

**TENSOR BASED MACHINE LEARNING FRAMEWORKS FOR
INTRUSION DETECTION IN THE PHYSICAL AND NETWORK
LAYERS OF CYBER-PHYSICAL SYSTEMS**

JOÃO PAULO ABREU MARANHÃO

**TESE DE DOUTORADO EM ENGENHARIA ELÉTRICA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA**

**FACULDADE DE TECNOLOGIA
UNIVERSIDADE DE BRASÍLIA**

**UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA**

**TENSOR BASED MACHINE LEARNING FRAMEWORKS FOR
INTRUSION DETECTION IN THE PHYSICAL AND NETWORK
LAYERS OF CYBER-PHYSICAL SYSTEMS**

JOÃO PAULO ABREU MARANHÃO

**ORIENTADOR: JOÃO PAULO C. LUSTOSA DA COSTA, PROF. DR.-ING.
COORIENTADOR: RAFAEL TIMÓTEO DE SOUSA JÚNIOR, PROF. DR.**

**TESE DE DOUTORADO EM ENGENHARIA
ELÉTRICA**

PUBLICAÇÃO: PPGEE.TD-176/21

BRASÍLIA/DF: ABRIL - 2021

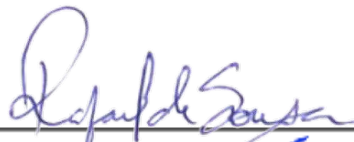
**UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA**

**TENSOR BASED MACHINE LEARNING FRAMEWORKS FOR
INTRUSION DETECTION IN THE PHYSICAL AND NETWORK
LAYERS OF CYBER-PHYSICAL SYSTEMS**

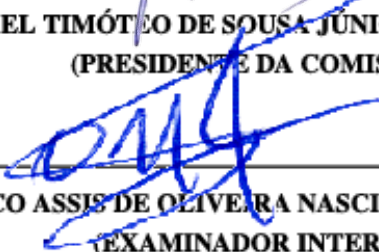
JOÃO PAULO ABREU MARANHÃO

**TESE DE DOUTORADO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA
DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA COMO PARTE DOS
REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR.**

APROVADA POR:



**RAFAEL TIMÓTEO DE SOUSA JÚNIOR, Dr., ENE/UnB
(PRESIDENTE DA COMISSÃO)**



**FRANCISCO ASSIS DE OLIVEIRA NASCIMENTO, Dr., ENE/UnB
(EXAMINADOR INTERNO)**



**EDISON PIGNATON DE FREITAS, Dr., UFRGS
(EXAMINADOR EXTERNO)**



**ANDRÉ LIMA FÉRRER DE ALMEIDA, Dr., UFC
(EXAMINADOR EXTERNO)**

Brasília, 19 de abril de 2021.

FICHA CATALOGRÁFICA

MARANHÃO, JOÃO PAULO ABREU

Tensor Based Machine Learning Frameworks for Intrusion Detection in the Physical and Network Layers of Cyber-Physical Systems [Distrito Federal] 2021.

xxv, 192p., 210 x 297 mm (ENE/FT/UnB, Doutor, Engenharia Elétrica, 2021).

Tese de Doutorado – Universidade de Brasília, Faculdade de Tecnologia.

Departamento de Engenharia Elétrica

- | | |
|-----------------------|---------------------------------------|
| 1. Machine Learning | 2. Multidimensional Signal Processing |
| 3. Drone Localization | 4. Network Attack Detection |
| I. ENE/FT/UnB | II. Título (série) |

REFERÊNCIA BIBLIOGRÁFICA

MARANHÃO, J. P. A. (2021). Tensor Based Machine Learning Frameworks for Intrusion Detection in the Physical and Network Layers of Cyber-Physical Systems , Tese de Doutorado , Publicação PPGEE.TD-176/21, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 192p.

CESSÃO DE DIREITOS

AUTOR: João Paulo Abreu Maranhão

TÍTULO: Tensor Based Machine Learning Frameworks for Intrusion Detection in the Physical and Network Layers of Cyber-Physical Systems .

GRAU: Doutor ANO: 2021

É concedida à Universidade de Brasília permissão para reproduzir cópias deste tese de doutorado e para emprestar ou vender tais cópias somente para propósitos acadêmicos e científicos. O autor reserva outros direitos de publicação e nenhuma parte dessa tese de doutorado pode ser reproduzida sem autorização por escrito do autor.



João Paulo Abreu Maranhão

Departamento de Engenharia Elétrica (ENE) - FT

Universidade de Brasília (UnB)

Campus Darcy Ribeiro

CEP 70919-970 - Brasília - DF - Brasil

Dedicated to my parents, Zélia and Luiz Alberto, who always encouraged me to overcome obstacles through life. To my wife, Anna Paola, thank you for your loving support and faith in me.

ACKNOWLEDGMENTS

Firstly, I would like to thank God for His guidance throughout the PhD period, and for having put the right people in my path.

To my supervisor and friend, Prof Dr.-Ing. João Paulo C. L. da Costa, for your considerable suggestions and support throughout the PhD period, as well as for giving your valuable time to listen with an open mind. Without your help and constant interest, the present work would not have been possible.

To my co-supervisor, Prof. Dr. Rafael T. de Sousa Jr., for accepting to assume my co-advisory and for providing valuable support in the activities related to the Graduate Program in Electrical Engineering (PPGEE) of the Department of Electrical Engineering (ENE) of the University of Brasília (UnB).

To the colleagues of the Graduate Program in Electrical Engineering (PPGEE) of the Department of Electrical Engineering (ENE) of the University of Brasília (UnB), who have supported me with suggestions and experiences during the last four years.

I am grateful for the technical and computational support of the Decision-Making Technologies Laboratory - LATITUDE, of the University of Brasilia (UnB), which is supported by CNPq - Brazilian National Research Council (Grants 312180/2019-5 PQ-2, BRICS2017-591 LargEWiN, and 465741/2014-2 INCT on Cybersecurity), by CAPES - Brazilian Higher Education Personnel Improvement Coordination (Grants PROAP PPGEE/UnB, 23038.007604/2014-69 FORTE, and 88887.144009/2017-00 PROBRAL), by FAP-DF - Brazilian Federal District Research Support Foundation (Grant 0193.001366/2016 UIoT, and Grant 0193.001365/2016 SSDDC), by the Brazilian Ministry of the Economy (Grant 005/2016 DIPLA, and Grant 083/2016 ENAP), by the Institutional Security Office of the Presidency of Brazil (Grant ABIN 002/2017), by the Administrative Council for Economic Defense (Grant CADE 08700.000047/2019-14), by the General Attorney of the Union (Grant AGU 697.935/2019), and by DPI/DPG/UnB - Deconates of Research and Innovation and Postgraduate Studies of the University of Brasilia.

I am grateful for the people who reviewed this manuscript and suggested ways to improve its readability, especially Prof. Dr. Francisco Assis de Oliveira Nascimento, Prof. Dr. Edison Pignaton de Freitas and Prof. Dr. André L. F. de Almeida.

To the Department of Science and Technology (DCT) of the Brazilian Army, for providing another learning opportunity in my career as a Military Engineer. In addition, I would like to especially thank the Chief of the Systems Development Center (CDS), Gen Div Eduardo Wolski, for believing in me when authorizing the challenge of pursuing a doctorate.

To my parents, Zélia and Luiz Alberto, who always encouraged me to face the challenges of life. You taught me to keep the faith and never give up. I thank you for all the unconditional love and care you have provided me throughout my life, especially in times of greatest difficulty.

To my beloved daughter, Lara, who is about to join us to make our lives even happier!

Last but not least, I would like to thank my wife, Anna Paola, for your loving support, affection and patience throughout the PhD period. Your life of persistence, dedication and professionalism inspired me to never give up and to overcome obstacles through life, always doing my best. You have made me a better person. I love you!

ABSTRACT

Title: Tensor Based Machine Learning Frameworks for Intrusion Detection in the Physical and Network Layers of Cyber-Physical Systems

Author: João Paulo Abreu Maranhão

Supervisor: João Paulo C. Lustosa da Costa, Prof. Dr.-Ing.

Co-Supervisor: Rafael Timóteo de Sousa Júnior, Prof. Dr.

Graduate Program in Electrical Engineering

Brasília, April 19th, 2021

Cyber-Physical Systems (CPS) are physical and engineered systems used to control and monitor the physical environment by integrating sensors, control processing units and communication devices. Usually, CPS are applied on safety-critical applications, such as defense systems, manufacturing and traffic control, as well as critical control infrastructures, for instance, electric power, water resources and communication systems. In this sense, the development of highly accurate intrusion detection systems applied to CPS is crucial, such that these critical applications can be efficiently managed and controlled. The scope of this thesis is the security of CPS at the physical and network layers, particularly regarding the localization and identification of Unmanned Aerial Vehicles in multipath environments, as well as the Distributed Denial of Service attack detection.

