THESE DE DOCTORAT DE L'ETABLISSEMENT UNIVERSITE BOURGOGNE FRANCHE-COMTE

PREPAREE A L'UNIVERSITE DE BOURGOGNE

Ecole doctorale n°37

Sciences Pour L'ingénieur et Microtechniques

Doctorat en Informatique

Par

Taki Eddine Toufik DJAIDJA

Advancing the Security of 5G and Beyond Vehicular Networks through AI/DL

Thèse présentée et soutenue à Nevers, le 21/02/2024.

Composition du Jury :

M, Beylot, André-Luc	Professeur, Institut de Recherche en Informatique de Toulouse	Président du jury
Mme, Bouzefrane, Samia	Professeur, Conservatoire National des Arts et Métiers	Rapporteur
M, Mosbah, Mohamed	Professeur, Université de Bordeaux	Rapporteur
M, Djahel, Soufiene	Professeur, Coventry University	Examinateur
M, Senouci, Sidi-Mohammed	Professeur, Université Bourgogne Franche-Comté	Directeur de thèse
M, Ghamri-Doudane, Yacine	Professeur, Université La Rochelle	Codirecteur de thèse
M, Brik, Bouziane	Maître de conférence, Université Bourgogne Franche-Comté	Encadrant
M, Boualouache, Abdelwahab	Chercheur, Université de Luxembourg	Encadrant

Titre: Sécuriser les réseaux 5G véhiculaire et au-delà grâce à l'IA/AP

Mots clés: 5G-V2X, Systèmes de Détection d'Intrusion, Intelligence Artificielle, Apprentissage Profond

Résumé: L'émergence des réseaux de cinquième génération (5G) et des réseaux véhiculaire (V2X) a ouvert une ère de connectivité et de services associés sans précédent. Ces réseaux permettent des interactions fluides entre les véhicules, l'infrastructure, et bien plus encore, en fournissant une gamme de services à travers des tranches de réseau (slices), chacune adaptée aux besoins spécifiques de ceux-ci. Les générations futures sont même censées apporter de nouvelles avancées à ces réseaux. Cependant, ce progrès remarquable les expose à une multitude de menaces en matière de cybersécurité, dont bon nombre sont difficiles à détecter et à atténuer efficacement avec les contre mesures actuelles. Cela souligne la nécessité de mettre en oeuvre de nouveaux mécanismes avancés de détection d'intrusion pour garantir l'intégrité, la confidentialité et la disponibilité des données et des services.

Un domaine suscitant un intérêt croissant à la fois dans le monde universitaire qu'industriel est l'Intelligence Artificielle (IA), en particulier son application pour faire face aux menaces en cybersécurité. Notamment, les réseaux neuronaux (RN) ont montré des promesses dans ce contexte, même si les solutions basées sur l'IA sont accompagnées de défis majeurs.

Ces défis peuvent être résumés comme des préoccupations concernant l'efficacité et l'efficience. Le premier concerne le besoin des Systèmes de Détection d'Intrusions (SDI) de détecter avec précision les menaces, tandis que le second implique d'atteindre l'efficacité en termes de temps et la détection précoce des menaces.

Cette thèse représente l'aboutissement de nos recherches sur l'investigation des défis susmentionnés des SDI basés sur l'IA pour les systemes 5G en général et en particulier 5G-V2X. Nous avons entamé notre recherche en réalisant une revue de la littérature existante. Tout au long de cette thèse, nous explorons l'utilisation des systèmes d'inférence floue (SIF) et des RN, en mettant particulièrement l'accent sur cette derniere technique. Nous avons utilisé des techniques de pointe en apprentissage, notamment l'apprentissage profond (AP), en intégrant des réseaux neuronaux récurrents et des mécanismes d'attention. Ces techniques sont utilisées de manière innovante pour réaliser des progrès significatifs dans la résolution des préoccupations liées à l'amélioration de l'efficacité et de l'efficience des SDI. De plus, nos recherches explorent des défis supplémentaires liés à la confidentialité des données lors de l'utilisation des SDIs basés sur l'AP. Nous y parvenons en exploitant les algorithmes d'apprentissage fédéré (AF) les plus récents.

Title: Advancing the Security of 5G and Beyond Vehicular Networks through AI/DL

Keywords: 5G-V2X, IDS, AI, DL

Abstract: The emergence of Fifth Generation (5G) and Vehicle-to-Everything (V2X) networks has ushered in an era of unparalleled connectivity and associated services. These networks facilitate seamless interactions among vehicles, infrastructure, and more, providing a range of services through network slices, each tailored to specific requirements. Future generations are even expected to bring further advancements to these networks. However, this remarkable progress also exposes them to a myriad of security threats, many of which current measures struggle to detect and mitigate effectively. This underscores the need for advanced intrusion detection mechanisms to ensure the integrity, confidentiality, and availability of data and services.

One area of increasing interest in both academia and industry spheres is Artificial Intelligence (AI), particularly its application in addressing cybersecurity threats. Notably, neural networks (NNs) have demonstrated promise in this context, although AI-based solutions do come with inherent challenges. These challenges can be summarized as concerns about effectiveness and efficiency. The former pertains to the need for Intrusion Detection Systems (IDSs) to accurately detect threats, while the latter involves achieving time efficiency and early threat detection.

This dissertation represents the culmination of our research findings on investigating the aforementioned challenges of AI-based IDSs in 5G systems in general and 5G-V2X in particular. We initiated our investigation by conducting a comprehensive review of the existing literature. Throughout this thesis, we explore the utilization of Fuzzy Inference Systems (FISs) and NNs, with a specific emphasis on the latter. We leveraged state-of-the-art NN learning, referred to as Deep Learning (DL), including the incorporation of recurrent neural networks and attention mechanisms. These techniques are innovatively harnessed to making significant progress in addressing the concerns of enhancing the effectiveness and efficiency of IDSs. Moreover, our research delves into additional challenges related to data privacy when employing DL-based IDSs. We achieve this by leveraging and experimenting state-of-the-art federated learning (FL) algorithms.

UBFC UNIVERSITE BOURGOGNE FRANCHE-COMTE

Université Bourgogne Franche-Comté
32, avenue de l'Observatoire
25000 Besançon, France

To my beloved parents, Kaddour and Wahiba;
To my sisters and brothers;
And to my Aunt Farida.

Acknowledgements

I would like to start by giving absolute praise and glory to *God*. I consider myself blessed and fortunate to witness the successful completion of this dissertation.

A special thanks to my supervisor, Pr. Sidi-Mohammed Senouci, for granting me the opportunity to pursue my doctorate and for his consistent guidance and contributions.

I'm also deeply grateful to Pr. Yacine Ghamri-Doudane, my co-supervisor, and to Dr. Bouziane BRIK and Dr. Abdelwahab Boualouache, my advisors, for their encouragement and invaluable insights. Your guidance has been indispensable.

I extend sincere thanks to the esteemed members of the thesis commitee Prof. Samia Saad-Bouzefrane , Prof. Mohamed Mosbah , Prof. André-Luc Beylot and Prof. Soufiene Djahel for accepting to evaluate this thesis and to be part of the jury.

To all who stood by me throughout this journey. I'm truly grateful for your support.

Contents

List of Figures

List of Tables

Acronyms

3GPP 3^{rd} Generation Partnership Project. 12, 13, 15

5G-V2X 5G Vehicle-to-Everything. 1, 2, 5, 12, 26, 35, 36, 39, 40, 44, 50, 52, 60, 61, 107

AE Auto Encoders. 67

AI Artificial Intelligence. 2–6, 11, 16–19, 23, 25, 28–31, 67, 107, 109, 110

AIDS Application-based Intrusion Detection System. 2, 4, 6, 27, 30, 47, 49–53, 59, 61, 108

AS Application Server. 2, 4, 14, 15, 25, 26, 29, 62, 65

BP Backpropagation. 23, 113

BPTT Backpropagation Through Time. 23

C-I-A Confidentiality, Integrity, and Availability. xiii, 1, 3, 4, 7, 25, *Glossary:* C-I-A

C-V2X Cellular Vehicle-to-Everything. 12, 15

CAM Cooperative Awareness Message. xi, 4, 16, 26, 30, 36, 39, 40, 44, 45, 47, 49–53, 59–61, 108

CBR Channel Busy Ratio. 42, 43

CDF Cumulative Distribution Function. xi, 45, 46

CN Core Network. 13–15, 25, 26

CNN Convolutional Neural Network. 51, 61, 67–69

CPU Central Processing Unit. 27, 28, 30

DDoS Distributed Denial-of-Service. 26, 29, 76, 79

DENM Decentralized Environmental Notification Messages. 16, 37, 108

DL Deep Learning. xii, 2–6, 22, 28, 29, 50, 51, 53, 58, 65, 68–70, 72–75, 79, 85–87, 89, 91–93, 95, 97–99, 103, 109

DoS Denial-of-Service. 25, 26, 29, 30, 76, 94

DPI Deep Packet Inspection. 29

DSRC Dedicated short-range communication. 11, 12

DT Deision Trees. 18, 67

E2E End-to-End. 15

eMBB Enhanced Mobile Broadband. 14, 15

ETSI European Telecommunications Standards Institute. 11, 13, 110

FedAvg Federated Average. xiii, 5, 85, 86, 88, 89, 91, 92, 100, 102, 103, 105, 108

FedPer Federated Personalizing. xii, 5, 85–92, 95, 100, 102, 103, 106, 108

FedProx Federated with the Proximal term. xiii, 5, 85–89, 97, 100, 102, 103, 105

FFNN Feed-Forward Neural Network. 20, 21, 54, 72, 74, 95

FIS Fuzzy Inference System. xi, 4–6, 17–19, 23, 28, 35, 37, 41, 42, 47, 107

FL Federated Learning. 5, 6, 23, 85–89, 91–93, 99, 100, 102, 103, 108, 109

FN False Negative. 23, 115, 116, *Definition:* FN

FNR False Negative Rate. 23, 58, *Definition:* FNR

FP False Positive. 23, 115, 116, *Definition:* FP

FPR False Positive Rate. 23, 58, *Definition:* FPR

GD Gradient Descent. 23, 55

GDPR General Data Protection Regulation. 86

GPS Global Positioning System. 52, 53

GRU Gated Recurrent Unit. xi, xiii, 21, 22, 50, 54, 55, 57–59, 61, 67, 68, 73, 77, 79, 91, 107, 108, 111, 112

HIDS Host-based Intrusion Detection System. 2–4, 6, 27, 30, 35, 37, 47, 61, 107

HTTP Hypertext Transfer Protocol. 25

IDS Intrusion Detection System. xi, 2, 5, 11, 24–31, 80, 81, 85, 89, 92, 107–110

IID Independent and Identically Distributed. xii, 85, 87, 92, 95, 97–103

IoT Internet of Things. 91

IP Internet Protocol. 29, 30, 70–72, 76

KNN K-Nearest Neighbors. 18, 28, 29, 67, 68

LogReg Logistic Regression. 67

LR Learning Rate. xxi, 23, 54, 56, 57, 77, 87, 97

LSTM Long Short-Term Memory. xi, xiii, 21, 22, 50, 51, 54, 55, 57–59, 61, 67–69, 73, 77, 78, 80, 91, 107, 108, 111, 112

LTE Long-Term Evolution. 12, 15

LTE-V2X Long-Term Evolution Vehicle-to-Everything. 12, 35, 36

LuST Luxembourg SUMO Traffic. 52

MAE Mean Absolute Error. 23, *Definition:* MAE

MIMO Multiple Input Multiple Output. 13

ML Machine Learning. 18, 19, 28, 29, 50, 51, 68–70, 89, 91–93

MLP Multi-Layer Perceptron. xiii, 21, 56–58, 60, 67, 68

mMTC Massive Machine-Type Communications. 14

MSE Mean Squared Error. 23, *Definition:* MSE

NB Naive Bayes. 18, 50, 51, 67

NF Network Function. 3, 13, 14, 29, 92

NFV Network Function Virtualization. 11, 14, 29, 110

NIDS Network-based Intrusion Detection System. xi, xii, 2–6, 27, 29, 30, 62, 65–75, 77, 79–83, 85–87, 91, 93, 94, 103, 106, 108–110

NN Neural Network. ix, xiii, xxi, 4–6, 17–23, 28, 29, 68, 69, 75, 95, 96, 107, 108, 110, 113, 115, 116

Non-IID Non-Independent and Non-Identically Distributed. xii, 85–88, 92, 95, 98, 100, 101, 103, 104

NPV Negative Predictive Value. 23, 58, *Definition:* NPV

NR-V2X New Radio Vehicle-to-Everything. 12

NS Network Slice. 13–16, 26, 29, 30, 65, 83, 85–87, 92, 94, 95, 103, 108–110, 113

NWDAF NetWork Data Analytics Function. 13, 92, 93

OSI Open Systems Interconnection. xiii, 3, 4, 7

PCAP Packet CAPture. 70

PPV Positive Predictive Value. 23, 58, 103, 105, 115, *Definition:* PPV

ProSe Proximity Services. 15

PRR Packet Reception Ratio. xi, 44–46

QoS Quality of Service. 13

RAN Radio Access Network. 13–15, 25, 26

RC Re-selection Counter. 36, 37, 39, 41–43, 46

relu Rectified Linear Unit. 20, *Definition:* relu

RF Random Forest. 18, 50, 51, 67, 68

RNN Recurrent Neural Network. xi, xii, xxi, 4–6, 20, 21, 23, 49–51, 53–56, 58–61, 65–67, 69, 70, 72–75, 79–82, 91, 107, 108, 111

RRI Resource Reservation Interval. 36

RSSI Received Signal Strength Indicator. 30, 52, 53

RSU Road Side Unit. 15, 51

RU Ressource Unit. 36–39, 41–44

SCI SL Control Information. 36, 38, 39

SDN Software Defined Networking. 11, 14, 26, 29

SGD Stochastic Gradient Descent. 23, 54, 56, 75, 77, 97

SL SideLink. 2–4, 6, 15, 25, 26, 30, 35, 38, 39, 49, 52, 107, 110

SPAT Signal Phase and Timing. 16, 108

SPS Semi-Persistent Scheduling. xi, 4, 35–37, 42, 44, 46, 47

SVM Support Vector Machines. 18, 28, 29, 50, 51, 67

TCP Transmission Control Protocol. 29, 68, 71, 72, 76, 78, 79

TN True Negative. 23, 115, 116, *Definition:* TN

TNR True Negative Rate. 23, 58, *Definition:* TNR

TP True Positive. 23, 58, 103, 105, 115, 116, *Definition:* TP

TPR True Positive Rate. 23, 58, 103, 105, 115, *Definition:* TPR

TTL Time-To-Live. 71

UDP User Datagram Protocol. 71, 72, 76, 82

UPF User Plane Function. 13, 29, 65, 92

uRLLC Ultra-Reliable Low Latency Communications. 14, 15

V2I Vehicle-to-Infrastructure. 15, 49, 51

V2N Vehicle-to-Network. 15, 25, 26, 30

V2P Vehicle-to-Pedestrian. 15, 49

V2V Vehicle-to-Vehicle. 15, 35, 47, 49

V2X Vehicle-to-Everything. 1–6, 11–13, 15, 16, 25, 26, 29, 30, 36, 38, 49, 61, 107–110

VAE Variational Auto Encoders. 67

VEINS Vehicular Network Simulation. 52

VeReMi Vehicular Reference Misbehavior. 52, 53, 56

VNF Virtual Network Function. 14, 26, 29, 30, 92, 93, 110

vRAN Virtual Radio Access Network. 14

WAVE Wireless Access in Vehicular Environments. 11

ZSM Zero touch network & Service Management. 110

List of Symbols

d_{in}, d_{out}	Perceptron/RNN cell input/output dimensions
W, V, b	Learnable weight (parameter) matrix/vector
θ	NN model weight matrix
X	dataset
x	data point in X, x can be scalar or vector $x = [x_{t_t}]$
L	Sequence length, when x is a sequence $t \in [1, L]$
$f()$	NN output function
y	target value for input x
$\hat{y}$	NN outputed value for input x, $\hat{y} = f(x, \theta)$
$g()$	activation function
h_{t_t}	Recurrent Neural Network (RNN) hidden state at time t
m	Number of layers
$\mathcal{L}()$	Loss function
η	Learning Rate (LR)
B	Batch size
E	Number of epochs

List of Publications

Journal Papers

[Dja+23] Taki Eddine Djaidja et al. "Early Network Intrusion Detection Enabled by Attention Mechanisms and RNNs". Submitted to: *IEEE Transactions on Cognitive Communications and Networking* (2023).

[Dja+24a] Taki Eddine Toufik Djaidja et al. "Federated learning for 5G and beyond, a blessing and a curse- an experimental study on intrusion detection systems". In: *Computers & Security* 139 (2024), p. 103707. ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2024.103707. URL: https://www.sciencedirect.com/science/article/pii/S0167404824000087.

[Dja+24b] Taki Eddine Toufik Djaidja et al. "Time-efficient detection of false position attack in 5G and beyond vehicular networks". In: *Computer Networks* (2024), p. 110461. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2024.110461. URL: https://www.sciencedirect.com/science/article/pii/S1389128624002937.

Conference Papers

[Bou+22] Abdelwahab Boualouache et al. "Deep Learning-based Intra-slice Attack Detection for 5G-V2X Sliced Networks". In: *2022 IEEE 95th Vehicular Technology Conference: (VTC2022-Spring)*. 2022, pp. 1–5. DOI: 10.1109/VTC2022-Spring54318.2022.9860373.

[Dja+22a] Taki Djaidja et al. "Adaptive Resource Reservation to Survive Against Adversarial Resource Selection Jamming Attacks in 5G NR-V2X Distributed Mode 2". In: *ICC 2022 - IEEE International Conference on Communications*. 2022, pp. 3406–3411. DOI: 10.1109/ICC45855.2022.9839023.

[Dja+22d] Taki Eddine Djaidja et al. "DRIVE-B5G: A Flexible and Scalable Platform Testbed for B5G-V2X Networks". In: *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*. 2022, pp. 2800–2805. DOI: 10.1109/GLOBECOM48099.2022.10001231.

Seminars

[Dja+21] Taki Eddine Djaidja et al. “Système d’Inférence floue pour adapter dynamiquement le temps de réservation des ressources/amélioration de la sécurité”. In: *FuturMob’21 DriveLab*. 2021.

Introduction

Thesis Context

Over the years, humanity has decisively overcome many obstacles and constraints related to time, space, and security in the field of information and communication. Technological advancements and progress in nearly all equipment and devices reflect the evolution of human needs and requirements. Thus, means of communication and networking have evolved from one era to another in order to meet these needs.

Presently, we are entering a new era enabled by modern mobile networks, known as 'connecting everything, anywhere'. This encompasses communication on the ground, in the air, and even in space, thereby paving the way for innovative applications in diverse domains, including smart cities, intelligent transportation systems, e-health services, and immersive augmented and virtual reality experiences. These advancements are made possible by the fifth generation of cellular networks, known as 5G, which provides a robust framework capable of meeting the networking demands that earlier generations could not fulfill. Ongoing research into future generations like 6G promises to further enhance 5G networks, ensuring continued progress in our increasingly connected world.

One sector that garners significant attention from both academia and industry is the intelligent transportation sector. It covers various transportation-related applications that leverage vehicular communications, which are facilitated by 5G networks, referred to as 5G Vehicle-to-Everything (5G-V2X) communications. The 'X' in Vehicle-to-Everything (V2X) encompasses communications between road users, including vehicles, road infrastructure, and pedestrians, as well as communication with network-hosted applications. The primary aim of this sector is to advance road safety, improve transport efficiency, and reduce environmental impacts, all with the vision of achieving fully autonomous vehicles and transportation systems.

While the potential benefits of 5G-V2X communications in the intelligent transportation sector are immense, security remains a paramount concern. As vehicles become increasingly connected and reliant on these networks, they become vulnerable to cyber-attacks and privacy breaches, which could compromise the *availability* and *integrity* of vehicular applications and services, potentially leading to disastrous accidents and endangering human lives. This goes against the intended use of these vehicular applications. Moreover, breaches in *confidentiality* pose significant problems, including the exposure of private data concerning drivers and business owners.

To preserve the Confidentiality, Integrity, and Availability (C-I-A) proprieties within the realm

of 5G-V2X networks and their applications, a variety of preventive security measures need to be deployed. These measures include encryption and authentication techniques, as well as network and control measures designed to prevent external attackers from compromising the network. Additionally, Intrusion Detection Systems (IDSs) play a central role. IDSs operate by *monitoring* the behavior of internal network nodes trying to identify and detect cyber-attacks initiated by internal 'malicious' nodes that have evaded the preventive security measures.

An IDS can monitor behavior at various levels. It can operate at the physical (or virtual) *hosts* level, including entities such as vehicles, servers, or virtual machines running applications. This type of IDS is known as Host-based Intrusion Detection System (HIDS). HIDS monitors network data and host resources to detect potential threats. There is also a subclass of HIDS called Application-based Intrusion Detection System (AIDS), which focuses specifically on monitoring applications at the host level. On the other hand, there is another type of IDS known as Network-based Intrusion Detection System (NIDS). A NIDS is responsible for protecting a set of hosts by detecting attacks at the network level, before they reach the hosts.

In recent years, the effectiveness of IDSs has seen substantial improvement thanks to the integration of Artificial Intelligence (AI) and Deep Learning (DL). AI-powered IDS models provide an intelligent approach to address cybersecurity threats. Notably, DL models have demonstrated impressive abilities in recognizing intricate patterns, greatly aiding IDS in the detection process.

Researchers have long been exploring the task of developing reliable IDSs. This quest persists today. Yet, in the era of 5G-V2X, characterized by the widespread adoption of connected vehicles and the evolving landscape of cybersecurity threats, the demand for reliable IDSs becomes even more imperative. When considering the specific case of 5G-V2X, it offers two communication options: SideLink (SL) communications for direct communication between road entities and the infrastructure-based communication via the 5G network components. Therefore, it is essential to deploy HIDS (and eventually AIDS) monitoring systems at the vehicle level to monitor communications and applications not routed via the 5G network infrastructure. Additionally, at the level of the 5G network, both HIDS(/AIDS) and NIDS are to be deployed as a complement to protect the 5G network and Application Servers (ASs).

Thankfully, AI-driven models offer the promise of delivering reliable IDSs. Within this context, this thesis is aligned with this trend, with its contributions centered on IDSs within the realm of 5G-V2X and beyond.

Objectives and Research Questions

This thesis explores the intersection between 5G(-V2X), AI and IDS. Our research interest centers around the question: (**RQ**)'How can AI be effectively and efficiently leveraged to secure 5G-V2X (and beyond) networks?' This question revolves around two key aspects: effectiveness and efficiency. We aspire for AI-based IDS models to excel in the detection process, as evidenced by various performance metrics. Additionally, these models must exhibit time efficiency, given the context of vehicular networks requiring low-latency communication. The AI model should not introduce significant overhead that could compromise the low-latency property of these networks.

Based on **RQ**, three research sub-questions have been developed and addressed in this thesis as follows:

- (**RQI**) How can AI ensure the integrity and reliability of transmitted messages in 5G-V2X communications?
- (**RQII**) How can AI facilitate real-time early-stage threat detection in 5G(-V2X)?
- (**RQIII**) How can AI techniques, particularly data-driven ones, be implemented while guaranteeing user and business privacy in 5G(-V2X)?

RQI focuses on the various types of transmitted messages in the context of 5G-V2X communications. The aim is to investigate how AI can ensure the reliable delivery and integrity of transmitted messages, whether they are between vehicles or via the 5G network. Any compromise of this security property should be detected with a high degree of confidence. **RQII** pertains to the efficiency of these AI-based detection models. Specifically, these models should be capable of identifying potential cyber attacks as soon as signs of such attacks appear within the network. **RQIII** addresses a specific challenge within data-driven AI models that utilize data as their knowledge base: how to deliver these models without compromising the privacy of various network users.

Within our contributions, we aim to address and provide solutions to these three sub-questions, while all our research findings are geared towards answering the central research query (**RQ**).

Thesis Contributions

Motivated by the need to address the traced research questions, we propose approaches to advance the state of the art in AI-based intrusion detection systems in the era of 5G and beyond V2X networks.

Our contributions can be categorized into two main categories: (1) HIDSs deployed at the vehicle level to monitor potential threats primarily initiated through SL communications, and (2) NIDSs positioned at the level of the 5G network. HIDSs, which should be implemented at the cloudified 5G level, including those HIDSs protecting 5G Network Functions (NFs), as well as servers and virtual machines that support the 5G infrastructure, are not covered in this thesis.

Following this categorization, the contributions are grouped into two parts. In Part II, we address the issues related to jamming attacks in SL communications (Chapter 4) and the problem of message forgery in V2X (Chapter 5). As for Part III, we focus on early network intrusion detection (Chapter 6) and the challenge of privacy in distributed DL model learning in 5G networks (Chapter 7). One should note that we begin this thesis by presenting the key concepts and backgrounds in Part I, which comprises three chapters (Chapter 1, Chapter 2, and Chapter 3).

To better position our research works, we can align other taxonomies for our contributions. From the perspective of attack nature and the C-I-A proprieties, attacks targeting *availability* are addressed in chapters 4, 6, and 7, while attacks targeting *integrity* are addressed in chapter 5. Chapter 6 also contains some attacks on *confidentiality*, such as port scanning. Furthermore, chapter 7 intervenes to further preserve *confidentiality* propriety by addressing data privacy concerns. Regarding the Open Systems Interconnection (OSI)[1] networking layers, in chapter 4, we delve into

[1] conceptual framework used to standardize and describe the functions of a networking system

attacks against the physical layer. Chapter 5 is dedicated to addressing attacks against the application layer, while Chapter 6 and 7 focuse on tackling attacks against the transport layer protocols.

Furthermore, each contribution will provide responses to the research questions. Table 1 summarizes the relationship between the chapters, research questions, C-I-A proprieties, OSI model and publications in this thesis. The following paragraphs provide a concise overview of the contributions of this thesis.

In our first research contribution, presented in Chapter 4, we addressed a vulnerability that is present in 5G-V2X SL communications. The used protocol, named Semi-Persistent Scheduling (SPS), specifies that a vehicle must autonomously reserve radio resources to transmit Cooperative Awareness Messages (CAMs), which serve as the foundation for numerous vehicular applications. These reserved resources are held for a certain amount of time. We found that this protocol is vulnerable to adversarial radio resource selection attacks, where the attacker exploits information about resource reservation to interfere with that resource, preventing the vehicle from sending its messages. This attack directly affects the reliability of vehicular applications, which is closely related to our research sub-question **RQI**, where we aim to leverage AI to ensure the reliability property. This chapter's contribution focuses on two steps: proposing and developing an HIDS for attack detection, and enhancing the HIDS with resilience mechanisms. While the detection mechanism relies on a feedback approach, the resilience mechanism employs a technique that enables attack evasion. We utilized Fuzzy Inference System, an appropriate AI mechanism for decision-making and adaptive systems, to enhance the SPS protocol, rendering it adaptive and resilient to the aforementioned attack. Our findings demonstrate that when a vehicle is under attack, 85% of its CAMs fail to be properly transmitted to the vehicles, leaving only 10% to 15% correctly received with suitable reception. Our HIDS effectively improves this rate to 60%.

Chapter 5 addresses the issue of message forgery attacks, where malicious vehicles exhibit misbehaving behavior and alter their CAMs to disrupt the system. Detecting these attacks is typically accomplished through an AIDS that checks the consistency of CAMs. This is where AI play a crucial role in addressing this complex task. However, a sub-problem arises due to the stringent latency requirements of these systems. The proposed AI model for detecting these attacks must operate in real-time and avoid introducing computational overhead that could compromise latency property. The cited problems are directly related to **RQI**, which focuses on the integrity of transmitted messages, and **RQII**, which addresses the efficiency and real-time detection of the proposed AIDS. In this second contribution, we propose an AIDS that utilizes Recurrent Neural Networks (RNNs), an advanced class of Neural Networks (NNs) adept at handling sequential data, enabling it to incorporate the historical behavior of nodes and effectively analyze longer CAMs sequences for consistency checks. Additionally, we emphasize the evaluation of computational overhead by providing theoritical complexity analysis of the proposed DL model. Our contributions reveal noteworthy findings, we demonstrate that longer CAMs sequences analyzed lead to enhanced performance, while related approaches rely on fixed-length sequences. Furthermore, we showcase that our proposed method exhibits the computational complexity of a single inference at each CAM inspection, while related works necessitate analyzing the entire "fixed-length" sequence for each CAM inspection.

The contribution chapter 6 addresses NIDSs aiming to secure 5G ASs (vehicular applications). Current approaches analyze network traffic packets, where they collect a sequence of packets, called

a Network Flow, aggregate it into one data point, and analyze it AI/DL models. We found that this approach requires the Network Flow to be terminated before the analysis can be performed, which may delay detection time. This limitation is related to **RQII**, requiring AI based NIDSs to perform early attack detection, given that attack traces may appear before the session ends. In this third contribution, we will exploit the sequential property of Network Flows, which is overlooked in current approaches that aggregate Network Flows into single data points. Our proposed NIDS will utilize RNNs to capture the sequential nature of Network Flows, and it will be equipped with an attention mechanism, an advanced mechanism that enhances the capabilities of RNNs in dealing with sequential data. This mechanism allows our NIDS to focus on the specific packets within a Network Flow that are most likely to indicate an intrusion. We demonstrate that our approach reduces the number of packets required for intrusion detection and significantly decreases the time needed for detection. These findings hold significant promise for enhancing NIDSs.

The contribution in Chapter 7 addresses an issue related to DL based NIDS which require substantial data to be effective. To achieve this, service and application providers need to collaborate and pool their data to train their DL models. However, due to privacy concerns, they cannot directly exchange their raw data. Therefore, they rely on a distributed approach called Federated Learning (FL), which enables them to collaboratively train their DL models without sharing their raw data. The commonly used FL approach, FedAvg, faces a challenge when dealing with heterogeneous data from different business partners, as is often the case in 5G networking data. This challenge, known as statistical heterogeneity, needs to be addressed to provide answers to **RQIII**. In this fourth contribution, we explore novel state-of-art algorithms in FL, specifically FedProx, FedPer and SCAFFOLD, and evaluate their effectiveness in addressing the challenge of statistical heterogeneity in the context related to 5G-V2X. Our findings demonstrate that these state-of-the-art algorithms enable effective collaborative distributed DL, even in the presence of statistical heterogeneity, unlike the commonly used FedAvg algorithm.

Thesis Organisation and Outline

The paragraphs in this section provide an outline of the thesis structure.

Chapter 1: 5G V2X and Beyond This chapter delves into the essential components of 5G, including its key technologies, from 5G-V2X communications to the various applications of V2X, all aimed at providing a comprehensive understanding of the main aspects of 5G-V2X.

Chapter 2: some backgrounds on Artificial Intelligence This chapter serves as an introduction to AI and provides essential background information on the two AI methods used in this thesis, namely, Fuzzy Inference System and Neural Networks, which readers can refer to while reading the contributions.

Chapter 3: Intrusions and Intrusion Detection in 5G-V2X This chapter summarizes network intrusions in 5G-V2X, defines IDSs, provides taxonomies for the types of IDSs, explains the nature of the data used, and discusses the different used models. Furthermore, it presents state-of-the-art approaches in IDSs in the context of 5G-V2X.

Chapter 4: Coping with adversarial radio resource selection in SideLink V2X communications In this chapter, we present our solution to counter adversarial resource selection attacks. We introduce our HIDS that leverages a Fuzzy Inference System (FIS) to effectively detect and mitigate these attacks. Additionally, we present detailed results and performance analyses to demonstrate the effectiveness of our proposed HIDS.

Chapter 5: Detecting message forgery attacks in V2X communications This chapter delves into message forgery attacks in V2X and SL communications. We introduce our novel RNN-based AIDS capable of effectively detecting these integrity attacks. Additionally, we conduct a thorough analysis and comparison of the time complexity of our approach, showcasing its efficiency.

Chapter 6: Early network intrusion detection in 5G networks This chapter tackles the challenge of early intrusion detection in NIDSs. We propose a novel approach that utilizes attention mechanisms and RNNs to achieve earlier and more accurate intrusion detection. Additionally, we conduct extensive evaluation to assess the efficiency of our approach on various performance metrics.

Chapter 7: Distributed DL-Based NIDSs in 5G networks This chapter delves into state-of-the-art FL algorithms to enable privacy-preserving DL based NIDSs. We evaluate these algorithms on their ability to address the challenge of statistical heterogeneity and discuss our findings.

Finally, the thesis concludes with the Conclusion, and Future Research chapter, which summarizes and discusses our research findings and outlines directions for future contributions and improvements.

Author's Guidance for a More Effective Reading Experience

Thank you for your interest in our work. To help you navigate through this manuscript, we provide some essential information:

- **Glossary for Key Definitions**: To assist you in understanding the technical terms and concepts used in the manuscript, we have provided a comprehensive *glossary* section. Furthermore, validation metric formulas used in our contributions are detailed in the *extended glossary* section.
- **PyTorch Integration**: Throughout the different Neural Networks (NNs) model training process, we conducted our experiments using the PyTorch framework. Furthermore, when explaining NNs and DL, it is presented in the PyTorch tensor style. You can find more information about PyTorch on their official website `pytorch.org`.
- **Hardware Configuration**: Our system, which is used throughout this thesis for the assessment of AI models, has the following configuration: Intel Core i7-10700 processor, 32GB of RAM, and Nvidia RTX 3070 graphics card.

Table 1: Relationship between chapters, Research Questions / C-I-A props. / OSI model and publications in this thesis.

Part	Chapter	Main Theme	rel. to RQs	rel. to C-I-A	rel. to OSI	Publication
Part II	Chapter 4	Coping with adversarial radio resource selection in SideLink V2X communications	**RQI**	A	Physical layer	[Dja+22a]
	Chapter 5	Detecting message forgery attacks in V2X communications	**RQI** and **RQII**	I	Application layer	[Dja+24b]
Part III	Chapter 6	Early network intrusion detection in 5G networks	**RQII**	C&A	Transport layer	[Dja+23]
	Chapter 7	Distributed DL-Based NIDSs in 5G networks	**RQIII**	C&A	Transport layer	[Dja+24a]

Part I

Concepts and Backgrounds

Chapter 1

5G V2X and Beyond

Introduction

As previously mentioned in the Introduction chapter, our thesis explores the dynamic intersection of three domains, namely 5G-Vehicle-to-Everything (V2X), Artificial Intelligence (AI), and Intrusion Detection Systems (IDSs). In this part of the manuscript, we aim to provide a comprehensive context and introduction to these fundamental concepts starting first, in this chapter, by exploring the essential components of 5G and 5G-V2X, which encompass their key technologies, 5G-V2X communications, and the diverse range of V2X applications.

The remainder of this chapter is organized as follows: Section 1.1 provides a historical review of vehicular networks and the technologies adopted in the evolution of vehicular networks over time. Section 1.2 delves into 5G networks, detailing their architecture and key technologies, which include SDN, NFV, and the Network Slicing paradigm. Section 1.3 explores vehicular communication within the 5G-V2X framework and its various applications.

1.1 Historical Review

Vehicular networks are a class of wireless networks wherein the communicating nodes consist of vehicles and road users, all equipped with wireless devices. These networks play a crucial role in the development of intelligent transportation systems, facilitating the exchange of real-time information among various transportation nodes. Their primary objective is to enhance road safety through driving assistance, enabling cooperative awareness, and supporting collision avoidance. Furthermore, vehicular networks help optimize transport efficiency, improve the driving experience, and reduce energy consumption and environmental impacts.

Vehicular networks have a history dating back to the early 2000s. These networks relied on Dedicated short-range communication (DSRC) technology, which uses an amended WiFi version, the IEEE 802.11p protocol. The DSRC standards and regulations are known as European Telecommunications Standards Institute (ETSI) ITS-G5 in Europe and Wireless Access in Vehicular Environments (WAVE) in the United States [Li12].

DSRC technology offered considerable potential for enhancing safety features in transportation

systems. It gained substantial interest from diverse automotive industries. However, it is unfortunate that its widespread utilization has not been realized as expected. In 2016, the European Commission acknowledged that vehicular network technologies "are far from being used at their full potential despite the benefits they could bring" [16].

In 2017, a novel vehicular network technology appeared in 3^{rd} Generation Partnership Project (3GPP) Release 14 [3GP17]. This technology, termed as Cellular Vehicle-to-Everything (C-V2X) utilizes cellular networks in place of WiFi for communication, that time, the C-V2X used the 4G Long-Term Evolution (LTE) network hence its name was Long-Term Evolution Vehicle-to-Everything (LTE-V2X). C-V2X holds the potential to revolutionize vehicular communications by offering an extended and more ubiquitous communication range compared to DSRC. Furthermore, one of its key advantages is the ability to enable communication between vehicles, even in scenarios where direct line-of-sight communication is obstructed.

The choice between adopting C-V2X or DSRC technology has triggered debates, with regulatory authorities, automotive industry players, and telecommunications stakeholders not arriving at a unanimous agreement on the preferred option [18].

In 2019, 3GPP released Release 16 [20], which proposed that V2X networks would rely on 5G technology, known as 5G Vehicle-to-Everything (5G-V2X) or New Radio Vehicle-to-Everything (NR-V2X). The communication performance of 5G surpasses that of LTE and DSRC. It is designed to offer reliable, ultra-low latency communication—a pivotal requirement in V2X communications, enabling the emergence of ambitious vehicular use cases, including remote driving and autonomous driving.

With the emergence of 5G-V2X technology, a significant portion of stakeholders in the transportation and telecommunications industries are transitioning towards embracing C-V2X communications, i.e., V2X based on 5G and beyond. Hence, numerous articles in journals and magazines dating back to 2019 discuss the transition to 5G and beyond technologies in the transportation sector [MA20]. Indeed, 5G is revolutionary, and the technologies it brings are laying the groundwork for future networks like 6G.
In the following sections, we present the underlying concepts of 5G networks, as well as 5G-V2X and beyond.

1.2 5G and Beyond: Key technologies

5G is the 5^{th} generation of cellular mobile networks, designed to offer an adaptable and customizable network platform that supports a wide array of services extending far beyond the realm of transportation systems. This vision encompasses the integration of smart cities, e-health services, and immersive augmented and virtual reality experiences, all with the ambition of enabling ubiquitous connectivity—the 'everything, anywhere' paradigm.

The heterogeneous requirements of the aforementioned services and applications necessitate the mobile network to demonstrate lower latency, higher speeds, and the capacity to accommodate more connected devices compared to existing network technologies. 5G fulfills these requirements, thanks to the technologies employed and an innovative paradigm known as Network Slicing. In the next subsections, we will first explain the components of the 5G network and then delve into the concept of Network Slicing and its key enablers.

1.2.1 5G Components

Mobile networks are composed of two parts: the Radio Access Network (RAN) and the Core Network (CN). With the advent of 5G, advancements have been introduced to the technologies employed in preceding generations. While reading the rest of this chapter, please refer to Figure 1.1, which illustrates the architecture of 5G components, Network Slices (NSs), and V2X communications in the context of 5G.

RAN refers to the set of radio technologies and protocols used by mobile devices and network base stations to enable wireless connectivity. Specifically, the 5G RAN enables more efficient use of spectrum and improved coverage. It incorporates technologies like massive Multiple Input Multiple Output (MIMO) antennas, beamforming, and full duplex trasmissions [LL21].

