Research Directions," *IEEE Network Magazine*, vol. 34, no. 2, pp. 186 – 194, Mar./Apr. 2020.

[44] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, "On the Properties of Neural Machine Translation: Encoder–Decoder Approaches," in *Proc. of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, Oct. 2014, pp. 103–111.

[45] J. S. Hunter, "The Exponentially Weighted Moving Average," *Journal of Quality Technology*, vol. 18, no. 4, pp. 203 – 210, Oct. 1986.

[46] X. N. Lam, T. Vu, T. D. Le, and A. D. Duong, "Addressing Cold-Start Problem in Recommendation Systems," in *Proc. of the 2nd Int. Conf. on Ubiquituous Information Management and Communication (ICUIMC'08)*, Jan. 2008, pp. 208 – 211.

[47] C. T. Nguyen, N. V. Huynh, N. H. Chu, Y. M. Saputra, D. T. Hoang, D. N. Nguyen, Q.-V. Pham, D. Niyato, E. Dutkiewicz, and W.-J. Hwang, "Transfer Learning for Wireless Networks: A Comprehensive Survey," *Proceedings of the IEEE*, vol. 110, no. 8, pp. 1073 – 1115, Aug. 2022.

[48] A. Y. Khinchin, *Mathematical Methods in the Theory of Queuing*. Dover Publications, 2013.

[49] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, "Youtube Traffic Characterization: A View from the Edge," in *Proc. of the 7th ACM SIGCOMM Conf. on Internet Measurement (IMC'07)*, Oct. 2007, pp. 15 – 28.

[50] S. S. Salunke, *Selenium Webdriver in Python: Learn with Examples*. CreateSpace Independent Publishing Platform, 2014.

[51] H. Yu, D. Zheng, B. Y. Zhao, and W. Zheng, "Understanding User Behavior in Large-Scale Video-on-Demand Systems," *ACM SIGOPS Operating Systems Review*, vol. 40, no. 4, pp. 333 – 344, Oct. 2006.

[52] P. Liashchynskyi and P. Liashchynskyi, "Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS," *arXiv*, vol. abs/1912.06059, 2019.

[53] L. Li, K. G. Jamieson, A. Rostamizadeh, E. Gonina, J. Ben-tzur, M. Hardt, B. Recht, and A. S. Talwalkar, "A System of Massively Parallel Hyperparameter Tuning," in *Proc. of the 3rd Machine Learning and Systems Conf. (MLSys)*, March 2020, pp. 230 – 246.

Chafika Benzaïd is currently a senior research fellow at Faculty of Information Technology and Electrical Engineering, University of Oulu, Finland. Between Nov. 2018 and Dec. 2021, she has been senior researcher at School of Electrical Engineer, Aalto University, Finland. Prior to that, she was working as an associate professor at Computer Science Department, University of Sciences and Technology Houari Boumediene (USTHB), Algeria. She received her Engineer degree in software engineering with distinction, Magister and "Doctorat ès Sciences" degrees in Programming & Systems from USTHB, in 2000, 2003 and 2009, respectively. Dr Benzaid's research interest lies in the field of 5G/6G, SDN, Network Security, AI Security, and AI/ML for zero-touch security management. She is an ACM professional member. She serves/served as a TPC chair and member for several international conferences and as a reviewer for several international journals.

Tarik Taleb is currently a Professor at University of Oulu, Finland. He is the founder and director of the MOSA!C Lab (www.mosaic-lab.org). Between Oct. 2014 and Dec. 2021, he was a Professor at Aalto University. Prior to that, he was a senior researcher and 3GPP standards expert at NEC Europe Ltd., Germany. He also worked as assistant professor at Tohoku University, Japan. He holds a B.E. degree in information engineering, and M.Sc. & Ph.D. degrees in information sciences from Tohoku University. His research interests lie in the field of Telco Cloud.

Ashkan Sami is a Professor of Computer Science at Edinburgh Napier University and head of Computer Science subject group which is one of the largest computer science departments in U.K. Previously, Ashkan was the head of CSE and IT department and professor at Shiraz University. Ashkan teaches and conducts research on Cyber Security, Empirical Software Engineering, Applied AI and Data Science. He obtained his B.S. from Virginia Tech; U.S.A. and PhD in 2006 from Tohoku University, where his PhD became a Japanese national project and earned him a tenured faculty position at Tohoku University; Japan. He has led various interdisciplinary and transdisciplinary research teams which focuses on themes of current social problems to, create products or services and publishes in quality venues. Dr. Sami's research has been presented in media outlets like BBC Technology, The Register and professional sites like Stack Exchange blogs.

Othmane Hireche eceived his Engineer and Master degrees from the University of Science and Technology Houari Boumediene (USTHB), Algeria, in 2009 and 2011, respectively. He is currently an IT specialist - Infrastructure at Nanoform, Finland and working toward the PhD degree at USTHB. Between 2019 and 2022, he has been a Project Engineer at MOSA!C Lab, Aalto University, Finland. Prior to that, he was a Research Support Engineer with the Research Center on Scientific and Technical Information (CERIST), Algeria. His research interests include self-driving networks, programmable data planes, blockchain, and AI-based zero touch security management.