Unmanned Aerial Vehicles (UAV) are remotely piloted aircrafts very popular for personal, commercial and public-safety applications. However, multiple drone-related accidental and intentional incidents have been increasingly reported, such as mid-air collisions in airports, smuggling of illicit materials and illegal surveillance. Therefore, solutions for localizing and identifying UAVs have been intensively investigated in the literature. As a first important contribution, a framework based on the joint application of multidimensional signal processing and machine learning schemes is proposed in order to accurately localize and identify intruding UAVs within a controlled air space. According to simulations, the proposed approach considerably outperforms its competing schemes in terms of several evaluation metrics.

In addition, the security of Cyber-Physical Systems at the network layer can be compromised by malicious activities, such as the Distributed Denial of Service (DDoS) attacks. Machine learning (ML) based solutions have been intensively investigated in order to automatically identify malicious patterns in the incoming data traffic. Since supervised ML approaches require training with large datasets, which present inherent multidimensional structures, the development of a DDoS attack detection system which exploits both machine learning and multidimensional signal processing techniques is crucial. As a second contribution, we propose a novel framework for DDoS attack detection, applying the above-mentioned techniques, where corrupted datasets are used during the training and testing phases. Simulation results show that the proposed approach outperforms the competitor

methods considering accuracy, detection rate and false alarm rate.

Still regarding DDoS attack detection when corrupted datasets are used for training and testing, as a third contribution, a noise-robust multilayer perceptron (MLP) architecture based on the Higher Order Singular Value Decomposition (HOSVD) algorithm is introduced. In our proposed scheme, the average value of the common features among dataset instances is iteratively filtered out via the HOSVD algorithm, providing more robustness against data corruption. The effectiveness of our scheme is validated through comparison with state-of-the-art methods. According to experiment results, the proposed approach outperforms its competing techniques in terms of accuracy, detection rate and false alarm rate.

Due to the considerable results shown in multiple domains, Convolutional Neural Networks (CNN), one of the most popular and efficient deep learning based techniques, present an outstanding potential for detecting DDoS attacks. In this sense, as our last contribution, two novel CNN based architectures for DDoS attack detection are introduced. The proposed schemes have multiple parallel branches, each of which composed by several CNNs. The CNN outputs from consecutive parallel branches are concatenated in order to enrich feature diversity and, consequently, a better recognition ability is achieved by the detection model. Experiment results confirm that the proposed approaches outperform their competing schemes when several performance evaluation metrics are considered.

Keywords: Machine Learning, Deep Learning, Multidimensional Signal Processing, Drone Localization, Distributed Denial of Service Attack Detection, Multilayer Perceptron, Convolutional Neural Networks, Multiple Denoising.

RESUMO

Título: Arquiteturas de Aprendizado de Máquina Baseadas em Tensores para Detecção de Intrusão nas Camadas Física e de Enlace de Sistemas Ciber-Físicos

Autor: João Paulo Abreu Maranhão

Orientador: João Paulo C. Lustosa da Costa, Prof. Dr.-Ing.

Coorientador: Rafael Timóteo de Sousa Júnior, Prof. Dr.

Programa de Pós-Graduação em Engenharia Elétrica

Brasília, 19 de abril de 2021

Sistemas Ciber-Físicos (do inglês Cyber-Physical Systems, ou CPS) são sistemas físicos e de engenharia usados para controlar e monitorar ambientes físicos, integrando sensores, unidades de processamento de controle e dispositivos de comunicação. Geralmente, os CPS são utilizados em aplicações críticas de segurança, tais como sistemas militares de defesa, fábricas e controle de tráfego, bem como em infraestruturas críticas, por exemplo, usinas para geração de energia elétrica, represas de abastecimento de água e sistemas de telecomunicações. Nesse sentido, o desenvolvimento de sistemas de detecção de intrusão de alta precisão para Sistemas Ciber-Físicos é fundamental, de modo que tais aplicações críticas possam ser gerenciadas e controladas com eficiência e confiabilidade. O escopo desta tese é a segurança das camadas física e de rede de CPS, particularmente em relação à localização e identificação de Veículos Aéreos Não Tripulados não autorizados, bem como à detecção de ataques de Negação de Serviço Distribuídos.

Veículos Aéreos Não Tripulados (VANTs) são aeronaves pilotadas remotamente muito populares em aplicações pessoais, comerciais e de segurança pública. No entanto, vários incidentes acidentais e intencionais relacionados com VANTs têm sido cada vez mais relatados, por exemplo, em colisões aéreas em aeroportos, contrabando de materiais ilícitos e espionagem. Portanto, soluções para localização e identificação de VANTs têm sido intensamente investigadas na literatura. Como uma primeira contribuição, uma arquitetura baseada na aplicação conjunta de técnicas de processamento de sinais multidimensionais e de aprendizado de máquina é proposta com a finalidade de localizar e identificar, com precisão, VANTs intrusos no interior de espaços aéreos controlados. De acordo com as simulações, a abordagem proposta supera consideravelmente seus esquemas concorrentes em termos de várias métricas de avaliação.

Além disso, a segurança na camada de rede em Sistemas Ciber-Físicos pode ser comprometida por diversas atividades maliciosas, como ataques de negação de serviço distribuídos (do inglês Distributed Denial of Service, ou DDoS). As soluções baseadas em aprendizado de máquina (ML) têm sido intensamente investigadas para identificar automaticamente padrões maliciosos no tráfego de entrada de dados. Como as abordagens de ML supervisionadas exigem treinamento com grandes conjuntos de dados, os quais apresentam estruturas inerentemente multidimensionais, o desenvolvimento de um sistema de detecção de ataques DDoS

explorando técnicas de aprendizado de máquina e de processamento de sinais multidimensionais é fundamental. Como uma segunda contribuição, propomos uma nova arquitetura para detecção de ataques DDoS, utilizando as referidas técnicas, onde conjuntos de dados corrompidos são utilizados durante as fases de treinamento e teste. Resultados de simulação mostram que a abordagem proposta supera os métodos concorrentes em termos de acurácia, taxa de detecção e taxa de falso alarme.

Ainda em relação à detecção de ataques DDoS, como uma terceira contribuição, este trabalho propõe uma nova arquitetura de Multilayer Perceptron (MLP), robusta à presença de dados de treinamento corrompidos, baseada na técnica Higher Order Singular Value Decomposition (HOSVD). Em nosso esquema proposto, o valor médio dos atributos comuns entre as amostras do conjunto de dados é filtrado iterativamente por meio do algoritmo HOSVD, fornecendo mais robustez contra a presença de dados corrompidos. A eficácia do esquema proposto é validada por meio da comparação com métodos do estado da arte equivalentes. A abordagem proposta apresenta desempenho superior em comparação às técnicas do estado da arte concorrentes em termos de acurácia, taxa de detecção e taxa de falso alarme.

Por fim, devido aos excelentes resultados mostrados em vários domínios do conhecimento, as Redes Neurais Convolucionais (do inglês Convolutional Neural Networks, ou CNN), uma das técnicas de aprendizado profundo mais populares e eficientes, apresentam um excelente potencial para detecção de ataques DDoS. Nesse sentido, como nossa última contribuição, duas novas arquiteturas baseadas em CNNs para detecção de ataques DDoS são apresentadas. Tais esquemas apresentam múltiplos ramos em paralelo, onde cada um deles é composto por várias CNNs em série. As saídas de CNNs pertencentes a ramos paralelos consecutivos são concatenadas com a finalidade de enriquecer a diversidade de atributos e, conseqüentemente, uma melhor capacidade de reconhecimento é alcançada pelo modelo de detecção. Resultados de simulações confirmam que as abordagens propostas superam os esquemas concorrentes quando várias métricas de avaliação de desempenho são consideradas.

Palavras-chave: Aprendizado de Máquina, Aprendizado Profundo, Processamento de Sinais Multidimensionais, Localização de Drones, Detecção de Ataques de Negação de Serviço Distribuída, Perceptron Multicamadas, Redes Convolucionais Neurais, Atenuação de Ruído Múltipla.