On the other hand, the CN is responsible for access and session management, authentication, security, and network traffic routing/forwarding. It implements a set of Network Functions (NFs) designed to accomplish the tasks mentioned above, e.g., the User Plane Function (UPF) in 5G CN is in charge of the data plane and is responsible mainly for traffic routing and forwarding. The 5G CN introduced new NFs that did not exist previously, enabling novel capabilities. For instance, the NetWork Data Analytics Function (NWDAF) was introduced to gather data from user equipment and network functions. This data can be utilized for analytics aimed at enhancing Quality of Service (QoS). The NFs are standardized in the 3GPP technical specifications [20].

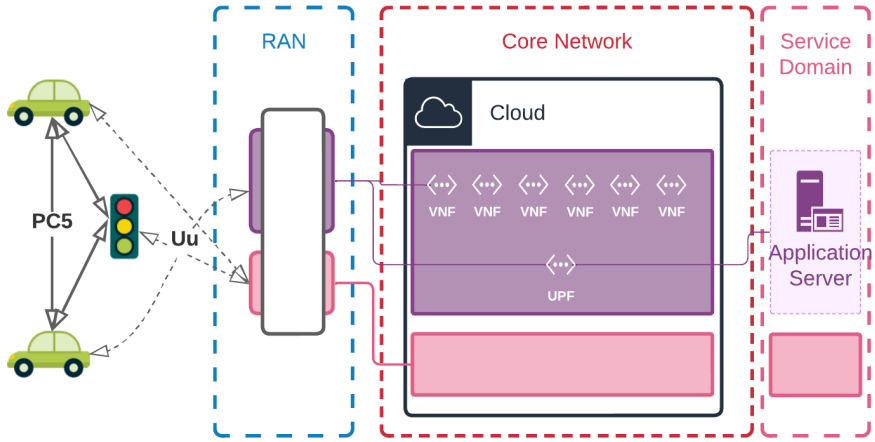


Figure 1.1: 5G-V2X architecture

1.2.2 Network slicing

The Network Slicing paradigm is considered a pivotal enabler for 5G. It stands as one of the most innovative technologies within the realm of 5G and future networks. According to 3GPP and ETSI [17; 22], Network Slicing is a paradigm in which logical network partitions are created with specific network capabilities and characteristics to serve and support particular services. A NS (or partition) can be defined as an independent network deployed, either fully or partially, on shared resources.

These resources may include components such as computing, storage, networking resources, and spectrum.

A NS tailors the functionalities of both the CN and RAN to precisely meet the distinctive demands posed by various applications and services. The primary enablers of Network Slicing are Network Function Virtualization (NFV) and Software Defined Networking (SDN):

- **NFV** leverages the concept of network softwarization, where NFs formerly deployed in dedicated devices are replaced by software. This eliminates the dependency between the NF and the hardware on which it is deployed. These NFs are designed to run as virtual machines/containers[1] on standard servers or cloud platforms. The NFs in 5G CN are virtualized and deployed as Virtual Network Functions (VNFs). Similarly, NFV is also applied to the RAN, where baseband functions are executed as Virtual Radio Access Networks (vRANs).

- **SDN** is an emerging technology that aims to simplify networking and make networks programmable. In the context of sliced 5G networks, the key role of SDN is to ensure, through its controller, VNF chaining paths and responding to network outages.

Thanks to NFV and SDN, the deployment of NSs is automated, customized and made reliable. The revolution of 5G is here, where 5G has brought advancements not only in new RAN and CN technologies but also in its function as an orchestration platform. The orchestration in 5G is responsible for performing several functions within the NS. These functions include instantiating the NS, replicating the number of VNFs to scale the NS, placing the VNF either in the cloud or at the edge near the end user, to reduce latency. All of these functions are aimed at fulfilling the requirements of the delivered service. These service requirements might be dynamic, meaning that they may need to change over time and across different geographical locations to ensure optimal performance and resource utilization.

5G and beyond networks enhance the traditional roles of network operators but also reshape the collaborative dynamics involving cloud providers and business customers. In this evolved ecosystem, enterprises no longer passively receive connectivity; instead, they actively participate in shaping their network environment. A NS concerns a business customer, and the technical requirements of the service offered [Ela+19]. As such, 5G introduces standardized NS types, covering:

1. Enhanced Mobile Broadband (eMBB) for services requiring high data rates,

2. Massive Machine-Type Communications (mMTC) for services supporting a large number of connected devices,

3. Ultra-Reliable Low Latency Communications (uRLLC) for services having stringent latency and reliability requirements.

Moreover, a single service may exhibit varying requirements across these three categories. The service type defines the technical parameters for the NS, also known as the "slice blueprint". For instance, the blueprint may include details such as the required number of replications of a certain NF and the utilization of edge computing in specific regions. Furthermore, the hosted Application Server (AS) (service) is integrated within the 5G NS, operating as a VNF that becomes a component of the NS's architecture. This integration extends to the orchestration procedures. This approach

[1] executable software packages that encapsulate an application and its dependencies. Unlike virtual machines, containers share the host operating system's kernel and are more lightweight.

is termed End-to-End (E2E) Network Slicing, signifying that the NS encompasses the RAN, CN, and AS.

An illustrative scenario where the network is sliced according to business/technical requirements is described in the following. This one deliberately concerns V2X—a sector of particular interest to us. Consider an enterprise offering two transportation-related services: one involving eMBB (e.g., music and video streaming) and the other focusing on uRLLC (e.g., remote driving). In this situation, the enterprise would be required to create two distinct NSs having blueprints suitable for the uRLLC and eMBB services, respectively. In a parallel scenario, another enterprise might offer the same uRLLC service. This second enterprise would deploy its own NS with the same technical requirements as the first one (uRLLC slice).

In the following subsection, we go further into V2X communications and their applications.

1.3 V2X Communications and Applications

As we stated in the Introduction chapter, the X in V2X signifies the intercommunication between vehicles and all entities present in the road environment and beyond, in the network. This includes interactions with fellow vehicles Vehicle-to-Vehicle (V2V), pedestrians and cyclists Vehicle-to-Pedestrian (V2P), the road infrastructure such as Road Side Units (RSUs) Vehicle-to-Infrastructure (V2I), and vehicles consuming internet services Vehicle-to-Network (V2N). These communications can be achieved via the cellular network infrastructure. One of the new technologies brought to mobile cellular networks by LTE is the support of Proximity Services (ProSe), allowing for device-to-device communication directly through a SideLink (SL) air interface. This technology is an enabler for C-V2X at the time it was introduced. By leveraging this technology, the latency in communication among neighboring road users (such as vehicles, pedestrians, and infrastructure) gets significantly reduced compared to the conventional approach, where communication between vehicles necessitates a multi-step process involving the cellular network (vehicle-to-network-to-vehicle).

As a result, C-V2X (including LTE, 5G, and future generations) supports two modes of communication:

- Direct Communications (SL communications): This mode is commonly used for (V2V, V2P, V2I) communications. For this SL communication, vehicles use an interface called PC5.
- Communication via Network (V2N): This mode operates via the Uu interface.

A Vehicular application employs either one or both of these modes, depending on its required specifications. Moreover, each application appears as a NS in a 5G and beyond V2X environment.

In parallel to the network technology evolution, with the advent of 5G, vehicular applications have seen significant development. This has opened up new business opportunities for vehicle manufacturers and transportation-related businesses. 3GPP [3GP20] categorizes vehicular applications into two categories: Safety-related apps and Non-safety-related apps.

Safety-related applications Safety apps offer enhanced safety for drivers, vehicles, pedestrians, and all road users. A plethora of services are offered, such as assisting the driver in situations with limited visibility, like driving in foggy conditions or executing emergency braking after a turn. Furthermore, as transportation systems evolve, autonomous driving can also be included in the

realm of safety apps since it has to ensure safe driving. The messages used in these applications are standardized to ensure that all vehicles can understand and respond appropriately to these messages, regardless of their model. Here are examples of these messages:

- Cooperative Awareness Message (CAM): These messages are broadcasted periodically by vehicles to indicate their status, such as direction, speed, and heading.
- Decentralized Environmental Notification Messages (DENM): are event-based messages, asynchronously used to inform about specific events, such as the occurrence of an accident.
- Signal Phase and Timing (SPAT): These messages are transmitted by the infrastructure and include information about traffic light states and the time remaining until a traffic light changes.

Non-safety-related applications Non-safety-related services include those that aim to enhance the driving experience, traffic efficiency, and environmental friendliness. The applications in this category are not standardized and may use safety-related messages to provide the proposed services. Vehicle manufacturers can offer convenience services such as remote support and automated parking to vehicle owners. Other applications can be proposed by service providers other than the manufacturers, which are attractive to vehicle drivers or passengers. These may include traffic optimization, gas consumption reduction, listening to music, playing video games, or watching videos.

Conclusion

This chapter explained the concepts of 5G and beyond vehicular networks, with a particular emphasis on the paradigm of Network Slicing. This paradigm enables the creation of customized "virtual" networks known as NSs, which host transportation-related applications and services enabled through vehicular communications. These NSs are customized to meet the stringent requirements of such applications, including low latency and reliability. Additionally, this chapter discussed the use of 5G for V2X.
The next chapter of this part will explain the second keyword of the concepts that we emphasize, which is Artificial Intelligence. AI is a cornerstone not only in this thesis but also in many aspects of future network design, including 5G.

Chapter 2

Artificial Intelligence

Introdution

During the past decades, researchers have aimed to showcase AI's capacity to successfully address various tasks. Presently, we are witnessing the appearance of several applications relying on AI, including those related to computer vision and natural language processing; Additionally, ongoing research persists in these fields and various other domains where the full potential has yet to be realized. Notably, network management and security hold particular interest for us.
In this context, this chapter will provide a concise introduction to AI and its techniques. Section 2.1 will delve into AI techniques in general, while Section 2.2 will focus on explaining Fuzzy Inference Systems. In Section 2.3, we will turn our attention to Neural Networks, with a particular emphasis on the learning aspect of Neural Networks.

2.1 Artificial Intelligence in a nutshell

AI, a concept that emerged in the early 1960s, is defined in [Kur00] as:

> *The art of creating machines that perform functions, and which require intelligence when performed by people*

Thus, AI is considered as the research field that attempts to understand how humans think and build intelligent entities, referred to as computers or agents, that can 'think and act humanly'[RN09]. This includes problem-solving, knowledge representation and reasoning and learning [Ric17].

AI involves the utilization of various techniques contributing to its advancement. These techniques span across various fields within AI including:

- Problem-solving: This consists of solving problems by searching through numerous possible solutions, especially when the search space becomes vast and an exhaustive search is not feasible. AI techniques for this purpose include heuristics, evolutionary algorithms (meta-heuristics), and game theory techniques.
- Knowledge representation and reasoning: Reasoning involves finding a path from premises to conclusions using a chain of deduction rules. This includes the use of formal logic to represent

information and various types of logical inference, such as fuzzy inference and probabilistic inference, as reasoning systems.

- Learning (a.k.a Machine Learning (ML)): involves automatically learning to recognize complex patterns and making decisions based on *data*. Techniques used in ML include Deision Trees (DT), Random Forest (RF), Naive Bayes (NB), Support Vector Machines (SVM), K-Nearest Neighbors (KNN), and Neural Networks (NNs).

We will not cover all of the AI techniques in depth; instead, we will focus on techniques that we used in our contributions. We will be providing comprehensive insights into Fuzzy Inference Systems (FISs) and NNs, with a special focus on the latter.

2.2 Fuzzy Inference Systems

Linguistic statements are the tools humans employ to depict a wide array of objects and scenarios. However, these statements often incorporate imprecise and ambiguous notions. For instance, determining whether 20 km/h qualifies as "high" speed or "slow" speed demonstrates the difficulty of precise formalization. This poses a challenge for representing such notions using Boolean logic[1], which fails to represent this nuanced "knowledge".

In response to these complexities, fuzzy logic emerged as a solution in the field of AI, introduced in [Zad65]. Fuzzy logic offers a framework to handle concepts characterized by uncertainty and imprecision. Unlike the binary nature of boolean true/false values, fuzzy logic permits the flexibility to represent partial truth, allowing for an accurate reflection of the real-world ambiguities that linguistic statements often encapsulate [Ric17; Rut08]. Fuzzy logic is utilized in various fields, including intrusion detection and prevention, as a framework that helps in making decisions involving security threats. It enables the system to make decisions such as determining whether an attack is occurring or not and selecting appropriate mitigation strategies.

Fuzzy logic is based on fuzzy set theory, which consists of elements possessing a degree of membership to a fuzzy set within a continuous range between 0 and 1. This is in contrast to crisp sets, where membership is binary (either 1 or 0), denoting the membership (or non-membership) of elements to the set.

Membership functions form the Fuzzy Sets; a membership function μ_A maps the values of the universe of discourse (U) – representing the range of possible values of a fuzzy concept (e.g., speed) – to their degree of membership in the fuzzy term A (for instance, "high", "slow").

$$\mu_A : U \to [0, 1] \tag{2.2.1}$$

The membership functions have various graphical representations, including triangular, trapezoidal, and Gaussian forms. Additionally, there is the singleton membership function, which exclusively assigns a value of 1 to a single point within U, and all other points as 0.

A FIS represents an intelligent decision-making technique [Sin+13] that takes crisp inputs belonging to different fuzzy concepts and processes them using a collection of fuzzy rules, resulting in crisp outputs. FIS comprise four main modules:

[1] Boolean logic is a system of formal logic that operates on binary values (true/false), using operations like AND, OR, and NOT.

- Fuzzifier: fuzzification consists of transform transforming an input crisp value into corresponding fuzzy membership values associated with the fuzzy terms within a given concept. The fuzzifier applies this fuzzification process to all inputs crisp values.
- Fuzzy rules: a set of combinatorial rules that encapsulate the relationships between fuzzy input variables and fuzzy outputs. Formulated by domain experts, these rules form the knowledge base of the inference system.
- Inference Engine: involves applying the relevant fuzzy rules to the fuzzy input and then aggregating them to infer the fuzzy output.
- Defuzifier: converts the fuzzy output from the inference engine into a crisp output.

The summary of the process in FISs, which encompasses these four modules, is depicted in Figure 2.1.

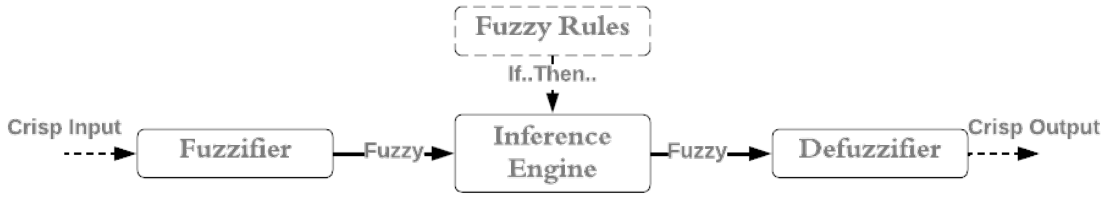


Figure 2.1: Fuzzy Inference System process.

2.3 Neural Networks

NNs, currently one of the most prominent topics in AI, involve the creation of 'artificial' neurons designed to simulate the information processing of the human brain. NNs are a subset of ML that specifically focuses on modeling complex patterns and relationships in data. In the following subsections, we will describe the backgrounds and the process of training and evaluating NNs.

A perceptron, an artificial neuron, is a computational entity that takes an input, applies weights to them, and produces an output based on an activation function. A perceptron unit is a function that calculates a single scalar value. Several perceptron units can be combined to compute complex functions, thus constituting a perceptron with many inputs and potentially multiple outputs [Ric17]. The output calculated by a perceptron is computed as follows:

$$\hat{y} = g(\theta^\top * \langle 1, x \rangle) \tag{2.3.1}$$

where:

- $\hat{y}$ is the output vector of dimension/shape $(d_{out},)$ where $d_{out} \geq 1$.
- $\langle 1, x \rangle$ is the input vector preceded by a bias element set to one, which enables an additional adjustable parameter in the perceptron unit known as the bias. The input vector x has a shape of $(d_{in},)$, thus the final shape of $\langle 1, x \rangle$ is $(d_{in} + 1,)$.

- θ is the weight matrix with a shape of $(d_{in}+1, d_{out})$,

$$\theta = \begin{pmatrix} b_1 & b_2 & \cdots & b_{d_{out}} \\ w_{1,1} & w_{1,2} & \cdots & w_{1,d_{out}} \\ \vdots & \vdots & \ddots & \vdots \\ w_{d_{in},1} & w_{d_{in},2} & \cdots & w_{d_{in},d_{out}} \end{pmatrix} = \begin{bmatrix} b \\ W \end{bmatrix}$$

  where b_j is the bias weight for the perceptron unit j. The weighted sum ($\sigma = \langle \sigma_j \rangle$) is calculated by the multiplication $\theta^\top * \langle 1, x \rangle$, which is equal to:

$$\sigma_j = \sum_{i=1}^{d_{in}} x_i * w_{i,j} + b_j$$

- g is the activation function. The basic activation function is the positive/negative function, which outputs (for the perceptron unit j) 1 if the weighted sum (σ_j) is positive, and zero otherwise. We distinguish several other element-wise[2] functions, which could be linear or non-linear, such as tanh, sigmoid and Rectified Linear Unit (relu).

After mathematical simplification to isolate the bias, we can express the formula 2.3.1 as follows:

$$\hat{y} = g(W^\top x + b) \tag{2.3.2}$$

Where W and b have the shapes of (d_{in}, d_{out}) and $(d_{out},)$, respectively.

Figure 2.2 illustrates graphically a perceptron with multiple outputs, in this case, 2.

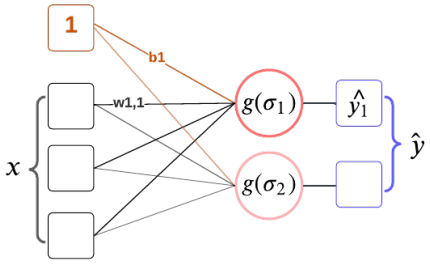


Figure 2.2: Illustration of a perceptron with multi outputs.

A NN is set of perceptrons connected together to form a network, we distinguish two ways to do that, named Feed-Forward Neural Networks (FFNNs), and Recurrent Neural Networks (RNNs). which we explain in the following two subsections.

[2] A function that is performed separately on each element of a data structure, such as an array or a matrix

2.3.1 Feed-Forward Neural Networks

A FFNN is a unidirectional network, which means that data exclusively flows from the network's ingress towards its egress. The basic perceptron, discussed earlier, follows this feed-forward architecture. Moreover, an advanced version of the perceptron exists, known as a Multi-Layer Perceptron (MLP). Unlike the 'single-layer' perceptron, an MLP consists of multiple layers that participate in generating the output, enabling the network to compute more complex functions. The architecture of MLP consists of three distinct perceptron layers: the input layer, one or more hidden layers, and the output layer. The output of MLP is calculated as follows—note that the perceptron formula (Equation 2.3.1) is represented as $f(x, \theta)$.

$$\hat{y} = f^n(\cdots f^1(f^0(x, \theta^0), \theta^1) \cdots, \theta^n) \tag{2.3.3}$$

where f^0 is the input perceptron, $f^1 \cdots f^{n-1}$ are the hidden perceptron layers and f^n is the output layer.

2.3.2 Recurrent Neural Networks

A RNN, which is another variant of NNs, operates by feeding its outputs back into its own inputs. This recurrence property signifies that the output generated at each step depends on the information from the preceding step. This recursive mechanism empowers the NN to handle the sequential characteristics that may be present in the data.

The input (x) of a RNN is a sequence of length (L), sequences have different lengths; $x = [x_{t_1} \cdots x_{t_L}]$, (x_{t_t}) have the shape of $(d_{in},)$ each element of a sequence (x_{t_t}) is passed to a RNN cell to produce the output $(\hat{y_{t_t}})$.

The RNN has a memory vector, known as the hidden state at time t, with shape $(d_{out},)$ and an initial value h_{t_0} of zero. The formula for calculating the hidden state h_{t_t} is as follows:

$$h_{t_t} = tanh(W^\top * x_{t_t} + b_x + V^\top * h_{t_{t-1}} + b_h) \tag{2.3.4}$$

Where W and V are weight matrices of shapes (d_{in}, d_{out}) and (d_{out}, d_{out}), respectively. b_x and b_h are the bias vectors of shape $(d_{out},)$. The output vector is obtained by passing the h_{t_t} through a FFNN, which may have one or multiple layers.

$$\hat{y_{t_t}} = f(h_{t_t}, \theta) \tag{2.3.5}$$

Note that the shape of the output $\hat{y_{t_t}}$ can be different from the output d_{out} of the RNN cell.

An RNN model may include multiple layers of RNN cells. In such a model, each layer $(i, i \leq m)$, except for the first layer $(i = 1)$, takes the output $h_{t_t}^{i-1}$ of the previous layer as input, and the weight matrix W^i of the layer has a shape of (d_{out}, d_{out}). The calculation of $\hat{y_{t_t}}$ involves using the output of the final layer $h_{t_t}^m$, where m is the total number of layers in the model.

Several variants of the RNN have been proposed, including the Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) models, which enhance the capabilities of the 'basic' RNN.

LSTM The LSTM, first proposed in [HS97], has a more complex architecture that involves a new state named the cell state, which serves as the long-term memory of the network. The cell C_{t_t} is updated based on the output of four gates: the input gate i_{t_t}, forget gate f_{t_t}, cell gate g_{t_t}, and output gate o_{t_t}. These gates serve as control gates that regulate the flow of information in the cell; for more details please refer to [SSB14]. With the introduction of C_{t_t}, the hidden state h_{t_t} is updated as follows:

$$i_{t_t} = \sigma(W_i^\top * x_{t_t} + b_{ix} + V_i^\top * h_{t_{t-1}} + b_{ih}) \tag{2.3.6}$$
$$f_{t_t} = \sigma(W_f^\top * x_{t_t} + b_{fx} + V_f^\top * h_{t_{t-1}} + b_{fh}) \tag{2.3.7}$$
$$g_{t_t} = tanh(W_g^\top * x_{t_t} + b_{gx} + V_g^\top * h_{t_{t-1}} + b_{gh}) \tag{2.3.8}$$
$$o_{t_t} = \sigma(W_o^\top * x_{t_t} + b_{ox} + V_o^\top * h_{t_{t-1}} + b_{oh}) \tag{2.3.9}$$
$$C_{t_t} = f_{t_t} \odot C_{t_{t-1}} + i_{t_t} \odot g_{t_t} \tag{2.3.10}$$
$$h_{t_t} = o_{t_t} \odot tanh(C_{t_t}) \tag{2.3.11}$$

$\odot$ denotes element-wise multiplication, σ denotes the sigmoid activation function, where $W_{i,f,g,o}$ and $V_{i,f,g,o}$ are weight matrices of shapes (d_{in}, d_{out}) and (d_{out}, d_{out}), respectively. $b_{ix,fx,gx,ox}$ and $b_{ih,fh,gh,oh}$ are the bias vectors of shape $(d_{out},)$. The calculation of the output vector y_{t_t} remains the same as described in equation 2.3.5.

GRU GRU was introduced in 2014 as an alternative to the complex LSTM architecture [Chu+14]. GRU has simpler structure and fewer parameters than LSTM, and it can still handle long-term dependencies. GRU also has gates that control the flow of information named reset gate r_{t_t}, update gate z_{t_t}, and new gate n_{t_t}. The h_{t_t} is calculated as follows:

$$r_{t_t} = \sigma(W_r^\top * x_{t_t} + b_{rx} + V_r^\top * h_{t_{t-1}} + b_{rh}) \tag{2.3.12}$$
$$z_{t_t} = \sigma(W_z^\top * x_{t_t} + b_{zx} + V_z^\top * h_{t_{t-1}} + b_{zh}) \tag{2.3.13}$$
$$n_{t_t} = tanh(W_n^\top * x_{t_t} + b_{nx} + rt * (V_n^\top * h_{t_{t-1}} + b_{nh})) \tag{2.3.14}$$
$$h_{t_t} = (1 - z_{t_t}) * n_{t_t} + z_{t_t} * h_{t_{t-1}} \tag{2.3.15}$$

It is worth mentioning that both LSTM and GRU models can handle multi-layer networks.

Later in this manuscript, we will frequently make use of the term 'model'. By this, we are referring to the NN architecture and its associated properties. These properties encompass its type, the number of layers, the quantity of perceptron units within each layer, the activation functions employed in each layer, as well as the entirety of its weights (θ matrix).

2.3.3 Learning in Neural Network

The NN learning process in NNs, also referred to as Deep Learning (DL), where the term 'deep' signifies that the NN consists of multiple layers. DL involves adjusting the model's parameters (weights) to better approximate the desired function using input data. This weight optimization task can be described as a minimization problem:

$$\min_w \mathcal{L}_x(W), x \in X, w \in \theta \tag{2.3.16}$$

Where $\mathcal{L}$ is the Loss function, it serves to assess how closely the model's output value ($\hat{y} = f(x, \theta)$) matches the actual target value (y) associated with the input data (x). Various functions can be used for this purpose, such as mean squared error Mean Squared Error (MSE), Mean Absolute Error (MAE), and Cross-Entropy.

The objective is to minimize the loss function. To achieve this, a NN uses the Gradient Descent (GD) optimization algorithm to find the optimal set of weights for the NN model. This process involves iteratively adjusting the weights using the gradient of the loss function. At each iteration, the adjusted weights are calculated as follows:

$$w^{new} = w^{old} - \eta \times \nabla \mathcal{L}(W) \tag{2.3.17}$$

Where $\nabla \mathcal{L}(W)$ is the gradient of the Loss function, and η is the Learning Rate (LR), which is a constant. At the first iteration, the weights are initialized randomly.

There are three types of GD. Batch gradient descent uses the mean of all individual losses for each weight update iteration. On the other hand, Stochastic Gradient Descent (SGD) uses one instance at each iteration. Mini-batch gradient descent uses small sets of instances called mini-batches of fixed size.

Furthermore, to calculate the gradient of the Loss function, especially in the hidden layers, we employ the Backpropagation (BP) algorithm. This algorithm propagates errors backward through the network, utilizing the partial derivatives of the gradient in the last layer (starting from the output layer) to calculate the gradient for the preceding layer. This process is repeated recursively until the input layer is reached. For RNNs, a related algorithm called Backpropagation Through Times (BPTTs) is used. This algorithm involves unrolling the RNN through time and than applying 'standard' BP.

It is worth noting that there exist other paradigms in deep learning, such as transfer learning and Federated Learning, which we will explain when we use them in Chapter 7.

The learning process implies the specification of several hyperparameters, including the Loss function ($\mathcal{L}$), LR η, the mini-batch size (B), and the number of learning iterations, which we commonly refer to as epochs (E).

The BP algorithm facilitates weight adjustments, enabling the model to accurately produce target values for various input scenarios, making it well-suited for both classification and regression tasks. To assess the model's performance, the dataset X is divided into training X_{train} and testing X_{test} datasets. X_{train} is used during the learning phase, while X_{test} is employed to evaluate the model's generalization performance on unseen data during training. Various metrics, including True Positive (TP), True Negative (TN), False Positive (FP), False Negative (FN), Accuracy, True Positive Rate (TPR), False Positive Rate (FPR), True Negative Rate (TNR), Negative Predictive Value (NPV), False Negative Rate (FNR), Positive Predictive Value (PPV), and F1Score, are defined in the *extended glossary* of the manuscript, along with their corresponding formulas.

Conclusion

In this chapter, we introduced the concept of Artificial Intelligence and its various fields. We discussed the techniques we use in our contributions, which are FISs and NN techniques. Additionally, we provided the mathematical background for these topics. We recommend that the reader refer

back to this chapter as we will make references to it in various parts of the thesis contributions. The next chapter will be the last one in Part I and will delve into the topic of Intrusion Detection Systems, which is our third keyword of focus.

Chapter 3

Intrusions and Intrusion Detection in 5G-V2X

Introduction

In this chapter, we explore the landscape of Network Intrusions and Intrusion Detection Systems (IDSs), providing a taxonomy of IDS categories and discussing current approaches. The chapter is organized into three sections: Section 3.1 introduces Network Intrusions, offering examples within the context of 5G. Section 3.2 highlights and presents a taxonomy of IDSs. In Section 3.3, we investigate the diverse approaches to IDSs focusing on those leveraging AI, all in the context of detecting Network Intrusions within 5G(-V2X) networks.

3.1 Network Intrusions in 5G-V2X

The principles of Confidentiality, Integrity, and Availability (C-I-A) are essential requirements that network systems must ensure. They serve to guarantee that data and resources can only be viewed, modified, and accessed by authorized parties. Network Intrusions or cyberattacks, on the other hand, represent malicious activities that attempt to compromise these C-I-A properties.

In the context of 5G-V2X, Network Intrusions target both the road nodes (vehicles and infrastructure) and the 5G network elements (RAN, CN, and AS). Some of these attacks exploit vulnerabilities specific to V2X communications and vehicular applications, while others target the 5G network components. Thus, cyberattacks can be executed by attackers or malicious nodes through direct V2X SL communications or via V2N communications.

In the following paragraphs, we will provide examples of attacks categorized according to the C-I-A triad.

Availability Denial-of-Service (DoS) attacks are notable for targeting the availability of a resource, achieved by overwhelming it with a flood of network traffic or disruptive requests. The attack exploits specific weaknesses in the system or network protocols. For example, crashing an HTTP server by sending a massive number of HTTP requests, thus preventing legitimate users

from accessing it. Another scenario occurs in 5G NSs and can exploit vulnerabilities or misconfigurations related to isolation properties in a multi-tenancy[1] environment, where a malicious NS owner exhausts its own resources to affect other NS that share the same infrastructure with this malicious NS. Denial-of-Service (DoS) attacks executed through V2X SL communications can lead to the depletion of vehicle resources. Additionally, DoS attacks through cellular communications, including V2N, or the internet can strain the 5G infrastructure, including cloud resources and servers hosting the CN and RAN, as well as NS (VNFs, SDN controller), and even the AS.

Radio jamming is a form of DoS attack characterized by intentional interference with the wireless medium, which prevents access to various services. Both direct V2X communications and V2N communications are vulnerable to this attack. These attacks can be executed through various methods, including inducing interference and dropping packets by employing adversarial radio resource selection.

Distributed Denial-of-Service (DDoS) attacks involve a distributed DoS approach, wherein attackers collaborate to leverage the largest possible number of machines, including compromised ones, to carry out the attack.

Confidentiality Attacks on confidentiality encompass various threats, including sniffing, which involves eavesdropping on communications to illicitly collect private data from direct V2X and V2N communications. Another method is port scanning, which is employed to identify open ports and running services.

Integrity Integrity attacks primarily focus on the integrity of vehicular application messages. These attacks include message forgery, where malicious vehicles can alter transmitted messages (e.g., CAM) to mislead other vehicles within the transportation systems. Attackers may also conduct Sybil attacks by creating fake identities to transmit deceptive messages.

It's worth noting that an attack may consist of a combination of malicious activities. For example, it is possible to realize a DoS attack through code injection. Therefore, the list of attacks cited above is not exhaustive but rather includes common attacks.

To detect and mitigate Network Intrusions, several defense mechanisms and protocols exist. Notably, firewalls and encryption play crucial roles as the first layer of defense. They function as preventive measures, either allowing or blocking users' access to the network or system and preventing certain actions. Nonetheless, the network system remains vulnerable to infiltration by both malicious insiders and potential outsiders seeking to impersonate authorized users. The effectiveness of the initial security layer proves inadequate in detecting and countering these threats [PP18]. Thus, IDSs represent the secondary line of defense to detect these evolving threats. The following sections will introduce IDS and discuss the various approaches to IDS in 5G-V2X.

3.2 Intrusion Detection Systems

IDSs actively monitor the activities of users who have successfully passed through the initial defense layer [PP18]. IDS are defined in [Bac99] as:

[1] the capability of an infrastructure instance to serve multiple tenants, which can be applications or NSs, while maintaining isolation between them.

> *Intrusion detection is the process of **monitoring** the **events** occuring in a computer system or network, **analyzing** them for signs of security problems.*

This definition encompasses three elements: the act of *monitoring*, the *events* under observation, and the subsequent *analysis*. Building upon these elements, a taxonomy of IDSs can be dressed. This taxonomy is summarized in Figure 3.1.

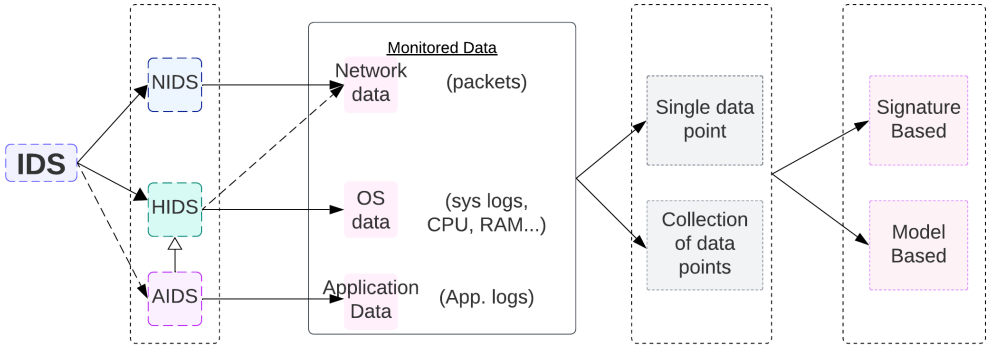


Figure 3.1: Intrusion Detection Systems taxonomy.

When determining what to **monitor**, IDS are categorized into three categories: network-based, host-based, and application-based IDS.

- Host-based Intrusion Detection System (HIDS) protect a single host machine, virtual machine, or container. They monitor various aspects, including the host's ingress and egress network traffic, the host's operating system logs, and the host's resources (CPU, RAM, disk).
- Network-based Intrusion Detection System (NIDS) monitors the network traffic in an entire network (several hosts) and can be deployed in routers, switches, and gateways.
- Application-based Intrusion Detection System (AIDS), is a sub-category of HIDS that focuses on protecting specific applications or services. They rely on monitoring the activity logs generated by the application.

The information and raw data collected, whether from a host, a network, or an application, are used to generate **events**, which are categorized as follows:

- Single data point events, such as one network packet, an observed Central Processing Unit (CPU) consumption value, or a user changing their password.
- Collection of data points events, which can be a sequence of exchanged packets, CPU consumption history, or a user's sequence of activities (e.g., logging in, changing passwords, requesting confidential data). It's important to note that a collection of data points can be aggregated and then interpreted as a single data point event.

Lastly, these events undergo **analysis**, which involves detecting potential security issues within them. This process has two approaches:

- Signature-Based IDS: This method involves comparing and matching the event against a predefined set of known malicious patterns or signatures. For example, it identifies a packet containing specific characters or observes CPU consumption exceeding a predefined threshold within a certain time frame.
- Model-Based IDS: This approach focuses on identifying events that deviate from the expected 'normal' behavior model, categorizing them as anomalies. For instance, it can detect a user logging in from a location different from their usual one. Furthermore, this method can be extended to recognize models of known malicious activities.

In recent decades, researchers have delved into the application of AI methods for IDSs, in conjunction with manually designed security policies by experts. The integration of AI contributes to enhance the effectiveness of intrusion detection, whether employing a signature-based or a model-based approach.

In the context of signature-based IDS, various approaches have leveraged fuzzy logic and FISs for the design and generation of signature rules. [MK20] reviewed articles have explored these approaches. Additionally, other proposed solutions also make use of frequent pattern mining algorithms [Siv+23] and decision trees [KT03].

On the other hand, model-based IDS primarily rely on learning AI tachniques (ML). In the literature, we find the use of ML techniques such as KNN, SVMs, and NNs. The article [HSB16] reviewed the use of these techniques in the context of IDS.

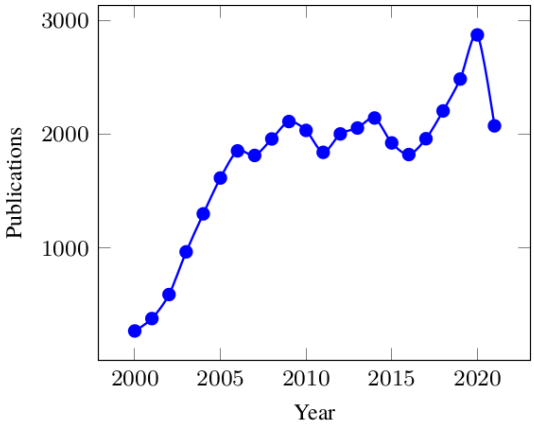


Figure 3.2: IDS stats [Lav+22]

The year 2016 marked a significant increase in research articles focused on IDS, as depicted in Figure 3.2. This rise in interest can be attributed to the emergence of DL. DL, known for its exceptional generalization capabilities, is employed to improve detection quality by reducing false positives in intrusion classification. Ongoing research is dedicated to achieving zero-day[2] intrusion

[2]are Network Intrusions that take advantage of undisclosed vulnerabilities, the targeted system have no mechanism to defend against

detection. Additionally, addressing the efficiency challenge in IDS, chapter 6 in this manuscript will focus on early detection.

IDS are also influenced by the concept of NFV. Initially, they were implemented as "physical" components, but now they are virtualized and deployed as VNFs in modern networks like cloud environments or 5G and beyond networks. The management and orchestration of IDS NFs also heavily depend on AI to enhance resource allocation and overall efficiency, optimizing their placement.

3.3 Overview of IDS Existing Approaches

In the following, we will list the approaches used to design IDSs with a particular focus on the context of 5G-V2X and approaches using AI techniques. These approaches are grouped according to the taxonomy described above, which includes the Host/Network/Application IDS, the nature of the events monitored, and the detection analysis approach.

NIDS NIDSs are used for network-based attacks, and their effectiveness in detecting threats at lower Transmission Control Protocol (TCP)/Internet Protocol (IP) stack levels, such as flooding DoS/Distributed Denial-of-Service (DDoS) attacks and port scanning, has been demonstrated. In the context of 5G-V2X, NIDSs can be deployed near the UPF when the objective is protecting the AS. Similarly, they can also be placed alongside the SDN controller when safeguarding NS components, such as NFs.

NIDSs rely on network packets, where the packets are analyzed individually using techniques such as Deep Packet Inspection (DPI) to detect patterns indicative of malicious activity. This approach presents two drawbacks: first, payload encryption makes it challenging to perform effective analysis [AMA+11]; and second, analyzing the payload takes considerable time, resulting in significant computational overhead. As a result, this approach may not be the ideal choice for real-time monitoring but it can be highly effective for passive network analysis and auditing.

Flow-based NIDSs, unlike the previous approach, do not analyze each packet individually. Instead, they construct a sequence of exchanged packets between a source and a destination, referred to as a Network Flow. This approach aligns with IDS that treat and analyze network events as collections of data points (as mentioned in paragraph 3.2). In many proposed research works [BM16; Sar+20; IA19; BB23], Network Flows are commonly aggregated into single data points that contain statistical information about the packets within the Network Flow. This information includes statistics on packet counts, arrival time, length, direction, and flags.

NIDSs employ both signature-based analysis and model-based methods. Signatures are commonly used in DPI techniques where the database of known attack signatures is continuously updated. Additionally, signatures can also be applied to aggregated Network Flow attributes for malicious flow detection purposes. In contrast, model-based methods are commonly employed in flow-based NIDS. Previous research has explored various ML algorithms, including SVM, KNN, and NNs. However, current research focuses on NNs and DL [Sho+18; AMK17; Yan+23], proposing modern and complex architectures to enhance the effectiveness and efficiency of flow-based NIDSs.

These ML/DL models are used as binary classifiers, distinguishing between "normal" and "attack" traffic, or as multi-class classifiers, categorizing traffic as "normal" or identifying the specific type of attack if present. In academic research, these models are trained using publicly available

datasets, such as NSL KDD [Tav+09], UNSW NB 2015 [MS15], CICIDS 2017 [SLG18], CICIDS 2019 [Sha+19], 5G NNIDS[Sam+22]. These datasets consist of aggregated Network Flow data, with each data point comprising numerous attributes derived from statistical analysis. In terms of attacks, these datasets encompass a diverse range of attack types.

HIDS The term "host" can encompass various elements, including the vehicle (and generally the different users' devices) and the various components of the 5G network, such as base stations, cloud infrastructure, virtual machines, containers, and servers. This highlights the diverse array of data and resources that can be monitored at the host level.

The host has access to network data from both IP-based communications and non-IP-based communications (V2X SL communications). Similar techniques used in NIDSs can be applied to the host's IP network data. Additionally, physical layer data can be effectively utilized to identify radio jamming attacks in both direct V2X and V2N communications, as outlined in the works[NGU+20; Hus+23]. Application layer data typically find applications in AIDS, and some approaches incorporate physical layer data within AIDS for enhanced detection capabilities.

HIDS also monitors resource utilization. For instance, during a DoS attack, CPU usage can be significantly affected. Detecting such attacks involves resource monitoring. HIDS monitors various resources such as CPU, memory, and I/O devices. In the context of 5G-V2X, this type of IDS is deployed at the vehicle level and at the NS level. Furthermore, infrastructure providers can also use it to monitor their physical resources to detect multi-tenancy-related attacks, this concept is exemplified in a this work [SM19] . The strategies employed include predefined threshold-based rules, with some already integrated into the operating system. More modern approaches involve model-based methods that learn "normal" resource consumption patterns of processes and detect deviations from these norms.

Additionally, the monitoring of operating system call logs constitutes another technique employed in HIDSs. These logs are leveraged to identify commands attempting to exploit system vulnerabilities, as reviewed in [Liu+18]. This category of HIDS often employs signature-based approaches.

AIDS are primarily designed to utilize application data and detect application-level anomalies. Techniques such as application log analysis are employed to monitor applications and services (e.g., VNF logs analysis). In the context of V2X applications, [Hei+19] has reviewed the approaches to AIDS and categorized them into two types: node-centric and data-centric. The former type analyzes a vehicle's behavior by verifying its compliance with protocol and use case specifications. The latter focuses on application data, ensuring its plausibility and consistency. Both approaches can interpret single data points, sequences of messages, or statistical data inferred from a sequence. They are commonly employed to analyze a node's behavior and the trustworthiness of its messages, such as CAMs. For enhanced detection effectiveness, these approaches may incorporate network data, such as using RSSI of received messages to infer distance and comparing it with the distance calculated from the sender's position in the CAM. Current research is focused on data-centric approaches for detecting attacks, such as message forgery and sybil attacks, with an emphasis on AI-based model solutions.

Conclusion

This chapter has provided an exploration of Network Intrusions and IDSs. It introduced the concept of IDSs and offered a taxonomy based on three categorizations. These categorizations are based on where monitoring is placed (Host, Network, Application), how data is treated (single data point or collection of data points), and how the monitoring analyses the data.
This chapter serves as the concluding segment of Part I of this manuscript, effectively bridging the preceding chapters on 5G-V2X and AI. Within this context, IDSs approaches are discussed, highlighting the intersection between three core concepts: 5G-V2X, AI, and IDSs.

Next, we will proceed to part II of this manuscript, where we will begin addressing the research questions introduced in the manuscript's Introduction. This part encompasses our contributions to Host-based Intrusion Detection Systems (HIDSs).

Part II

Host-based Intrusion Detection Systems (HIDSs)

Chapter 4

Coping with adversarial radio resource selection in SideLink V2X communications

Introduction

In this chapter, we present the first contribution of the second part of our manuscript, which is dedicated to Host-based Intrusion Detection Systems. The contribution in this chapter is an attempt to address a specific type of distributed jamming attack that occurs in V2V SideLink (SL) communications, which is adversarial radio resource selection attack. To achieve this, we propose a HIDS with a collaborative approach for detecting the occurrence of this attack. Additionally, we present a mitigation scheme to alleviate its impact.

The subsequent sections of this chapter are structured as follows: The next Section reminds the communication protocol used in V2V SL communications and discusses the vulnerabilities of the Semi-Persistent Scheduling (SPS) protocol to adversarial resource selection attacks, and Section 4.2 summarizes our contribution. Section 4.3 reviews related works. Section 4.4 discusses the methodology in three steps: first exploring the attacker's strategy and the intuitive solutions to it, then explaining the proposed HIDS, and finally the mitigation strategy that leverages Fuzzy Inference Systems (FISs). Section 4.5 discusses the simulation results before concluding the chapter.

4.1 Context

The allocation of radio resources for SideLink (SL) communications can occur through two distinct approaches, namely centralized and distributed allocation, as documented in [Har+21].
In centralized mode, formerly referred to as Mode 3 in LTE-V2X and currently denoted as Mode 1 in 5G-V2X, the cellular base station is responsible for managing radio resource allocations for vehicles direct communications. However, this mode is not commonly used in practice due to the limited cellular coverage in certain areas.

Furthermore, the signaling process between the vehicle and the base station for resource alloca-

tion introduces an additional delay that may not align with the stringent latency demands of the V2X environment [AKG18]. On the contrary, in the distributed mode, designated as Mode 2 in 5G-V2X (and formerly referred to as Mode 4 in LTE-V2X), vehicles take on the responsibility of allocating their radio resources. This mode can operate even when cellular coverage is unavailable. It allows for self-radio resource allocation without reliance on a base station, thus helping to reduce the additional communication delay associated with centralized modes.

This distributed resource assignment is performed using the sensing-based algorithm named Semi-Persistent Scheduling (SPS) [3GP19]. The SPS algorithm leverages information about neighboring vehicles' resource usage patterns to allocate radio resources. Once a vehicle identifies an available resource, it reserves that specific frequency resource for transmitting a certain number of consecutive messages, such as CAMs. In such a context, the resource reservation duration depends mainly on both Resource Reservation Interval (RRI) and Re-selection Counter (RC) [Gar+21].

The RRI is the time interval between two consecutive messages (packets); for instance, the RRI is set at 100 ms during the exchange of CAMs. RC represents the number of transmissions a vehicle is permitted before being required to select a new Ressource Unit (RU). It is a value randomly chosen from the range of 5 to 15 before each resource reservation. After each transmission, the value of RC decrements until it reaches zero. In this case, either the reserved resource will be kept (with a probability p_k), or a new resource reservation procedure will be initiated. [MG17]. The described resource scheduling procedure is illustrated in Figure 4.1.

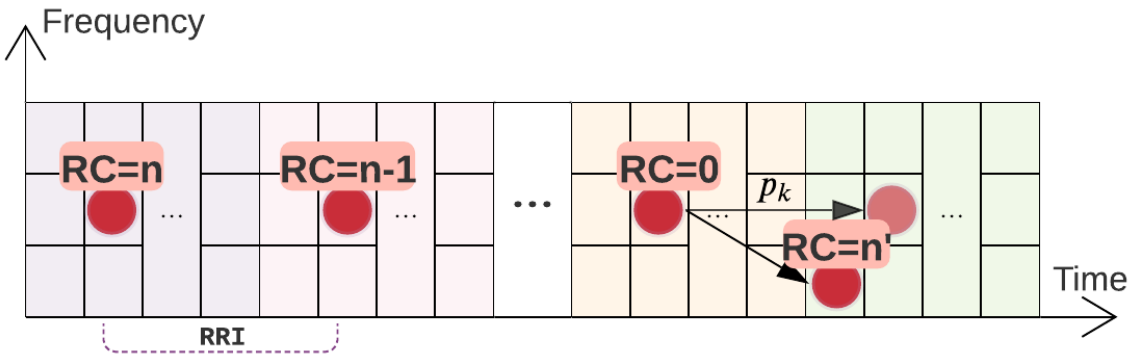


Figure 4.1: Resource scheduling procedure.

Utilizing the resource reservation patterns of neighboring nodes can significantly reduce communication packet collisions[1], especially in the context of distributed allocation. However, it's important to be aware that malicious nodes, can also exploit this aspect to launch adversarial resource selection attacks, which are a type of jamming attack that compromises the availability property. The scenario of this attack is illustrated in Figure 4.2.
A malicious vehicle may intercept resource reservation information from a nearby vehicle and use it to transmit on the same resource. This information can be readily deduced from the SL Control Information (SCI) header associated with the transmitted packets, as depicted in the left part of

[1]occurs when two or more packets collide and interfere with each other as they attempt to transmit over the same RU, leading to data corruption or loss.

Figure 4.2. Consequently, the victim vehicle becomes unable to send its subsequent messages to its neighbors, as shown in the right part of Figure 4.2. This results in isolating the victim vehicle from its neighbors, all without the victim being aware that it's under attack. Moreover, if dropped packets include emergency and alarm information (DENMs), the adversarial resource selection attack, may cause devastating effects in the vehicular network and lead to accidents and fatalities.

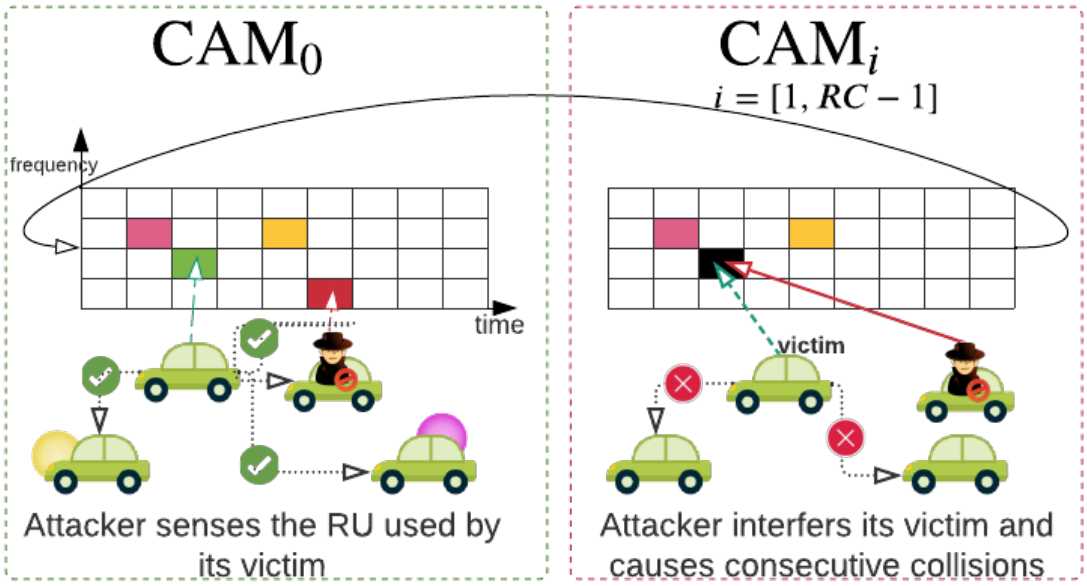


Figure 4.2: Adversarial resource selection jamming attacks.

4.2 Contribution Summary

We propose an enhanced SPS scheme designed to defend against adversarial resource selection attacks. Our approach addresses attacks initiated by single nodes as well as distributed attacks carried out through the cooperation of multiple smart attackers.
The proposed detection mechanism functions as a HIDS since it is deployed at the vehicle level. Detection is accomplished collaboratively with neighboring vehicles through a feedback mechanism that informs vehicles about collisions.
Uniquely within this particular contribution, as opposed to others, we introduce a mitigation strategy aimed at preventing an attack once it has been detected. We leverage a Fuzzy Inference System (FIS) to devise an effective defense policy [Ben+19; Tam+20]. Specifically, we dynamically adjust the RU reservation time (RC) based on the context, i.e., being attacked or not.

4.3 Related Works

In [Xu+04], the authors proposed two defense mechanisms to avoid interference and jamming attacks in wireless networks: channel surfing and spatial retreat. The former relies on continuously switching the channel when being attacked, while the latter is suitable for mobile nodes that can move to a safe place outside the interference zone. However, spatial retreat is not suitable for vehicular networks due to the constrained mobility of vehicles.

Existing works can be classified into three main categories: (*i*) works exploring jamming attacks in wireless networks [Xu+04]; (*ii*) works studying channel surfing approaches against these attacks in vehicular networks [NGU+20; YK21]; (*iii*) works addressing how to reduce packet collisions [JCK19; WS20].

To mitigate jamming attacks in vehicular networks, the authors of [NGU+20] suggested random channel switching among available service channels, without considering attack detection in this channel surfing-based approach. Similarly, the authors in[YK21] introduced an evasive approach to counter adversarial ressource selection attacks. Their detection mechanism relies on feedback from the first neighboring vehicle using a RU in the same frequency domain as the victim, indicating collisions in the SCI. Upon detecting an attack, the victim switches to a random sub-frame within the same sub-channel for its RU. This reduces the attacker's ability to predict the victim's resource usage but increases the occurrence of legitimate collisions, as an unreserved RU is chosen each time. Moreover, this approach does not account for the possibility of multiple attackers; another attacker could impersonate the feedback provider and disrupt the detection process.

Other works aimed to enhance and improve the distributed mode in V2X SL communications. In [JCK19], the authors proposed a method to reduce continuous collisions by reserving and alternating between multiple resources. In [WS20], the authors presented a mechanism to reduce collisions by allowing vehicles to provide explicit feedback on channel conditions and acknowledge successfully decoded radio resources. Additionally, they designed a candidate resource selector to extend the sensing range and minimize hidden terminal situations.

Our specific focus is on adversarial radio resource selection jamming attacks in 5G-V2X. Unlike the works mentioned earlier, we address both the detection and mitigation of these attacks in scenarios involving single and multiple smart jammers.

4.4 Methodology

In this section, we outline our effective defense strategy for countering adversarial resource selection jamming attacks. We begin by considering the attackers' strategy for maximum damage. Next, we introduce a feedback-based detection strategy as the initial step of our defense, followed by our approach to mitigate these attacks.

4.4.1 Attacker strategy and intuitive solutions

The primary goal of adversarial resource selection attacks is to maximize consecutive packet collisions against a victim, preventing nearby vehicles from decoding the victim's messages. An attacker,

being an internal node, has all the necessary information for a successful attack through gathered information from SCI and CAMs.

Through the received CAMs, an attacker targets a nearby vehicle that is taking the same road. Once the target is selected, the attacker tries to keep the victim in range by matching its speed and following it. The attacker senses the used resource and reservation time from SCI and then schedules its CAMs on it. The attacker continues to sense the victim's information to detect any changes in the victim's resource allocation (for re-selection or evasion). When such a change occurs, the attacker switches to the newly allocated resource. An attack can last for a duration equivalent to the entire reservation time, precisely $RC-1$, during which the victim is only able to successfully transmit the first CAM, as shown in Figure 4.2.

In the case of multiple attackers, they ensure that the victims are distinct. So, the number of jammed nodes is equal to $min(N_{attackers}, N_{legitimates})$. Consequently, there are $N_{legitimates}$-$N_{attackers}$ legitimate nodes transmitting messages correctly (except for some normal occurrences of legitimate collisions). Despite the potential harm posed to the targeted vehicles, attackers are often categorized as non-aggressive jammers. This classification is based on the belief that their impact on the system is relatively minor. The rationale for this categorization is further elucidated in later sections, with supporting evidence presented in the simulation results, as illustrated in Figure 4.5.

The first intuitive solution is to empower targeted vehicles to choose a new RU after each transmitted CAM. This approach sets the value of the RC equal to 1. By doing so, we effectively thwart attackers from identifying the RU employed by victims, rendering the attack ineffective and futile. Unfortunately, this solution does come with the drawback of increased legitimate packet collisions since there is no resource reservation step. Another approach is to make the RC adaptive. When a vehicle comes under attack, it initiates multiple consecutive RU re-selections after each period. In this case, the attacker may attempt to compete and locate the victim but will struggle to keep up and will eventually give up. To further reduce legitimate collisions, the victim can gradually increase its reservation period. However, if a new attack is detected, it repeats the process once more.

4.4.2 Attack detection strategy

In the context of broadcast 5G-V2X transmissions, a blind re-transmission approach is employed to enhance reliability, as recommended in [3GP20]. This technique is particularly useful for managing packet collisions. However, it's important to note that the RU utilized is included in the SCI. As a result, the network remains susceptible to adversarial resource selection, which can impact the system's security.

Another vulnerability in SL communications is the inability of the victim to detect attacks due to the half-duplex mode and the broadcast nature of communication. This limitation prompted the development of our feedback mechanism for collision (or attack) detection. In cases of packet collisions, neighboring nodes may detect the transmitted CAM due to a high received signal on the Reference Unit RU, but they may fail to decode it, making the transmitter (T_x) unknown. Thanks to the reservation mechanism, neighboring vehicles can consult the table of recently received CAMs. If the RU was reserved prior, T_x can be easily and confidently identified. Subsequently, the close neighboring vehicles within communication range of the victim need to provide feedback reporting

the occurrence of the collision to T_x.

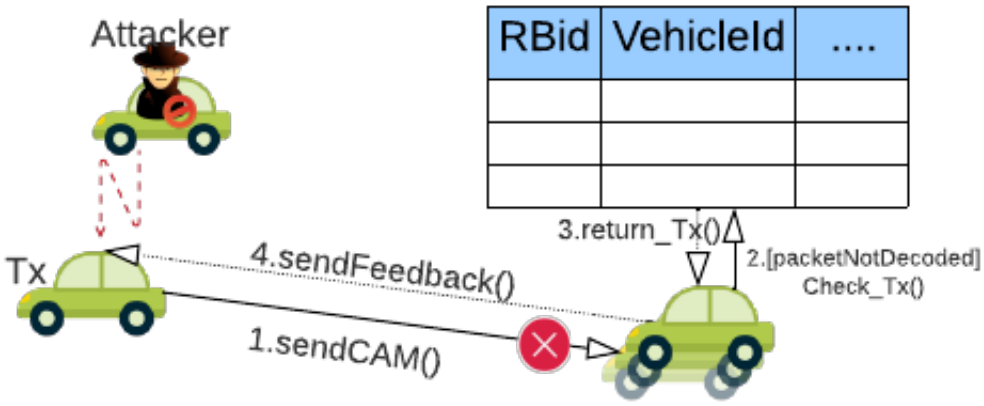


Figure 4.3: Communication diagram: feedback sending.

We assume that the feedback is sent in unicast mode (supported by 5G-V2X). However, the feedback is aborted if it is considered obsolete, i.e., the RU was not or is no longer used by the vehicle receiving the feedback. We also assume that the feedback is always correctly received.

If the feedback ratio of a transmitted CAM is greater than a threshold (ϕ), the vehicle will consider the packet as dropped and initiate an immediate resource re-selection. The feedback ratio is calculated by dividing the number of received feedback on the number of neighboring vehicles (which is the number of the last received CAMs). As shown in steps 3 and 5 of Algorithm 1, the threshold is used both to reduce false positive alarms resulting from infrequent legitimate collisions and to address trust concerns arising from potential malicious nodes injecting false feedback (this type of attacker is not considered in this contribution).

Algorithm 1 FeedBackListener

Input: fb: FeedBackMessage
number_feedback: Array[][]
$number\_close\_neighbors$: Integer
Output: number_feedback
1: **if** Check_RU(fb.RU$_{id}$) **then**
2: number_feedback[fb.RU$_{id}$,fb.t] ++
3: $\alpha \leftarrow \frac{number\_feedback[fb.RU_{id}, fb.t]}{number\_close\_neighbors}$
4: **if** $\alpha > \phi$ **then**
5: RC $\leftarrow$ 0
6: **end if**
7: **end if**

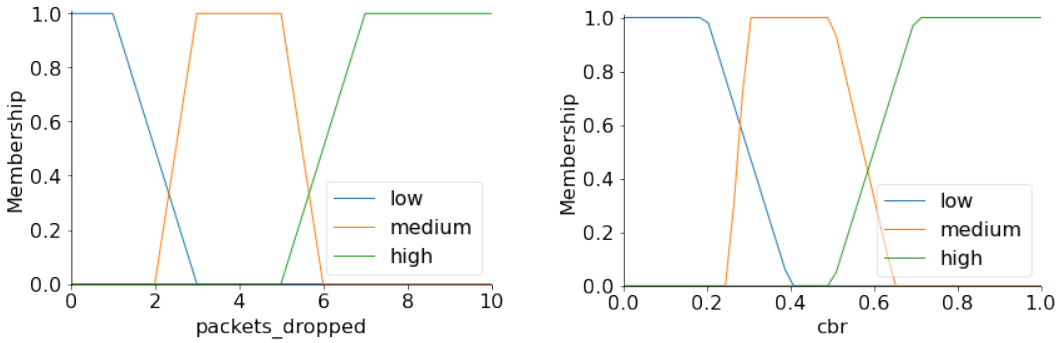
4.4.3 Attack mitigation strategy

After detecting an attack, the victim vehicle must update its RC value to select a new radio resource. To accomplish this, we have developed a FIS that determines the reservation period for each subsequently selected RU. This innovative resource reservation scheme is detailed in Algorithm 2 where, in step 4 of the algorithm, the RC value is updated according to our designed FIS.

Algorithm 2 Ressource Reservation

Require: $RC == 0$
{SPS Scheme}
1: channelSensing()
2: RU ← ressourceSelection()
3: cbr ← calculateCBR()
{Updating RC}
4: RC ← fis(number_feedbacks[:,t-Λ:t],cbr)

The remaining paragraphs in this sub-section will explain our FIS. For the fuzzifier, we consider two inputs that we deem relevant for the corresponding output (RC value). The fuzzy sets are shown in Figure 4.4.



(a) Number of dropped packets

(b) CBR

(c) RC

Figure 4.4: Fuzzy sets.

- **Number of Dropped Packets**: the number of dropped packets in the observation interval ([t-Λ, t]), helps classifying the collisions as legitimate or malicious, through three fuzzy terms: low, medium, and high.
- **Channel Busy Ratio (CBR)**: represents the time ratio the channel is sensed as busy on the total observation time, as defined in SPS scheme. The higher the CBR is, the more vehicles will struggle to find a RU, and hence the channel congestion. CBR depends mainly on vehicle density.
- **RC value**: represents the output of our FIS. We distinguish four fuzzy terms: One (fuzzy singleton), Low, Normal, and High. One is used when trying immediately to escape from the attacker, Low represents the gradual increase towards the Normal state as defined in the standards [Har+21], High RC values are greater than 10.

Our strategy is to enable vehicles to escape by sequentially changing resources. However, our goal is also to prevent constant resource reselection when legitimate collisions occur. This typically happens in congested traffic situations where either CBR is high or the change of RU is frequent.

As for the inference engine, we have established a set of fuzzy rules for decision-making, which were validated through tests. Table 4.1 illustrates the set of fuzzy rules we developed. These rules

are designed to help our inference system strike the right balance between the need to sequentially change RU to escape attacks and the need to remain on the same RU when legitimate collisions occur.

If the number of packets dropped by the vehicle is considered high, it definitely indicates an ongoing attack, regardless of the CBR value. In this case, the only solution is to set RC to one in order to evade the attacker. However, if the number of packets dropped by the vehicle is considered medium, it is challenging to determine whether they are being dropped due to an attack or due to legitimate collisions. In such cases, it is necessary to check the CBR value to devise the optimal strategy:

- High CBR: Changing RC will likely cause collisions with other neighboring vehicles since the probability of finding an available RU is low.
- Medium CBR: The available RUs will allow the vehicle to escape the attack or avoid legitimate collisions by gradually diminishing RC to increase the rate of re-selection.
- Low CBR: The system will not suffer any side effects, therefore it is better to put RC to one in order to clear out any kind of collisions (malicious or legitimate).

In the case of a low number of packets dropped, the value of RC remains normal unless the CBR is low. In such a scenario, a higher value is assigned to RC, which, in turn, slows down the re-selection procedure.

Inputs		**Output**
Dropped Packets	**CBR**	**RC**
HIGH	-	ONE
MEDIUM	HIGH	NORMAL
MEDIUM	LOW	ONE
MEDIUM	MEDIUM	LOW
LOW	HIGH	NORMAL
LOW	MEDIUM	NORMAL
LOW	LOW	HIGH

Table 4.1: List of fuzzy rules.

For de-fuzzification, we used the centroid method which is calculated as follows:

$$x^* = \frac{\int \mu(x) x dx}{\int \mu(x) dx}$$

where:

- x^* the output value;
- $\mu(x)$ is the RC membership value for the point x.

4.5 Simulation and Results

To validate the proposed scheme and analyze its performances, we used LTE-V2V simulator [Cec+17]. It implements the sensing-based SPS scheme used in LTE-V2X mode 4, which is similar to 5G-V2X mode 2. We performed the simulation over 20 seconds on a 2 km road of 3 lanes per direction. The vehicles follow Poisson distribution in their positioning with a density $\rho \in \{50, 75, 100, 125, 150\}$ vehicles per km. The vehicles send CAMs packets at 10Hz frequency with a transmission power of 23 dBm. Two main 2 sub-channels per sub-frame are used by vehicles to send their packets and hence 200 RU.
We compared our scheme performance to the SPS scheme according to the Packet Reception Ratio (PRR). PRR represents the ratio between the number of vehicles that correctly received CAMs and the total number of vehicles within the communication range of the transmitter vehicle. We set the number of attackers to $N_{attackers} \in \{0, 1, 5, 10\}$ in each scenario, where the number of victim nodes is equal to the number of attackers. A packet is presumed to be dropped if $\phi \geq 0.3$. We found that this value is effective in managing the false alarm rate. However, in this contribution, we will not delve into further details, as we do not cover cases where attackers are capable of sending false feedback. The observation period λ is set to 20 sub-frames.

Figure 4.5 shows the impact of the adversarial resource selection jamming attacks on the average PRR value when varying both vehicle density (ρ) and the number of attackers. It is noteworthy that in the case of zero attackers, the reduction in PRR is primarily attributed to legitimate collisions. However, we observe minimal differences in PRR values between scenarios with no attackers and scenarios with a single attacker. This contrast becomes noticeable only in low-density scenarios, ranging from 50 to 75 vehicles. Furthermore, we notice a consistent decrease in PRR as vehicle density increases across all tested cases.

In high-density scenarios, the disparities in PRR become negligible, with an approximately 2% difference between scenarios with 0 and 5 attackers and a 4% difference between scenarios with 0 and 10 attackers in each case. Remarkably, the average PRR value remains above 87% for all considered scenarios, regardless of vehicle density. These findings clearly emphasize the limited effectiveness of these attacks on the overall system, which is why they are called non-aggressive.

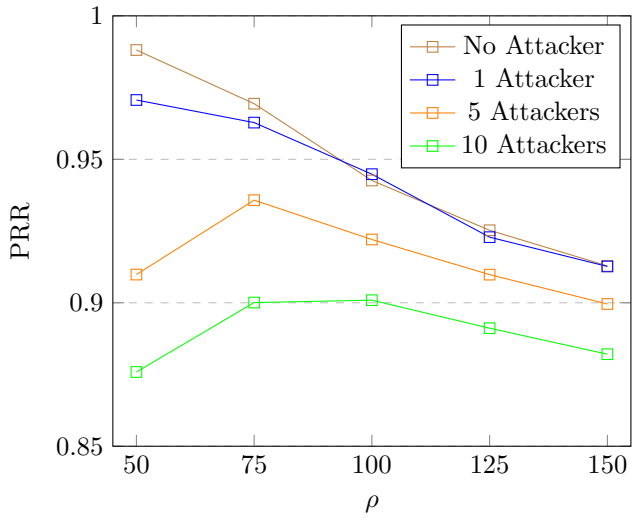


Figure 4.5: Impact of adversarial resource selection attacks on PRR.

To assess the effectiveness of our proposed approach, we conducted an experiment in a scenario with a vehicle density of $\rho = 150$ vehicles/km. We analyzed the PRR for three different vehicles:(i) vehicle A: safe (unattacked), (ii) vehicle B: subjected to attacks and did not implement our approach, and (iii) vehicle C: Under attack but implemented our approach.

The results are presented in Figure 4.6, which shows the Cumulative Distribution Function (CDF) of PRR for each CAM sent by the different vehicles.

The findings reveal a significant disparity in PRR among the three vehicles. Vehicle B, which did not employ our approach, exhibited poor performance, with only 10% of its packets achieving a PRR of one. This contrasts with Vehicles A and C, which achieved PRR values of 60% and 55%, respectively.

Vehicle B was severely affected by the attacks, with 85% of its packets experiencing a PRR below 20%, effectively isolating the vehicle from the network. In contrast, Vehicle C, which implemented our approach, achieved a PRR similar to that of a safe vehicle (Vehicle A). Impressively, 65% of Vehicle C's packets achieved a PRR greater than 80% (compared to 77% for safe vehicles).

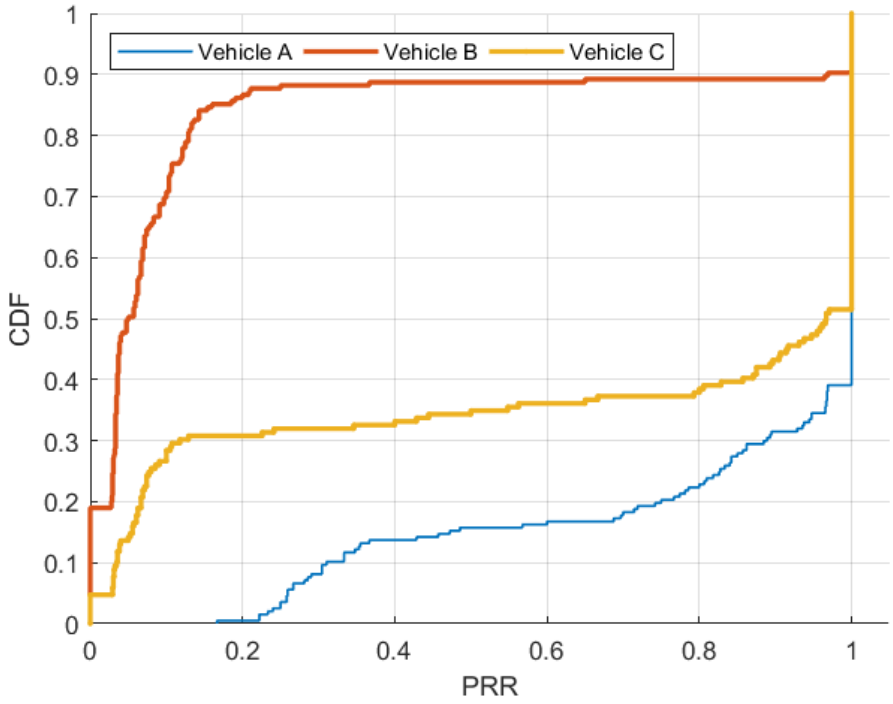
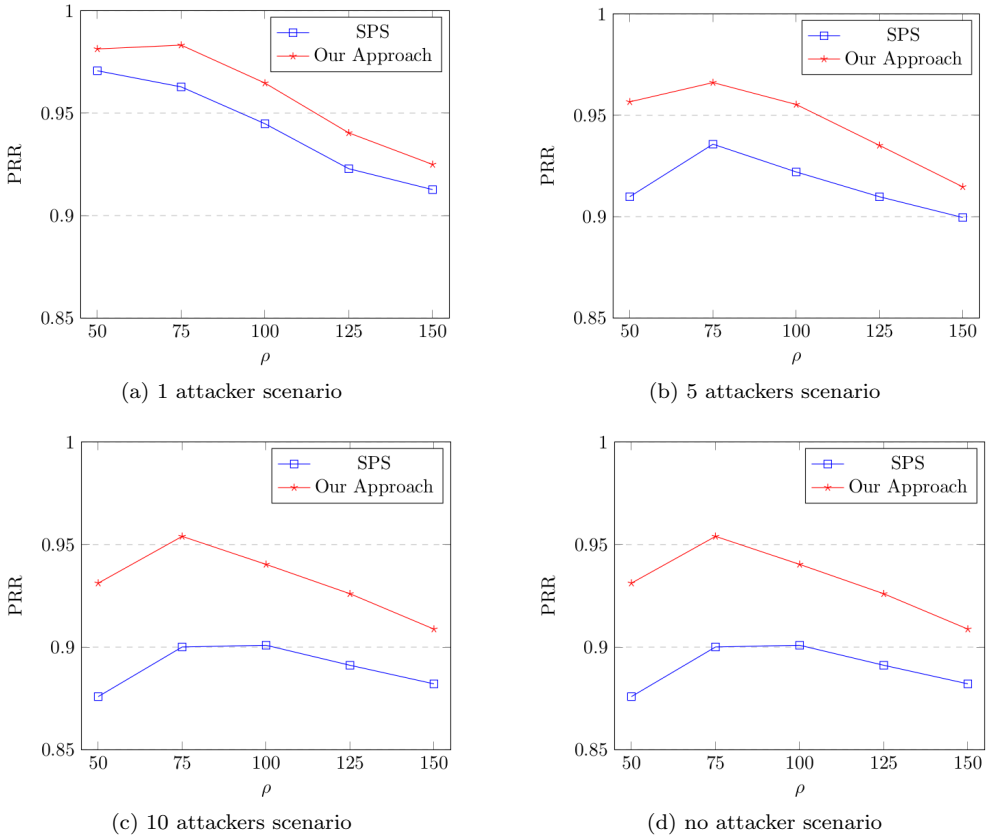


Figure 4.6: CDF of PRR.

Figure 4.7 compares the SPS scheme to our approach across different attack scenarios. Regardless of the number of attackers, our approach consistently achieves a higher PRR compared to the SPS scheme, demonstrating its effectiveness in defending against adversarial resource selection attacks.

Even in scenarios with no attackers (Figure 4.7d), we observe an improvement in PRR. This enhancement is a result of the reduction in consecutive legitimate collisions, achieved through the feedback mechanism that triggers re-selection in vehicles. Additionally, the improved adaptability of the RC, which now considers congestion situations, contributes to this performance improvement.



(a) 1 attacker scenario

(b) 5 attackers scenario

(c) 10 attackers scenario

(d) no attacker scenario

Figure 4.7: SPS vs. our defending approach.

Conclusion

In this chapter, we proposed a HIDS that addresses adversarial resource selection jamming attacks. Additionally, we introduced a mitigation algorithm based on FIS to counter these attacks, demonstrating the effectiveness of the detection protocol. We also demonstrated how our proposed solutions improve the SPS algorithm by minimizing the number of legitimate collisions, thus ensuring better availability.
The next chapter will delve into the integrity property of CAMs, where we will develop our AIDS to deal with message forgery attacks in V2V communications.

Chapter 5

Detecting message forgery attacks in V2X communications

Introduction

The previous contribution aimed to ensure the reliability of CAMs against jamming attacks. In this chapter, our focus shifts to awareness messages integrity. Indeed, CAMs can be altered, and for safety reasons, when a vehicle receives these awareness messages, it should be able to detect their legitimacy. To address this issue, this chapter proposes an Application-based Intrusion Detection System (AIDS) that inspects CAMs and detects message forgery attacks. Our approach leverages RNNs and incorporates historical data to analyze the behavior of vehicles for effective detection. Furthermore, it approach guarantees low computational overhead to ensure efficient detection. We will demonstrate these two findings in this chapter.

The remainder of this chapter is organized as follows. Section 5.1 provides more context and details about the attack. Section 5.2 summarizes our contributions. Section 5.3 comprehensively reviews existing approaches to AIDS that deal with message forgery. The considered attack scenario, as well as the dataset used, are described in Section 5.4. We detail our designed RNN-based approach in Section 5.5. Section 5.6 presents and discusses the performance of our approach, and then we conclude the chapter.

5.1 Context

Cooperative Awareness Messages (CAMs), as introduced in Section 1.3, constitute the fundamental component of various safety applications in 5G-V2X and beyond, where vehicles periodically exchange information regarding their position, velocity, heading, and other mobility details with surrounding vehicles, pedestrians, and road infrastructure [ADC21]. These messages necessitate a 90% reliability rate and require a maximum latency of 100 ms [TM21]. Given these stringent requirements, the direct (SL) communication mode (V2V/V2I/V2P) is employed [Sed+23].

In a previous section of this manuscript (Section 3.1), we discussed message forgery in vehicular

network attacks, which are attacks on the integrity property. In such attacks, a malicious node can manipulate its CAMs, potentially leading to confusion or misinformation among nearby vehicles, pedestrians, or infrastructure entities. The malicious node can have various motivations, including economic gain, disruption of road traffic, or even causing road accidents. This specific type of attack is often referred to as false information injection, as documented in [ASJ19].

Real-time detection of message forgery is primordial for vehicular networks, given that CAMs arrive at a frequency of 10Hz, and needs to be analyzed in time by the vehicles. The detection system needs to accurate and precise in identifying the type of fake received messages to enable effective mitigation policies.

5.2 Contributions summary

To address the previously mentioned challenges of real-time and accurate intrusion detection, we propose a novel AIDS that relies on a data-centric approach, performing consistency checks (refer to Section 3.3 for details about AIDS approaches). The proposed approach leverages DL, specifically RNNs, to perform a classification task on received CAMs, distinguishing between benign and tampered messages.

RNNs can embed historical knowledge into their hidden states and convey it over time, allowing the AIDS to infer the class of the received CAM, while taking into account the past behavior of the vehicle; it is demonstrated through experiments that including historical data improved the performances of the proposed detection model. The longer the historical data sequence considered, the better the performance of the model. Furthermore, the AIDS keeps track of the hidden states and performs one RNN cell feed per received CAM. This allows for a more efficient detection process. The main contributions of this chapter are summarized as follows:

1. An innovative RNN-based model to detect misbehavior in 5G-V2X networks, utilizing either LSTM or GRU. The performance of the model is extensively evaluated by varying meta-data such as the number of layers and the dimension of the RNN cell,
2. A review of the effect of incorporating historical data in AIDSs deployment in 5G-V2X, providing insights into how this can enhance the effectiveness of attacks detection.
3. Extensive discussions on the effectiveness of the proposed model, including theoretical complexity analysis and comparison with existing approaches.

5.3 Related works

Various AIDS approaches have been employed in the literature utilizing different algorithms and classifiers like rules-based classifiers and ML-based approaches (e.g., SVM, RF, NB, DL) [ASJ19]. As mentioned in the introduction section, AIDS in this context can be categorized into two categories: node-centric and data-centric. For the scope of this contribution, we will focus on data-centric approaches and emphasize the ML/DL techniques due to their potential and high performance in various related fields. Data-centric based models are fed with **plausibility** or **consistency** inputs to perform the classification task more accurately. Plausibility checks ensure the correctness of an

input based on drawn evidences, while consistency checks verify the absence of contradiction among a set of inputs.

Authors in [Sin+19] proposed a binary-class ML algorithm for message forgery detection in vehicular networks. Their model takes input in the form of received coordinates, received speed, relative coordinates between the sender and receiver, and relative speed on the 3D plan X,Y,Z. The proposed approach is plausibility-based AIDS, where each received message is classified as normal or malicious. The authors investigated two classifiers: SVM and logistic regression, they concluded their paper by stating that SVM achieved better performance than logistic regression. [SL21] proposed a data-centric ML-based approach. The authors introduced two plausibility checks, namely location plausibility and movement plausibility. The location plausibility is determined by investigating the acceleration, and the movement plausibility verifies if the vehicle remains idle. These two plausibility checks are then aggregated with the received position and fed to the ML model. The authors implemented several ML classifiers, including SVM, RF and NB. The results demonstrated that, due to the incorporation of plausibility checks, their proposed approach can achieve high-quality detection.