SUMMARY

1	INTRODUCTION.....	1
1.1	OVERVIEW OF CYBER-PHYSICAL SYSTEMS.....	1
1.2	MOTIVATION	2
1.3	RESEARCH AIM, OBJECTIVES AND CONTRIBUTIONS	6
1.4	NOTATION	11
1.5	THESIS ORGANIZATION	12
2	TENSOR BASED FRAMEWORK FOR UAV LOCALIZATION AND IDENTIFICATION IN MULTIPATH ENVIRONMENTS	13
2.1	MOTIVATION	14
2.2	DATA MODEL	17
2.3	PROPOSED TENSOR BASED FRAMEWORK FOR UAV LOCALIZATION AND IDENTIFICATION	21
2.3.1	DECORRELATION AND MULTIPLE DENOISING PREPROCESSING	23
2.3.2	DIRECTION OF ARRIVAL ESTIMATION	25
2.3.3	MULTIPATH COMPONENTS GROUPING	26
2.3.4	LINE-OF-SIGHT ESTIMATION	30
2.3.5	TRIANGULATION	32
2.3.6	UAV IDENTIFICATION	36
2.3.7	MAJORITY VOTING	37
2.4	COMPUTATIONAL COMPLEXITY.....	38
2.5	SIMULATION RESULTS.....	40
2.5.1	LOCALIZATION MODULE	41
2.5.2	IDENTIFICATION MODULE	46
2.5.3	DISCUSSION	50
2.6	SUMMARY	52
3	TENSOR MULTIPLE DENOISING BASED FRAMEWORK FOR DISTRIBUTED DENIAL OF SERVICE ATTACK DETECTION	53
3.1	MOTIVATION	54
3.2	DATA MODEL	57
3.3	PROPOSED TENSOR MULTIPLE DENOISING BASED FRAMEWORK FOR DDoS ATTACK DETECTION	59
3.3.1	DATA PREPROCESSING	60
3.3.2	DATASET SPLITTING.....	60
3.3.3	PROPOSED EXTENDED MUDe TECHNIQUE	61

3.3.4 MACHINE LEARNING SUPERVISED CLASSIFICATION	63
3.4 COMPUTATIONAL COMPLEXITY	64
3.5 SIMULATION RESULTS	66
3.5.1 DDoS ATTACK DATASETS	66
3.5.2 RESULTS	67
3.5.3 DISCUSSION	74
3.6 SUMMARY	80
4 NOISE-ROBUST MULTILAYER PERCEPTRON ARCHITECTURE FOR DISTRIBUTED DENIAL OF SERVICE ATTACK DETECTION	82
4.1 MOTIVATION	83
4.2 PROPOSED NOISE-ROBUST MLP ARCHITECTURE FOR DDoS ATTACK DETECTION	85
4.2.1 PROPOSED FEATURE EXTRACTION METHOD APPLIED ON DATA CLASSIFICATION	85
4.2.1.1 TRAINING PHASE	87
4.2.1.2 TESTING PHASE	90
4.2.2 PROPOSED MLP ARCHITECTURE	92
4.2.2.1 TRAINING PHASE	92
4.2.2.2 TESTING PHASE	94
4.2.2.3 SUMMARY	94
4.3 COMPUTATIONAL COMPLEXITY	96
4.4 SIMULATION RESULTS	98
4.4.1 DDoS ATTACK DATASETS	98
4.4.2 RESULTS	98
4.4.3 DISCUSSION	105
4.5 SUMMARY	108
5 TENSOR FEATURE MAP CONCATENATION BASED CONVOLUTIONAL NEURAL NETWORK SCHEMES FOR DISTRIBUTED DENIAL OF SER- VICE ATTACK DETECTION	109
5.1 MOTIVATION	110
5.2 PROPOSED TENSOR FEATURE MAP CONCATENATION BASED CNN SCHEME FOR DDoS ATTACK DETECTION	113
5.2.1 DATA PREPROCESSING	114
5.2.2 PROPOSED TFMCB-CNN SCHEME	115
5.2.2.1 PROPOSED TENSOR FEATURE MAP CONCATENATION BLOCK ..	115
5.2.2.2 FINAL CONCATENATION	120
5.2.2.3 FLATTEN	120
5.2.2.4 MLP	120

5.2.3 PROPOSED TFMCB-CNN SCHEME IMPROVED VIA MAJORITY VOT- ING	122
5.2.3.1 FLATTEN	122
5.2.3.2 MLP	123
5.2.3.3 SMV BLOCK	124
5.3 COMPUTATIONAL COMPLEXITY	124
5.4 SIMULATION RESULTS	126
5.4.1 DDoS ATTACK DATASETS	127
5.4.2 RESULTS	128
5.4.3 PROCESSING TIME	138
5.4.4 DISCUSSION	140
5.5 SUMMARY	143
6 CONCLUSION.....	144
6.1 CONCLUSIONS	145
6.2 FUTURE WORKS	148
A LOW-RANK MATRIX AND TENSOR BASED APPROXIMATION TECH- NIQUES.....	151
A.1 SINGULAR VALUE DECOMPOSITION (SVD).....	151
A.2 HIGHER ORDER SINGULAR VALUE DECOMPOSITION (HOSVD)	154
A.3 HIGHER ORDER ORTHOGONAL ITERATION.....	156
B TRIANGULATION	157
C RF SENSING BASED DRONE IDENTIFICATION AND DDoS ATTACK DATASETS.....	165
C.1 RF SENSING BASED DRONE IDENTIFICATION DATASET.....	165
C.2 DDoS ATTACK DATASETS	166
D PERFORMANCE EVALUATION METRICS	170
E SIMULATION PARAMETERS ADOPTED FOR ML AND DL CLASSI- FICATION ALGORITHMS	172
REFERENCES.....	173

LIST OF TABLES

1.1	Incidents involving UAVs between 2015 and 2021 over the world.....	3
1.2	The world’s largest DDoS attacks between 2012 and 2020.	7
2.1	Related works summary.....	17
2.2	Data from the estimated symbols matrix $\hat{\mathbf{S}}_u \in \mathbb{C}^{6 \times 3}$	28
2.3	Correlation coefficient between $\hat{\mathbf{S}}_u(i, :)$ and $\hat{\mathbf{S}}_u(j, :)$ for $i, j = 1, \dots, 6$ and $i \neq j$	29
2.4	Data from the estimated symbols matrices $\hat{\mathbf{S}}_u^q \in \mathbb{C}^{3 \times 4}$ for $q = 1, 2$	31
2.5	Number of raw samples and segments for each drone model present in the DroneRF dataset.	36
2.6	Computational complexity of the following models: (1) proposed framework, (2) tensor based ESPRIT with SS, and (3) matrix based ESPRIT with SS.....	40
2.7	RMSE of the estimated spatial frequency, considering different configurations of the antenna array based UAV localization system.	42
2.8	Macro-averaged values of accuracy, detection rate and false alarm rate of the proposed framework and its competitor schemes for identifying three different types of drones (Bebop, AR and Phantom), considering SIR between 0 dB and 20 dB.	49
2.9	Values of accuracy, detection rate and false alarm rate of the proposed framework and its competitor schemes for identifying three different drone models (Bebop, AR and Phantom), considering ET and LDA classifiers.	50
3.1	Related works summary.....	56
3.2	Computational complexity of the following schemes: (1) proposed extended MuDe technique with HOOI, (2) proposed extended MuDe technique with HOSVD, (3) HOOI, (4) HOSVD, and (5) SVD.	66
3.3	DDoS attack types used in this chapter as well as the corresponding number of instances extracted from the CIC-DDoS2019 dataset.	67
3.4	DDoS attack types used in this chapter as well as the corresponding number of instances extracted from the NSL-KDD dataset.	67
3.5	Performance evaluation, considering different LRA schemes, for the following models: (1) proposed framework with HOOI, and (2) proposed framework with HOSVD.....	69

3.6	Mean training times (in seconds), considering different LRA schemes, for the following models: (1) proposed framework with HOOI, and (2) proposed framework with HOSVD.	70
3.7	Performance evaluation of the proposed framework for different instance configurations.....	71
3.8	Mean training times (in seconds) of the proposed framework for different instance configurations.	71
3.9	Performance evaluation of the proposed framework considering different MOS techniques.	72
3.10	Performance comparison with related works, considering the NSL-KDD and CIC-DDoS2019 datasets.....	79
4.1	Related works summary.....	84
4.2	Computational complexity of the following schemes: (1) proposed feature extraction technique, (2) HOSVD, and (3) HOOI.	98
4.3	DDoS attack types used in this chapter, as well as the corresponding number of instances extracted from each dataset.	99
4.4	Performance evaluation and mean training times (in seconds) of the proposed technique for different instance configurations.....	100
4.5	Performance evaluation under real time attacks.....	104
4.6	Noise-robustness evaluation results.	105
4.7	Performance comparison with related works, considering the CIC-IDS2017 and CIC-DDoS2019 datasets.	106
5.1	Related works summary.....	112
5.2	Computational complexity of the following schemes: (1) Proposed TFMCB-CNN-MV, (2) Proposed TFMCB-CNN, (3) Parallel-CNN-MV, (4) Parallel-CNN, and (5) Serial-CNN.	126
5.3	DDoS attack types used in this chapter, as well as the corresponding number of instances extracted from the CIC-DDoS2019 and CIC-IDS2017 datasets. ...	127
5.4	Layer types, output shapes and simulation parameters at each block of the proposed TFMCB-CNN-MV scheme for $l = 1, \dots, L$	129
5.5	Layer types, output shapes and simulation parameters at each block of the proposed TFMCB-CNN scheme for $l = 1, \dots, L$	130
5.6	Performance evaluation metrics of the proposed TFMCB-CNN-MV and TFMCB-CNN schemes, as well as its competitor approaches, when the number of branches ranges from 1 to 5.....	133
5.7	Performance evaluation metrics of the proposed TFMCB-CNN-MV and TFMCB-CNN schemes, as well as its competitor approaches, when the number of dense layers ranges from 1 to 5.	134