In the context of consistency checks in AIDS, [Gro+21] proposed a semi-supervised[1] approach to a binary classification of received messages in the context of V2I communication, where detection is performed at the infrastructure, or RSU, level. The approach leverages edge computing, and the detection model is based on a RNN, specifically LSTM. The model takes a fixed-length sequence of CAMs, considering both position and velocity. The model tries to regenerate the sequence of the positions and is trained only on legitimate data; the aim was to learn the legitimate behavior pattern. After training the LSTM network, the authors determined the reconstruction error threshold in a supervised[2] manner. They used a test dataset containing both normal and malicious CAM sequences to select this threshold. The opted threshold is then used by AIDS to classify the received CAMs sequence as either normal or abnormal. This approach is limited to binary classification. Additionally, the fixed-length sequences used by the RNN model can be a drawback, as it only allows for learning consistency within this specific time window, and not across the entire original sequence. The approaches in the literature that aim to verify the coherence of sequences are mainly based on the idea of reconstruction of sequences as in [Gro+21]. In [All+21; HCT22], the input sequence is first passed through a Convolutional Neural Network (CNN) layer, which captures the spatial relationships between inputs, and then through the RNN layers. However, this approach has additional drawbacks, including the need for all sequence classification after each received message, which causes more computational overhead. Thus it may not meet time-sensitive requirements.

Compared to existing literature, our research work aims to perform CAMs classification while considering the historical data to ensure the integration of consistency verification. Moreover, our approach overcomes the limitation of fixed-length sequences, allowing the use of the entire sequence for classification. Another advantage of our approach is its low computational cost, as we only need to use one RNN cell fed per received CAM. Thus, our approach represents a significant improvement over existing methods.

[1] ML/DL model is trained on a combination of labeled and unlabeled data.
[2] ML/DL model is trained on a labeled dataset, where the input data is paired with corresponding output labels.

5.4 Attack scenario and Dataset

This section explains the attacker model and describes the dataset constructed for the proposed AIDS.

5.4.1 Attack scenario

The 5G-V2X scenario involves multiple moving vehicles that communicate with each other using a PC5 interface [Gar+21] since our focus is SL communications. Additionally, vehicles are equipped with a Global Positioning System (GPS) sensor that continuously provides position and velocity information in the X, Y, and Z planes. This information is used to generate CAMs, which are periodically broadcasted via the PC5 interface to neighboring vehicles [Baz+21]. Moreover, vehicles also receive CAMs sent by surrounding vehicles. Misbehaving vehicles, acting as attackers, are part of the vehicular network that is being studied. A misbehaving vehicle claims in its CAMs information that contradicts the ground-truth information obtained through GPS. We consider the attack scenarios as defined in [HLK18]. The misbehaving vehicle is able to perform four different types of misbehavior:

- *Constant*: Predefined position coordinates are used,
- *Constant offset*: A predefined offset is added to the position coordinates,
- *Random*: Random position coordinates are generated,
- *Random offset*: A random offset is added to the position coordinates,
- *Eventual stop*: The vehicle initially behaves normally but eventually stops and repeatedly transmits its current position.

The Vehicular Reference Misbehavior (VeReMi) dataset is a reference dataset used in the literature for detecting misbehavior in vehicular networks. It was simulated using the Vehicular Network Simulation (VEINS) simulator within the Luxembourg SUMO Traffic (LuST) scenario. The dataset includes the five attacks described above. The data is generated after several repetitions of each simulation scenario. A simulation scenario is defined by the attack type, vehicle density, and density of malicious vehicles within the simulation. As a result, the data generated through one simulation contains both legitimate (normal) and malicious data. VeReMi dataset contains three types of data:

- *GPS data* (*type2*) in the dataset: the position and speed on the plane X,Y,Z received by vehicle (v_{id}) through the GPS,
- *Generated CAM (type4) or ground-truth data*: a received CAM is identified by the concatenated key (s_{id},message_id), where s_{id} represents the sender's ID, the CAM message contains the data generated in (type2). *type4* messages contains also the nature of the sender vehicle: legitimate or malicious, and the specific type of malicious behavior,
- *Received CAM (type3)*: a received CAM is identified by the concatenated key (s_{id},message_id,r_{id}), where s_{id} and r_{id} represent the sender and receiver ids, respectively. The (s_{id},message_id) is the message generated in (*type4*); in case of a misbehaving vehicle, the sent data is altered and forged. The *type3* message also includes the Received Signal Strength Indicator (RSSI), which is a measure of the signal power received by (r_{id}) vehicle.

5.4.2 Dataset Construction

Our work aims to utilize sequential data. The generated sequences allow capturing the temporal dynamics in the raw data contained in VeReMi. Therefore, we made certain modifications to the raw VeReMi dataset. In the constructed dataset, a data point x_{t_t} is defined in Equation 5.4.1.

$$\begin{aligned} x_{t_t} :<& 3\_\text{t}, 3\_r_{id}, 3\_s_{id}, 3\_\text{m\_id}, 3\_pos_{x,y}, 3\_spd_{x,y}, 3\_rssi, \\ & 2\_pos_{x,y}, 2\_spd_{x,y}, 4\_tag > \end{aligned} \tag{5.4.1}$$

The prefix {1,2,3}_ represents the source type of data in raw VeReMi; the data point x_{t_t} is created using type 3 data, 3_t is the receiving time, ($3\_r_{id}, 3\_s_{id}, 3\_\text{m\_id}$) are the sender id, receiver id, and the message-id, respectively. $3\_pos_{x,y}, 3\_spd_{x,y}$ is the received CAM, contains the sender's (s_id) claimed position and speed on X,Y. 3_rssi is the RSSI value. $2\_pos_{x,y}, 2\_spd_{x,y}$ are the receiver's current position and speed. This is the recent data that r_id received through GPS. 4_tag represents the type of the sending vehicle, obtained from the type 4 data for the key (s_{id},m_id).

A vehicle with the ID s_{id} broadcasts CAMs to its neighboring vehicles in a simulation. If a vehicle with the ID r_{id} is within the communication range of vehicle s_{id}, it will receive a sequence of consecutive CAMs for a certain period of time in the interval ($[t_0, t_L]$), where t_0 is the time of the first received message, and t_L is the time of the last received message before r_{id} is no longer in communication range of s_{id}. We define this sequence of messages as the communication sequence between vehicles s_{id} and r_{id}. The communication sequence is then transformed to form a sequence x of time-ordered x_{t_t}, where $x = \{x_{t_t}/t \in [0, L]\}$, L is the length of the sequence, and each x_{t_t} represents a data point defined above. The constructed dataset (X) contains a set of x sequences. For the VeReMi dataset, we observed that the maximum sequence length was 100.

5.5 Methodology: RNN-based AIDS for 5G-V2X and Beyond

In this research approach, we describe the proposed DL-based AIDS model designed to be deployed at the vehicle (host) level. This model is responsible for analyzing the receiving message before it is processed by the receiving vehicle. To achieve this, the model takes into consideration the history of CAMs received from the same sender that precedes the currently received message. By doing so, the model is able to analyze the consistency of the sender's CAMs, which is made possible using an RNN model.

Figure 5.1 illustrates how the model classifies each received CAM from the sender, while also taking into account the history of CAMs embedded in the hidden state of RNN. This enables the model to detect inconsistencies quickly and efficiently.

The DL model takes an input (x_{t_t}) a representation of the t^{th} received CAM and produces its class label as an output (p_{t_t}). The class label corresponds to either a legitimate CAM or one of the five defined attacks in the attacker model: Constant, Constant offset, Random, Random offset, and Eventual stop.

The input x_{t_t} is the features vector:

$$< \text{pos}_x, \text{pos}_y, \text{spd}_x, \text{spd}_y, \text{rssi}, \text{dst} >$$

The features $\text{pos}_x, \text{pos}_y, \text{spd}_x$, and spd_y represent the claimed position and speed of the sender along the X and Y planes, while *dst* denotes the Euclidean distance between the receiver and the sender's reported position.

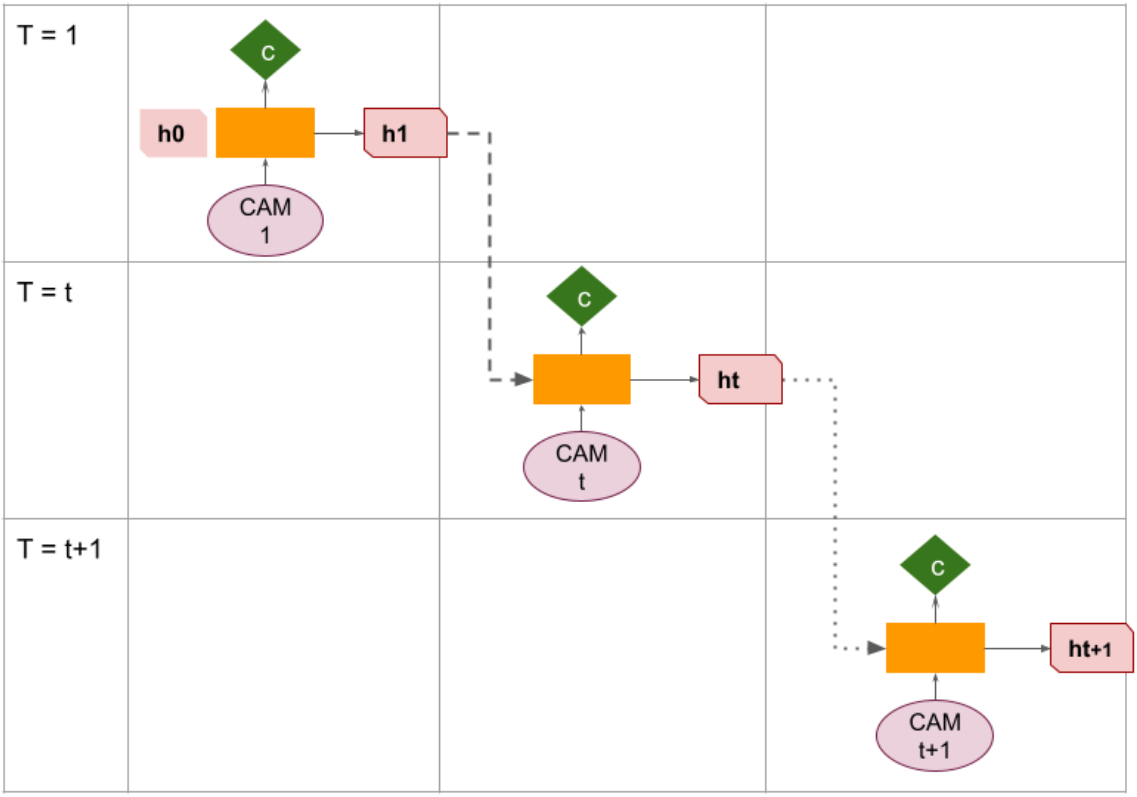


Figure 5.1: Inference process.

The input vector (x_{t_t}) is projected using a FFNN with the aim of mapping it to a higher-dimensional space. The projected input may reveal more complex patterns and relationships that were not visible in the original input space. The projected input ($\widehat{x}_{t_t}$) has the shape (d_{in},), and is calculated as follows:

$$\widehat{x}_{t_t} = W^\top * x_{t_t} + b \tag{5.5.1}$$

W is a learnable weight matrix of shape ($||x_{t_t}||, d_{in}$) where $||x_{t_t}|| = 6$ and b is the bias vector of shape (d_{in},). The input $\widehat{x}_{t_t}$ is fed to an RNN network along with its previous hidden state ($h_{t_{t-1}}$). The RNN model can be either a GRU or LSTM and have m layers ($m \geq 1$). After the feeding process, the model will produce the output y_{t_t} and h_{t_t} the hidden state produced by the RNN cells. The hidden state h_{t_t} generated at time t is used for generating the prediction at $t + 1$.

The decoder's output vector has a shape of (k,), where k is the number of classes, and each element of the output vector $\hat{y}_{t_t}$ represents the probability of the input x_{t_t} belonging to the corresponding class i. The predicted class p_{t_t} is the arg($max(\hat{y}_{t_t})$) which is the index of the maximum value of the vector $\hat{y}_{t_t}$. The architecture of the proposed model is shown in Figure 5.2.

To train our model and minimize the Loss function, we employed SGD as a learning Optimiser. Cross-Entropy is used as a Loss function. To optimize the convergence, we adopted a LR (η) scheduling technique. The scheduling technique adjusts the η during training, η is decayed by

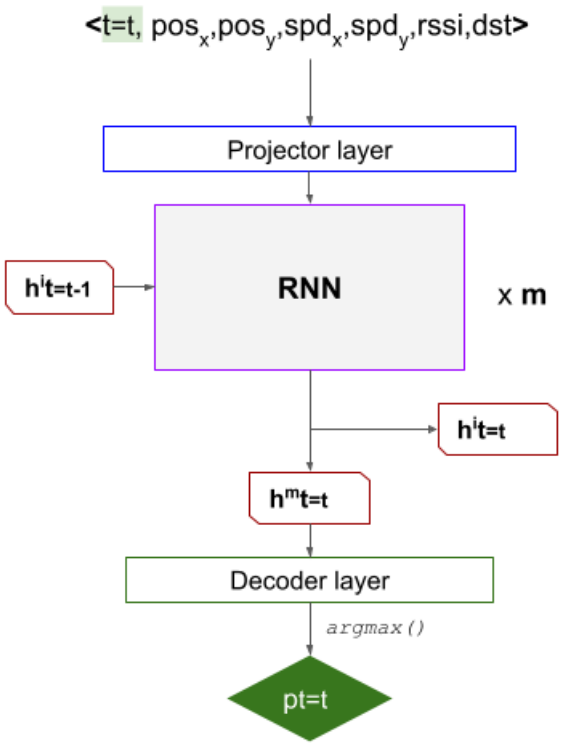


Figure 5.2: RNN(LSTM/GRU) Model Architecture.

a factor of γ every step_size iterations. Furthermore, we opted for mini-batch GD in which the training data is divided into small batches of size (B), and the model parameters are updated after each batch. Since the sequences in the dataset have varying lengths, they are left-padded with zeros ($x_0 = \overrightarrow{0}$) within each batch to match the maximum sequence length in that batch; when calculating the Loss, the padded values are ignored. The explained model training process is repeated for (E) iterations.

5.5.1 Data pre-processing

We used the dataset constructed as described in Subsection 5.4.2. However, to facilitate the training process, we split a subset from the larger original dataset. As 80% of the data represents legitimate behavior, we took steps to address the potential impact of imbalanced data during the training phase. Specifically, we ensured that we sampled data classes with identical distributions to avoid the dominance of one class over the other.

This data (X) is divided into two sets, with 80% of the data allocated for training (X_{train}) and 20% allocated for testing (X_{test}). The total number of sequences and data points are displayed in Table 5.1. Additionally, this table provides this information for each class label. Note that the number of data points corresponds to the sum of sequences lengths.

Table 5.1: Train/Test data distribution

Label	Train		Test	
	sequences	data points	sequences	data points
Benign	11166	197570	2800	39031
Constant	11679	207007	2995	42565
Constant offset	11784	207244	2934	41433
Random	11778	206289	2935	42213
Random offset	11764	206457	2889	41241
Eventual stop	11731	206199	2917	39424
Total	69902	984859	17470	245907

Table 5.2: Training hyper-parameters

parameter		value
DL model	$\|\|x_{t_t}\|\|$	6
	d_in	256
	d_out	256
	m	1
	k	6
LR	η	$1 * 10^{-1}$
	γ	$5 * 10^{-2}$
	step_size	50
Train	Optimiser	SGD
	Loss	Cross-Entropy
	B	32
	E	150

Each $x_{t_t} \in x$, where x is a sequence from the constructed dataset. The *dst* is calculated as the Euclidean distance between $3\_pos_{x,y}$ and $4\_pos_{x,y}$.

Furthermore, based on the VeReMi dataset, the X and Y coordinates fall within the range of (X_{min}, Y_{min}) to (X_{max}, Y_{max}), where $X_{min} = 2300, Y_{min} = 5200, X_{max} = 6400$, and $Y_{max} = 6400$. The positions' coordinates in x_{t_t} are re-scaled to the interval $[0, K], K \in \mathbb{N}$, and this is calculated as follows: $(\text{pos}_x = \frac{3\_pos_x - X_{min}}{X_{max} - X_{min}} * K)$ and $(\text{pos}_y = \frac{3\_pos_y - Y_{min}}{Y_{max} - Y_{min}} * K)$, we set K to 10. The process of re-scaling ensures efficient computations and numerical stability.

5.5.2 Model training

The training hyper-parameters are summarized in Table 5.2. Furthermore, to demonstrate the effectiveness of RNN-based approaches, we have included a MLP model as a baseline. The MLP model takes the data points x_{t_t} as input to perform the classification task. The MLP model comprises five hidden layers with dimensions of 32, 64, 128, 64, and 32, respectively. The remaining training parameters for the MLP model are similar to those used in Table 5.2.

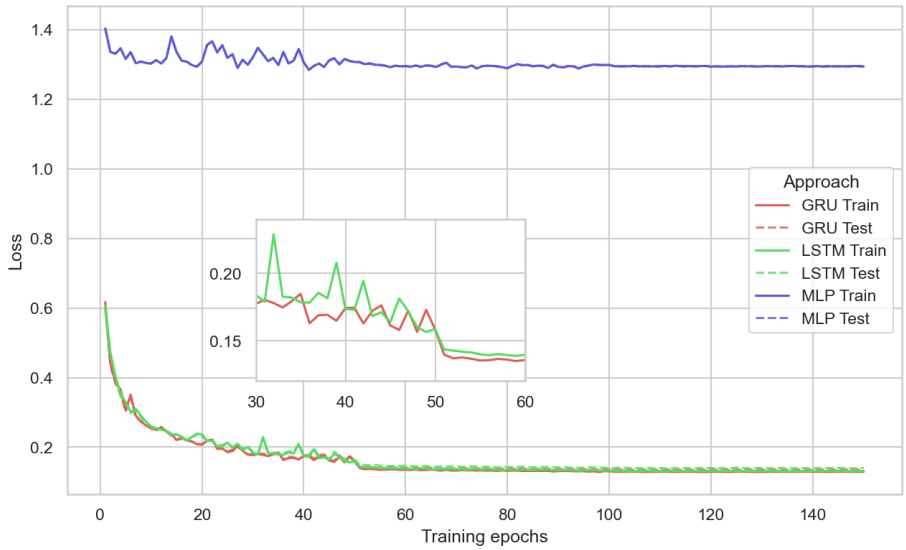


Figure 5.3: Training Loss curve.

5.6 Results

This section presents the performance evaluation of our approach, including an analysis of the impact of sequential data and a discussion of the theoretical complexity[3].

5.6.1 AIDS classification performances

Figure 5.3 displays the train/test loss curves for three models: GRU, LSTM, and MLP. The GRU and LSTM models show a similar convergence pattern, with their training and testing losses decreasing steadily over time without showing any signs of over-fitting. They ultimately achieve good results with a loss value of approximately 0.15. On the other hand, the MLP model appears to have difficulty learning the task, as indicated by its oscillating loss curves and higher loss value (approximately 1.15).

Figure 5.3 also shows the benefits of using LR scheduling. The implementation of LR scheduling helped the model to converge to a better solution, as evidenced by the results obtained at epochs 50 and 100, with the most significant improvement observed at epoch 50. In fact, we can see a decrease in the loss at epoch 50 when the LR is decayed, indicating that the model was able to learn more efficiently with the adjusted η value.

The results of the training history provide valuable insights into the performance of models. Figure 5.4 illustrates the accuracy of the models mentioned above on the test dataset during the training epochs. It can be observed that the GRU and LSTM models converge to produce high accuracy, whereas the MLP model does not. The accuracy metric confirms that the GRU and LSTM models are capable of effectively learning the patterns in the data and producing accurate predictions (approximately 95%), while the MLP model appears to struggle in capturing the

[3]refers to the computational resources (time or space), required by an algorithm.

Table 5.3: GRU, LSTM and MLP classification performances on train data.

Label	GRU							LSTM							MLP						
	TPR	TNR	PPV	NPV	FPR	FNR	F1Score	TPR	TNR	PPV	NPV	FPR	FNR	F1Score	TPR	TNR	PPV	NPV	FPR	FNR	F1Score
Benign	0.89	0.97	0.84	0.98	0.03	0.11	0.86	0.89	0.97	0.84	0.98	0.03	0.11	0.86	0.64	0.89	0.53	0.93	0.11	0.36	0.58
Constant	1.00	1.00	0.99	1.00	0.00	0.00	0.99	1.00	1.00	0.99	1.00	0.00	0.00	0.99	0.67	0.92	0.62	0.93	0.08	0.33	0.64
Constant offset	0.99	0.99	0.97	1.00	0.01	0.01	0.98	0.99	0.99	0.96	1.00	0.01	0.01	0.98	0.64	0.84	0.45	0.92	0.16	0.36	0.53
Random	1.00	1.00	1.00	1.00	0.00	0.00	1.00	1.00	1.00	1.00	1.00	0.00	0.00	1.00	0.94	0.96	0.83	0.99	0.04	0.06	0.88
Random offset	0.94	1.00	0.99	0.99	0.00	0.06	0.97	0.94	1.00	0.99	0.99	0.00	0.06	0.96	0.10	0.97	0.42	0.84	0.03	0.90	0.16
Eventual stop	0.84	0.98	0.88	0.97	0.02	0.16	0.86	0.84	0.98	0.88	0.97	0.02	0.16	0.86	0.22	0.85	0.23	0.84	0.15	0.78	0.22

complexity of the data, achieving a reduced accuracy of only 55%. In addition to the accuracy metric, there are several other metrics that we may consider when evaluating the performance of the classifier DL models. These metrics include TP, TPR, PPV, and the F1Score. These metrics are measured for each class label and are summarized in Table 5.3. A F1Score close to 1 indicates high precision and recall, which means that the model has correctly identified most of the positive instances and correctly rejected most of the negative instances. We notice that the RNN (LSTM and GRU) models have succeeded and performed well for all the class labels, guaranteeing high TPR and F1Score.

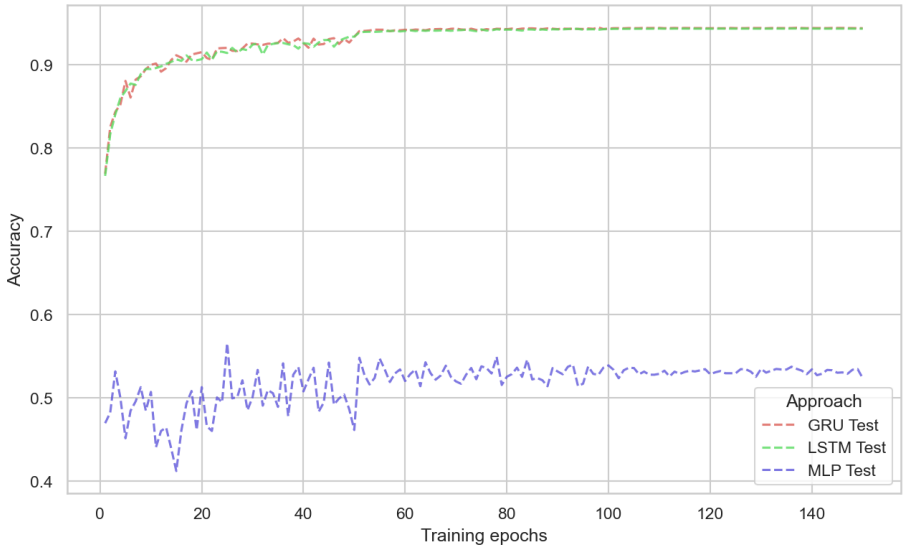
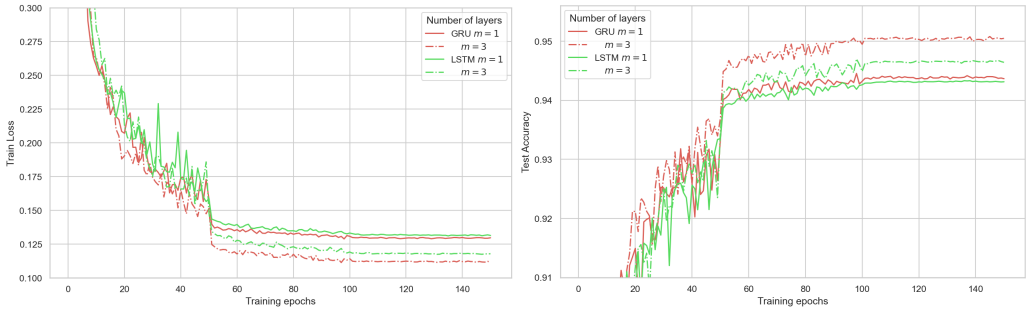


Figure 5.4: Testing Accuracy curve.

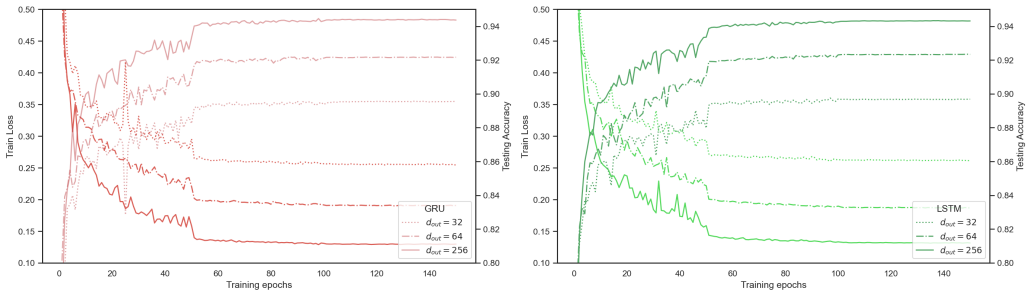
Figure 5.5a displays the Loss curves of GRU and LSTM models with varying the number of layers ($m = \{1, 3\}$). The models with $m = 3$ (dashed lines) minimize the loss more effectively than those with $m = 1$ (solid lines). As the depth of the model increases, the performance improves; the finding aligns with the accuracy curves already demonstrated in Figure 5.5b.



(a) Loss (b) Accuracy

Figure 5.5: Loss and Accuracy, Impact of the number of layers (m) parameter.

Figure 5.6 illustrates the effect of varying dimensions of the hidden state vector. The Loss curves and accuracy of both GRU and LSTM are shown in Figure 5.6b and Figure 5.6a, respectively. The role of the hidden states is to embed historical knowledge, and increasing their size improves the model's ability to remember more information. The larger the models, the better performance.



(a) GRU (b) LSTM

Figure 5.6: Loss and Accuracy, Impact of (d_out) parameter.

5.6.2 Impact of sequential data

To demonstrate the necessity of incorporating historical data in the deployment of consistency-based AIDS, we assess the performance of our model in classifying the t^{th} received CAM. To accomplish this, we utilized our test dataset and calculated the F1Score for each class on the data points received at time t for each CAM. Since the maximum observed sequence length was 100, the value of t varies between 1 and 100.

Figure 5.7 summarizes the obtained results. The lines in the graph depict the average F1Score for each RNN model, while the colored contour indicates the confidence interval of the F1Score, which is calculated as the average F1Score plus or minus the standard deviation[4] of the F1Score for the six class labels. The figure illustrates that at $t = 1$, the classification quality is low, similar

[4]statistical measure that quantifies the amount of variation or dispersion in a set of data points.

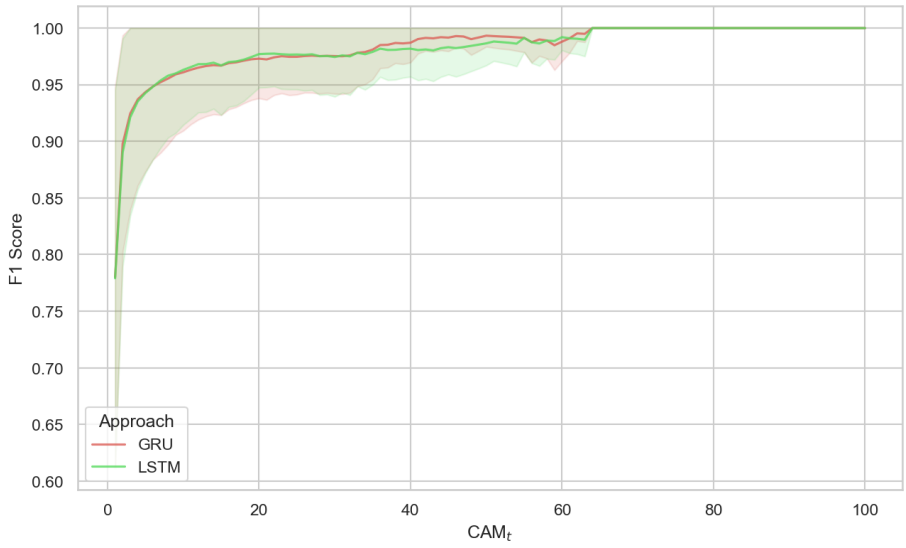


Figure 5.7: F1Score, Classifying t^{th} received CAM.

to the performance obtained by the MLP. This is because both the MLP and RNN models, in this case, perform a data point classification without taking any previous data points into account. As time progresses, the F1Score for both RNN models increases rapidly, and the confidence interval becomes narrower. This is because the RNN models analyze the consistency of the received CAMs over time, resulting in more accurate classification. When $t > 60$, it can be observed that the RNN achieves an F1Score of 1, indicating that the model is able to achieve perfect classification with 100% accuracy and 0% false positive rate. The performance improved linearly over time and reached a perfect model performance after approximately 60.

These findings demonstrate the effectiveness of RNN models in detecting legitimate/malicious CAMs by analyzing the consistency over time.

On the one hand, the study indicates that including longer historical data improves model performance significantly. On the other hand, related works frequently adopt windowing approaches in which the model analyzes only a limited number of messages (windows with a fixed length, W_{size}), and this length is generally short (24, 20 and 20 in [Gro+21],[All+21] and [HCT22] respectively). Nonetheless, we believe that using the windowing approach limits by disregarding historical data (older than W_{size}) that could provide insight for the classification of CAMs.

5.6.3 AIDS time complexity analysis

5G-V2X systems require stringent latency, especially when it comes to cooperative awareness applications. Vehicles receive CAMs at a high frequency and from multiple neighboring vehicles. The transmission capabilities introduced in 5G and beyond networks can meet these requirements. However, deploying security-related mechanisms is generally costly and is not exempt from this. In fact, analyzing CAMs creates additional computation overhead and thus increases the vehicle's end-to-end latency.

To demonstrate the efficiency of the suggested models, analyzing their theoretical time complexity is the most effective method. It offers valuable insights and enables a fair comparison among different approaches.

To determine the precise complexity of the models, we assume that scalar addition, multiplication, as well as calculating the exponential and hyperbolic tangent functions (tanh), are all considered primitive operations belonging to $\mathcal{O}(1)$. Additionally, it should be noted that $d\_in = d\_out = d$ and $m = 1$ (refer to Table 5.2).

$$\text{Complexity}_{\text{model\_gru}} = d[12d + 2x + 2k + 9] \tag{5.6.1}$$

$$\text{Complexity}_{\text{model\_lstm}} = d[16d + 2x + 2k + 13] \tag{5.6.2}$$

The computational complexity of classifying a data point (CAM_t) using GRU and LSTM models can be determined using Equations (5.6.1) and (5.6.2), respectively. The details are provided in Appendix A. Both models belong to $\mathcal{O}(1)$, making them computationally efficient. However, GRU has fewer gates and requires fewer primitive operations ($4d^2+d$) than LSTM, which is advantageous in latency-sensitive domains like V2X. Therefore, we have chosen the GRU model as it can provide lower end-to-end latency due to its simple architecture. Compared to works cited in related works, that rely on sequence classification, where they used the window sliding approach and pass it through a CNN layer, then a RNN network to reconstruct the sequence and finally calculate the reconstruction error, we will calculate the time complexity.

We aim to compare the computational efficiency of our method to approaches cited in related works ([All+21; HCT22]). The approaches rely on sequence classification using a window sliding approach, a CNN layer, and an RNN network to reconstruct the sequence and calculate the reconstruction error. To simplify, we will replace the CNN layer with a concatenation of the projected data points within the window, where the window has a length of W_{size}. Therefore, the computational complexity can be expressed as $W_{size} * \text{Complexity}_{\text{gru/lstm}}$, plus the complexity of calculating the reconstruction error, both of which have a time complexity of $\mathcal{O}(Wsize)$. As a result, the overall time complexity is also $\mathcal{O}(W_{size})$.

It is worth noting that a constant time complexity, $\mathcal{O}(W_{size})$, is considered efficient in most cases. However, our proposed approach offers an even greater efficiency with a time complexity that is W_{size} times less than approaches proposed in [All+21; HCT22], making it suitable for AIDS deployment in the context of 5G-V2X networks.

Conclusion

In this contribution, we introduced a novel data-centric AIDS designed for 5G and beyond vehicular networks. Our model is based on RNNs allowing for consistency verification by gradually integrating the entire historical data sequence for classification. We have demonstrated that longer historical sequence lengths lead to better performance. This is particularly an interesting finding because existing approaches rely on fixed-length sequences, which may limit their performance. Furthermore, our proposed approach has the advantage of low computational cost, as it only requires one RNN cell per received CAM. We have proven this through theoretical time complexity analysis. As a result, we believe that the approach proposed in this chapter represents a significant improvement over existing methods.

This chapter marks the culmination of Part II, summarizing our contributions to HIDSs implemented at the vehicular level in the context of 5G-V2X. Our contributions have encompassed

enhancing both the reliability and integrity aspects of these systems. Moving forward, the subsequent part will delve into Network-based Intrusion Detection Systems (NIDSs) in 5G networks, focusing on safeguarding Application Servers (ASs).

Part III

Network-based Intrusion Detection Systems (NIDSs)

Chapter 6

Early network intrusion detection in 5G networks

Introduction

This chapter discusses our contribution to Network-based Intrusion Detection Systems (NIDSs) in 5G and beyond networks. The contribution aims to protect Application Servers (ASs) at the NS level, which can include vehicular applications or any other services. Our NIDS is deployed at the gateway level (UPF) and inspects malicious traffic.
Furthermore, and more importantly, our contribution addresses the issue of delays in network intrusion detection in the current approaches. Thus, we propose a novel approach capable of early intrusion detection. Our proposed model incorporates DL techniques, utilizing RNNs, along with attention mechanisms. This architecture takes advantage of the sequential nature of packets within Network Flows to enable early attack detection.

The remainder of this chapter is organized as follows, the following section discusses the context, and then the contributions are summarized. Section 6.3 provides a comprehensive review of existing approaches in the field of NIDSs. Section 6.4 presents the proposed approach, including the preparation of data, a detailed description of the proposed Attention RNN model, and an explanation of the training phase. Section 6.5 presents the datasets utilized in the research. The classification performances of the proposed model are also demonstrated. Additionally, this section examines the early detection capabilities of the model.

6.1 Context

In Section 3.3 we introduced approaches of NIDSs, the aim is to analyze incoming and outgoing traffic in real-time, looking for patterns and behaviors that may indicate suspicious activities. NIDSs trigger alerts so that the mitigation component takes the appropriate countermeasures to prevent the intrusion.

The Network Flow starts when the source sends its first packet and has two termination conditions: either a termination signal is sent in the packet, or a timeout is reached [USB17]. Commonly

in research works, the sequence is then aggregated into a single data point that contains statistical information about the packets, such as their arrival time, length, direction, and flags. Flow-based NIDS, which will be referred to in this chapter as conventional flow-based NIDS, perform the classification task only when the Network Flow termination condition is met.

However, a Network Flow can potentially expose some abnormal traffic traces before its termination. Therefore, the flow type can be inferred early, which makes waiting for its termination unnecessary and results in delayed detection. This can further harm the network. In another case, if the last packet of a Network Flow is sent after a certain amount of time, before reaching the timeout and not including a termination signal, the NIDS ignores the termination of the flow and must wait until the timeout is reached to perform the classification. This introduces more delay in taking appropriate action against the malicious user, which gives them more time to carry out further malicious activities.

The scenarios described above underscore a significant challenge that, to the best of our knowledge, remains inadequately addressed in the existing literature. This challenge concerns the efficacy of NIDSs in detecting malicious traffic at the earliest possible stage.
A NIDS must detect suspicious traffic in its nascent form to proactively prevent the persistence of malicious traffic flows in the network. Therefore, we believe that future research efforts should focus on developing preemptive approaches that minimize the time required to detect malicious traffic in the context of flow-based NIDS.

6.2 Contribution Summary

Sequence-based classifiers, such as RNNs and attention mechanisms, can help address the aforementioned challenge. These techniques are particularly advantageous for handling sequential data, as they can process and identify sequential patterns within a sequence. Incorporating RNNs into NIDS can significantly improve flow classification accuracy by preserving the temporal (sequential) dimension.

RNN-based NIDS can better capture temporal dependencies of network packets within a Network Flow, thereby enhancing its ability to detect and respond to malicious behavior. Furthermore, attention mechanisms allow the RNN to selectively focus on parts of the input sequence that are most relevant to sequence classification task [NZY21]. This is done by assigning a weight to each packet header of the input flow, which reflects its importance to the current flow class. By allowing the RNN to focus on the most relevant parts of the input sequence at each time step, attention mechanisms have the potential to improve the performance of RNNs.

In this contribution, we present a novel approach for flow-based NIDS. Our approach utilizes packet headers and incorporates the sequential nature of packets within Network Flows. We propose an attention-based RNN model, which we demonstrate to be highly effective in achieving high classification accuracy. Furthermore, we showcase the remarkable capabilities of our proposed models in addressing the challenge of early attack detection. To the best of our knowledge, our contribution is the first to tackle this aspect in the literature, making it a significant contribution to the field of network security.

6.3 Related Works

Table 6.1: Summary of Related Works

Work	Approach			Time Series	Class.[1]		AI model	Dataset	Detection Time	Cons
	Packet	Flow			Binary	Multiclass				
		Headers	Payload							
[Far+22]	✓		✓	✗		✓	KNN, Adaboost, MLP/CNN	[SLG18; MS15]	packet arrival	- NIDS Computing overhead/Payload encryption - Does not take context into consideration
[Has+22]	✓		✓	✗		✓	KNN			
[BM16]		✓		✗	✓		LogReg/ SVM/ NB/ RF	[Tav+09]	Flow termination	-Waiting for Network Flow termination condition.
[Sar+20]		✓		✗	✓		DT	[23c]		
[IA19]		✓		✗	✓		MLP	[Tav+09]		
[Sho+18]		✓		✗		✓	AE	[Tav+09]		
[AMK17]		✓		✗		✓	MLP	[Tav+09]		
[Yan+23]		✓		✗		✓	CNN	[Tav+09; SLG18]		
[BB23]		✓		✗	✓	✓	RF/ MLP / DT / LogReg	[MS15]		
[Zİ20]		✓		✗	✓		AE/ VAE	[SLG18]		
[Min+21]		✓		✗	✓		AE	[Tav+09; SLG18; MS15]		
[Wu+22]		✓		✗		✓	Transformers	[SLG18; Sha+19]		- Misusing of sequence based models (RNNs/ Transformers)
[Jia+20]		✓		✗		✓	CNN+ BiLSTM	[Tav+09; SLG18]		
[Tan+19b]		✓		✗	✓		GRU	[Tav+09; SLG18]		
[WL21]		✓		✗	✓		Transformers+ CNN	[Sha+19]		
[Kha21]		✓		✗	✓		CNN+ RNN	[SLG18]		

[1] Classification Type

[UM22]		✓		✗	✓	✓	LSTM/ biLSTM/ GRU	[Hin+20; Kor+18; Tav+09]		
[NAS20]		✓		✗		✓	LSTM	[23d]		
[Sun+20]		✓	✓	✓		✓	CNN+ LSTM	[SLG18]		- Payload analysis (encryption) - Detection time not discussed
[Han+23]		✓	✓	✓	✓		Transformers	[SLG18; Shi+12]		
[Tan+19a]	/	/	/	✓	✓		Transformers	[SLG18]	periodic (0.5s)	- Can not identify the attacker/ and the number of attacks in the network

The problem of NIDS has been a topic of great interest to researchers, and its development is still ongoing. Researchers constantly strive to develop new techniques and architectures to address the challenges and issues related to NIDSs.