5.8	Performance evaluation metrics of the proposed TFMCB-CNN-MV and TFMCB-CNN schemes, as well as its competitor approaches, when the batch size ranges from 128 to 1024.	135
5.9	Performance evaluation metrics of the proposed TFMCB-CNN-MV and TFMCB-CNN schemes, as well as its competitor approaches, when the convolution filter size ranges from 2×2 to 5×5	136
5.10	Performance evaluation metrics of the proposed TFMCB-CNN-MV and TFMCB-CNN schemes against state-of-the-art ML and DL classification algorithms. ...	136
5.11	Performance comparison with related works, considering the CIC-IDS2017 and CIC-DDoS2019 datasets.	137
B.1	Coordinates of the q -th UAV, computed via triangulation with the u -th and v -th URAs, according to the relative positions shown in Fig. B.2 and B.3.	163
B.2	Coordinates of the q -th UAV, computed via triangulation with the u -th and v -th URAs, according to the relative positions shown in Fig. B.4 and B.5.	164
C.1	NSL-KDD dataset features used in Chapter 3.	167
C.2	CIC-IDS2017, CIC-IDS2018 and CIC-DDoS2019 dataset features used in Chapters 3 to 5.	169
C.3	Capturing days and network traffic present in the CIC-IDS2017, CIC-IDS2018 and CIC-DDoS2019 datasets.	169
D.1	Confusion matrix.	170
E.1	Main simulation parameters adopted for the state-of-the-art ML and DL classification algorithms used in this thesis.	172

LIST OF FIGURES

1.1	Typical CPS architecture	2
1.2	Number of DDoS attacks through the forecast period of 2018-2023 [44].	5
1.3	Top 10 countries, in the 4th quarter of 2020, by: (a) number of DDoS attacks. (b) number of DDoS targets [45].	5
1.4	DDoS-related statistics in the 4th quarter of 2020. (a) Distribution of botnet C&C servers by country. (b) Distribution of DDoS attacks by type [45].	6
1.5	Overview of the complete solution for intrusion detection in CPSs proposed in this thesis. (a) DDoS attack infrastructure. (b) Local Area Network (Tar- get). (c) UAV localization and identification system.....	11
2.1	(a) Overview of the antenna array based UAV localization system. (b) The azimuth angle at the u -th antenna array corresponds to the acute angle be- tween the y -direction and the projection of the LOS onto the xy -plane, i.e, $\hat{\phi}_{u,\text{los}}^q = 360^\circ - \hat{\varphi}_{u,\text{los}}^q$	23
2.2	Elevation and azimuth angles for four different relative positions between the u -th URA and the q -th UAV: (a) $0^\circ < \hat{\varphi}_{u,\text{los}}^q < 90^\circ$. (b) $90^\circ < \hat{\varphi}_{u,\text{los}}^q < 180^\circ$. (c) $180^\circ < \hat{\varphi}_{u,\text{los}}^q < 270^\circ$. (d) $270^\circ < \hat{\varphi}_{u,\text{los}}^q < 360^\circ$	24
2.3	Block diagram of the proposed framework, which is composed by two main modules: (a) UAV localization, and (b) UAV identification. Each block presents the respective references to steps of Algorithm 1 as well as equa- tions of Subsections 2.3.1 to 2.3.7.....	25
2.4	LOS plus NLOS signals from two UAVs received at an antenna array. (a) UAVs 1 and 2 localized at similar distances from the antenna array. (b) UAV 2 is closer to the antenna array than UAV 1.....	27
2.5	Localization of the q -th UAV via triangulation techniques, considering four different relative positions between the u -th and v -th URAs, as well as the detected drone.	33
2.6	Four different configurations of the proposed antenna array based UAV lo- calization system used in simulations. (a) triangle. (b) square. (c) pentagon. (d) hexagon.....	43
2.7	(a) RMSE of the estimated spatial frequency versus SIR (dB). (b) RMSE of the estimated position coordinates versus SIR (dB).....	44
2.8	(a) RMSE of the estimated spatial frequency versus number of antennas per dimension. (b) RMSE of the estimated position coordinates versus number of antennas per dimension.....	45

2.9	RMSE of the estimated spatial frequency versus number of samples. (b) RMSE of the estimated position coordinates versus number of samples.	45
2.10	RMSE of the estimated spatial frequency versus number of UAVs. (b) RMSE of the estimated position coordinates versus number of UAVs.	46
2.11	a) Mean processing time (s) versus number of samples. (b) Mean processing time (s) versus number of UAVs.	47
3.1	The process of construction of a three-dimensional dataset tensor $\mathfrak{X} \in \mathbb{R}^{N_1 \times N_2 \times M}$ from the dataset matrix $\mathbf{X} \in \mathbb{R}^{M \times N}$	58
3.2	Different tensor foldings of the m -th dataset instance $\mathfrak{X}(:, \dots, m) \in \mathbb{R}^{N_1 \times \dots \times N_R}$ for $R=2, \dots, 6$	58
3.3	The proposed tensor multiple denoising based framework for DDoS attack detection. The extended MuDe technique is detailed in Blocks 3.1 to 3.10 with the respective references to steps of Algorithm 2 as well as equations of Subsection 3.3.3.	59
3.4	Plots of accuracy, as a function of the signal-to-noise ratio (dB), considering: (a) AB, (b) LDA, (c) LR, and (d) RFo classification algorithms.	73
3.5	Plots of detection rate, as a function of the signal-to-noise ratio (dB), considering: (a) AB, (b) LDA, (c) LR, and (d) RFo classification algorithms.	74
3.6	Plots of false alarm rate, as a function of the signal-to-noise ratio (dB), considering: (a) AB, (b) LDA, (c) LR, and (d) RFo classification algorithms.	75
3.7	Plots of accuracy, as a function of the training dataset size proportion, considering: (a) AB, (b) LDA, (c) LR, and (d) RFo classification algorithms.	76
3.8	Plots of detection rate, as a function of the training dataset size proportion, considering: (a) AB, (b) LDA, (c) LR, and (d) RFo classification algorithms. ...	77
3.9	Plots of false alarm rate, as a function of the training dataset size proportion, considering: (a) AB, (b) LDA, (c) LR, and (d) RFo classification algorithms. ...	78
3.10	Plots of mean training time (s), as a function of the training dataset size proportion, for AB, LDA, LR and RFo classification algorithms.	80
4.1	The concept of common features for multidimensional data. The original tensor $\mathfrak{Y} \in \mathbb{R}^{I_1 \times I_2 \times 5}$ consists of slices with the bright blue, bright green, gray, orange and pink colors. Each slice $\mathfrak{Y}(:, :, s) \in \mathbb{R}^{I_1 \times I_2}$ for $s = 1, \dots, 5$ corresponds to a combination of the base colors green, red and blue, given by $\mathbf{B}_G \in \mathbb{R}^{I_1 \times I_2}$, $\mathbf{B}_R \in \mathbb{R}^{I_1 \times I_2}$ and $\mathbf{B}_B \in \mathbb{R}^{I_1 \times I_2}$, respectively. Such base colors can be stacked along the 3rd dimension, generating the common feature tensor $\mathfrak{B} \in \mathbb{R}^{I_1 \times I_2 \times 3}$	86
4.2	Block diagram of the proposed feature extraction method applied on ML classification. (a) Training phase. (b) Testing phase.	88

4.3	Block diagram of the training phase of the proposed feature extraction technique considering a three-dimensional dataset tensor $\mathcal{X}^s \in \mathbb{R}^{N_1 \times N_2 \times M}$. The diagram presents references to steps of Algorithm 3 as well as equations of Subsection 4.2.1.1.	91
4.4	The proposed MLP architecture for DDoS attack detection.	93
4.5	Block diagram of the low-rank approximation based MLP (training and testing phases).	95
4.6	Block diagram of the proposed common feature extraction based MLP (training phase).	96
4.7	Block diagram of the proposed common feature extraction based MLP (testing phase).	96
4.8	Plots of: (a) training and validation accuracy, and (b) training and validation cost, as a function of the number of epochs, for the proposed MLP architecture.	101
4.9	(a) Mean weight as a function of the number of epochs. (b) Section of the MLP representing the proposed weights within the dotted red rectangle.	101
4.10	Acc, DR and FAR as a function of the noise level and number of hidden layers.	102
4.11	Acc, DR and FAR as a function of the number of hidden layers and training size proportion under noise-free conditions.	103
4.12	Network topology for simulating real time attacks.	103
4.13	(a) Training time (s) versus number of hidden layers. (b) Testing time (s) versus number of hidden layers.	104
5.1	Block diagram of the proposed TFMCB-CNN scheme.	115
5.2	Detailed view of the proposed Tensor Feature Map Concatenation block, including a zoomed view of the inputs and outputs of the $\text{CNN}_{l,k}$ and $\text{concl}_{l,k}$ blocks at the bottom.	116
5.3	Inputs and outputs of the Convolution, Batch Normalization, Activation, Pooling and Dropout blocks within $\text{CNN}_{l,k}$	116
5.4	Dense layers and Softmax function within the MLP block.	121
5.5	Block diagram of the proposed TFMCB-CNN-MV scheme.	122
5.6	Examples of block diagrams of the following models: (a) Proposed TFMCB-CNN-MV, (b) Proposed TFMCB-CNN, (c) Parallel-CNN-MV, (d) Parallel-CNN, and (e) Serial-CNN.	132
A.1	(a) SVD of the dataset matrix $\mathbf{X} \in \mathbb{R}^{M \times N}$. The rank d is used to truncate the matrices $\mathbf{U} \in \mathbb{R}^{M \times M}$, $\mathbf{\Lambda} \in \mathbb{R}^{M \times N}$ and $\mathbf{V} \in \mathbb{R}^{N \times N}$ to the signal subspace. (b) Denoised dataset matrix $\tilde{\mathbf{X}} \in \mathbb{R}^{M \times N}$, corresponding to the product $\mathbf{U}_s \mathbf{\Sigma}_s \mathbf{V}_s^H$, where $\mathbf{U}_s \in \mathbb{R}^{M \times d}$, $\mathbf{V}_s \in \mathbb{R}^{N \times d}$ and $\mathbf{\Sigma}_s \in \mathbb{R}^{d \times d}$	153

- A.2 (a) HOSVD of the dataset tensor $\mathbf{X} \in \mathbb{R}^{N_1 \times N_2 \times N_3}$. The multilinear rank (d_1, d_2, d_3) is used to truncate the core tensor $\mathcal{G} \in \mathbb{R}^{N_1 \times N_2 \times N_3}$ and the factor matrices $\mathbf{U}_r \in \mathbb{R}^{N_r \times N_r}$ for $r = 1, \dots, 3$. (b) Denoised dataset tensor $\tilde{\mathbf{X}} \in \mathbb{R}^{N_1 \times N_2 \times N_3}$, corresponding to the product $\mathcal{G}^{[s]} \times_1 \mathbf{U}_1^{[s]} \times_2 \mathbf{U}_2^{[s]} \times_3 \mathbf{U}_3^{[s]}$, where $\mathcal{G}^{[s]} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ and $\mathbf{U}_r^{[s]} \in \mathbb{R}^{N_r \times d_r}$ for $r = 1, \dots, 3$ 155
- B.1 (a) Triangulation technique used to localize the q -th UAV by using the u -th and v -th URAs. (b) Triangulation technique, projected onto the xy -plane. 158
- B.2 Localization of the q -th UAV via triangulation, projected onto the xy -plane, considering positions 1 to 6 ($x_u < x_v$). 159
- B.3 Localization of the q -th UAV via triangulation, projected onto the xy -plane, considering positions 7 to 12 ($x_u < x_v$)..... 160
- B.4 Localization of the q -th UAV via triangulation, projected onto the xy -plane, considering positions 1 to 6 ($x_u > x_v$). 161
- B.5 Localization of the q -th UAV via triangulation, projected onto the xy -plane, considering positions 7 to 12 ($x_u > x_v$)..... 162

LIST OF SYMBOLS

Latin symbols

$\mathbf{X}, \mathbf{Y}, \mathbf{Z}$	Matrices
$\mathbf{x}, \mathbf{y}, \mathbf{z}$	Vectors
$\mathcal{X}, \mathcal{Y}, \mathcal{Z}$	Tensors
x, y, z	Scalars

Subscript symbols

$[\mathcal{X}]_{(r)}$	The r -th mode unfolding matrix of \mathcal{X}
-----------------------	--

Superscript symbols

$\{\cdot\}^C$	Concatenation
$\{\cdot\}^H$	Hermitian
$\{\cdot\}^T$	Transposition

Operator symbols

$EV\{\cdot\}$	Returns the eigenvalues of a matrix
$E\{\cdot\}$	Returns the expectation value of a function
$\operatorname{argmax}\{\cdot\}$	Returns the argument that gives the maximum value from a target function.
$\operatorname{diag}\{\cdot\}$	Transforms its argument vector into the main diagonal of a diagonal matrix
$\max\{\cdot\}$	Returns the highest number in a numeric data set.
$\min\{\cdot\}$	Returns the lowest number in a numeric data set.
$\operatorname{mode}\{\cdot\}$	Returns the most frequently occurring number in a numeric data set.
$\operatorname{vec}\{\cdot\}$	Transforms its argument into a vector
$\mathcal{O}[\cdot]$	Returns the asymptotic behavior of a function

Product symbols

\circ	Outer product
\diamond	Khatri-Rao product
\odot	Hadamard product
\otimes	Kronecker product
\times_r	The r -th mode product between a tensor and a matrix

LIST OF ACRONYMS

AB	AdaBoost.
Acc	Accuracy.
AIC	Akaike's Information Theoretic Criteria.
CIC	Canadian Institute for Cybersecurity.
CLDAP	Connectionless Lightweight Directory Access Protocol.
CPS	Cyber-Physical System.
CSV	Comma-Separated Values.
DDoS	Distributed Denial of Service.
DFT	Discrete Fourier Transform.
DL	Deep Learning.
DMZ	Demilitarized Zone.
DNS	Domain Name System.
DoA	Direction of Arrival.
DoD	Direction of Departure.
DoS	Denial of Service.
DR	Detection Rate.
DT	Decision Trees.
EDC	Efficient Detection Criterion.
ESPRIT	Estimation of Signal Parameters via Rotational Invariance Techniques.
ET	Extra Trees.
EVD	Eigenvalue Decomposition.
FAR	False Alarm Rate.
FN	False Negative.
FNR	False Negative Rate.
FP	False Positive.
FPR	False Positive Rate.
Gbps	Gigabits per second.
HL	Hidden Layers.
HOOI	Higher Order Orthogonal Iteration.

HOSVD	Higher Order Singular Value Decomposition.
HTTP	HyperText Transfer Protocol.
ICMP	Internet Control Message Protocol.
IDS	Intrusion Detection System.
IoT	Internet of Things.
IP	Internet Protocol.
kNN	k-Nearest Neighbors.
LAN	Local Area Network.
LDA	Linear Discriminant Analysis.
LDAP	Lightweight Directory Access Protocol.
LOIC	Low Orbit Ion Cannon.
LOS	Line-of-Sight.
LR	Logistic Regression.
LRA	Low-Rank Approximation.
MCC	Matthews Correlation Coefficient.
MDL	Minimum Description Length.
MIMO	Multiple Input Multiple Output.
ML	Machine Learning.
MLP	Multilayer Perceptron.
MOS	Model Order Selection.
MSSQL	Microsoft Structured Query Language.
MuDe	Multiple Denoising.
N/A	Not Available.
NaN	Not a Number.
NB	Naïve Bayes.
NetBIOS	Network Basic Input/Output System.
NIDS	Network Intrusion Detection System.
NL	Noise Level.
NLOS	Non Line-of-Sight.
NTP	Network Time Protocol.
NTSC	National Television Standard Committee.
OSI	Open Systems Interconnection.

Prec	Precision.
R2L	Remote to Local.
RAM	Random Access Memory.
Rec	Recall.
ReLU	Rectified Linear Unit.
RF	Radio Frequency.
RFo	Random Forest.
RLA	Relative Loss of Accuracy.
RLDR	Relative Loss of Detection Rate.
RMSE	Root Mean Square Error.
RPCA	Robust Principal Component Analysis.
RSS	Received Signal Strength.
SAGE	Space Alternating Generalized Expectation Maximization.
SDN	Software Defined Network.
SDR	Software Defined Radio.
SIR	Signal-to-Interference Ratio.
SMOTE	Synthetic Minority Oversampling Technique.
SNMP	Simple Network Management Protocol.
SNR	Signal-to-Noise Ratio.
SS	Spatial Smoothing.
SSDP	Simple Service Discovery Protocol.
SURE	Stein's Unbiased Risk Estimator.
SVD	Singular Value Decomposition.
Tbps	Terabits per second.
TCP	Transmission Control Protocol.
TDoA	Time Difference of Arrival.
TFTP	Trivial File Transfer Protocol.
TN	True Negative.
TP	True Positive.
TSP	Training Size Proportion.
U2R	User to Root.
UAV	Unmanned Aerial Vehicle.
UDP	User Datagram Protocol.
UNB	University of New Brunswick.