Some NIDS treat each network packet as a separate data point that must be analyzed and classified. This analysis can be based on the packet's header or payload. Many existing NIDS methods use signatures, which typically involve testing whether the packet matches a header, port, or payload attack condition using regular expressions, please refer to [Kum] for details about signatures-based NIDS. Recent works use advanced techniques, especially DL/ML based approaches. [Far+22] experimented ML for a packet classification task, the authors have proposed a tool that parses packet payloads into a fixed-size byte vector. The resulting payload vector is then labeled and tested using several ML algorithms, including RF, KNN, Adaboost, MLP, and CNN. In the same context, [Has+22] proposed a novel approach for packet classification inspired by techniques used in natural language processing. The approach utilizes an embedding technique that learns a vector representation of the payload. A NN is trained to learn byte embedding from the surrounding bytes in the same payload. The packet payload bytes are then aggregated to obtain a payload embedding for the packet, which is passed to a KNN model for final packet classification. However, packet-based approaches have several drawbacks. Firstly, payload encryption makes it difficult to perform effective analysis. Secondly, analyzing the payload takes considerable time. Moreover, these methods do not take any contextual information into account, which means they may fail to detect abnormal traffic that consists of a set of packets where each packet separately is benign, but the whole traffic is malicious.

To overcome these limitations, the new NIDSs are based on Network Flows. A Network Flow is a continuous packet stream representing a communication session between a source and a destination. A Network Flow begins when the source sends its first packet and terminates when one of two conditions is met: either a termination signal is sent in the packet, such as the FIN signal in TCP, or a timeout is reached. Usually, the timeout period is set to 60 seconds [23a].

Most flow-based NIDSs utilize packet headers to extract relevant information. The data extracted from packet headers is then consolidated into a single data point containing statistical information about the packets. This statistical information includes the packet's arrival time, length, direction, and flags. The aggregated vector may contain over 100 features. In the context section

(Section 6.1), we referred to this type of NIDS as conventional NIDS.

In the literature, a variety of models and classifiers have been developed for flow based intrusion detection, and recent research has focused on utilizing advanced ML/DL techniques. ML techniques are employed in various ways, including supervised learning. In binary-class supervised NIDS, the classifier is trained to learn and predict whether the flow is benign or malicious [BM16; Sar+20; IA19]. In contrast, in multi-class supervised NIDS, the classifier is trained to predict the type of attack [Sho+18; AMK17; Yan+23; BB23]. Another approach is semi-supervised learning, where the ML/DL model is trained only on benign traffic and is then used to determine whether a given flow belongs to the benign class. Any flow that does not fit this class is considered as an attack [Zİ20; Min+21].

Conventional flow-based NIDSs rely on flow termination conditions. As stated in the introduction, it is important to note that these approaches can result in increased delays. Moreover, when aggregating the Network Flow (sequence of packets) into a single data point, there is a loss of the temporal dimension inherent in the sequence. This loss of temporal information can impact the performances of the model [Sun+20; Han+23].

It is worth noting that certain sequence-based DL models like RNNs and transformers[2] have been misused in some flow-based NIDS works. Approaches discussed in [Wu+22; Jia+20; Tan+19b; WL21; Kha21; UM22; NAS20] treat the flow aggregated vector as a sequence and feed it into a sequence-based DL model. We strongly believe that these approaches are conceptually flawed and should be reconsidered.

Recent proposals for flow-based NIDS have suggested leveraging time-series data instead of flow aggregation to enhance performance. The authors of [Sun+20] proposed a model that processes flow packets, including both the header and payload, in a 2D format consisting of a sequence of packets and their corresponding features. The model then passes this input to a CNN layer, with the resulting output being fed into a LSTM layer for flow classification. Additionally, [Han+23] proposed a transformer-based model for flow classification, which considers both the header and payload of a packet. The payload is a vector containing the frequency counts of its bytes. The sequence of packets in the flow is fed to a transformer model for classification. [Sun+20; Han+23] demonstrated the effectiveness of using sequence-based deep learning models, with [Han+23] in particular highlighting the advantages of using transformers, which rely on attention mechanisms to improve performance. However, both approaches rely on packet payload, making them prone to computational overhead and payload encryption issues.

[Tan+19a] attempted to address the detection time issue and highlighted that relying on termination timeouts for flows can cause delays in the detection process. The authors have proposed a framework that continuously monitors the state of the network to determine if it is experiencing any anomalies. The proposed approach periodically checks various network parameters to identify potential deviations from expected network behavior. The authors have considered 55 features, including statistics related to the number and length of packets, as well as the active flows in the network. The NIDS performs inference every 0.5 seconds to detect anomalies in the behaviour. The framework leverages a transformer model, which performs binary classification to determine whether the current network behavior is normal or not. The model considers the network's previous states during the last 5 seconds. Through comparison with other models, the authors have demonstrated the effectiveness of their transformer-based approach. The proposed framework is effective in detecting the presence of an attack in the network, but it does not provide information about the attacker's identity or attack type. Additionally, the approach cannot determine whether

[2] type of NN architecture that uses attention mechanisms.

there is only one attack or multiple attacks being performed, nor can it identify whether there is a single attacker or multiple attackers involved. These limitations can make it challenging for the mitigation module to take the appropriate actions.

Table 6.1 presents a comprehensive overview of the works discussed earlier. It encompasses key information such as the ML/DL models utilized, classification type (binary or multi-class), and the datasets employed. Furthermore, the table provides a concise summary of the limitations associated with each of the works.

To overcome the aforementioned limitations, we introduce a flow-based NIDS that focuses on packet headers and considers the sequential nature of Network Flows. Our approach capitalizes on the benefits offered by attention mechanisms, showcasing their effectiveness in early flow classification.

6.4 Methodology

In this section, we provide an overview of our proposed flow-based NIDS. We delve into the various components and steps involved in its implementation. We cover the data processing phase, the utilization of RNNs and attention mechanisms, as well as the DL-model training process.

6.4.1 Transformation of the network traffic data to network flows

Network security and monitoring data is collected and stored in the form of *pcap* files. Packet CAPture (PCAP) represents a file format used to store network traffic data that has been captured by a network sniffer tool. These *pcap* files serve as repositories for detailed information about individual network packets, including their headers and payloads. The *pcap* files are processed by various tools to extract aggregated flows and convert them into CSV format. Examples of such tools include Argus[3], NetFlow[4], CiCFlowmeter[5]. These tools are commonly employed in different open-source datasets for network analysis. However, the aforementioned tools primarily focus on providing aggregated flow-level information from *pcap* files. Although these tools are widely used in the literature for flow-based NIDSs, they are not suitable for NIDSs that rely on packet sequentiality within flows.

Given the importance of processing packet-level sequentiality for flow-based NIDS, we have created a Python-based tool called *py_flows*. This tool segments the fully captured raw network traffic files (provided in many open-source datasets), into individual network traffic files. Each segmented file contains the packet sequences associated with their respective flow.

A flow session is uniquely identified by the source IP address and port, destination IP address and port, and the protocol used. Traditionally, when flows surpass the timeout period, they are subdivided into multiple sub-flows. However, in our work, we take a different approach by preserving the entire session flow without splitting it.

The flow labeling is conducted in a semi-manual manner. Open-source network intrusion datasets provide metadata on how the datasets were generated. These metadata include attack information such as the attacker's IP address, the victim's IP address, the type of attack, and the

[3] openargus.org
[4] cisco.com
[5] github.com/ahlashkari/CICFlowMeter

time when the attack was initiated. The dataset's metadata are utilized for labeling the flow files generated by the *py_flows* tool. These labels serve the purpose of identifying, whether a flow is tagged as benign or corresponds to a specific type of attack. The dataset containing the labeled flows can be used to train NIDS classifiers.

6.4.2 Data preparation and pre-processing

The data set generated from the previous step consists of a collection of labeled flows in pcap format. In our approach, we used ScaPy[6] library to read the packets within each flow and extract the header data. The extracted features are:

- Packet relative time: represents the arrival time of a packet (x_{t_t}) relative to the first packet (x_{t_1}) in the flow. This feature is expressed in seconds and calculated as time(x_{t_t})−time(x_{t_1}),
- Inter arrival time, refers to the time interval between the arrival of a packet (x_{t_t}) and its preceding packet ($x_{t_{t-1}}$) in the flow sequence,
- Packet direction, can be categorized as either forward or backward. Forward (fwd) packets are the ones sent by the flow initiator, determined by the source of the first packet in the sequence. Conversely, backward (bwd) packets refer to the ones sent by the destination,
- Destination port: the flow destination port,
- Packet length,
- Packet payload length,
- Time-To-Live (TTL), a field in the IP header specifing the number of routers that a packet can pass through before it is discarded,
- Protocol: IP protocol, TCP or User Datagram Protocol (UDP),
- TCP flags, which are control bits found in the TCP packet header. There are 8 flags (PSH,SYN,RST,FIN,ACK,URG,ECE,CWR) and please refer to [Edd22] for details on TCP/IP networking.

After extracting features from *pcap* files, they are saved in a *csv* file, with each file representing a labeled sequence of the extracted features.

To prepare the data for the model training, we performed encoding and normalization techniques on the features. We used the Z score function to normalize the packet length and packet payload length. Additionally, since the TTL field is an integer ranging from 0 to 255, we normalized it using the min-max function to scale it within the range of 0 to 1. Regarding the destination port, which is a categorical type field, we decided to retain only the commonly used destination ports found in the dataset, such as 22, 80, 8080, and 5353. It is important to note that these ports may vary depending on the testbed used to generate the data and the servers/services deployed. After identifying the common ports, we proceeded to encode them. As for the remaining ports that were not part of the common set, we assigned them a default code. Due to the variable lengths of packets

[6] scapy.net

in flow sequences, we adopted a simplification approach by truncating the sequences to a maximum length, denoted as MAX LENGTH (= 128).

At this stage, the data is prepared as input to the NIDS model. The dataset consists of a collection of Network Flows, where each flow is represented as a tuple $< x, label >$. In this representation, $x = \{x_{t_t}\}/t \in [1, L]$ is the sequence of packet headers belonging to the flow, L is the length of the flow (number of packets) $L = ||P||/L \in [1, \text{MAX LENGTH}]$. A packet header, denoted as x_{t_t}, has the following structure:

$$x_{t_t} :<\text{rt}, \text{iat}, \text{fwd}, \text{bwd}, \text{d\_port}, \text{len\_p}, \text{len\_payload}, \text{ttl}, \\ \text{proto}, f_p, f_s, f_r, f_f, f_a, f_u, f_e, f_c >$$

(rt), (iat), (len_p), (len_payload), (ttl) represents the packet's relative time, inter arrival time, packet length, payload length and the time-to-live, respectively. fwd and bwd are boolean variables that represent the packet's direction, proto represents the type of IP protocol (TCP or UDP), The flags $f_p, f_s, f_r, f_f, f_a, f_u, f_e$ and f_c are boolean variables that indicate control bits of the TCP packet header. If the packet is a UDP packet, then these flags are set to false.

6.4.3 Deep Learning Model

This subsection provides a description of our DL model, which consists of three key modules: a projection layer, an encoder module using an RNN with attention mechanism, and a decoder layer. The model's architecture overview is illustrated in Figure 6.1.

Projection Layer

The model operates by accepting a sequence x as its input. To start, each individual input $x_{t_t} \in x$ is projected through a FFNN layer.

The FFNN layer is followed by a dropout() function, serving as a regularization technique that aim to prevent over-fitting. The dropout layer randomly sets a fraction of the input tensor elements to zero, during training with a specified dropout probability, denoted as p_{dropout}.

The primary objective of this projection is to map the inputs to a higher-dimensional space, enabling richer representation and capturing more complex patterns. The projected input ($\widehat{x}_{t_t}$) has the shape $(d,)$, and is calculated as follows:

$$\widehat{x}_{t_t} = \text{dropout}(W^{\top}_{\text{projector}} * x_{t_t} + b) \tag{6.4.1}$$

$W_{\text{projector}}$ is a learnable weight matrix of shape $(||x_{t_t}||, d)$ and $b_{\text{projector}}$ is the bias vector of shape $(d,)$.

AttnRNN Encoder

The purpose of this block is to process the projected packets within the flow sequence and encode it in the form of a latent representation, extracting meaningful and important information from the flow sequence. This representation will then be used by the decoder layer to classify the input sequence effectively. Our encoder building block consists of RNN layers and an attention module, which we detail in the subsequent paragraphs.

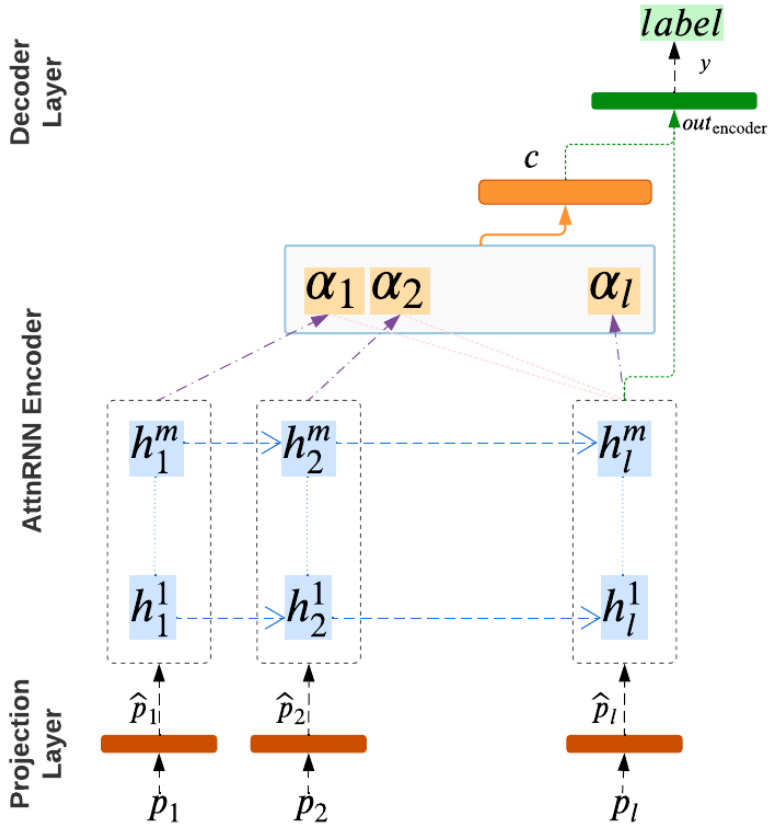


Figure 6.1: AttnRNN Model Architecture.

RNN block The RNN component, depicted in blue in Figure 6.1, maintains an internal hidden state that enables it to retain and use information from previous data points within a sequence.

The RNN cell accepts as input a projected input $\widehat{x}_{t_t}$ with shape $(d,)$. The formula for calculating the hidden state h_{t_t} is as follows:

$$h_{t_t} = tanh(W^\top * \widehat{x}_{t_t} + b_x + V^\top * h_{t_{t-1}} + b_h) \tag{6.4.2}$$

Where W and V are learnable weight matrices of shapes (d, d). b_x and b_h are the bias vectors of shape $(d,)$. Please note that in this contribution, we set $d_{in} = d_{out} = d$ for the RNN cell shapes.

In this contribution, we will utilize LSTM and GRU, both LSTM and GRU rely on the concept of hidden states, they also incorporate additional gates and internal states to facilitate their functioning. Please refer to Section 2.3.2 for further details on these LSTM and GRU.

Attention module Attention mechanisms are designed to improve the performance of neural networks when processing sequential data. When applied to a DL-based NIDS model, attention mechanisms enable the DL model to assign weights to the packets in the flow sequence, which helps to selectively focus on the most important and relevant packets during the classification process.

The key idea behind attention is to create a context vector (c) and provides it as an additional input to the decoder. This context vector is calculated by combining the hidden states of the encoder h^m, which represent the input flow sequence, with attention weights that determine the importance of each hidden state h_i^m. The context vector is calculated as follows:

$$c = (\alpha * h^m)^\top \tag{6.4.3}$$

α is the vector of the attention weights, it has the shape of (1,L) where L is the flow sequence length, and its calculation is shown in Equation 6.4.4; h^m is the hidden states of the RNN's last layer (m), $h^m = \{h_{t_t}^m / t \in [1, L]\}$. The context vector c has shape of (d,).

$$\alpha = \{\text{softmax}(\alpha_{t_t}) / t \in [1, L]\} \tag{6.4.4}$$

α_{t_t} is the attention weight of the input x_{t_t}, it quantifies its importance and the relevance to predicted flow. There are several methods for calculating attention weights in the context of attention mechanisms. the commonly used methods include content base attention, additive attention, location attention, general attention, dot product attention, and scaled dot product attention, refer to [Wen18] for further details.

Our DL model utilizes additive attention, a technique that was first introduced in [BCB16]. This approach is robust in handling sequences of varying lengths and provides interpretable weight scores. The attention weights are parameterized by a FFNN layer, which is jointly trained with the other components of the model. The attention weight α_{t_t} of the packet x_{t_t} is calculated as follows:

$$\alpha_{t_t} = v_{\text{attn}}^\top * tanh(W_{\text{attn}}[h_{t_L}^m; h_{t_t}^m]^\top) \tag{6.4.5}$$

Where v_{attn} and W_{attn} are learnable weight matrices of shapes $(d, 2d)$ and $(d,)$, respectively.

Decoder Layer

The last stage of the classification process involves decoding the encoded representation of the flow and inferring the flow's class. The output of the encoder building block, denoted as out_{encoder}, is passed through a FFNN layer, the final output vector $\hat{y}$ is computed as follows:

$$\hat{y} = U^\top * out_{\text{encoder}} + b_y \tag{6.4.6}$$

Where U is a learnable weight matrix and b_y is the bias vector.

In the case where the encoder does not include the attention module, out_{encoder} corresponds to the last hidden state h_L^m of the RNN. On the other case, if the encoder does have an attention module, then out_{encoder} is obtained by concatenating the RNN's output and the context vector (c), resulting in $out_{\text{encoder}} = [h_L^m, k]$. Consequently, the shape of U is either (d, k) or $(2d, k)$, depending on the specific configuration of the model, where k represents the number of classes.

The output vector y has a shape of (k,). Each element $\hat{y}_z$ in the output vector corresponds to the probability of the input flow belonging to the corresponding class z. The predicted class (predicted_label), is determined by the NIDS based on the index (z) that corresponds to the highest probability value in the vector $\hat{y}$.

6.4.4 DL Model Training

To train our NN model, we used the backpropagation algorithm to compute gradients and update the model's learnable parameters θ (including $W_{\text{projector}}$, $b_{\text{projector}}$, W^j, V^j, v_{attn}, W_{attn}, U, and b_y). We employed SGD as our optimization algorithm. Cross-Entropy is used as a Loss function. Additionally, to address the issue of gradient explosion that can occur in RNNs, we used the gradient clipping technique. Gradient clipping promotes more stable updates and aids in preventing the occurrence of infinity values in the gradients (pytorch NaN[7]).

Given that the sequences in the dataset have different lengths, we apply right-padding with zeros ($x_0 = \overrightarrow{0}$) within each batch to align them with the maximum sequence length present in that batch.

When computing the attention context vector, we apply a masking technique to ignore the padded values (x_0) in the flow sequence. As a result, these padded points do not contribute to the calculation of attention weights for the context vector. Similarly, during the calculation of the Loss, the padded values are also disregarded and not taken into account. The explained model training process is repeated for (E) iterations.

6.5 Performances Analysis

This section explains the datasets used in our approach, the DL training environment, performance metrics such as Accuracy, and discusses the early detection capabilities of our attnRNN NIDS.

Table 6.2: CIC-IDS2017 dataset overview

Label	N. of packets	N. of flows	
Benign	5628248	248587	56.71%
Portscan	217530	107692	24.57%
DDoSLOIT	1265657	45168	10.30%
DoSHulk	2137508	14108	3.22%
DoSGoldenEye	106177	7574	1.73%
DoSSlowhttptest	37924	4212	0.96%
FTP-Patator	110256	3958	0.90%
DoSslowloris	45510	3835	0.87%
SSH-Patator	136073	2464	0.56%
Botnet	9862	735	0.17%

6.5.1 Datasets

In order to showcase the strength and effectiveness of our proposed approach, we selected two modern and up-to date intrusion detection datasets that provide full network packet capture records and the corresponding labels. These datasets are CIC-IDS2017 [SLG18] and 5G-NIDD [Sam+22]. For both datasets, we followed the process outlined in sub-section 6.4.1 to extract the flow sequences

[7] stands for Not a Number, a special value that represents undefined numbers

Table 6.3: 5G-NIDD dataset overview

Label	N. of packets	N. of flows	
BENIGN	624154	75625	40.51%
Goldeneye	900385	27467	14.71%
TCPConnect	20341	20032	10.73%
UDPScan	15921	15890	8.51%
Torshammer	555319	15837	8.48%
SYNscan	10064	10014	5.36%
SYNScan	10054	10009	5.36%
SYNflood	26458	7566	4.05%
Slowloris	61916	4247	2.27%

from the packet capture records. The labeling phase varies depending on the nature of the available metadata. To prepare the labeled flow sequences for model input, we proceeded to pre-process them as outlined in sub-section 6.4.2. Tables 6.2 and 6.3 present the labels, as well as the number of packets in the original dataset and the extracted flows for datasets CIC-IDS2017 and 5G-NIDD, respectively. Each dataset (X) is divided into two sets, with 80% of the data allocated for training (X_{train}) and 20% allocated for testing (X_{test}).

CIC-IDS2017

The CIC-IDS2017 dataset was proposed by the Canadian Institute of Cybersecurity and has gained significant popularity in the literature. It has been referenced over 1000 times and is widely regarded as a reference dataset in the field of network intrusion detection. The CIC-IDS2017 dataset encompasses both benign network traffic and various types of attacks, such as Bot, DDoS, DoS, Patator, PortScan, Web Attack, Heartbleed, and Infiltration. Due to the limited representation of certain classes in the dataset, some of these classes were excluded from this contribution. The website of the dataset provides the necessary metadata for labeling, which includes information such as the attack lunch time and IP addresses of the attackers.

5G-NIDD

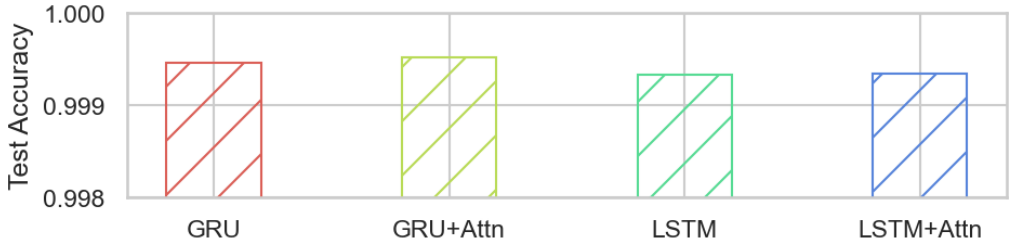
The 5G Network Intrusion Dataset (5G-NIDD) is a recently developed dataset designed for intrusion detection in 5G networks. This dataset was generated using the 5G Test Network at the University of Oulu in Finland. The data in this dataset is collected from the gateways deployed in two base stations. It consists of various types of network traffic, including benign traffic and different types of attacks: ICMP Flood, UDP Flood, SYN Flood, HTTP Flood, Slowrate DoS, SYN Scan, TCP Connect Scan, and UDP Scan. The dataset is organized in such a way that each attack is represented by a separate (*.pcap*) file. Additionally, the IP address of the attacker is (10.41.150.68). This metadata was utilized to accurately label the extracted flow sequences within the dataset.

6.5.2 Model Training Environment

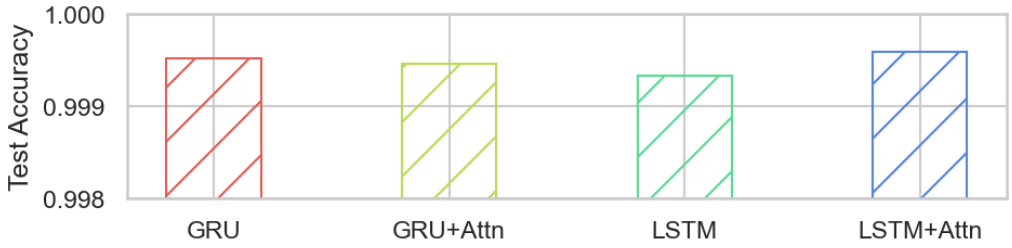
The training hyper-parameters are summarized in Table 6.4.

Table 6.4: Training hyper-parameters

parameter		value
DL model	d	128
	m	1
	k	**10** CIC-IDS2017/ **9** 5G-NIDD
	p_{dropout}	0.2
LR	η	$1 * 10^{-3}$
Train	Optimiser	SGD
	Loss	Cross-Entropy
	B	32
	E	10



(a) CIC-IDS2017 Dataset



(b) 5G-NIDD Dataset

Figure 6.2: NIDS Classification accuracy on test data

6.5.3 Classification Accuracy

Figure 6.2a presents the test accuracy of various experimented models, including LSTM, LSTM with Attention, GRU, and GRU with Attention, on the CIC-IDS2017 dataset. The corresponding results for the 5G-NIDD dataset can be found in Figure 6.2b. It is evident from both datasets that the models achieved remarkably high accuracy rates (99%), which aligns with previous findings in the literature. Furthermore, the inclusion of attention mechanisms in the models resulted in slight performance enhancements. The utilization of attention context vector aided the learning process, contributing to these improvements. The GRU with Attention model demonstrated superior performance on the CIC-IDS2017 dataset, whereas the LSTM with Attention model outperformed others

on the 5G-NIDD dataset. Therefore, we will select these respective models to conduct further evaluations on each dataset.

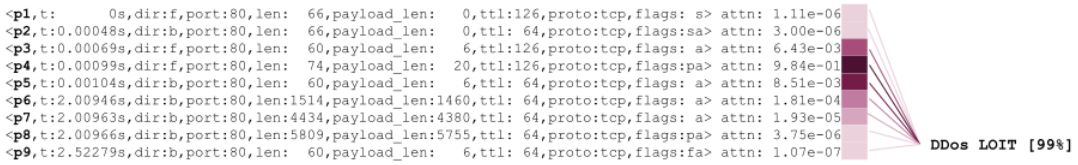


Figure 6.3: Visualizing a Network Flow, its predicted class, and the attention weights.

6.5.4 Visualizing Attention

Figure 6.3 illustrates a flow sequence extracted from the CIC-IDS2017 dataset. This particular flow is a TCP flow, with a destination port of 80. It consists of nine (9) packets, and each packet within the flow is represented by its header. The total duration of the flow is 2.52 seconds. Notably, the LSTM attention model accurately classified this flow, assigning it the label DDoSLOIT. The figure also depicts the attention weights (α). These weights are used to calculate the context vector (c) as shown in Equation 6.4.3.

The context vector (c) is then concatenated with the LSTM output of the last packet (h_{t_9}). This concatenated representation is subsequently fed into the decoder for the classification task.

We observe that the forth and fifth packet (x_{t_4}, x_{t_5}) possesses the highest attention weights among all the packets in the flow. This particular observation suggests that x_{t_4} and x_{t_5} significantly influenced the decoder's classification decision, leading to the assignment of the DDoSLOIT label. This finding suggests that the presence of (x_{t_4}, x_{t_5}) in the flow may contain crucial information related to the underlying attack. Additionally, the high attention weights observed in the flow can be interpreted as an indication of the presence of hidden traces of the attack within in these specific packets.

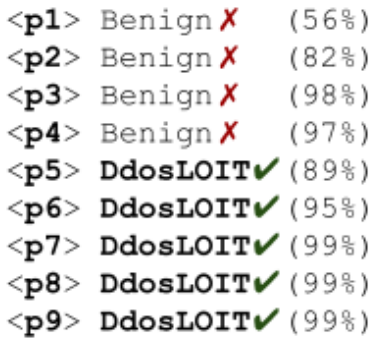


Figure 6.4: Step-by-step predicted class of a Network Flow

To further explore the notion that some packets in the flow contains relevant traces of the

produced labels, we employ a step-by-step classification. For each packet in the flow, denoted as x_{t_t} where $t \leq L$, we feed the hidden state (h_{t_t}) and the context vector (c) calculated using the attention weights (α_{t_j}) up to the t-th packet to the decoder $\hat{y}$. This step-by-step classification allows us to observe how the encoder progressively represents the flow, and see how the decoder decodes this representation.

In this specific case, the observed flow analysis reveals interesting behavior. At the beginning of the flow, the NIDS initially classified the flow as Benign. The decoder indicated that the traffic appeared to be normal with a high level of confidence. This initial classification is expected since the first three packets contain the TCP handshake exchange, making it difficult to infer any attack solely from these packets. Therefore, the benign classification is reasonable given the limited information available in the initial packets. However, a shift occurred when the 5th packet (x_{t_5}) was observed. At this point, the NIDS reevaluated the flow and reclassified it as DDoS LOIT. Initially, the classification confidence was 89%, indicating a fair level of certainty. As the analysis progressed with each subsequent packet, the NIDS became increasingly confident in its classification, reaching a high confidence level of 99%.

These observations are consistent with our previous analysis of attention weights. When examining the attention weights across all the packets, we found that the 4th and 5th packets has the highest attention weight, indicating its significance in determining the label DDoSLOIT. The DL model successfully generated this label by decoding the information utilizing h_{t_5} and the context vector c obtained through the attention mechanism, which incorporates the attention weights α_{t_j} for $j \leq 5$.

Our model successfully detected the attack immediately following the occurrence of packet (x_{t_5}). Early detection of the attack can be achieved if an alert is triggered after (x_{t_5}). As a result, the remaining four packets (from packet x_{t_6} to x_{t_9}) of the flow would not take place. Additionally, the analyzed flow represents a TCP connection that is closed with a FIN flag. Conventional flow-based NIDS can only analyze this flow once it has been terminated, as indicated by the event (x_{t_9}) approximately 2.52 seconds after the initiation of the flow. In contrast, our approach enables detection to be carried out much earlier, precisely right after (x_{t_5}), at approximately 0.001 seconds.

6.5.5 Early detection capabilities

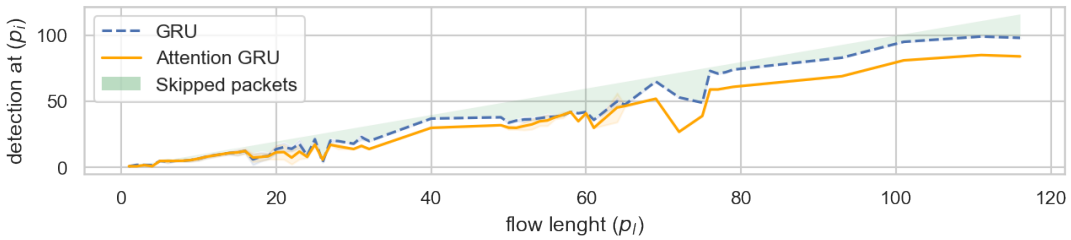
The previous sub-section highlighted the early flow detection capabilities in comparison to conventional NIDS from two perspectives: the requirement of fewer packets for attack detection and the ability to classify flows at an earlier stage. In this sub-section, we will generalize these observations based on the experimental datasets.

Packets required for attack classification

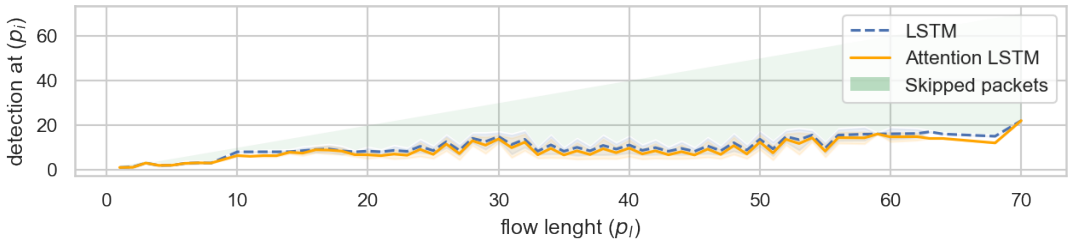
Figure 6.5 provides an overview of the performance concerning sufficient initial successive packets required for correct attack flow classification.

The x-axis represents the flow length, which corresponds to the number of packets in each flow. Each element on the x-axis groups flows of the same length together. On the other hand, the y-axis represents the number of initial packets required for the NIDS to detect an attack.

Conventional flow based ids analyses the flow using all its packets, hence its graph is represented by the diagonal line. The dashed lines represents the RNN model while the continuous line represents the RNN model with attention mechanism. The RNN model is GRU for CIC-IDS2017

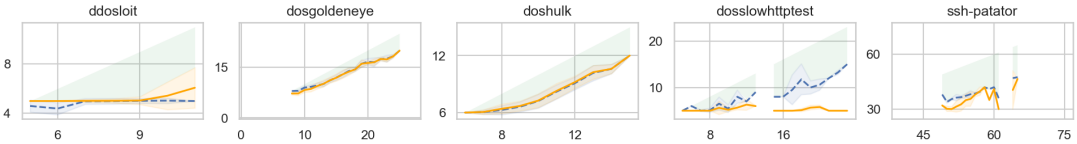


(a) CIC-IDS2017

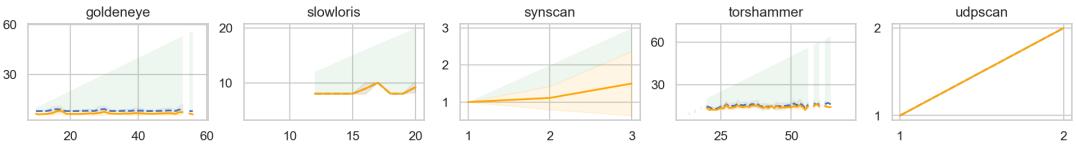


(b) 5G-NIDD

Figure 6.5: Number of initial packets required for correct attack classification. AttnRNN (orange line), RNN (blue line), Conventional flow-based NIDS (green line)



(a) CIC-IDS2017

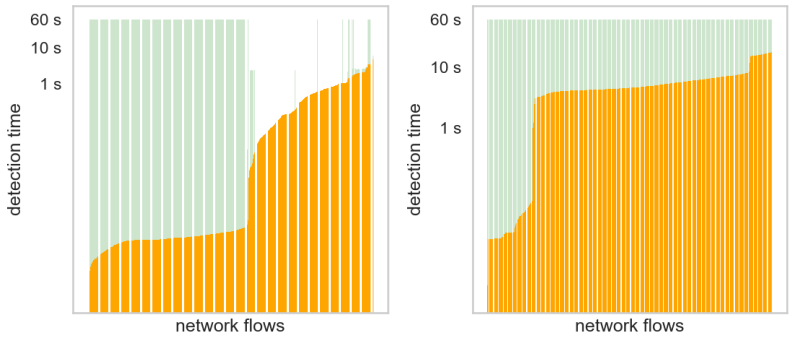


(b) 5G-NIDD

Figure 6.6: Number of initial packets required for correct attack classification of a selected attack classes. AttnRNN (orange line), RNN (blue line), Conventional flow-based NIDS (green line)

(displayed in 6.5a), and LSTM for 5G-NIDD (displayed in 6.5b), these were the selected models based on the accuracy metric as shown in figure1.

In both scenarios, both the RNN and AttnRNN models demonstrated the capability to achieve early classification. The plots representing their respective results clearly illustrate data points positioned below the diagonal line, indicating their advancements compared to conventional flow-based IDS. Moreover, the Attn RNN model demonstrated a slight improvement over the RNN



(a) CIC-IDS 2017 (b) 5G-NIDD

Figure 6.7: Overall Attack Detection Time. AttnRNN (orange bars), Conventional flow-based NIDS (green bars)

model, highlighting the effectiveness of attention mechanisms in enhancing performance.

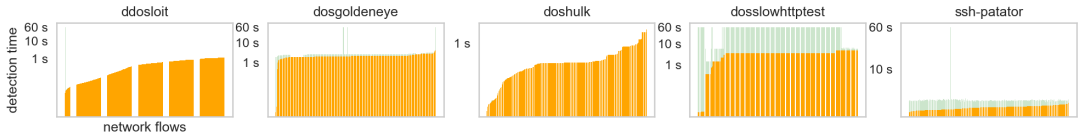
In figure 6.5, the green-colored area represents the remaining skipped packets, which signifies the number of packets that were ignored in consequence of early attack detection. The green-colored packets represent malicious traffic that can be prevented from occurring within the network. The green area appears narrower in 6.5a compared to 6.5b. This observation indicates that in the 5G-NIDD dataset, attacks were more readily classified at an early stage where the attack patterns were relatively easier to identify.

Figure 6.6 highlights attack labels, the sub-figures provide further evidence that the proposed models (RNN and AttnRNN) can enable early detection; certain attacks are detected earlier than others, this observation suggests that the characteristics of the attack and its corresponding traces vary depending on the specific type of attack. One interesting attack label worth noting is the udpscan. Our models demonstrate similar performance to the conventional model in detecting this type of attack. The udpscan attack, characterized by only two packets, is passive in nature. Interestingly, proposed models cannot infer the attack based on the first packet, possibly due to its resemblance to normal traffic behavior. Consequently, both our RNN and attnRNN models require the two packets of the flow to identify the udpscan attack.

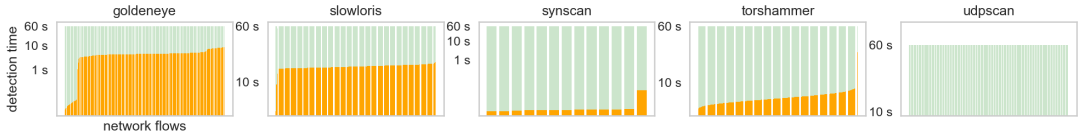
Detection time

Figure 6.7 illustrates the detection time, which refers to the moment at which the model identifies a flow as an attack. It is important to note that the model inference time has not been taken into account in this representation. The x-axis represents flows as data points, while the bars represent the duration of each flow. The green bars depict the detection time of the conventional flow-based IDS, while the orange bars represent the attnRNN model. Sub-Figure 6.7a showcases the performance results on the CIC-IDS2017 dataset, and Sub-Figure 6.7b displays the performance on the 5G-NIDD dataset.

The conventional flow-based NIDS performs detection once the flow is terminated, which occurs under two conditions: either when the FIN flag is sent or when the timeout period (60s) is reached. In both datasets (represented by sub-figures 6.7a and 6.7b), AttnRNN demonstrates a reduced



(a) CIC-IDS2017



(b) 5G-NIDD

Figure 6.8: Attack Detection Time for a set of Labels. AttnRNN (orange bars), Conventional flow-based NIDS (green bars) of a selected attack classes

detection time. This figure highlights the effectiveness of AttnRNN in minimizing the needed time (in seconds) to classify an attack flow. Minimizing the attack detection time can effectively limit the duration of an attacker's presence within the network.

Figure 6.8 presents the detection results for specific attack labels from both datasets. Although AttnRNN successfully reduces the number of packets in the doshulk (shown in sub-figure 6.7a) attack compared to conventional NIDS, both approaches show similar detection times. This is primarily due to the extremely short flow duration (around 1 second) and the small inter-arrival time between packets within the flow.

In the case of a UDP scan (shown in sub-figure 6.7b), conventional NIDS systems wait for the timeout period as the flow is UDP. As mentioned previously, all approaches require two packets of the flow for accurate detection. However, AttnRNN demonstrates the ability to detect the UDP scan immediately after the occurrence of the second packet, whereas conventional NIDS systems have to wait until the end of the flow, resulting in a significantly longer detection time (around 60 seconds compared to just a few milliseconds).

To summarize this subsection, our proposed model is capable of promptly classifying malicious flows as soon as an attack trace emerges. As discussed in paragraph 6.5.5, our model can reduce the number of packets that need to be analyzed. Additionally, as shown in paragraph 6.5.5, our model significantly decreases the time required for flow detection. These findings related to our model capabilities offer multiple advantages, including minimizing the impact of attacks, reducing computational overhead associated with analyzing skipped packets, improving attack response time, and preventing further malicious activities within the network.

Conclusion

In this chapter, we addressed the challenge of early detection of network intrusions. We proposed a NIDS that utilizes packet headers. Our proposed NIDS incorporates attention mechanisms and RNNs to leverage the sequential nature of Network Flows. We evaluated the effectiveness of our approach by conducting experiments on two up-to-date network intrusion datasets. Furthermore,

we discussed the ability of our model for early detection and demonstrated its advantages from two perspectives: (*i*) our approach reduces the number of initial required packets to classify the flow, (*ii*) it reduces the time needed for detection compared to existing flow-based NIDS approaches. The proposed NIDS shows great promise in enhancing the security and resilience of networks across different environments, including 5G and beyond networks.

In this contribution, the proposed NIDS is trained and deployed on a per NS basis. This means that each NS owner is responsible for training and deploying its own model. However, in a scenario with a massive deployment of NSs, it becomes evident that collaboration among NSs is necessary to construct a model that can learn from data across different NSs. Unfortunately, sharing data for this purpose is not feasible due to privacy and business concerns. The next chapter will delve into this problem, exploring the challenges of training NIDS collaboratively without sharing data.

Chapter 7

Distributed DL-Based NIDSs in 5G networks

Introduction

In this chapter, we delve into the collaboration of 5G NSs owners to develop a global DL-based NIDSs without the need to share NS data due to privacy reasons. Fortunately, Federated Learning (FL) offers this possibility. However, this approach presents challenges when dealing with heterogeneous networks, where local NS data exhibits Non-Independent and Non-Identically Distributed (Non-IID) distributions. This chapter aims to conduct comprehensive experiments, evaluating state-of-the-art Federated Learning (FL) algorithms under both Independent and Identically Distributed (IID) and Non-Independent and Non-Identically Distributed (Non-IID) data distribution scenarios. The considered FL algorithms include Federated Average (FedAvg), Federated with the Proximal term (FedProx), Federated Personalizing (FedPer), and SCAFFOLD.
We will compare these FL models against both centralized and locally trained models. Through this comparative analysis, we will draw meaningful conclusions and engage in a discussion to provide insights for building collaborative IDSs.

The rest of this chapter is organized as follows. Section 7.1 provides more details about the problematic of using of FL in heterogeneous networks. Section 7.2 summarizes our contributions. Section 7.3 explains the FL aggregation algorithms. Section 7.4 provides an overview of related works on using FL for NIDSs. Section 7.5 outlines our methodology, details the dataset used, and its partitioning for the IID/Non-IID scenarios, and specifies the DL models, including centralized, FL, and local models. The experimental evaluations of these models are presented in Section 7.6. Finally, we conclude the chapter.

7.1 Context

Service owners in 5G and beyond networks must manage their Network Slices (NSs) and design security mechanisms for their NSs. This responsibility is inherited from the shared responsibility model used in the cloud computing, where the physical infrastructure owner is responsible for the

underlying infrastructure, and the cloud client (in our context, the NS owner) is responsible for their data and applications.

To design a DL based NIDS, NS owners should contribute their data to construct a global dataset used for training the DL model, this method is known as centralized training.
In practice, however, the centralized approach has proven inconceivable because the Network Flows dataset contains sensitive information about companies and NS users. This raises serious privacy issues, which are further regulated by legislation (e.g., General Data Protection Regulation (GDPR) [VB17]), and labeled datasets can disclose that an attack has occurred within the NS, and NS owners are reluctant to share this information. Therefore, sharing the data to build a global dataset is resisted by individual NS owners. Even so, NSs must collaborate in order to deploy effective NIDS; unfortunately, DL models trained independently at the NS level using local NS data may miss relevant samples of malicious and benign traffic, resulting in a decrease in the effectiveness of the learning model.

To deal with this challenge, the community has been considering distributed techniques to perform training, without requiring data sharing across NS. The advent of Federated Learning (FL) [McM+16], a framework that allows decentralized training, has been a catalyst for NIDS research [Lav+22].
FL consists of performing local training for different clients, with the clients communicating just their local model parameters to a coordinator, this approach ensures privacy since local data is not shared. The coordinator performs an aggregation procedure, then this aggregated model is returned back to clients; the training process is repeated until convergence. In [McM+16], the federated average (FedAvg) aggregation function was introduced; it simply averages the clients' models weights in the aggregation phase, to obtain the global model at each training round. However, FL faces a number of challenges that impede its use in the industry, including concerns about statistical heterogeneity (non iid-ness), communication efficiency, and security and privacy. The authors in [Li+20; Rah+21; Wah+21; Ma+22] reviewed these challenges and possible solutions found in the literature.

To recap, the main objective of FL is to deliver a DL model with similar performances to the centralized one. Statistical heterogeneity concern compromises this objective, and the FedAvg aggregation function has shown its limitation in this realistic scenario, suffering from a weight divergence from an optimal model that leads to unstable convergence[Kar+19].

In light of these challenges, new aggregation methods have been introduced, including FedProx[Li+18] and SCAFFOLD[Kar+19] as two novel aggregation techniques, that solve the high weight divergence caused by Non-IID data by using controlling mechanisms in client updates. The FedPer[Ari+19] aggregation approach tackles the issue, by personalizing DL models via transfer learning.

7.2 Contribution Summary

In order to contribute to the development of FL-based NIDS capable of addressing the aforementioned issue, this study investigates FL and the aggregation methods by which it can be utilized to enable collaborative DL-based NIDS in a 5G and beyond slicing environment.

The contribution in this chapter aims to delve into state-of-the-art FL algorithms, namely FedProx, FedPer, and SCAFFOLD, designed to address the challenge of weight divergence in heterogeneous networks. It focuses on two primary settings: one where data is uniformly distributed across NSs (IID), and another, more realistic scenario where data distribution is non-uniform (Non-IID) among NSs. In this context, this chapter explores situations where malicious traffic may manifest in one NS while being absent in another. The chapter also conducts a comparative analysis of the performance of these models in comparison to locally trained models and centralized global models, a comparison that has been overlooked in existing research works. Furthermore, within this chapter, we present a framework for the training and deployment of FL-based NIDSs within the context of 5G and beyond networks. The experimental study reveals that FL can be viewed as a "curse" because the exchanging models involve communication overhead but do not improve detection Accuracy and performance compared to locally trained models. Fortunately, state-of-art approaches could overcome convergence problems in Non-IID data, a "blessing" for future FL-based models to enable effective distributed learning.

7.3 Background: Federated Learning

Federated Learning is a distributed DL approach with the aim of building a single global model from data stored on multiple end devices (or entities); devices are referred as clients. The model training is conducted at the client level, and only model weights are exchanged across clients.

The learning of FL was formulated in [McM+16] as a minimization function:

$$\min_{w} \sum_{k=1}^{K} \frac{n_k}{n} \mathcal{L}_x(W), x \in X^k \tag{7.3.1}$$

Where K is the number of clients, $\mathcal{L}_x(w)$ represents the Loss function of input x where the output is predicted by the model having $\theta = w$, X^k is the client's k local dataset with $n_k = |X^k|$ and $n = \sum n_k$.

FL was initially designed to reduce the communication overhead [McM+16]. This communication efficiency is achieved by sharing model weights, rather than client-generated large-sized raw data. Additionally, FL can guarantee more data privacy since the client's local data will always reside at the client level. However, one of the primary objectives of FL is to show classification/prediction results comparable to models trained in a centralized fashion. This can be challenging because a single set of client data is not supposed to reflect the distribution of data in the domain.

Starting from an initial DL model, the process begins by training separate models locally for a certain number of iterations (E), each client using its own data. These respective models are then communicated to a central node, and aggregated using an aggregation function to obtain a global model. This process is called a round. At the end of each round, clients receive the aggregated model, and the process is repeated for a certain number of rounds (R). The majority of FL models rely on client-server architecture, in which the server (or coordinator) is in charge of initializing the model and performing aggregation at each round. However, other topologies, such as peer-to-peer architecture, are also possible. Algorithms 3, 4 summarize the FL training procedure for both the server-side (coordinator-) and client-sides, respectively. Regarding the client procedure (Algorithm 4), η represents the LR, $\nabla\mathcal{L}$ is the gradient of the Loss function ($\mathcal{L}_x(w)$).

In the next sub-sections, we will explore four aggregation algorithms, namely FedAvg, FedProx, FedPer, and SCAFFOLD [Li+18; Ari+19; Kar+19].

Algorithm 3 FL Server (Coordinator)

Input: *clients*,r
Output: w_r
1: $w_0 \leftarrow$ init()
2: **for** $t = 1$ to r **do**
3: $\quad w_t^k \leftarrow$ clients.local_train(t,w_{t-1}) // Executed in parallel
4: $\quad w_t \leftarrow$ **aggregate**(w_t^k)
5: **end for**
6: **return** w_r

FedAvg

Federated Average (FedAvg) is the first averaging algorithm proposed in [McM+16]. The approach consists of simply averaging the weights of the different models communicated by the clients. As shown in (Equation 7.3.2), the global model w_t at round t is calculated by averaging $n_k w_t^k$, this line will replace the line 4 in algorithm 3.

$$w_t \leftarrow \frac{1}{n} \sum_{k=1}^{K} n_k w_t^k \tag{7.3.2}$$

FedProx[Li+18]

Federated with the Proximal term (FedProx) was suggested to deal with heterogeneous networks; seeking to encounter FedAvg's limitations in Non-IID situations. FedProx changed the objective function on the client side, by adding a proximal term (μ) to regulate the direction of weights at the client level, and prevent the client's model to deviating from the global model communicated, by the coordinator at the beginning of the round. The line 5 in algorithm 4 will be:

$$w_t^k \leftarrow w_t^k + \eta(\nabla \mathcal{L}_x(w_t^k) + \frac{\mu}{2}||w_t^k - w_{t-1}||^2) \tag{7.3.3}$$

Where w_{t-1} is the global aggregated model of the previous round $(t-1)$, μ is the proximal term and $||w_t^k - w_{t-1}||^2$ is an L2 norm which measures the distance between the model during the local training and the previous global model communicated.

SCAFFOLD[Kar+19]

Stands for the stochastic controlled averaging algorithm. Its main aim is to reduce the client variance induced by heterogeneity in client updates. In this approach, both the clients and server (coordinator) procedures are updated and are shown in algorithms 5 and 6. The server's model and all the clients' models are equipped with control variates, c and c_k, respectively, the difference (c_k-c) estimates client (k)'s weight drift, and used to control the client update (line 6 in algorithm 6).

Algorithm 4 FL Client Local Train

Input: t,w_{t-1},X^k,E,B
Output: w_t^k
$\mathcal{X} \leftarrow$ getBatches(X^k,B)
$w_t^k \leftarrow w_{t-1}$
for $e = 1$ to E **do**
  for χ in $\mathcal{X}$ **do**
    $w_t^k \leftarrow w_t^k + \eta \nabla \mathcal{L}_x(w_t^k)$, $x \in \chi$
  end for
end for
return w_t^k

Algorithm 5 SCAFFOLD Server

Input: *clients*,r
Output: w_r
$w_0 \leftarrow$ init()
$c_0 \leftarrow 0$
for $t = 1$ to r **do**
  $(\Delta w_t^k, \Delta c_t^k) \leftarrow$ clients.local_train(t,w_{t-1},c_{t-1})
  $(\Delta w_t, \Delta c_t) \leftarrow \frac{1}{K} \sum_{k=1}^{K} (\Delta w_t^k, \Delta c_t^k)$
  $w_t \leftarrow w_{t-1} + \gamma \Delta w_t$
  $c_t \leftarrow c_{t-1} + \Delta c_t$
end for
return w_r

FedPer[Ari+19]

Federated Personalizing (FedPer) is a transfer learning-inspired approach. It involves modifying the training process of FL by integrating transfer learning technique. This approach is compatible with various FL aggregation algorithms, including FedAvg and FedProx. In Federated Personalizing (FedPer), the model is divided into the following: base and personalized layers. Only the base layers are communicated and aggregated in the server, while the personalized layers are trained locally; the architecture is shown in Fig. 7.1.
Consequently, the FedPer aggregation updates the algorithm executed at the client level; at the first round the clients initialize their local weights, and during the training the used model is the concatenation of base and personalized layers. The algorithm 7 showcases the procedure.

7.4 Related Works

Given the success of DL-based models in performing various decision and prediction tasks in communication and networking systems, the industry began thinking about collaborative ML techniques to break down the data sharing barrier. In this context, FL has been welcomed as an alternative framework to deliver global models trained in a distributed way, without data exchange.

Besides, following the emergence of DL, which significantly increased research interest in IDSs,

Algorithm 6 SCAFFOLD Client Local Train

Input: t,w_{t-1},c_{t-1},X^k,E,B
Output: Δw_t^k,Δc_t^k
if $(t = 1)$ **then** $c_0^k \leftarrow 0$ // Init
$\mathcal{X} \leftarrow$ getBatches(X^k,B)
$w_t^k \leftarrow w_{t-1}$
for $e = 1$ to E **do**
  for χ in $\mathcal{X}$ **do**
    $w_t^k \leftarrow w_t^k + \eta(\nabla \mathcal{L}_x(w_t^k) - c_{t-1}^k + c_{t-1})$, $x \in \chi$
  end for
end for
$c_t^k \leftarrow \nabla \mathcal{L}_x(w_{t-1})$, $x \in X^k$
$(\Delta w_t^k, \Delta c_t^k) \leftarrow (w_t^k - w_{t-1}, c_t^k - c_{t-1}^k)$
return Δw_t^k,Δc_t^k

Algorithm 7 Federated Personalizing (FedPer) Client Local Train

Input: t,w_{t-1},X^k,E,B
Output: w_t^k
if $(t = 1)$ **then**
  $wp_0^k \leftarrow init()$
end if
$\mathcal{X} \leftarrow$ getBatches(X^k,B)
$temp_t^k \leftarrow (w_{t-1}, wp_{t-1}^k)$
for $e = 1$ to E **do**
  for χ in $\mathcal{X}$ **do**
    $temp_t^k \leftarrow temp_t^k + \eta \nabla \mathcal{L}_x(temp_t^k)$, $x \in \chi$
  end for
end for
$(w_t^k, wp_t^k) \leftarrow temp_t^k$
return w_t^k

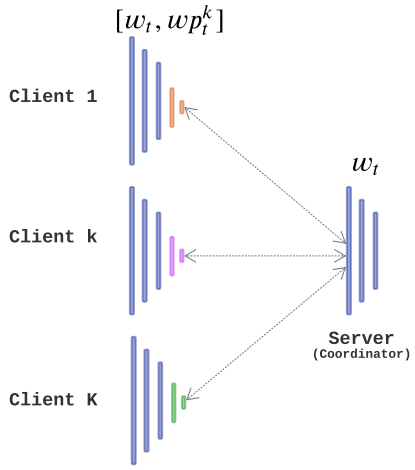


Figure 7.1: Federated Personalizing (FedPer) aggregation algorithm: layers in blue are the base layers w_t, the other colours represent the personalized layers wp_t^k. The server knows only base layers.

the appearance of FL in 2018 stimulated research efforts in collaborative NIDS [Lav+22]. Hence, a wide range of works have been proposed, addressing the integration of FL to NIDS were proposed.

In [Mot+22], the authors have proposed a RNN (LSTM and GRU) trained in a federated fashion to detect anomalies in Internet of Things (IoT) networks, using the ModBus network dataset [RM21]. Seven different models were trained while varying the window size of their time-series data; then these models were combined using a decision tree vote-based scheme to classify the traffic with high confidence.
Attack detection in wireless edge-enabled networks using FL, while minimizing communication costs and ensuring high detection performance was discussed in [Che+20b]. The authors leveraged attention mechanisms applied to GRU-based ML models in their FL training process. The use of attention is meant to increase the weight of important devices (clients) in the training phase.
In [Mot+22; Che+20b], they stated that FL models outperform the centralized approach. Conversely, in our study, the centralized model is assumed to be the optimal model with the best performances, and the experiments showed that (see section7.6), and we are attempting to get similar results using FL. In [Zha+21], the authors proposed a FL-based framework for NIDS in the IoT context. They used a semi-supervised scheme, where auto-encoders based models are trained using the FedAvg algorithm, across different IoT devices. The authors presented a method for calculating a global reconstruction error threshold for the traffic classification task. The authors used the NB-IoT dataset [Mei+18] for the experiments, where data is collected from 9 IoT devices attacked with mainly the same attacks. The FL approach outperformed the local approach, while having nearly the same performances as the centralized one, thanks to the introduced global reconstruction error threshold approach.
The authors in [Fan+20] expressed the need of personalizing distributed DL-based model, in the context of 5G IoT, due to IoT devices' heterogeneity. The authors presented a similar approach to

FedPer, by leveraging personalized layers, that are learned via transfer learning. At each training round, each client performs a train with a public shared dataset, communicates the model weights to the server to perform the aggregation (FedAvg). Then, the server returns the aggregated model to the clients, and each client performs a training epoch with its local data, on its personalized layers. The process is repeated for all training rounds. Furthermore, by utilizing a public dataset, that is shared by all clients, the approach is somehow inspired by knowledge distillation in FL. Four separate clients are simulating four different networks for the experimental phase; the authors used both IoTDataset [Kan+19] and NSLKDD [Tav+09] datasets for the clients' data, and CI-CIDS [SLG18] dataset for the public dataset. The authors revealed the good results obtained by their approach; nevertheless, the generalization aspect of each separate model is missing, as well as comprehensive comparison with models trained on local data only.
The authors in [Rah+20] investigated the implementation of IDS DL models, whether centralized, local, or based on FL. They evaluated the NSLKDD dataset in IID and Non-IID partitioning settings. The authors reported that FedAvg obtained comparable Accuracy to the centralized approach, while outperforming the on-device models. Besides, under Non-IID settings, FedAvg was marginally higher than on-device, and significantly exceeded by the centralized approach. In addition to their investigation, our work includes a pathological Non-IID scenario as well as multi-class DL models classifiers. Furthermore, we investigate various FL aggregation algorithms, that have been proposed to address the limitations of FedAvg in heterogeneous networks.

7.5 Methodology

In this section, we detail our methodology to study the FL-enabled IDS performance involving the use of DL, FL aggregation algorithms and the partitioning of IID and Non-IID datasets. Before we proceed, we first introduce the sliced 5G architecture as well as the system model we considered in our study.

7.5.1 Our Sliced 5G Architecture

In light of the success of DL in different decision-making tasks and the emergence of virtualization in 5G and beyond networks, the NWDAF was created to support the deployment of ML models pipeline in the network. NWDAF's primary function is to provide analytics to 5G NFs, which in turn make decisions based on the insights obtained from these analytic findings[Yua+22]. For intrusion detection, the NWDAF gathers network packets and transforms them to Network Flows. The Network Flow is then analyzed using the DL model, and the classification result is communicated to the security VNF, in order to devise the appropriate policy, for example, in the event of an attack.

One can imagine that the NS's security engineers gather elaborated flows from the NWDAF; the training is then performed after data labeling. The DL model training can be performed locally or collaboratively with other NSs, which include centralized and federated methods. Finally, the DL model is deployed at the NWDAF level, and the SecurityVNF will request the NWDAF to classify future network traffic. Fig. 7.2 depicts and illustrates the whole process that can be summarized as follows: 1. NWDAF collects traffic from UPF, 2. NWDAF generates Network Flows, 3. Security engineer gathers the unlabeled flows, 4. Security engineer labels the flows and sends them back to NWDAF, 5. Training phase: (*i*) Local training is done at the NWDAF level, (*ii*) Centralized training: labeled flows are transmitted to the coordinator (5.a), which performs training (5.b)

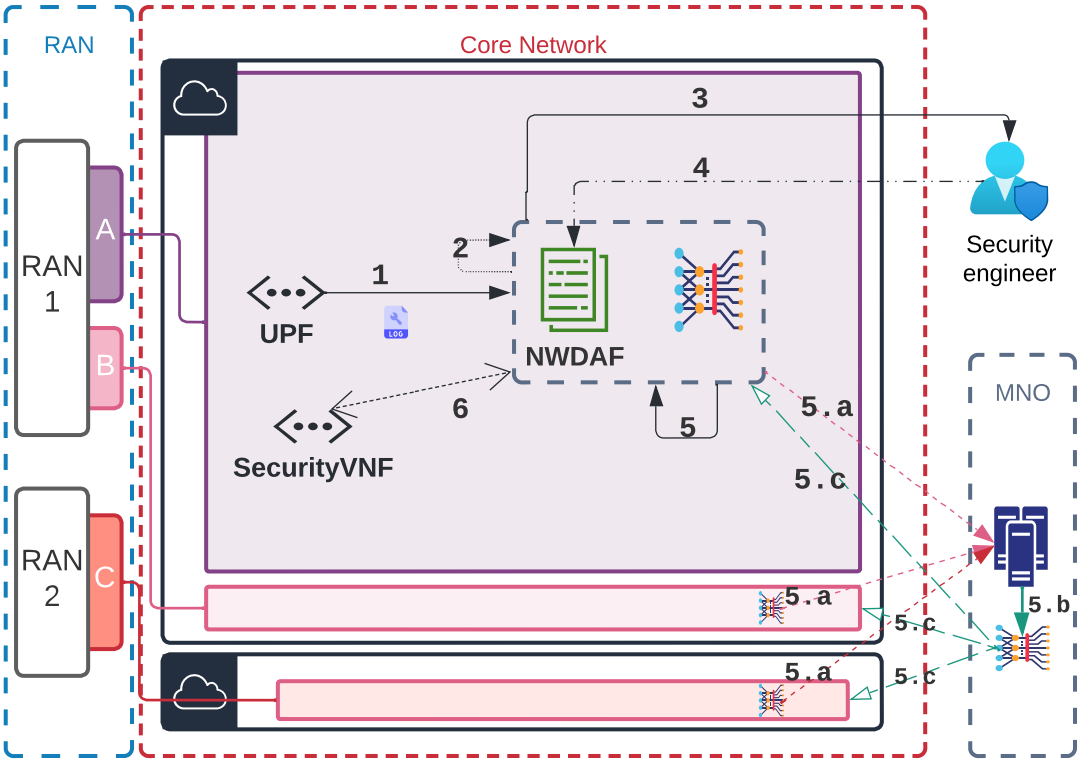


Figure 7.2: DL based NIDS in 5G and Beyond

and returns the final model (5.c), (*iii*) FL: for each round, model weights are transmitted to the coordinator (5.a), which performs aggregation (5.b) and returns the aggregated model (5.c), 6. SecurityVNF requests the NWDAF when new traffic arrives

In our system, we consider a set of K slices (or clients following the FL process), each slice c_k has a labeled Network Flow X^k dataset; where $X^k = X^k_{\text{train}} \cup X^k_{\text{test}}$, X^k_{train} and X^k_{test} refer to train and test set for client k, respectively.

We also define $M_{\text{centralized}}$ as the centralized model, which has $X^{\text{centralized}}_{\text{train}}$ and $X^{\text{centralized}}_{\text{test}}$ as training and test sets; where $X^{\text{centralized}}_{\text{train}} = X^1_{\text{train}} \cup X^2_{\text{train}} \cup \cdots \cup X^K_{\text{train}}$ and $X^{\text{centralized}}_{\text{test}} = X^1_{\text{test}} \cup X^2_{\text{test}} \cup \cdots \cup X^K_{\text{test}}$. For FL approaches, slices will act as clients, and we build three FL global models using three different aggregation algorithms (M_{fedavg}, M_{fedprox} and M_{scaffold}), and K personalized model M^k_{fedper}. Besides, M^k_{local} reflects the local models trained on X^k_{train} and X^k_{test}. In order to evaluate the different ML models M_x, we also consider a validation dataset. We evaluate the performance of the models (M_x) with this validation set, assumed as new traffic to evaluate future use.

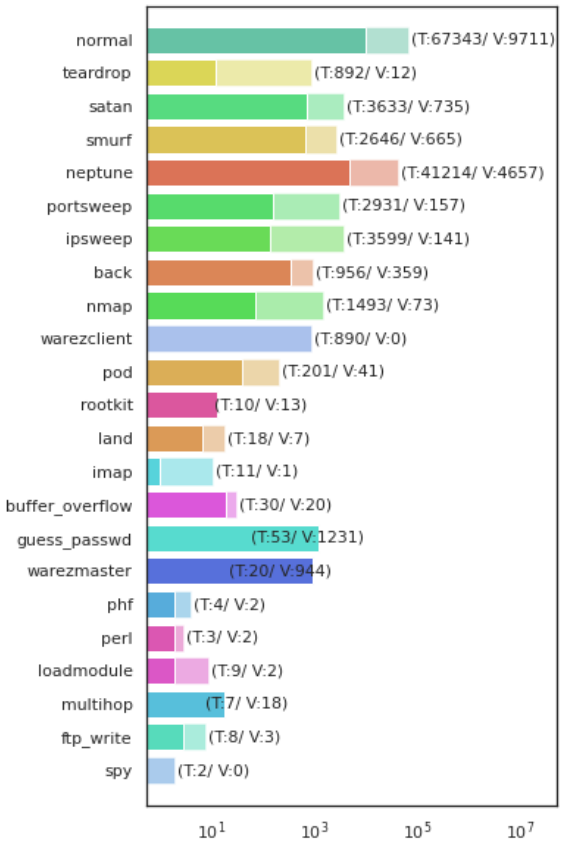


Figure 7.3: NSLKDD label distribution. T: Train / V : Validation datasets

7.5.2 Dataset Description and Pre-processing

In our study, we used the NSLKDD dataset [Tav+09], since it was widely used by the research community and is considered a reference dataset for flow-based NIDS. The primary data files of NSLKDD are "*KDDTrain+*" and "*KDDTest+*". The former will be used during the training phase, while the latter will be kept for the validation phase. The purpose of utilizing the validation dataset is to evaluate the performance of the various models using previously unseen data. The results related to this validation data are depicted in Figures 7.8 and 7.12.

The training dataset has 148,517 records with 41 features and covers 23 different labels. The dataset is mostly made up of normal and Neptune attack traffic, the latter is a flooding-based DoS attack.

Fig. 7.3 displays the label distribution of NSLKDD dataset. Features' nature is either Boolean, real, or categorical. In regard to data pre-processing, we used min-max normalization to convert real values to a range from 0 to 1. In our contribution, we assumed that the involved NSs communicate their minimum local values to enable global min-max normalization.
The dataset has three categorical features: protocol type, service, and flag. The categorical features

are encoded between 0 and 1 using the following encoding process, e.g. the protocol type has three values: 'tcp', 'udp', and 'icmp' and the corresponding value for each category equals to its order count divided by the number of categories (3), given that the order of counting starts from zero. Hence, the result is $\{0, \frac{1}{3}, \frac{2}{3}\}$. Flow labels are indeed encoded using one-hot encoding; at this stage, we can expect the DL model to have 41 input and 23 output sizes. The "*KDDTest+*" captured 17 new attack types. However, our research is limited to the 23 labels already present in the training set, and the new attack labels are removed from the validation set. In fact, the new attacks are convenient for binary classification (normal and abnormal), where researchers validate the efficiency of their models, by having them learn normal behavior and label any new attack pattern as abnormal. But, in our study, we seek to identify the type of attack, and for novel attack labels, we cannot make such predictions since we do not currently have that knowledge during the training phase.

The training dataset is partitioned into K subsets for the generating slices' (clients') data X^k. Partitioning scenarios are discussed in the next subsection.

7.5.3 Partitioning Scenarios

In our study, we consider eight slices ($K = 8$), where "*KDDTrain+*" dataset is partitioned according to two different scenarios. We chose this number of clients ($K = 8$) due to practical reasons. The NSLKDD dataset, although valuable, isn't very large. Including more clients could have made the dataset less representative. Thus, we decided that having eight clients would provide a manageable dataset size, allowing us to explore diverse scenarios effectively. The labels distribution for both scenarios are shown in Figs. 7.4 and 7.5.

IID Data-based scenario

In this scenario, the "*KDDTrain+*" data is randomly partitioned across the clients using a uniform distribution, assuming that each NS has known more or less the same nature of traffic, hence the NSs undergo the same attacks and with the same frequency.

Non-IID Data-based scenario

This pathological Non-IID scenario, is more realistic, where the assumption that NSs undergo the same type of attacks is not valid. So, in this scenario, we assume that normal traffic is more or less collected with the same distribution across NSs. However, for attacks, we assume that the traces of some attacks are only available in one NS and not in others.

7.5.4 NN Architecture and Training

Table 7.1 summarizes the NN architecture, which is used to build all the models M_x described previously. The NN model has six layers, each followed by a ReLu activation function and a batch normalization[1] layer, while the output layer is succeeded by a dropout layer. In the FedPer approach (M^k_{fedper}), the layers 5(*) and 6(*) are used as personalized layers as detailed in sub-section 7.3.

The majority of the rows in the training sets (X^k_{train} and $X^{\text{centralized}}_{\text{train}}$) are normal and neptune, as seen in Figs. 7.4 and 7.5. To handle this unbalanced data situation, we used weighted Cross-Entropy

[1] A technique used in NNs to improve training stability and speed by normalizing the input of each FFNN layer in a mini-batch to have zero mean and unit variance.

Table 7.1: NN (M_x Architecture)

N	**Layers**	d_{in}	
0	Linear	41	**input layer**
1	Linear ReLu BatchNorm	128 128	
2	Linear ReLu BatchNorm	256 256	
3	Linear ReLu BatchNorm	128 128	
3	Linear ReLu BatchNorm	128 128	
4	Linear ReLu BatchNorm	64 64	
5(*)	Linear ReLu BatchNorm	32 32	
6(*)	Linear ReLu dropout	23	**output layer**

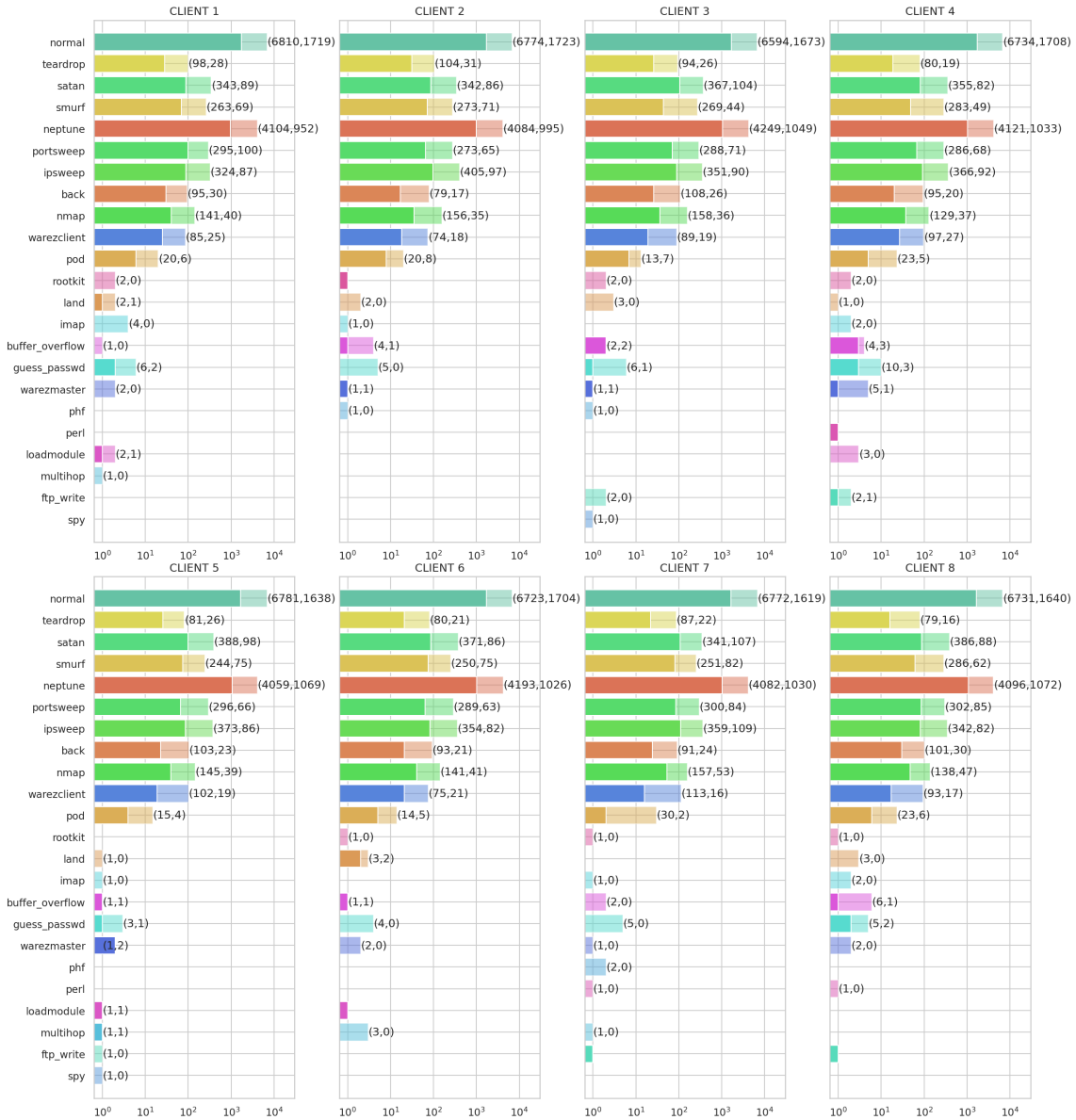


Figure 7.4: IID Scenario. Label distribution over K clients, (X^k_{train},X^k_{test})

Loss for the DL training function. We used SGD as the learning Optimiser, setting the LR (η) at $3e^{-4}$. Concerning the remaining training-related hyper-parameters, we set B to 64, the training rounds (R) to 200, the number of local epochs learning (E) to 1, the proximal term for FedProx (μ) to 0.3 and the step size for SCAFFOLD (γ) to 1. These hyper-parameters were selected following a series of tests using a generate-and-test methodology for hyper-parameter tuning.

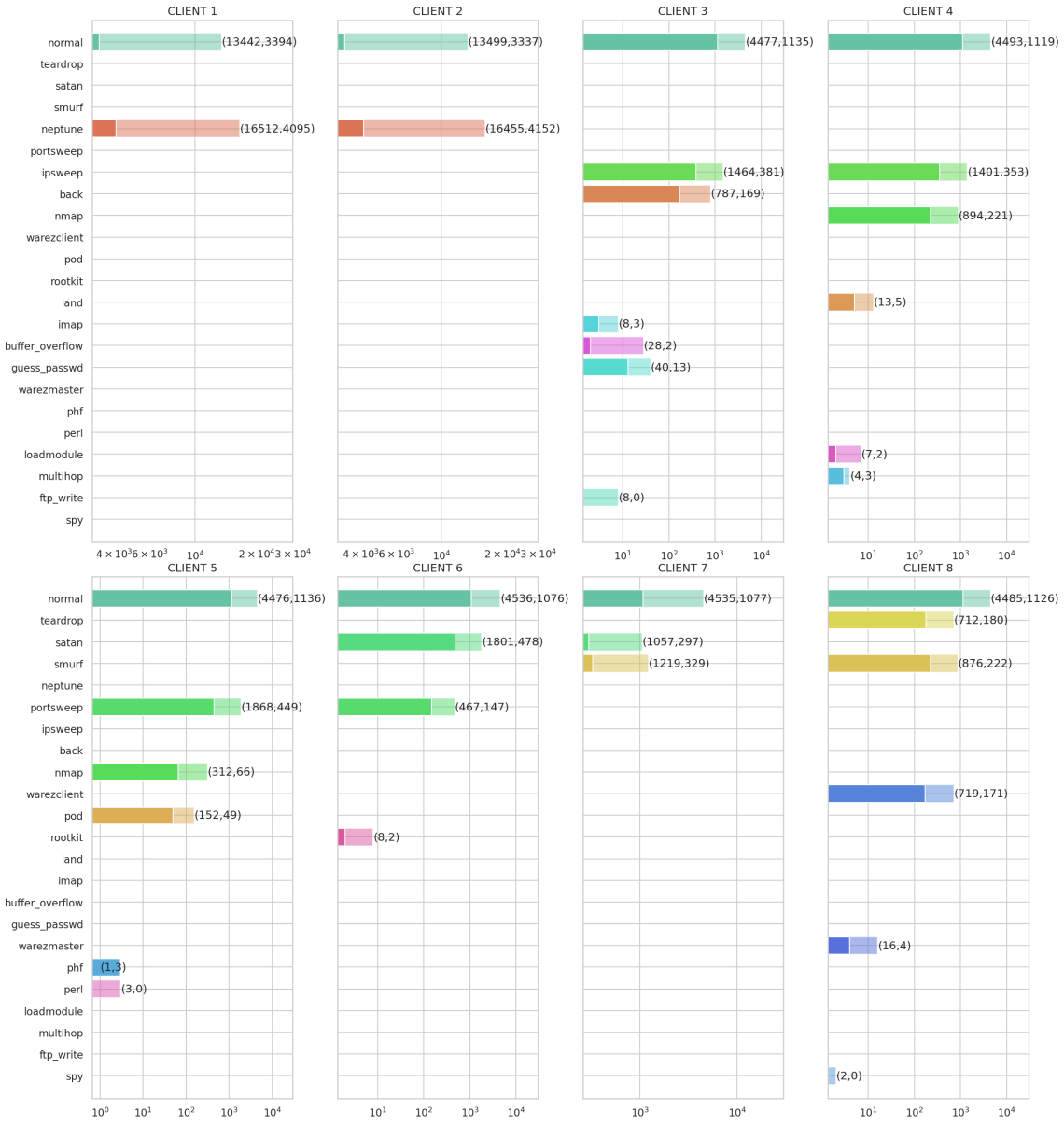


Figure 7.5: Non-IID Scenario. Label distribution over K clients, $(X^k_{\text{train}}, X^k_{\text{test}})$

7.6 Experimental Study and Results

This section presents experimental study and evaluates the performances of different DL models M_X on top of both IID and Non-IID scenarios and different aggregation algorithms.