XSS Cross-Site Scripting.

1 INTRODUCTION

1.1 OVERVIEW OF CYBER-PHYSICAL SYSTEMS

Cyber-Physical Systems (CPSs) consist of a set of networked components including sensors, control processing units and communication devices applied on the physical infrastructure monitoring and management [18]. CPSs are typically used on safety-critical applications, such as avionics, instrumentation, defence systems and controlling of critical infrastructures, for instance, electric and nuclear power plants, water resources and telecommunications systems [19]. Consequently, severe consequences can be imposed due to potential cyber and physical attacks, such as customer information leakage, extensive damages to the economy and destruction of infrastructures, endangering human lives [20].

As depicted in Fig. 1.1, typically a CPS architecture is composed by five layers, namely: physical layer, sensor/actuator layer, network layer, control layer, and information layer. First, the physical layer consists of the physical objects or processes monitored by CPSs. Next, the sensor/actuator layer is composed by sensors, which measure data obtained from the physical layer, and by actuators, which execute specific actions under the control of the above layers. For example, in the air traffic control, sensors receive measurement data collected from a sensor array based localization system, whereas actuators are used to neutralize unmanned aerial vehicles detected within the controlled airspace [21]. Following, the network layer is responsible to network sensors and actuators, as well as to connect the sensor/actuator and control layers through communication devices and protocols. Then, the control layer, through intelligent electronic devices, programmable logic controllers and remote terminal units, is responsible for the local distributed control action level. In addition, control layer forwards the measurement data to human operators in the information layer, which monitor the system and take actions whenever required [18].

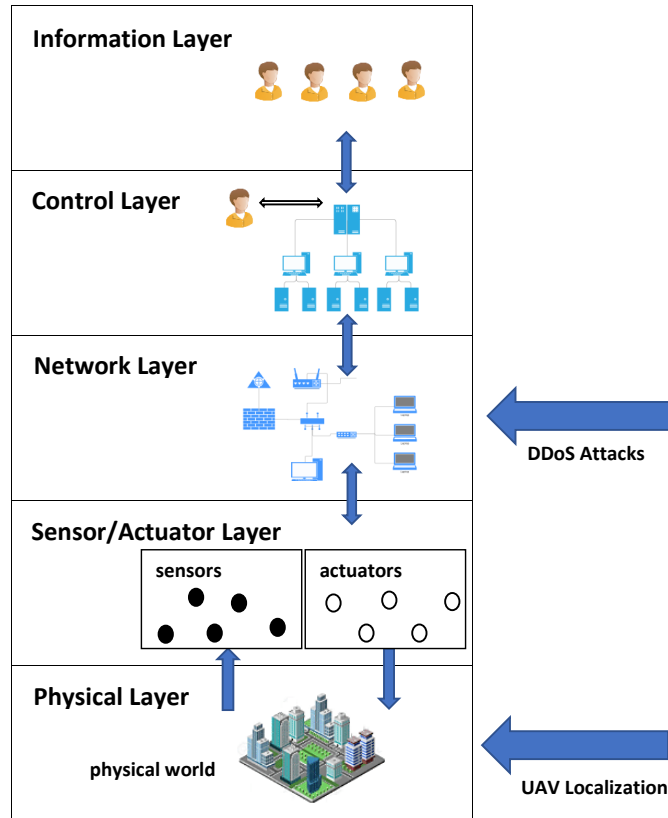


Figure 1.1 – Typical CPS architecture

1.2 MOTIVATION

Due to the reasons presented in Section 1.1, it is crucial to develop highly reliable Intrusion Detection Systems (IDSs) for CPSs such that safety-critical applications can be controlled and protected in an efficient way. Currently, IDSs present highly sophisticated designs, involving advanced signal processing techniques [8–10], as well as Machine Learning (ML) and Deep Learning (DL) based solutions [15, 22, 23]. Particularly considering the physical layer security of a CPS, a complex issue is the localization of unauthorized Unmanned Aerial Vehicles (UAVs) in controlled airspaces. UAVs are remotely piloted aircrafts very popular for personal, commercial, public-safety and military applications, including photography, delivery services, traffic monitoring, disaster monitoring, border patrol and intelligence gathering. Nonetheless, several drone-related accidental and intentional incidents have been increasingly reported, such as mid-air collisions in airports, smuggling of illicit materials, illegal surveillance and terrorist attacks [24]. Consequently, several strategies have been released by governments in order to prevent and combat security threats posed by UAVs, for example, by regulating drone activities or by investing in counter-drone technologies [25]. Table 1.1 summarizes some recent incidents involving UAVs around the globe from 2015 to 2021. Note that most of the reported incidents were related to unautho-

Table 1.1 – Incidents involving UAVs between 2015 and 2021 over the world.

Victim	Date	Characteristics
White House, Washington D.C., USA [33]	Jan., 2015	An inebriated off-duty employee for a government intelligence agency lost control of a drone, which crashed on the grounds of the White House, in Washington D.C., USA.
Power lines in West Hollywood, USA [34]	Oct., 2015	A drone crashed into power lines, causing a three-hour blackout of several hundred houses in West Hollywood, USA
Dubai Airport, United Arab Emirates [26]	Oct., 2016	Four drones invaded the Dubai International Airport, the third busiest in the world, causing an estimated loss of one million dollars after several flights were cancelled or diverted to other airports.
Space Needle, Seattle, USA [31]	Dec., 2016	A drone crashes into the Space Needle observation tower during New Year's Eve fireworks setup in Seattle, USA.
Golden State Race Series, USA [35]	May, 2017	A drone hits a tree and crashes into a cyclist's wheel during the Golden State Race Series in Rancho Cordova, USA.
Aircraft in Quebec, Canada [27]	Oct., 2017	An aircraft with eight people on board, heading to Quebec, Canada, was hit by a UAV at an altitude of 450 meters.
Shenzen, China [37]	Mar., 2018	Criminals busted by custom officers for using drones to smuggle \$79.8 million worth in smartphones to Shenzhen, China.
Prisons across England [38]	Mar., 2018	Ten people were charges for using drones to smuggle drugs and phones into prisons across England.
Gatwick Airport, Sussex, England [30]	Dec., 2018	Two drones were reportedly hovering near the Gatwick Airport in Sussex, England. Moreover, the drones mysteriously appeared whenever the airport tried to reopen. The airport was closed for 33 hours and more than 1,000 flights were cancelled.
Congonhas Airport, São Paulo, Brazil [29]	Jan., 2019	An unauthorized drone paralyzed the Congonhas Airport in São Paulo, Brazil, interrupting the landings and take-offs.
McCarran Airport, Las Vegas, USA [28]	Dec., 2019	Unauthorized drone imposes hazard to aircraft when landing at McCarran Airport in Las Vegas, USA.
US military drones, Syria [36]	Aug., 2020	Two US MQ-9 Reaper drones apparently collided and crashed in Syria, but possibly they were shot down by militants on the ground.
High-rise building, Sydney, Australia [32]	Jan., 2021	Drone used on a commercial photography job crashed into a high rise building in Sydney, Australia, after the pilot reportedly lost control of the drone.

alized flights near airports [26–30], accidental collisions [31–36] and smuggling [37, 38]. An important drone-related incident occurred in 2015, when an off-duty employee for a government intelligence agency lost control of a drone, which accidentally landed near the White House [33]. Such fact raised serious questions about security, since small UAVs could potentially be operated by terrorists against the President of the USA. In addition to the UAV localization task, the correct identification of intruding drones is also fundamental, since specific counter-measures can be more efficiently developed according to the drone brand and its flight mode at the time of the detection/localization.

On the other hand, Distributed Denial of Service (DDoS) attacks are considered a major security threat at the network layer level. In DDoS attacks, available resources of target systems are rapidly exhausted by extremely large volume of traffic launched by attackers in order to intentionally disrupt network services [39]. Consequently, the victim is forced to slow down, crash or shut down due to multiple connection requests during a period of time [40]. Nevertheless, since networks and servers became more robust in identifying network layer DDoS attacks, hackers responded by moving up the Open Systems Interconnection (OSI) model stack to higher layers [41]. For instance, several DDoS attacks exploit vulnerabilities present in the application layer, reproducing the behavior of legitimate customers and,

consequently, are not detected by most of the conventional IDSs [42]. In this context, several researches in the literature broadly classify DDoS attacks into three types: application layer attacks, resource exhaustion attacks, and volumetric attacks [43], which are described as follows.

- **Application Layer Attack:** in this type of attack, vulnerabilities present in the application are used by an attacker, making it inaccessible by legitimate users [43]. Instead of depleting the network bandwidth, the server resources, such as CPU, database, socket connections or memory, are exhausted by application layer attacks. In addition, such attacks present some subtleties which make them harder to detect and mitigate: they can be performed, for instance, through legitimate HyperText Transfer Protocol (HTTP) packets, with a low traffic volume, presenting high resemblance to flash crowds [41]. HTTP and Domain Name System (DNS) based DDoS attacks are examples of application layer attacks.
- **Resource Exhaustion Attack:** Similarly to the application layer attacks, hardware resources of servers, such as memory, CPU, and storage, are also depleted in this category. Nonetheless, resource exhaustion attacks are protocol-based malicious activities, since vulnerabilities in protocols are exploited. For example, in a SYN flood attack, a hacker exploits the Transmission Control Protocol (TCP) three-way handshake process. After receiving a high volume of SYN packets, the targeted server responds with SYN/ACK packets and leaves open ports to receive the final ACK packets, which never arrive. This process continues until all ports of the server are unavailable.
- **Volumetric Attack:** In this type of attack, the bandwidth of the target system is exhausted by a massive amount of traffic. Since such attacks are launched by using amplification and reflection techniques, they are considered as the simplest DDoS attacks to be employed [42]. User Datagram Protocol (UDP) flood and Internet Control Message Protocol (ICMP) flood can be cited as volumetric attacks.

According to the Cisco Annual Internet Report (2018–2023) [44], the total number of DDoS attacks will reach 15.4 million in 2023 globally, as it can be seen in Fig. 1.2. In addition, according to statistics generated by honeypots from the global cybersecurity company Kaspersky [45], in the last quarter of 2020 (Q4), the top 3 countries by number of DDoS attacks were China (58.95%), United States (20.98%) and Hong Kong (3.55%), as shown in Fig. 1.3a. Moreover, the top 10 countries list by number of DDoS targets was similar to the ranking by number of attacks, as it can be seen in Fig. 1.3b. China (44.49%), United States (23.57%) and Hong Kong (7.20%) once again held the top 3 positions. Still in accordance with [45], most of the command and control (C&C) servers used to master botnets in Q4 were located in the United States (36.30%), followed by Netherlands (19.18%) and

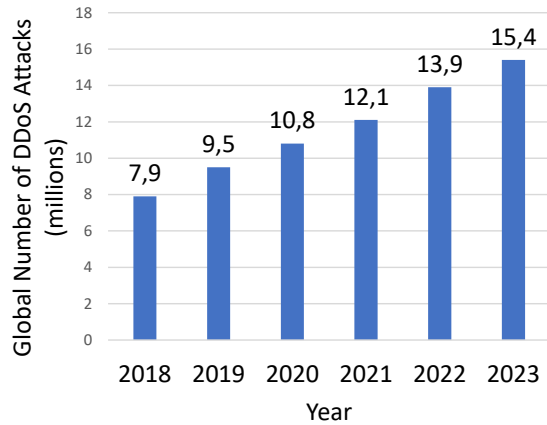


Figure 1.2 – Number of DDoS attacks through the forecast period of 2018-2023 [44].

Germany (8.22%), whereas the three main types of DDoS attack launched during the period were SYN flooding (78.28%), UDP flooding (15.17%) and TCP (5.47%). Such statistics are shown in Fig. 1.4a and 1.4b, respectively. Furthermore, the world’s most recent DDoS attacks with the highest impacts on the victim organizations, from 2012 to 2020, are illustrated in Table 1.2. Note that, in 2016, several massive DDoS attacks were launched by the Mirai botnet [46–48], composed by thousands of Internet of Things (IoT) devices infected by the Mirai malware. Moreover, it can be seen that most of the DDoS attacks targeted well-known information technology companies, including OVH [47], Dyn [48], CloudFlare [49], GitHub [50] and Amazon [51]. The largest DDoS attack recorded so far was carried out against the Google Cloud Team [52] in 2017, with a volume of 2.54 Tbps, but such attack was publicly revealed only in Oct., 2020.

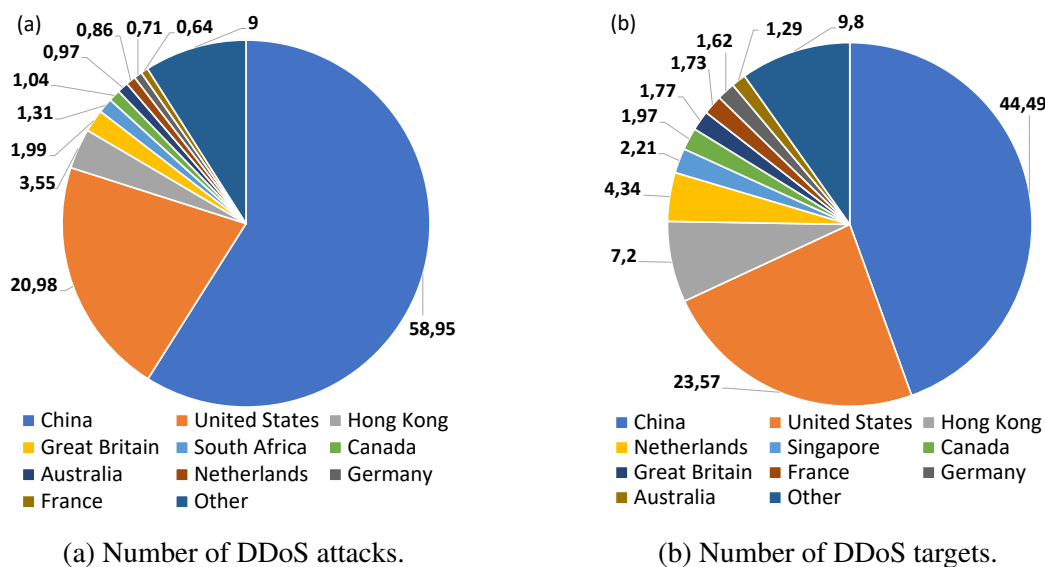


Figure 1.3 – Top 10 countries, in the 4th quarter of 2020, by: (a) number of DDoS attacks. (b) number of DDoS targets [45].

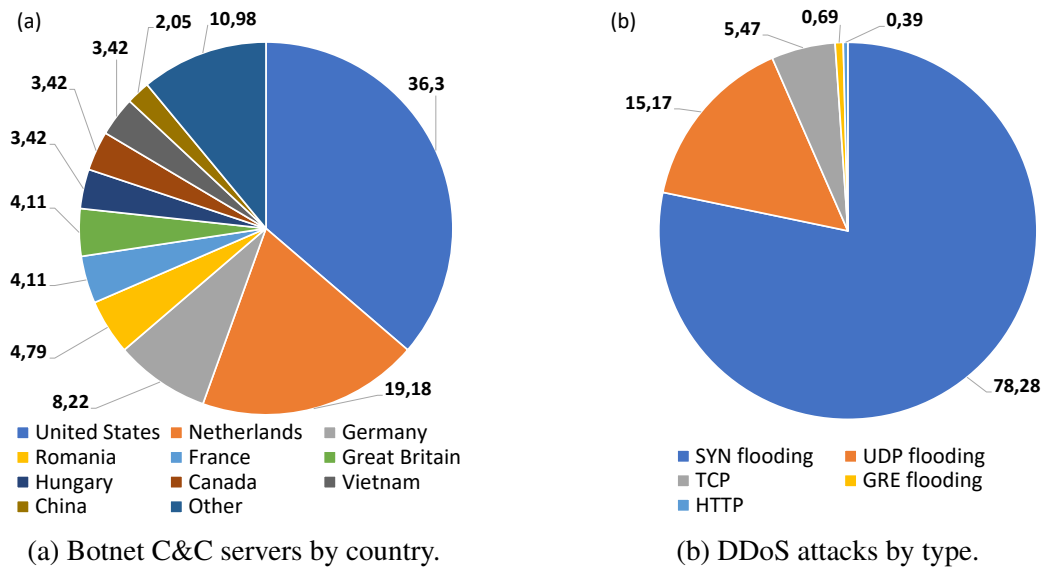


Figure 1.4 – DDoS-related statistics in the 4th quarter of 2020. (a) Distribution of botnet C&C servers by country. (b) Distribution of DDoS attacks by type [45].

1.3 RESEARCH AIM, OBJECTIVES AND CONTRIBUTIONS

The research aim of this thesis is to develop multiple Intrusion Detection Systems (IDS), based on the joint application of multidimensional signal processing techniques and machine learning/deep learning algorithms, in order to detect potential threats on the physical and network layers of Cyber-Physical Systems.