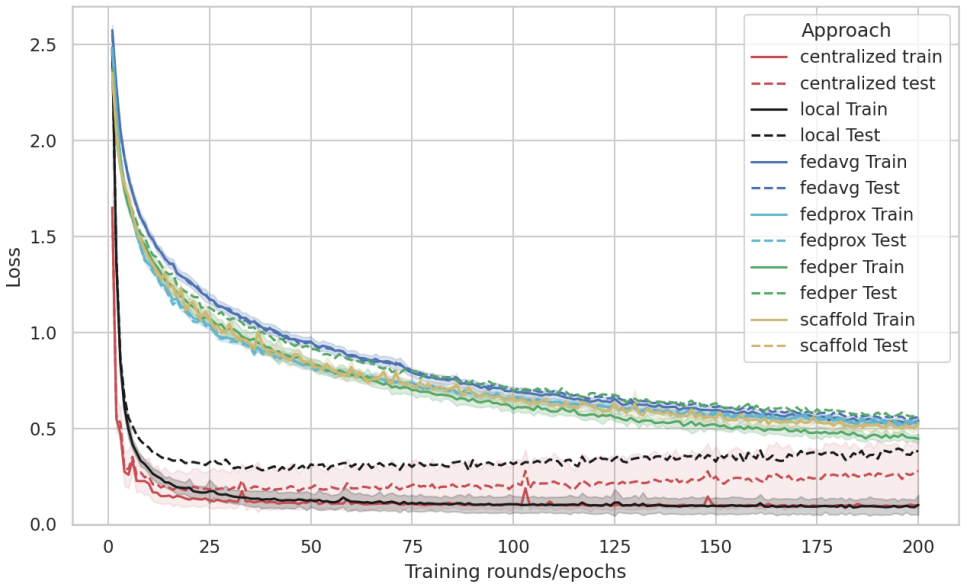


Figure 7.6: IID Loss.

7.6.1 IID-based Scenario

Fig. 7.6 depicts the training classification Loss, the weighted cross entropy as mentioned previously, during each training round/epoch of the different DL models (M_x) for IID scenario, with the lines and dashed lines in the graph representing the average Loss on the training (X^k_{train}) and test (X^k_{test}) sets at each round/epoch, respectively. Furthermore, the colored contour shows the confidence interval of the train Loss' average. It is calculated as the average plus-minus the standard deviation of the Loss in (X^k_{train}); this confidence interval presents insight about the Loss trend across all clients. The four FL models in this scenario converge at the same pace, which is promising. Another interesting aspect is that the confidence interval is so tight, that the Loss standard variation is ≈ 0. This implies that at each round, the Loss value is approximately the same for all clients, simply due to the similar distribution in this training scenario. We also remark that both centralized and local models converge at around 25th epoch. But, they then started to over-fit to the training data set, due to over-training of the model, which led to over-fitting of the common labels (normal and Neptune). Another reason is the weighted Loss function, which penalizes the error on the non dominant labels more. Furthermore, the centralized model outperforms the other models, despite the fact that they all have the same distribution. This is because the centralized model has a complete view of all the distinct traces of the different labels.

Fig. 7.7 displays the M_x's Accuracy during the training process. We clearly observe that both the centralized and local models provide the best results. The previously mentioned minor over-fitting between train and test does not appear, because the Accuracy calculation is not weighted, and non dominant labels have the least impact on this metric. In addition, most of the FL methods

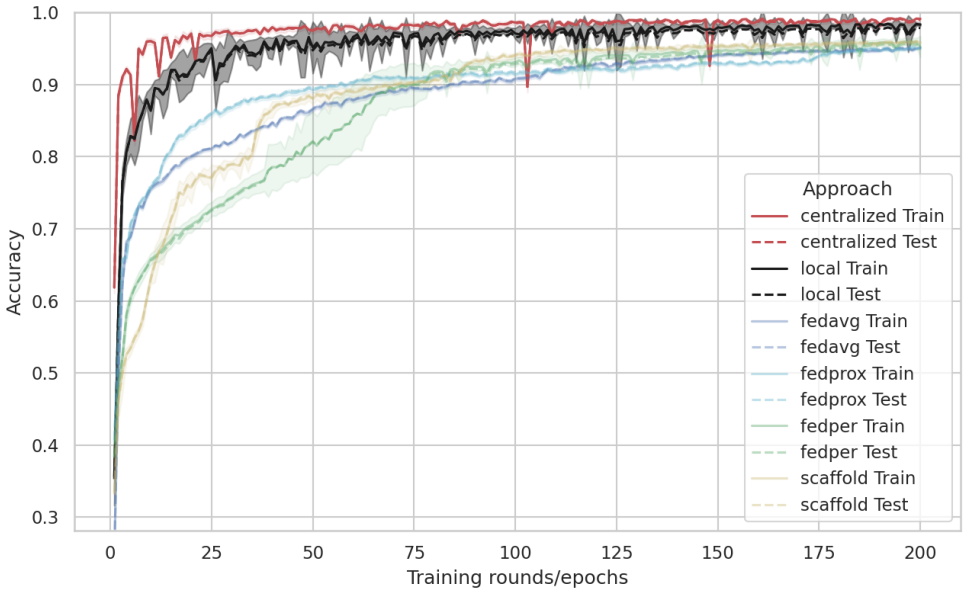


Figure 7.7: IID Accuracy.

progress toward the same acceptable Accuracy of 95 ($\approx 95\%$); FedPer and SCAFFOLD converge the slowest, however after a few rounds (25), they start to follow the converging tendency like other FL approaches.

The models' efficiency is evaluated by showing how they perform on a validation dataset. Accuracy in validation data is shown in Fig. 7.8. The Accuracy values of the global models ($M_{\text{centralised}}$,M_{fedavg}, M_{fedprox} and M_{scaffold}) are identical throughout all clients, while for FedPer and local models (M^k_{fedper}, M^k_{local}) they are proper to each client k. We can see that all models achieved the same performances ($\approx 80\%$). The federated techniques were able to deliver similar results as the centralized approach, which is the purpose of federated learning. However, same outcomes were also obtained by local training, putting into question the necessity of collaboration in this scenario.

7.6.2 Non-IID-based Scenario

Similarly to the previous scenario, we elaborate the same evaluation/analyse for the Non-IID scenario. The (M_x)s' train/test Loss is presented in Fig. 7.9. The average Loss (represented in the figure by the lines and dashed lines) for both FedAvg and FedProx stays unchanged at (≈ 2) and does not decrease. Secondly, after 100 rounds, the average standard deviation values for these two approaches are (≈ 0.8), indicating a large confidence interval. This signifies that the classification Loss is not uniform across clients, with certain clients achieving good error scores, while others do not. Meanwhile, SCAFFOLD shows lower train/test Loss rates (≈ 1) when compared to Fed-

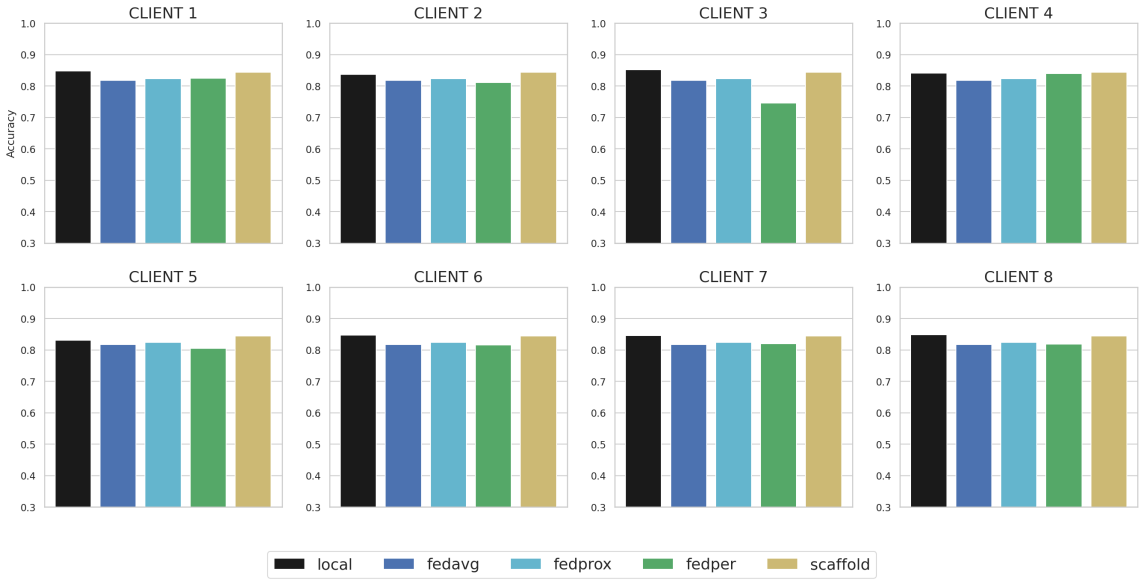


Figure 7.8: IID Accuracy on validation dataset.

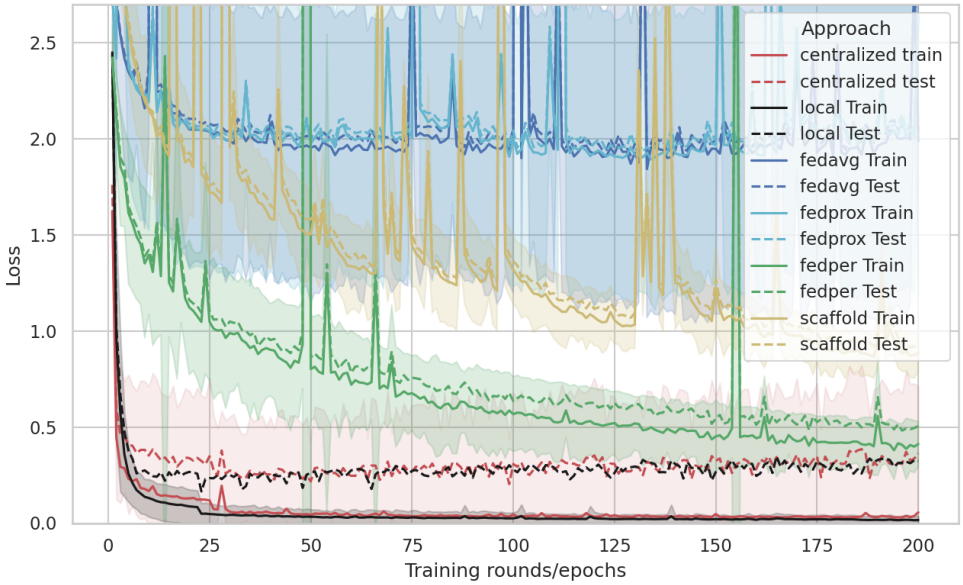


Figure 7.9: Non-IID Loss.

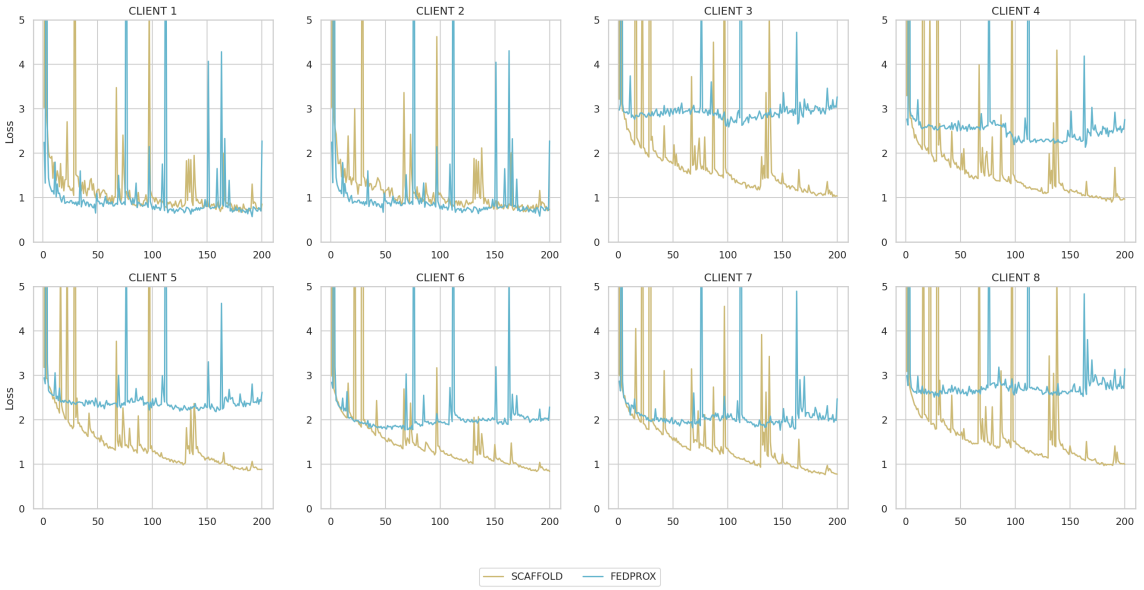


Figure 7.10: SCAFFOLD Vs FedProx Loss.

Prox and FedAvg. Additionally, it has the narrowest confidence interval, with an average standard deviation of (≈ 0.17), indicating that M_{scaffold} achieves almost same error rate for all clients.

Fig. 7.10 displays the training Loss over the K clients for FedProx and SCAFFOLD and clearly illustrates these findings. For SCAFFOLD, we can see that the Loss is descending equitably (to ≈ 1) for all clients, but FedProx is converging just for clients 1 and 2. However, the Loss sticks and does not decrease for other clients, indicating that the learning is poor. Both clients 1 and 2 contain normal and Neptune labels; FedProx is forcing to converge to a model that fits these two labels, while diverging from the optimal model that is expected to learn all labels.

FedPer has the lowest Loss when compared to other FL approaches. Each personalized model achieves the minimum Loss on its train/test data, with an average Loss of less than 0.5. These values were not previously obtained with other FL models. However, before the validation process, it is too early to make any conclusions. Local models also provide good results, with a narrower confidence interval than the centralized approach. This is because each client has a limited number of labels, this makes it easier for M^k_{local} to learn labels of $X^k_{\text{train}}/X^k_{\text{test}}$. For example, client 1 must learn just normal and Neptune. Also, for the same reasons indicated in the IID experiments (see subsection 7.6.1), the centralized model produces the best performance.

The Accuracy graph (Fig. 7.11) aligns to the Loss graph. The Accuracy of both centralized and local models is very close to 100%. Moreover, an interesting observation is that FedPer is showing comparable performances. FedAvg and FEDProx both have a large confidence interval and an Accuracy of approximately (80%); while SCAFFOLD reaches (90%) Accuracy with a small confidence interval, it outperforms both FedAvg and FedProx.

In the next paragraphs, we present and discuss the validation phase. To begin, we investigate personalized models (FedPer and local models). As shown in Fig. 7.12, M^k_{fedper} and M^k_{local} models achieve the same Accuracy rates over all clients K. $M^{1,2}_{\text{fedper}}$ and $M^{1,2}_{\text{local}}$ models learn well the Neptune

attack since it is frequent in $Tr_{1,2}$, but also in the validation dataset. This explains why $M^{1,2}_{\text{fedper}}$ and $M^{1,2}_{\text{local}}$ have high Accuracy rates on the validation data ($\approx 70\%$). Nonetheless, we notice that employing FedPer does not lead in any knowledge sharing between the different M^{k}_{fedper}, as seen by poor Accuracy rates for clients $c_3..c_8$. We find that the personalized layers (wp^k) trained via transfer learning force the M^{k}_{fedper} model to generate a model similar to the local one (M^{k}_{local}). This finding also explains why FedPer performs better across the different $X^{k}_{\text{train}}/X^{k}_{\text{test}}$; each personalized model M^{k}_{fedper} has lost its generalization ability and is over-fitting its locally known labels.

Table 7.2 contains the validation results for the remaining global models, including TP, TPR, PPV, and F1Score metrics for each label in the validation dataset. The three FL models perform well in learning the normal and Neptune labels, but only FedAvg and SCAFFOLD are able to do so for the Nmap label (label 3 in Table 7.2). SCAFFOLD outperforms FedAvg and FedProx for labels 4 to 7. For example, the Pod attack, which is not frequent in the training dataset, SCAFFOLD is able to learn effectively. However, labels 8-13 are unfortunately not learned by the FL models; while the centralized model identifies those labels adequately, the FL models ignore them. This demonstrates the current limitations of FL approaches in dealing with pathological Non-IID context.

Meanwhile, the centralized model fails to detect labels 14-20, which may be due to a variety of reasons including the pre-processing and encoding processes, the quantity/quality of training data, the DL architecture, etc. However, the purpose of this study is to compare the effectiveness of FL models with the centralized approach. The centralized approach is assumed to be optimal with satisfactory results (labels 1-13). In the previous paragraphs, we showed that SCAFFOLD outperformed the other FL-based aggregation approaches, and may be a solution for a stable convergence to an optimal model (which could be similar to the centralized). Even so, further enhancement of the control mechanisms is required to overcome SCAFFOLD's limitations (eg: labels 8-13).

Conclusion

In this chapter, we suggest an architecture for NIDSs in the context of multi-slice 5G and beyond networks, discuss and experiment with various state-of-the-art FL aggregation approaches to intrusion detection, and finally, emphasize the limitations of FL aggregation algorithms, namely FedAvg, FedPer, FedProx, and SCAFFOLD, as it stands at present.

We showed that for IID scenario, FL algorithms performed effectively. However, similar results were also obtained when each slice trained a model with only its local data. In reality, uniform sampling gives the same label distribution throughout NSs, resulting in equivalent models for both local and FL. The efficacy of FL, consequently, cannot be measured in this scenario.

In the second scenario, we considered a more realistic scenario in which the label distribution is not identical across FL clients (Non-IID scenario). We revealed that FedAvg converged for a limited number of clients' data while diverging in others, asserting that the model was forced to learn only a specific set of labels while ignoring the others. This results in poor Loss decreasing during training. FedProx surpassed FedAvg; however, FedProx, like FedAvg, has the same previous issue despite the fact that it has a controlling mechanism in place to deal with it. SCAFFOLD was the only global model that provided stable convergence across the FL clients; SCAFFOLD outperformed other FL approaches.

We also showed that FedPer training via transfer learning tends to replicate local models and

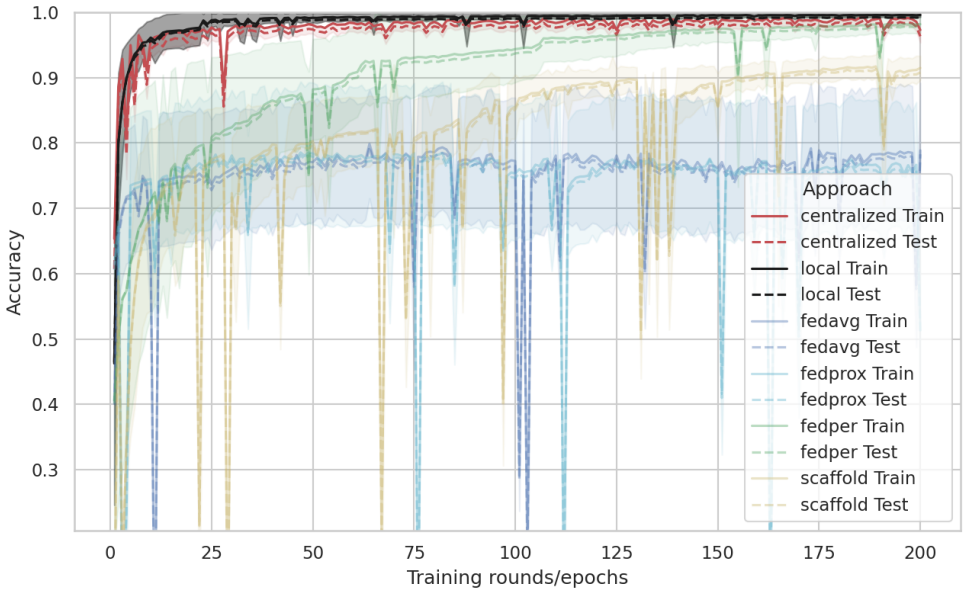


Figure 7.11: Non-IID Accuracy.

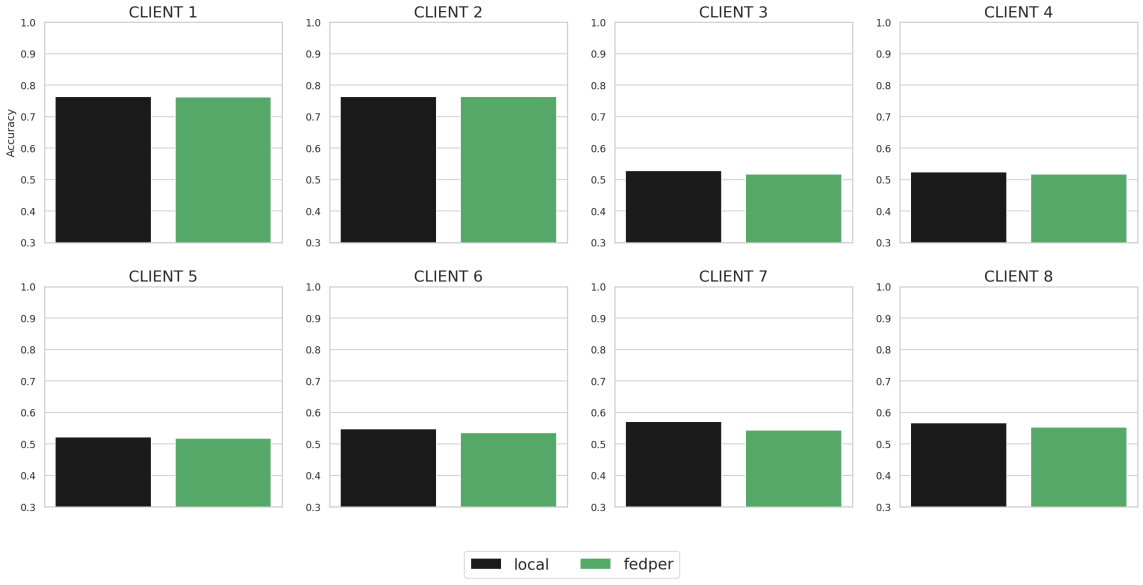


Figure 7.12: Local and FedPer Accuracy on validation dataset in Non-IID scenario.

Table 7.2: Detection metrics for FedAvg, FedProx, SCAFFOLD and Centralized models

label		N	FedAvg				FedProx				SCAFFOLD				CEN			
			TP	TPR	PPV	F1Score	TP	TPR	PPV	F1Score	TP	TPR	PPV	F1Score	TP	TPR	PPV	F1Score
1	normal	9711	9645	0.99	0.72	0.83	9642	0.99	0.71	0.83	9623	0.99	0.72	0.84	9199	0.95	0.82	0.88
2	neptune	4657	1457	0.31	1.00	0.48	1507	0.32	0.97	0.49	1426	0.31	1.00	0.47	4567	0.98	1.00	0.99
3	nmap	73	71	0.97	0.66	0.79	0	0.00	/	/	67	0.92	0.28	0.43	73	1.00	0.55	0.71
4	satan	735	397	0.54	0.68	0.60	372	0.51	0.85	0.63	490	0.67	0.82	0.74	457	0.62	0.74	0.68
5	prtswep	157	90	0.57	0.03	0.06	99	0.63	0.26	0.37	119	0.76	0.61	0.67	144	0.92	0.41	0.56
6	pod	41	0	0.00	/	/	0	0.00	/	/	25	0.61	0.60	0.60	35	0.85	0.73	0.79
7	land	7	3	0.43	0.33	0.38	1	0.14	0.08	0.10	7	1.00	0.00	0.01	5	0.71	1.00	0.83
8	smurf	665	0	0.00	0.00	/	0	0.00	/	/	8	0.01	0.53	0.02	656	0.99	0.97	0.98
9	back	359	2	0.01	0.67	0.01	0	0.00	0.00	/	0	0.00	/	/	324	0.90	0.58	0.71
10	ipswep	141	0	0.00	0.00	/	0	0.00	0.00	/	0	0.00	0.00	/	138	0.98	0.93	0.96
11	tear	12	0	0.00	0.00	/	0	0.00	0.00	/	0	0.00	0.00	/	11	0.92	0.21	0.34
12	phf	2	0	0.00	/	/	0	0.00	0.00	/	0	0.00	/	/	1	0.50	0.14	0.22
13	perl	2	0	0.00	0.00	/	0	0.00	/	/	0	0.00	/	/	1	0.50	0.09	0.15
14	rtkit	13	0	0.00	/	/	0	0.00	/	/	0	0.00	/	/	0	0.00	0.00	/
15	imap	1	0	0.00	/	/	0	0.00	0.00	/	0	0.00	/	/	0	0.00	0.00	/
16	buffer	20	0	0.00	0.00	/	0	0.00	0.00	/	0	0.00	0.00	/	0	0.00	0.00	/
17	guess	1231	0	0.00	0.00	/	0	0.00	0.00	/	0	0.00	0.00	/	0	0.00	0.00	/
18	mlthop	18	0	0.00	0.00	/	0	0.00	0.00	/	0	0.00	0.00	/	1	0.06	0.06	0.06
19	ldmdle	2	0	0.00	/	/	0	0.00	0.00	/	0	0.00	0.00	/	0	0.00	0.00	/
20	ftp	3	0	0.00	/	/	0	0.00	/	/	0	0.00	0.00	/	0	0.00	0.00	/

over-fit the local data. This approach is not yet mature since it loses generalization potential. Despite this, we can envision FedPer being a solution in cases where the same sample has different labels for different clients. This requires further investigation to locate these reflective labels and perform transfer learning to encounter this situation.

This chapter represents the culmination of our contribution to NIDSs. Next, we will delve into the conclusion of the manuscript, where we will summarize our findings, discuss our responses to the research questions posed at the beginning of this dissertation, and provide an overview of our contributions.

Conclusion, and Future Research

The research work in this thesis comprises research contributions aimed at advancing the knowledge and state of the art in 5G-V2X Intrusion Detection Systems by leveraging AI. In this regard, we have posed a primary research question (**RQ**) at the outset of this thesis, which revolves around the effectiveness and efficiency of these AI models. We have also formulated three sub-questions (**RQI**, **RQII** and **RQIII**) that we aim to answer satisfactorily.

We initiated our thesis by introducing fundamental concepts, specifically 5G-V2X and its associated key technologies, in Chapter 1. In Chapter 2, we also elucidated the background of the AI techniques utilized in our research, namely the Fuzzy Inference Systems and Neural Networks. Chapter 3 delves into network intrusions and IDSs in the context of V2X and 5G, as well as the state-of-the-art techniques employed in IDSs. These three chapters were consolidated as Part I of this thesis for organizational coherence.

Chapter 4 highlighted the vulnerability of the protocol used in SL communications to adversarial radio resource attacks, which compromise the availability of the cooperative awareness vehicular application. We demonstrated that this jamming attack can effectively isolate vehicles. This issue aligns with our **RQI**, which concerns ensuring the reliability of V2X aspects to meet the ultra-reliable communication requirements. Therefore, we have proposed a HIDS that performs both attack detection and mitigation. The detection mechanism relies on feedback, while a rule-based mitigation mechanism leverages Fuzzy Inference System. We demonstrated through simulations that our approach can mitigate the impact of the attack, even in scenarios involving multiple intelligent attackers collaborating to perform a distributed attack. This finding underscores the efficacy of employing AI techniques to safeguard communication systems against reliability-related attacks, and it opens the door for further exploration of AI methods like reinforcement learning in this context. One possible approach is to replace the threshold-based detection mechanism and the FIS resilience strategy deviser policy with a reinforcement learning model. This approach can be further enhanced by employing deep reinforcement learning.

In Chapter 5, our objective was to enhance the existing approaches in the literature by proposing an RNN-based method for detecting message forgery attacks in V2X SL communications. Our contribution revealed interesting conclusions for improving the accuracy of detecting such attacks through consistency checks. We found that integrating more information about vehicle behavior (history) could be beneficial. This integration was made possible by RNNs (LSTM, GRU), which can effectively capture long-term memory. These findings suggest that relying on relatively small fixed-length information, as used in the literature, limits the model's performance. This work can

be extended to cover messages other than CAMs that are vulnerable to these types of attacks, such as DENM and SPAT, and also to other information exchanged in various V2X applications. In such cases, it would involve the deployment of an AIDS implemented at the 5G NS level.

The contributions in Chapter 4 and 5, which are organized within Part II of this thesis, offer advanced attack detection and adaptive security measures. These enhancements aim to strengthen existing security protocols designed to safeguard availability and integrity, thereby aligning with our approach to addressing **RQI**.

Chapter 5 also illustrated our achievement in minimizing computational cost in contrast to existing works in the literature when it comes to addressing AIDS for message forgery attacks in the context of V2X. We reached this conclusion through theoretical complexity analysis, an approach that is often overlooked in favor of simulations these days. This low complexity is achieved by utilizing RNN in an auto-regressive manner, in contrast to existing approaches, which analyze a fixed-length sequence entirely at each step. This contribution raises awareness about optimization and emphasizes the importance of theoretical complexity analysis, when dealing with IDSs and particularly latency-sensitive environments. This chapter's contribution aligns with addressing **RQII** by enabling efficient inference and attack detection for NN-based AIDSs.

In Chapter 6, we addressed a challenge in current Network Intrusion Detection Systems (NIDS) approaches known as flow-based NIDS, which relies on network flow termination conditions. We explained the drawbacks of relying on such approaches, as they can enable attackers to transmit more malicious network packets and have more time to conduct potential cyber attacks on the network. As a result, we achieved a high level of effectiveness in detection accuracy. More importantly, we demonstrated the capabilities of our model in early detection. Furthermore, we showcased the effectiveness of attention mechanisms, which are state-of-the-art mechanisms in NNs and work well with RNNs like LSTM and GRU. We find that our approach advances research towards early-stage detection of network intrusions, a critical need, especially in the era of 5G and beyond networks with ubiquitous support for NSs and services. The contribution in this chapter have provided valuable insights into **RQII** by highlighting the significant potential of modern architectures in early-stage network intrusion detection. On the other hand, we believe that analyzing network intrusions by examining network packet headers, as we did in our approach, provides a more standardized method for comparing NIDSs research works in academia. This is in contrast to commonly used aggregated flows, where the definition of aggregated features may vary between different approaches or datasets.

Chapter 7 is dedicated to responding to **RQIII**, which investigates the use of FL for the development of NIDSs in multi NSs 5G networks. This research is based on NNs while ensuring the privacy of data within each NS. In this chapter, we addressed the limitations of current mechanisms and aggregation methods, particularly in cases of statistical heterogeneity. Such heterogeneity is common in 5G networks, reflecting the diversity of data that can be gathered from NSs. Our findings indicate that approaches like FedAvg, which are widely used in the literature, fail to converge in situations involving statistical heterogeneity. Furthermore, we observed that FedPer, which is based on transfer learning, tends to replicate local models and overfit the local slice data, thereby diminishing its generalization potential. Among the various FL approaches we explored, SCAFFOLD emerged as the only global model that consistently achieved stable convergence across different NSs and outperformed other FL methods.

However, it is essential to acknowledge that data-driven model-based NIDSs, in contrast to signature-based systems where signatures are shared among NSs owners without exposing any privacy data, present a significant challenge. Therefore, the sharing of knowledge in model-based data-driven IDS remains a significant and unresolved challenge. Fortunately, SCAFFOLD offers promising potential in this regard, thanks to its control mechanisms. The findings and results of our experimental studies are intended to raise awareness about the utility of FL, particularly in the context of IDS models.

In conclusion, this thesis has presented a pathway to address the proposed challenges related to network security in the context of 5G-V2X, and more broadly, within the framework of 5G networks where services are deployed in the form of NSs. Our research work encompasses host/application and network IDS, all of which leverage AI and DL techniques. Our contributions have shed light on the effectiveness and efficiency of our models, highlighting the advantages of transitioning to AI-based models for such tasks. All of this work has been aimed at attempting to solve the research questions posed in **RQI**,**RQII** and **RQIII**, thereby providing a response to the main research questions (**RQ**).

Limitations and Future Research

We can identify a common limitation in our research works and current literature. At this stage, these approaches are considered prototypes intended to demonstrate the potential of AI/DL in 5G networks. However, for models that are commercially viable and ready for deployment, a substantial amount of realistic data is required. Obtaining such data from commercial network operators in academia is challenging due to regulations and the sensitive nature of the data. As a result, we have turned to open-source datasets generated using testbeds to simulate the behavior of network users. While these datasets are useful, they do not perfectly replicate the characteristics of a realistic 5G dataset. To address these current limitations and fulfill the need for a realistic and up-to-date open-source dataset in this context, we have developed a testbed named *DRIVE-B5G* [Dja+22d] that emulates a 5G-V2X environment. This testbed enables to generate data and validate approaches, and we plan to utilize it in our future works.

Zero-day attacks remain a challenge in DL-based IDSs and are a well-known problem that several research works are addressing. These attacks involve detecting new attack types that were not previously known, often referred to as 'unknown unknowns'. Unfortunately, our research has shown that FL is currently struggling with sharing knowledge to classify the 'known unknowns', i.e., it faces difficulties in handling attacks that are known in one network slice, but not in another. We believe that the current limitations of FL to enable cognitive knowledge in realistic scenarios are not limited to cybersecurity applications, but may be present in all areas intended to rely on FL in 5G and beyond networks. Therefore, we argue that further studies are needed to improve FL aggregation approaches, which is an interesting research area for the future. One promising approach is FL aggregation based on control systems like SCAFFOLD.

Regarding improvements in our work, we have not yet tackled the aspect of prevention in the context of monitoring network intrusions, which comes after the detection phase. In preventing network intrusions, approaches extend beyond simply banning malicious nodes from the network.

In the context of message forgery attacks in SL communication, it becomes necessary to address false position messages to prevent road incidents. To achieve this, we propose an avenue for exploration: the correction of altered messages. This concept takes inspiration from computer vision algorithms, where NNs models can rectify corrupted portions of an image. A similar approach could be applicable in our context.

In our future work, we also plan to delve into the monitoring and orchestration of NIDSs within 5G NS. These NIDSs are deployed as Virtual Network Functions following the principles of NFV. Our objective is to optimize their placement and replication. These improvements will provide more comprehensive responses to our research questions.

The research work and the directions for future work are in alignment with the new vision of cellular networks, including 5G and beyond networks, such as the Zero touch network & Service Management (ZSM)[2] initiative proposed by ETSI. The aim of this initiative is to design fully autonomous networks capable of monitoring and optimizing NSs without requiring human intervention. This vision predicts the widespread deployment of 5G NSs in various sectors, including V2X, which will lead to an increase in the overall complexity of network management and orchestration.

In this visionary paradigm, it becomes increasingly evident that AI will serve as the cornerstone for virtually every facet. However, it is imperative to acknowledge that, from an attacker standpoint, the rise of AI also portends a transformation in the strategies employed by adversaries. Attackers may use AI to counter devised IDS and discover new vulnerabilities and attack vectors. We believe that in the future, there will be an ongoing game between AI agents, akin to a "whack-a-mole" game, where humans play a role in continually proposing new AI architectures and mechanisms to counteract attackers.

[3]

[2] www.etsi.org/technologies/zero-touch-network-service-management
[3] Source: istockphoto.com

Appendix A

Complexity formulas (Chapter 5)

This appendix provides additional information on the computational calculations discussed in Section 5.6. It is assumed that scalar addition, multiplication, as well as calculating the exponential and hyperbolic tangent functions, are all considered primitive operations belonging to $\mathcal{O}(1)$.

The proposed model architecture consists of three parts: the projector, the RNN cell (GRU or LSTM), and the decoder. Therefore, the computational complexity of the proposed model is the sum of the computational complexity of each part, named $\text{Complexity}_{\text{projector}}$, $\text{Complexity}_{\text{lstm/gru}}$, $\text{Complexity}_{\text{decoder}}$.

Projector complexity

The projector performs two computations (refer to Equation (5.5.1)):

- The product of two matrices with shapes $(d_{in}, ||x_{t_t}||)$ and $(||x_{t_t}||,)$ respectively. The computational complexity of this operation is $d_{in} * (2 * ||x_{t_t}|| - 1)$.
- An element-wise vector sum of length d_{in}, which requires d_{in} operations to be performed.

The final computational complexity of the projector is:

$$\text{Complexity}_{\text{projector}} = 2 * d_{in} * ||x_{t_t}|| \tag{A.0.1}$$

Decoder complexity

The computational complexity of computing the output vector (indicated in Equation (2.3.5)) with the $f()$ function being a linear function can be expressed as:

$$\text{Complexity}_{\text{decoder}} = 2 * d_{out} * k \tag{A.0.2}$$

RNN Complexity

The equation (2.3.4), has the form of $f(W * x_t + b_x + V * h_{t-1} + b_h)$, where W and V have shapes (d_{in}, d_{out}) and (d_{out}, d_{out}), respectively, and b has shape $(d_{out},)$. In the case of a multi-layer RNN,

the shape of W becomes (d_{out}, d_{out}). By taking the complexity of f as d_{out}, the complexity of Equation (2.3.4) is:

$$\begin{aligned}\text{Complexity}_{\text{rnn}} =& 2d_{out}[d_{out} + d_{in} + 1]+ \\ & (m-1)[2d_{out}(2d_{out}+1)]\end{aligned} \tag{A.0.3}$$

LSTM complexity Equations (2.3.6), (2.3.7), (2.3.8), and (2.3.9) have the same format as Equation (2.3.4). Therefore, the computational complexity equals $\text{Complexity}_{\text{rnn}}$. In Equation 2.3.10, two vector products and a vector sum are performed, where the vectors have a length of d_{out}. Hence, the computational complexity of this equation is $3d_{out}$. Similarly, in Equation 2.3.11, a product and a tanh function are performed. Thus, the computational complexity of this equation is $2 * d_{out}$. The complexity of LSTM is then:

$$\begin{aligned}\text{Complexity}_{\text{lstm}} =& 4 * \text{Complexity}_{\text{rnn}}+ \\ & m * (3 * d_{out})+ \\ & m * (2 * d_{out})\end{aligned} \tag{A.0.4}$$

GRU complexity Equations (2.3.12) and (2.3.13), similarly to Equation (2.3.4) have complexity of $\text{Complexity}_{\text{rnn}}$. The complexity of Equation (2.3.14) is expressed as $d_{out}(2 * d_{in} + 2 * d_{out} + 3) + (m-1)[d_{out} * (4d_{out} + 3)]$, and the complexity of Equation (2.3.15) is $m * (4 * d_{out})$. The complexity of GRU is then:

$$\begin{aligned}\text{Complexity}_{\text{gru}} =& 2 * \text{Complexity}_{\text{rnn}}+ \\ & d_{out}(2 * d_{in} + 2 * d_{out} + 3)+ \\ & (m-1)[d_{out} * (4d_{out} + 3)]+ \\ & m * 4 * d_{out}\end{aligned} \tag{A.0.5}$$

Glossary

C-I-A The CIA proprieties, known as security triad, ensures that data and resources can only be viewed, modified, and accessed by authorized parties. xiii, 1, 25

dropout A regularization technique used in NNs to prevent overfitting by randomly deactivating a fraction of neurons during each training iteration. 72, 95, 96

Fuzzy Set A concept that represents a set where each element has a degree of membership, ranging between 0 and 1. 18

Loss A mathematical function ($\mathcal{L}()$) that quantifies the error between the model outputed value and the actual target value. This function is intended to be minimized during the training of NN models. xi, xii, xxi, 23, 54–59, 75, 77, 87, 97, 99–103, 113

Network Flow A sequence of exchanged network packets between a source host and a destination. xi, 5, 29, 30, 65–70, 72, 78, 82, 86, 92, 93

Network Intrusion or cyberattack, represents malicious activity that attempt to compromise these C-I-A properties. 25, 26, 28, 31

Network Slicing A technology that allows the creation of logical network partitions (known as Network Slices) with specific network capabilities and characteristics to serve and support particular services. 11–16

Optimiser Algorithm used to minimize the Loss function during the training process by adjusting the model's parameters, BP is commonly used as an optimization algorithm. 54, 56, 77, 97

Z score A statistical calculation used to determine how many standard deviations a particular data point is from the mean of a dataset. 71

Extended Glossary

Accuracy A metric that measures the proportion of correct predictions or classifications made by a model out of the total predictions or classifications. $= \frac{TP+TN}{TP+TN+FP+FN}$. xii, 23, 75, 87, 92, 99–104

Cross-Entropy A loss function that measures the difference between predicted and actual probabilities using the formula $\mathcal{L} = -\sum_i^c (y^i * log(\hat{y}^i))$ where c is the number of classes. 23, 54, 56, 75, 77, 95

F1Score A metric that combines both precision (PPV) and recall (TPR) to provide a single value that balances the trade-off between the two. $= \frac{PPV*TPR}{PPV*TPR}$. xi, 23, 58–60, 103, 105

FN A false negative is an outcome where the model incorrectly predicts a specific class as negative when it is actually positive among multiple classes. 23

FNR The ratio of false negative predictions among all actual positive instances $= \frac{FN}{TP+FN}$. 23

FP A false positive is an outcome where the model incorrectly predicts a specific class as positive when it is actually negative among multiple classes. 23

FPR or fall out. The ratio of false positive predictions among all actual negative instances $= \frac{FP}{TN+FP}$. 23

MAE A loss function that averages the absolute value of the errors, $\mathcal{L} = \frac{1}{B*d_{out}} \sum_1^B \sum |y - \hat{y}|$ where B is the batch size. 23

MSE A loss function that averages the squares of the errors, $\mathcal{L} = \frac{1}{B*d_{out}} \sum_1^B \sum (y - \hat{y})^2$ where B is the batch size. 23

NPV The proportion of true negative predictions among all negative predictions. $= \frac{TN}{TN+FN}$. 23

PPV or precision. The proportion of true positive predictions among all positive predictions. $= \frac{TP}{TP+FP}$. 23

relu is an activation function used in NNs. The function takes an input and outputs the input itself if it is positive, and zero otherwise. Its formula is calculated as follows: $\text{relu}(\sigma_j) = \max(0, \sigma_j)$, where σ_j is the output of perceptron j. 20

sigmoid is a non-linear mathematical function used as an activation function in NNs. Its formula is calculated as follows: $\tanh(\sigma_j) = \frac{1}{1+e^{-\sigma_j}}$, where σ_j is the output of perceptron j. 20

tanh known as the hyperbolic tangent function, it is a mathematical function used as an activation function in NNs. Its formula is calculated as follows: $\tanh(\sigma_j) = \frac{e_j^{\sigma} - e^{-\sigma_j}}{e_j^{\sigma} + e^{-\sigma_j}}$, where σ_j is the output of perceptron j. 20, 61

TN A true negative is an outcome where the model correctly predicts all other classes as negative for a specific class. 23

TNR or specificity. The proportion of true negative predictions among all actual negative instances $= \frac{\text{TN}}{\text{TN+FP}}$. 23

TP A true positive is an outcome where the model correctly predicts a specific class as positive among multiple classes. 23

TPR or recall, sensitivity. The proportion of true positive predictions among all actual positive instances $= \frac{\text{TP}}{\text{TP+FN}}$. 23

Bibliography

[Zad65] L.A. Zadeh. "Fuzzy sets". In: *Information and Control* 8.3 (June 1965), pp. 338–353. DOI: 10.1016/s0019-9958(65)90241-x. URL: https://doi.org/10.1016/s0019-9958(65)90241-x.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-term Memory". In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[Bac99] Rebecca Bace. *Intrusion Detection*. English. Paperback. Sams Publishing, Dec. 22, 1999, p. 368. ISBN: 978-1578701858.

[Kur00] Ray Kurzweil. *The Age of Spiritual Machines: When Computers Exceed Human Intelligence*. English. Paperback. Penguin Books, Jan. 1, 2000, p. 388. ISBN: 978-0140282023.

[KT03] Christopher Kruegel and Thomas Toth. "Using Decision Trees to Improve Signature-Based Intrusion Detection". In: *Recent Advances in Intrusion Detection*. Ed. by Giovanni Vigna, Christopher Kruegel, and Erland Jonsson. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 173–191.

[Xu+04] Wenyuan Xu et al. "Channel Surfing and Spatial Retreats: Defenses against Wireless Denial of Service". In: *Proceedings of the 3rd ACM Workshop on Wireless Security*. WiSe '04. Philadelphia, PA, USA: Association for Computing Machinery, 2004, pp. 80–89. ISBN: 158113925X. DOI: 10.1145/1023646.1023661. URL: https://doi.org/10.1145/1023646.1023661.

[Fuc05] Andreas Fuchsberger. "Intrusion Detection Systems and Intrusion Prevention Systems". In: *Information Security Technical Report* 10.3 (Jan. 2005), pp. 134–139. DOI: 10.1016/j.istr.2005.08.001. URL: https://doi.org/10.1016/j.istr.2005.08.001.

[DM08] Roberto Di Pietro and Luigi V Mancini. *Intrusion detection systems*. Vol. 38. Springer Science & Business Media, 2008.

[GLF08] Moses Garuba, Chunmei Liu, and Duane Fraites. "Intrusion Techniques: Comparative Study of Network Intrusion Detection Systems". In: *Fifth International Conference on Information Technology: New Generations (itng 2008)*. 2008, pp. 592–598. DOI: 10.1109/ITNG.2008.231.

[Rut08] Leszek Rutkowski. *Computational Intelligence*. Springer Berlin Heidelberg, 2008. DOI: 10.1007/978-3-540-76288-1. URL: https://doi.org/10.1007/978-3-540-76288-1.

[RN09] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. English. Hardcover. Pearson, Dec. 1, 2009, p. 1152. ISBN: 978-0136042594.

[Tav+09] Mahbod Tavallaee et al. "A detailed analysis of the KDD CUP 99 data set". In: *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*. 2009, pp. 1–6. DOI: 10.1109/CISDA.2009.5356528.

[3GP11] 3GPP. *Release 17 Description*. Technical report (TR) 21.917, 2021. 3rd Generation Partnership Project (3GPP), 2011.

[AMA+11] Hashem Alaidaros, Massudi Mahmuddin, Ali Al Mazari, et al. "An overview of flow-based and packet-based intrusion detection performance in high speed networks". In: *Proceedings of the International Arab Conference on Information Technology*. 2011, pp. 1–9.

[Li12] Yunxin (Jeff) Li. "An Overview of the DSRC/WAVE Technology". In: *Quality, Reliability, Security and Robustness in Heterogeneous Networks*. Ed. by Xi Zhang and Daji Qiao. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 544–558.

[Shi+12] Ali Shiravi et al. "Toward developing a systematic approach to generate benchmark datasets for intrusion detection". In: *Computers & Security* 31.3 (2012), pp. 357–374. ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2011.12.012. URL: https://www.sciencedirect.com/science/article/pii/S0167404811001672.

[Sin+13] Harpreet Singh et al. "Real-Life Applications of Fuzzy Logic". In: *Advances in Fuzzy Systems* 2013 (2013), pp. 1–3. DOI: 10.1155/2013/581879.

[Chu+14] Junyoung Chung et al. *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. 2014. arXiv: 1412.3555 [cs.NE].

[Hof+14] Rick Hofstede et al. "Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX". In: *IEEE Communications Surveys & Tutorials* 16.4 (2014), pp. 2037–2064. DOI: 10.1109/COMST.2014.2321898.

[SSB14] Haşim Sak, Andrew Senior, and Françoise Beaufays. *Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition*. 2014. arXiv: 1402.1128 [cs.NE].

[MS15] Nour Moustafa and Jill Slay. "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)". In: *2015 Military Communications and Information Systems Conference (MilCIS)*. 2015, pp. 1–6. DOI: 10.1109/MilCIS.2015.7348942.

[BCB16] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2016. arXiv: 1409.0473 [cs.CL].

[BM16] Manjula C. Belavagi and Balachandra Muniyal. "Performance Evaluation of Supervised Machine Learning Algorithms for Intrusion Detection". In: *Procedia Computer Science* 89 (2016). Twelfth International Conference on Communication Networks, ICCN 2016, August 19– 21, 2016, Bangalore, India Twelfth International Conference on Data Mining and Warehousing, ICDMW 2016, August 19-21, 2016, Bangalore, India Twelfth International Conference on Image and Signal Processing, ICISP 2016, August 19-21, 2016, Bangalore, India, pp. 117–123. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2016.06.016. URL: https://www.sciencedirect.com/science/article/pii/S187705091631081X.

[HSB16] Yasir Hamid, M Sugumaran, and V Balasaraswathi. "IDS Using Machine Learning - Current State of Art and Future Directions". In: *British Journal of Applied Science Technology* 15.3 (Jan. 2016), pp. 1–22. DOI: 10.9734/bjast/2016/23668. URL: https://doi.org/10.9734/bjast/2016/23668.

[McM+16] H. Brendan McMahan et al. "Communication-Efficient Learning of Deep Networks from Decentralized Data". In: (2016). DOI: 10.48550/ARXIV.1602.05629. URL: https://arxiv.org/abs/1602.05629.

[16] *Study on the Deployment of C-ITS in Europe: Final Report*. Tech. rep. ED60721. DG MOVE, 2016.

[3GP17] 3GPP. *Release 14 Description*. Technical report (TR) 21.914, 2017. 3rd Generation Partnership Project (3GPP), 2017.

[AMK17] Akashdeep, Ishfaq Manzoor, and Neeraj Kumar. "A feature reduced intrusion detection system using ANN classifier". In: *Expert Systems with Applications* 88 (2017), pp. 249–257. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2017.07.005. URL: https://www.sciencedirect.com/science/article/pii/S0957417417304748.

[Cec+17] Giammarco Cecchini et al. "LTEV2Vsim: An LTE-V2V simulator for the investigation of resource allocation for cooperative awareness". In: *2017 5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*. 2017, pp. 80–85. DOI: 10.1109/MTITS.2017.8005625.

[MG17] Rafael Molina-Masegosa and Javier Gozalvez. "LTE-V for Sidelink 5G V2X Vehicular Communications: A New 5G Technology for Short-Range Vehicle-to-Everything Communications". In: *IEEE Vehicular Technology Magazine* 12.4 (2017), pp. 30–39. DOI: 10.1109/MVT.2017.2752798.

[Ric17] Kevin Knight & Elaine Rich. *Artificial Intelligence*. English. Paperback. Mc Graw Hill India, July 25, 2017. ISBN: 978-0070087705.

[17] *System architecture for the 5G System (5GS)*. Technical Specification (TS) 23.501 V1.4.0. 3rd Generation Partnership Project (3GPP), 2017.

[USB17] Muhammad Fahad Umer, Muhammad Sher, and Yaxin Bi. "Flow-based intrusion detection: Techniques and challenges". In: *Computers & Security* 70 (2017), pp. 238–254. ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2017.05.009. URL: https://www.sciencedirect.com/science/article/pii/S0167404817301165.

[VB17] Paul Voigt and Axel von dem Bussche. *The EU general data protection regulation (GDPR)*. en. 1st ed. Basel, Switzerland: Springer International Publishing, Aug. 2017.

[AKG18] Luis F. Abanto-Leon, Arie Koppelaar, and Sonia Heemstra de Groot. *Enhanced C-V2X Mode-4 Subchannel Selection*. 2018. arXiv: 1807.04819 [eess.SP].

[HLK18] Rens W. van der Heijden, Thomas Lukaseder, and Frank Kargl. *VeReMi: A Dataset for Comparable Evaluation of Misbehavior Detection in VANETs*. 2018. arXiv: 1804.06701 [cs.CR].

[Kor+18] Nickolaos Koroniotis et al. *Towards the Development of Realistic Botnet Dataset in the Internet of Things for Network Forensic Analytics: Bot-IoT Dataset*. 2018. arXiv: 1811.00701 [cs.CR].

[Li+18] Tian Li et al. *Federated Optimization in Heterogeneous Networks*. 2018. DOI: 10.48550/ARXIV.1812.06127. URL: https://arxiv.org/abs/1812.06127.

[Liu+18] Ming Liu et al. "Host-Based Intrusion Detection System with System Calls: Review and Future Trends". In: *ACM Comput. Surv.* 51.5 (Nov. 2018). ISSN: 0360-0300. DOI: 10.1145/3214304. URL: https://doi.org/10.1145/3214304.

[Mei+18] Yair Meidan et al. *N-BaIoT: Network-based Detection of IoT Botnet Attacks Using Deep Autoencoders*. 2018.

[PP18] CHARLES P. PFLEEGER and SHARI LAWRENCE PFLEEGER. *Security In Computing: 5Th Edition*. English. Paperback. PEARSON INDIA, 2018. ISBN: 978-9352866533.

[18] *Rapport du groupe de travail : technologies de communication pour les STI coopératifs*. Ministère de la transition écologique, France, 2018. URL: https://www.ecologie.gouv.fr/sites/default/files/Rapport%20GT%20technologies%20STI-vfin.pdf.

[Sal+18] Hojjat Salehinejad et al. *Recent Advances in Recurrent Neural Networks*. 2018. arXiv: 1801.01078 [cs.NE].

[SLG18] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization". In: *Proceedings of the 4th International Conference on Information Systems Security and Privacy*. SCITEPRESS - Science and Technology Publications, 2018. DOI: 10.5220/0006639801080116. URL: https://doi.org/10.5220/0006639801080116.

[Sho+18] Nathan Shone et al. "A Deep Learning Approach to Network Intrusion Detection". In: *IEEE Transactions on Emerging Topics in Computational Intelligence* 2.1 (2018), pp. 41–50. DOI: 10.1109/TETCI.2017.2772792.

[Wen18] Lilian Weng. "Attention? Attention!" In: *lilianweng.github.io* (2018). URL: https://lilianweng.github.io/posts/2018-06-24-attention/.

[3GP19] 3GPP. *Study on NR Vehicle-to-Everything (Release 16)*. Technical report (TR) 38.885, 2019. 3rd Generation Partnership Project (3GPP), 2019.

[ASJ19] Aljawharah Alnasser, Hongjian Sun, and Jing Jiang. "Cyber security challenges and solutions for V2X communications: A survey". In: *Computer Networks* 151 (2019), pp. 52–67. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2018.12.018. URL: https://www.sciencedirect.com/science/article/pii/S1389128618306157.

[Ari+19] Manoj Ghuhan Arivazhagan et al. *Federated Learning with Personalization Layers*. 2019. DOI: 10.48550/ARXIV.1912.00818. URL: https://arxiv.org/abs/1912.00818.

[Ben+19] Mohammed Anis Benblidia et al. "Ranking Fog nodes for Tasks Scheduling in Fog-Cloud Environments: A Fuzzy Logic Approach". In: *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*. 2019, pp. 1451–1457. DOI: 10.1109/IWCMC.2019.8766437.

[Ela+19] Salah Eddine Elayoubi et al. "5G RAN Slicing for Verticals: Enablers and Challenges". In: *IEEE Communications Magazine* 57.1 (2019), pp. 28–34. DOI: 10.1109/MCOM.2018.1701319.

[Hei+19] Rens Wouter van der Heijden et al. "Survey on Misbehavior Detection in Cooperative Intelligent Transportation Systems". In: *IEEE Communications Surveys & Tutorials* 21.1 (2019), pp. 779–811. DOI: 10.1109/COMST.2018.2873088.

[IA19] Ahmed Iqbal and Shabib Aftab. "A Feed-Forward and Pattern Recognition ANN Model for Network Intrusion Detection". In: *International Journal of Computer Network and Information Security* 11.4 (Apr. 2019), pp. 19–25. DOI: 10.5815/ijcnis.2019.04.03. URL: https://doi.org/10.5815/ijcnis.2019.04.03.

[JCK19] So-Yi Jung, Hye-Rim Cheon, and Jae-Hyun Kim. "Reducing Consecutive Collisions in Sensing Based Semi Persistent Scheduling for Cellular-V2X". In: *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*. 2019, pp. 1–5. DOI: 10.1109/VTCFall.2019.8891226.

[Kan+19] Hyunjae Kang et al. *IoT network intrusion dataset*. 2019. DOI: 10.21227/q70p-q449. URL: https://dx.doi.org/10.21227/q70p-q449.

[Kar+19] Sai Praneeth Karimireddy et al. *SCAFFOLD: Stochastic Controlled Averaging for Federated Learning*. 2019. DOI: 10.48550/ARXIV.1910.06378. URL: https://arxiv.org/abs/1910.06378.

[Pas+19] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[Rin+19] Markus Ring et al. "A survey of network-based intrusion detection data sets". In: *Computers & Security* 86 (2019), pp. 147–167. ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2019.06.005. URL: https://www.sciencedirect.com/science/article/pii/S016740481930118X.

[Roy+19] Abhijit Guha Roy et al. *BrainTorrent: A Peer-to-Peer Environment for Decentralized Federated Learning*. 2019. DOI: 10.48550/ARXIV.1905.06731. URL: https://arxiv.org/abs/1905.06731.

[SM19] Danish Sattar and Ashraf Matrawy. "Towards Secure Slicing: Using Slice Isolation to Mitigate DDoS Attacks on 5G Core Network Slices". In: *2019 IEEE Conference on Communications and Network Security (CNS)*. 2019, pp. 82–90. DOI: 10.1109/CNS.2019.8802852.

[Sha+19] Iman Sharafaldin et al. "Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy". In: *2019 International Carnahan Conference on Security Technology (ICCST)*. IEEE, Oct. 2019. DOI: 10.1109/ccst.2019.8888419. URL: https://doi.org/10.1109/ccst.2019.8888419.

[Sin+19] Pranav Kumar Singh et al. "Machine Learning Based Approach to Detect Position Falsification Attack in VANETs". In: *Security and Privacy*. Ed. by Sukumar Nandi et al. Singapore: Springer Singapore, 2019, pp. 166–178.

[Tan+19a] Mengxuan Tan et al. "A Neural Attention Model for Real-Time Network Intrusion Detection". In: *2019 IEEE 44th Conference on Local Computer Networks (LCN)*. 2019, pp. 291–299. DOI: 10.1109/LCN44214.2019.8990890.

[Tan+19b] Tuan Anh Tang et al. "Intrusion Detection in SDN-Based Networks: Deep Recurrent Neural Network Approach". In: *Deep Learning Applications for Cyber Security*. Springer International Publishing, 2019, pp. 175–195. DOI: 10.1007/978-3-030-13057-2_8. URL: https://doi.org/10.1007/978-3-030-13057-2_8.

[Zha19] Shunliang Zhang. "An Overview of Network Slicing for 5G". In: *IEEE Wireless Communications* 26.3 (2019), pp. 111–117. DOI: 10.1109/MWC.2019.1800234.

[3GP20] 3GPP. *Architecture enhancements for 5G System (5GS) to support Vehicle-to-Everything (V2X) services*. Technical Specification (TS) 23.287, 2020. 3rd Generation Partnership Project (3GPP), 2020.

[Che+20a] Shanzhi Chen et al. "Vision, Requirements, and Technology Trend of 6G: How to Tackle the Challenges of System Coverage, Capacity, User Data-Rate and Movement Speed". In: *IEEE Wireless Communications* PP (Feb. 2020), pp. 1–11. DOI: 10.1109/MWC.001.1900333.

[Che+20b] Zhuo Chen et al. "Intrusion Detection for Wireless Edge Networks Based on Federated Learning". In: *IEEE Access* 8 (2020), pp. 217463–217472. DOI: 10.1109/ACCESS.2020.3041793.

[Cho+20] Mostafa Zaman Chowdhury et al. "6G Wireless Communication Systems: Applications, Requirements, Technologies, Challenges, and Research Directions". In: *IEEE Open Journal of the Communications Society* 1 (2020), pp. 957–975. DOI: 10.1109/OJCOMS.2020.3010270.

[Fan+20] Yulin Fan et al. "IoTDefender: A Federated Transfer Learning Intrusion Detection Framework for 5G IoT". In: *2020 IEEE 14th International Conference on Big Data Science and Engineering (BigDataSE)*. 2020, pp. 88–95. DOI: 10.1109/BigDataSE50710.2020.00020.

[GC20] Amrita Ghosal and Mauro Conti. "Security issues and challenges in V2X: A Survey". In: *Computer Networks* 169 (2020), p. 107093. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2019.107093. URL: https://www.sciencedirect.com/science/article/pii/S1389128619305857.

[Hin+20] Hanan Hindy et al. *MQTT Internet of Things Intrusion Detection Dataset*. 2020. DOI: 10.21227/BHXY-EP04. URL: https://ieee-dataport.org/open-access/mqtt-internet-things-intrusion-detection-dataset.

[Jia+20] Kaiyuan Jiang et al. "Network Intrusion Detection Combined Hybrid Sampling With Deep Hierarchical Network". In: *IEEE Access* 8 (2020), pp. 32464–32476. DOI: 10.1109/ACCESS.2020.2973730.

[Li+20] Tian Li et al. "Federated Learning: Challenges, Methods, and Future Directions". In: *IEEE Signal Processing Magazine* 37.3 (2020), pp. 50–60. DOI: 10.1109/MSP.2020.2975749.

[Lu+20] Rongxing Lu et al. "5G Vehicle-to-Everything Services: Gearing Up for Security and Privacy". In: *Proceedings of the IEEE* 108.2 (2020), pp. 373–389. DOI: 10.1109/JPROC.2019.2948302.

[MK20] Mohammad Masdari and Hemn Khezri. "A survey and taxonomy of the fuzzy signature-based Intrusion Detection Systems". In: *Applied Soft Computing* 92 (July 2020), p. 106301. DOI: 10.1016/j.asoc.2020.106301. URL: https://doi.org/10.1016/j.asoc.2020.106301.

[MA20] Calabrese Michael and Nasr Amir. *The 5.9 GHz Band*. Accessed: 20-6-2023. OPEN TECHNOLOGY INSTITUTE, 2020. URL: https://www.newamerica.org/oti/reports/59-ghz-band/.

[NAS20] Sanchit Nayyar, Sneha Arora, and Maninder Singh. "Recurrent Neural Network Based Intrusion Detection System". In: *2020 International Conference on Communication and Signal Processing (ICCSP)*. 2020, pp. 0136–0140. DOI: 10.1109/ICCSP48568.2020.9182099.

[NGU+20] Huong NGUYEN-MINH et al. "Channel Surfing to Mitigate against Jamming Attacks on Safety Applications in Vehicular Networks". In: *2020 RIVF International Conference on Computing and Communication Technologies (RIVF)*. 2020, pp. 1–5. DOI: 10.1109/RIVF48685.2020.9140788.

[Rah+20] Sawsan Abdul Rahman et al. "Internet of Things Intrusion Detection: Centralized, On-Device, or Federated Learning?" In: *IEEE Network* 34.6 (2020), pp. 310–317. DOI: 10.1109/MNET.011.2000286.

[20] *Release description; Release 16*. Technical report (TR) 21.916, 2020. 3rd Generation Partnership Project (3GPP), 2020.

[El-+20] Zeinab El-Rewini et al. "Cybersecurity challenges in vehicular communications". In: *Vehicular Communications* 23 (2020), p. 100214. ISSN: 2214-2096. DOI: https://doi.org/10.1016/j.vehcom.2019.100214. URL: https://www.sciencedirect.com/science/article/pii/S221420961930261X.

[Sar+20] Iqbal H. Sarker et al. "IntruDTree: A Machine Learning Based Cyber Security Intrusion Detection Model". In: *Symmetry* 12.5 (2020). ISSN: 2073-8994. URL: https://www.mdpi.com/2073-8994/12/5/754.

[Sun+20] Pengfei Sun et al. "DL-IDS: Extracting features using CNN-LSTM hybrid network for intrusion detection system". In: *Security and Communication Networks* 2020 (Aug. 2020), pp. 1–11. DOI: 10.1155/2020/8890306.

[Tam+20] Nouredine Tamani et al. "On Link Stability Metric and Fuzzy Quantification for Service Selection in Mobile Vehicular Cloud". In: *IEEE Transactions on Intelligent Transportation Systems* 21.5 (2020), pp. 2050–2062. DOI: 10.1109/TITS.2019.2911860.

[WS20] Philip Wendland and Guenter Schaefer. "Feedback-Based Hidden-Terminal Mitigation for Distributed Scheduling in Cellular V2X". In: *2020 IFIP Networking Conference (Networking)*. 2020, pp. 549–553.

[Zİ20] Sultan Zavrak and Murat İskefiyeli. "Anomaly-Based Intrusion Detection From Network Flow Features Using Variational Autoencoder". In: *IEEE Access* 8 (2020), pp. 108346–108358. DOI: 10.1109/ACCESS.2020.3001350.

[ADC21] Ahmad Alalewi, Iyad Dayoub, and Soumaya Cherkaoui. "On 5G-V2X Use Cases and Enabling Technologies: A Comprehensive Survey". In: *IEEE Access* 9 (2021), pp. 107710–107737. DOI: 10.1109/ACCESS.2021.3100472.

[All+21] Tejasvi Alladi et al. "DeepADV: A Deep Neural Network Framework for Anomaly Detection in VANETs". In: *IEEE Transactions on Vehicular Technology* 70.11 (2021), pp. 12013–12023. DOI: 10.1109/TVT.2021.3113807.

[Baz+21] Alessandro Bazzi et al. "On the Design of Sidelink for Cellular V2X: A Literature Review and Outlook for Future". In: *IEEE Access* 9 (2021), pp. 97953–97980. DOI: 10.1109/ACCESS.2021.3094161.

[CJ21] Dylan Chou and Meng Jiang. "A Survey on Data-driven Network Intrusion Detection". In: *ACM Computing Surveys* 54.9 (Oct. 2021), pp. 1–36. DOI: 10.1145/3472753. URL: https://doi.org/10.1145/3472753.

[DJJ21] Anutusha Dogra, Rakesh Kumar Jha, and Shubha Jain. "A Survey on Beyond 5G Network With the Advent of 6G: Architecture and Emerging Technologies". In: *IEEE Access* 9 (2021), pp. 67512–67547. DOI: 10.1109/ACCESS.2020.3031234.

[Fer+21] Mohamed Amine Ferrag et al. "Federated Deep Learning for Cyber Security in the Internet of Things: Concepts, Applications, and Experimental Analysis". In: *IEEE Access* 9 (2021), pp. 138509–138542. DOI: 10.1109/ACCESS.2021.3118642.

[Gar+21] Mario H. Castañeda Garcia et al. "A Tutorial on 5G NR V2X Communications". In: *IEEE Communications Surveys & Tutorials* 23.3 (2021), pp. 1972–2026. DOI: 10.1109/COMST.2021.3057017.

[Gro+21] Harsh Grover et al. "Edge Computing and Deep Learning Enabled Secure Multitier Network for Internet of Vehicles". In: *IEEE Internet of Things Journal* 8.19 (2021), pp. 14787–14796. DOI: 10.1109/JIOT.2021.3071362.

[Har+21] Mehdi Harounabadi et al. "V2X in 3GPP Standardization: NR Sidelink in Release-16 and Beyond". In: *IEEE Communications Standards Magazine* 5.1 (2021), pp. 12–21. DOI: 10.1109/MCOMSTD.001.2000070.

[Kha21] Muhammad Ashfaq Khan. "HCRNNIDS: Hybrid Convolutional Recurrent Neural Network-Based Network Intrusion Detection System". In: *Processes* 9.5 (2021). ISSN: 2227-9717. URL: https://www.mdpi.com/2227-9717/9/5/834.

[Lan+21] Jan Lansky et al. "Deep Learning-Based Intrusion Detection Systems: A Systematic Review". In: *IEEE Access* 9 (2021), pp. 101574–101599. DOI: 10.1109/ACCESS.2021.3097247.

[LL21] Xingqin Lin and Namyoon Lee, eds. *5G and Beyond*. Springer International Publishing, 2021. DOI: 10.1007/978-3-030-58197-8. URL: https://doi.org/10.1007/978-3-030-58197-8.

[Min+21] Byeongjun Min et al. "Network Anomaly Detection Using Memory-Augmented Deep Autoencoder". In: *IEEE Access* 9 (2021), pp. 104695–104706. DOI: 10.1109/ACCESS.2021.3100087.

[NA21] M. Z. Naser and Amir H. Alavi. "Error Metrics and Performance Fitness Indicators for Artificial Intelligence and Machine Learning in Engineering and Sciences". In: *Architecture, Structures and Construction* (Nov. 2021). DOI: 10.1007/s44150-021-00015-8. URL: https://doi.org/10.1007%5C%2Fs44150-021-00015-8.

[NGO21] Gianfranco Nencioni, Rosario G. Garroppo, and Ruxandra F. Olimid. *5G Multi-access Edge Computing: Security, Dependability, and Performance*. 2021. DOI: 10.48550/ARXIV.2107.13374. URL: https://arxiv.org/abs/2107.13374.

[NZY21] Zhaoyang Niu, Guoqiang Zhong, and Hui Yu. "A review on the attention mechanism of deep learning". In: *Neurocomputing* 452 (2021), pp. 48–62. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2021.03.091. URL: https://www.sciencedirect.com/science/article/pii/S092523122100477X.

[Rah+21] K. M. Jawadur Rahman et al. "Challenges, Applications and Design Aspects of Federated Learning: A Survey". In: *IEEE Access* 9 (2021), pp. 124682–124700. DOI: 10.1109/ACCESS.2021.3111118.

[RM21] Ondrej Rysavy and Petr Matousek. *Modbus Dataset for ICS Anomaly Detection*. 2021. DOI: 10.21227/e1bc-3w91. URL: https://dx.doi.org/10.21227/e1bc-3w91.

[SL21] Prinkle Sharma and Hong Liu. "A Machine-Learning-Based Data-Centric Misbehavior Detection Model for Internet of Vehicles". In: *IEEE Internet of Things Journal* 8.6 (2021), pp. 4991–4999. DOI: 10.1109/JIOT.2020.3035035.

[TM21] Thien Thi Thanh Le and Sangman Moh. "Comprehensive Survey of Radio Resource Allocation Schemes for 5G V2X Communications". In: *IEEE Access* 9 (2021), pp. 123117–123133. DOI: 10.1109/ACCESS.2021.3109894.

[Wah+21] Omar Abdel Wahab et al. "Federated Machine Learning: Survey, Multi-Level Classification, Desirable Criteria and Future Directions in Communication and Networking Systems". In: *IEEE Communications Surveys & Tutorials* 23.2 (2021), pp. 1342–1397. DOI: 10.1109/COMST.2021.3058573.

[WL21] Haomin Wang and Wei Li. "DDosTC: A Transformer-Based Network Attack Detection Hybrid Mechanism in SDN". In: *Sensors* 21.15 (2021). ISSN: 1424-8220. URL: https://www.mdpi.com/1424-8220/21/15/5047.

[YK21] Youngjoon Yoon and Hyogon Kim. "An Evasive Scheduling Enhancement Against Packet Dropping Attacks in C-V2X Communication". In: *IEEE Communications Letters* 25.2 (2021), pp. 392–396. DOI: 10.1109/LCOMM.2020.3030811.

[Zha+21] Tuo Zhang et al. "Federated Learning for Internet of Things". In: *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*. SenSys '21. Coimbra, Portugal: Association for Computing Machinery, 2021, pp. 413–419. ISBN: 9781450390972. DOI: 10.1145/3485730.3493444. URL: https://doi.org/10.1145/3485730.3493444.

[Dan+22] Sedeng Danba et al. "Toward Collaborative Intelligence in IoV Systems: Recent Advances and Open Issues". In: *Sensors* 22.18 (2022). ISSN: 1424-8220. DOI: 10.3390/s22186995. URL: https://www.mdpi.com/1424-8220/22/18/6995.

[Dja+22b] Taki Eddine Djaidja et al. "Adaptive Resource Reservation to Survive Against Adversarial Resource Selection Jamming Attacks in 5G NR-V2X Distributed Mode 2". In: *ICC 2022 - IEEE International Conference on Communications*. 2022, pp. 3406–3411. DOI: 10.1109/ICC45855.2022.9839023.

[Dja+22c] Taki Eddine Djaidja et al. "DRIVE-B5G: A Flexible and Scalable Platform Testbed for B5G-V2X Networks". In: *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*. 2022, pp. 2800–2805. DOI: 10.1109/GLOBECOM48099.2022.10001231.

[Edd22] *Transmission Control Protocol (TCP)*. Tech. rep. Aug. 2022. DOI: 10.17487/rfc9293. URL: https://doi.org/10.17487/rfc9293.

[Far+22] Yasir Ali Farrukh et al. "Payload-Byte: A Tool for Extracting and Labeling Packet Capture Files of Modern Network Intrusion Detection Datasets". In: *2022 IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT)*. 2022, pp. 58–67. DOI: 10.1109/BDCAT56447.2022.00015.

[GR22] Bimal Ghimire and Danda B. Rawat. "Recent Advances on Federated Learning for Cybersecurity and Cybersecurity for Federated Learning for Internet of Things". In: *IEEE Internet of Things Journal* 9.11 (2022), pp. 8229–8249. DOI: 10.1109/JIOT.2022.3150363.

[Has+22] Mehedi Hassan et al. "Intrusion Detection Using Payload Embeddings". In: *IEEE Access* 10 (2022), pp. 4015–4030. DOI: 10.1109/ACCESS.2021.3139835.

[HCT22] Hsiao-Yuan Hsu, Nai-Hsin Cheng, and Chun-Wei Tsai. "A Deep Learning-Based Integrated Algorithm for Misbehavior Detection System in VANETs". In: *Proceedings of the 2021 ACM International Conference on Intelligent Computing and Its Emerging Applications*. ACM ICEA '21. Jinan, China: Association for Computing Machinery, 2022, pp. 53–58. ISBN: 9781450391603. DOI: 10.1145/3491396.3506509. URL: https://doi.org/10.1145/3491396.3506509.

[Lav+22] Léo Lavaur et al. "The Evolution of Federated Learning-based Intrusion Detection and Mitigation: a Survey". In: *IEEE Transactions on Network and Service Management* (2022), pp. 1–1. DOI: 10.1109/TNSM.2022.3177512.

[Ma+22] Xiaodong Ma et al. "A state-of-the-art survey on solving non-IID data in Federated Learning". In: *Future Generation Computer Systems* 135 (2022), pp. 244–258. ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2022.05.003. URL: https://www.sciencedirect.com/science/article/pii/S0167739X22001686.

[22] *Management and orchestration; Concepts, use cases and requirements*. Technical Specification (TS) 28.530, V17.2.0. European Telecommunications Standards Institute (ETSI), 2022.

[ME22] Innocent Mbona and Jan H. P. Eloff. "Detecting Zero-Day Intrusion Attacks Using Semi-Supervised Machine Learning Approaches". In: *IEEE Access* 10 (2022), pp. 69822–69838. DOI: 10.1109/ACCESS.2022.3187116.

[Mot+22] Viraaji Mothukuri et al. "Federated-Learning-Based Anomaly Detection for IoT Security Attacks". In: *IEEE Internet of Things Journal* 9.4 (2022), pp. 2545–2554. DOI: 10.1109/JIOT.2021.3077803.

[Sam+22] Sehan Samarakoon et al. *5G-NIDD: A Comprehensive Network Intrusion Detection Dataset Generated over 5G Wireless Network*. 2022. arXiv: 2212.01298 [cs.CR].

[UM22] Imtiaz Ullah and Qusay H. Mahmoud. "Design and Development of RNN Anomaly Detection Model for IoT Networks". In: *IEEE Access* 10 (2022), pp. 62722–62750. DOI: 10.1109/ACCESS.2022.3176317.

[Wu+22] Zihan Wu et al. "RTIDS: A Robust Transformer-Based Approach for Intrusion Detection System". In: *IEEE Access* 10 (2022), pp. 64375–64387. DOI: 10.1109/ACCESS.2022.3182333.

[Yua+22] Yachao Yuan et al. "Insight of Anomaly Detection with NWDAF in 5G". In: *2022 International Conference on Computer, Information and Telecommunication Systems (CITS)*. 2022, pp. 1–6. DOI: 10.1109/CITS55221.2022.9832914.

[BB23] Ahmed Ramzi Bahlali and Abdelmalik Bachir. "Machine Learning Anomaly-Based Network Intrusion Detection: Experimental Evaluation". In: *Advanced Information Networking and Applications*. Springer International Publishing, 2023, pp. 392–403. DOI: 10.1007/978-3-031-28451-9_34. URL: https://doi.org/10.1007/978-3-031-28451-9_34.

[BE23] Abdelwahab Boualouache and Thomas Engel. "A Survey on Machine Learning-based Misbehavior Detection Systems for 5G and Beyond Vehicular Networks". In: *IEEE Communications Surveys & Tutorials* (2023), pp. 1–1. DOI: 10.1109/COMST.2023.3236448.

[23a] *Cisco NetFlow Configuration*. Accessed: 10-5-2023. 2023.

[Han+23] Xueying Han et al. "Network intrusion detection based on n-gram frequency and time-aware transformer". In: *Computers & Security* 128 (2023), p. 103171. ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2023.103171. URL: https://www.sciencedirect.com/science/article/pii/S0167404823000810.

[Hus+23] Ahmed Hussain et al. "Jamming Detection in IoT Wireless Networks: An Edge-AI Based Approach". In: *Proceedings of the 12th International Conference on the Internet of Things*. IoT '22. Delft, Netherlands: Association for Computing Machinery, 2023, pp. 57–64. ISBN: 9781450396653. DOI: 10.1145/3567445.3567456. URL: https://doi.org/10.1145/3567445.3567456.

[23b] *Knowledge discovery in databases DARPA archive*. Accessed: 10-5-2023. 2023.

[23c] *Network Intrusion Detection Dataset*. Accessed: 10-5-2023. 2023.

[Sed+23] Roshan Sedar et al. "A Comprehensive Survey of V2X Cybersecurity Mechanisms and Future Research Paths". In: *IEEE Open Journal of the Communications Society* 4 (2023), pp. 325–391. DOI: 10.1109/OJCOMS.2023.3239115.

[Siv+23] S. Sivanantham et al. "Association Rule Mining Frequent-Pattern-Based Intrusion Detection in Network". In: *Computer Systems Science and Engineering* 44.2 (2023), pp. 1617–1631. DOI: 10.32604/csse.2023.025893. URL: https://doi.org/10.32604/csse.2023.025893.

[23d] *The CAIDA UCSD "DDoS Attack 2007" Dataset*. Accessed: 10-5-2023. 2023.

[Yan+23] Fengru Yan et al. "TL-CNN-IDS: transfer learning-based intrusion detection system using convolutional neural network". In: *The Journal of Supercomputing* (May 2023). DOI: 10.1007/s11227-023-05347-4. URL: https://doi.org/10.1007/s11227-023-05347-4.

[Kum] Sailesh Kumar. *Survey of Current Network Intrusion Detection Techniques*. Accessed: 10-5-2023.