To accomplish such general aim, this thesis addresses several research objectives, which are restructured into the following research questions:

1. How to efficiently localize and identify UAVs in multipath environments by jointly applying multidimensional signal processing techniques and machine learning (ML) algorithms?
2. How to efficiently detect DDoS attacks in a CPS network by jointly applying multidimensional signal processing techniques and machine learning algorithms, assuming that a noisy dataset is used for training and testing?
3. Following the same idea of the previous question, how to efficiently detect DDoS attacks in a CPS network, still applying multidimensional signal processing techniques on noisy datasets, but now using deep learning algorithms instead of ML techniques?
4. How to efficiently detect DDoS attacks in a CPS network by jointly applying multidimensional feature map concatenation based techniques and deep learning algorithms, but now assuming that a noiseless dataset is used for training and testing?

Table 1.2 – The world’s largest DDoS attacks between 2012 and 2020.

Victim	Date	Characteristics
U.S. banks [53]	Sept., 2012	Multiple DDoS attacks were launched against the websites of six U.S. banks, severely disrupting online and mobile banking services. A botnet called Brobot was used to carry out such attacks, which achieved a traffic load of up to 60 Gigabits per second (Gbps).
Occupy Central’s campaign [54]	Oct., 2014	Massive DDoS attacks with approximately 500 Gbps were launched against Occupy Central’s web hosting services in Hong Kong, as well as two independent sites, PopVote and Apple Daily, which supported Occupy Central’s cause.
CloudFlare cybersecurity provider [49]	Feb., 2014	With approximately 400 Gbps, the DDoS attack exploited a vulnerability in the Network Time Protocol (NTP). Despite it was launched at a single CloudFlare’s customers, such attack significantly degraded CloudFlare’s own network.
KrebsOnSecurity blog [46]	Sept., 2016	The 623 Gbps DDoS attack was launched by the Mirai botnet, which was composed by 600,000 compromised Internet of Things (IoT) devices.
OVH web hosting provider [47]	Sept., 2016	The DDoS attack was driven by 145,000 bots from a Mirai botnet, and overwhelmed an OVH customer with a volume of 1.1 Terabits per second (Tbps) during seven days.
Dyn DNS service provider [48]	Oct., 2016	Multiple high-profile Dyn’s client websites, such as GitHub, HBO, Twitter, Reddit, PayPal, Netflix, and Airbnb, were rendered after Dyn’s DNS infrastructure was knocked offline. A DDoS attack of 1.5 Tbps was launched by millions of IP addresses associated to the Mirai botnet.
Google Cloud Team [52]	Sept., 2017	With a volume of 2.54 Tbps, the DDoS attack was carried out by four Chinese internet service providers, targeting thousands of Google’s IP addresses. It is the largest DDoS attack recorded to date, but it was publicly revealed by Google only in Oct., 2020.
GitHub [50]	Feb., 2018	A massive DDoS attack was launched from a thousand different autonomous systems across tens of thousands of unique endpoints by using vulnerable memcached servers, with a volume of 1.35 Tbps during about 20 minutes.
U.S. based service provider customer [55]	Feb., 2018	A 1.7 Tbps reflection/amplification DDoS attack was targeted at a customer of a U.S. based service provider and reported by NETSCOUT Arbor’s DDoS mitigation service providers.
Amazon’s AWS shield service [51]	Feb., 2020	Based on CLDAP reflection, the DDoS attack caused three days of elevated threat during a single week in Feb., 2020, with a traffic load of up to 2.3 Tbps.

In this sense, the aforementioned reasons motivated us to find an appropriate solution for efficiently detecting the threats posed by unauthorized UAVs and Distributed Denial of Service attacks on the physical and network layers of Cyber-Physical Systems, respectively. By taking several novel and different approaches, based on tensor signal processing and machine learning techniques, this thesis can be used as a reference to design state-of-the-art intrusion detection systems by organizations and governments in order to prevent such threats.

Each chapter of this thesis is presented as self-contained as possible such that each research question is presented and discussed independently, allowing more freedom to the reader. At the beginning of each chapter, an overview as well as a detailed list of the contributions are introduced. In addition, Appendices A to E present basic concepts related to the schemes proposed in Chapters 2 to 5.

The first question is addressed in Chapter 2, where the problem of UAV localization and identification in multipath environments is discussed. Usually, a sensor array based localization system is composed by several antenna arrays, where each array receives line-of-sight and non line-of-sight signals emitted from UAVs positioned within the system coverage area. The main goal of such system is to blindly provide the position coordinates of each UAV by applying signal processing techniques on the measurement data collected by its antenna ar-

rays. There are two main issues in this approach. First, since multipath environments are considered, each antenna array receives correlated signals, which reduces the rank of the received signal matrix. Second, antenna arrays must apply denoising schemes on the incoming signals such that the UAV position coordinates are computed more accurately. Therefore, a multidimensional antenna array based framework is proposed such that the structure inherently multidimensional of the data is better exploited by adopting the tensor representation. In addition, a multiple denoising preprocessing scheme is included in order to decorrelate and increase the signal-to-noise ratio of the received signals more efficiently. In summary, the four major contributions of Chapter 2 are:

1. A tensor based framework for localizing multiple UAVs randomly positioned within a region of interest. Since tensors provide a more natural approach to store and manipulate multidimensional data, their UAV localization performance outperforms its matrix based counterpart.
2. The inclusion of a recent denoising preprocessing scheme known as Multiple Denoising (MuDe). Such technique drastically attenuates the noise present in the signals received by each antenna array, which improves the UAV localization performance in terms of the estimated spatial frequency and estimated position coordinates.
3. The inclusion of a machine learning based UAV identification module in order to identify the drone model and its flight mode at the time of the detection/localization. Several ML algorithms are trained and tested by using the recent DroneRF dataset for validation.
4. The performance of the proposed UAV localization and identification model is evaluated using numerical simulations. According to the obtained results, the proposed scheme outperforms state-of-the-art localization techniques.

Next, Chapter 3 deals with the second research question. In order to obtain higher performance, machine learning based Network Intrusion Detection System (NIDS) must be trained with massive amount of data. Nonetheless, a potential drawback consists of the presence of noise in such large datasets, which can be a consequence, for example, of false data injection attacks performed on publicly available datasets. Such fact can degrade the performance of the ML classifier, reducing its reliability and efficiency. In this regard, a novel noise-robust architecture for DDoS attack detection is proposed in Chapter 3. First, the multiple denoising preprocessing scheme is applied for noise attenuation of each dataset instance. In sequel, low-rank approximation is performed in order to denoise the complete dataset. The three major contributions of Chapter 3 are:

1. An extension of the recent MuDe algorithm in order to attenuate the noise present in the instances of DDoS attack detection datasets. Given the outstanding performance

of MuDe to denoise measurement data collected in sensor arrays, such scheme shows a good potential for DDoS attack dataset denoising.

2. The inclusion of a second denoising stage performed by a Low-Rank Approximation (LRA) technique such that a higher degree of noise reduction is achieved, with significant gain on the overall DDoS attack detection performance.
3. The performance of the proposed scheme is validated through numerical simulations by using samples extracted from the CIC-DDoS2019 and NSL-KDD benchmark datasets. According to the obtained results, the proposed framework achieves satisfactory performance, with considerable values of accuracy, detection rate and false alarm rate compared with state-of-the-art low-rank approximation techniques.

The third research question is addressed in Chapter 4. Following the same idea of the previous chapter, a noise-robust Multilayer Perceptron (MLP) based architecture is proposed in order to efficiently detect DDoS attacks when NIDSs are trained with datasets corrupted by Gaussian noise. The research contributions presented in Chapter 4 are:

1. A novel feature extraction method applied on data classification is proposed such that the average value of the common features among dataset instances is iteratively filtered out via the HOSVD algorithm, providing to the NIDS more robustness against data corruption.
2. A noise-robust MLP architecture for DDoS attack detection which applies the technique cited in the previous item is also introduced. The best parameters used for dataset filtering are dynamically computed in order to minimize the errors between the expected and predicted classifications.
3. The performance of the proposed MLP is validated through numerical simulations by using samples extracted from the CIC-DDoS2019, CIC-IDS2018 and CIC-IDS2017 benchmark datasets. According to the simulation results, the proposed scheme outperforms state-of-the-art low-rank approximation techniques in terms of accuracy, detection rate and false alarm rate.

Finally, the fourth research question is dealt in Chapter 5. Two novel tensor feature map concatenation based Convolutional Neural Network (CNN) architectures for DDoS attack detection are proposed. Such schemes present multiple parallel branches, which are composed by several CNNs. The CNN outputs from consecutive parallel branches are concatenated in order to enrich feature diversity and, consequently, a better recognition ability is achieved by the NIDSs. Chapter 5 presents the following research contributions:

1. A tensor feature map concatenation based CNN architecture for DDoS attack detection is proposed. In this approach, multiple branches, composed by CNN basic building blocks alternately positioned with concatenation modules, are placed in parallel such that the outputs from CNNs of consecutive branches are concatenated and sent to the following CNN within each branch. Finally, the outputs from all branches are concatenated and forwarded to the same flattening and multilayer perceptron blocks, where samples are classified as legitimate traffic or DDoS attack.
2. A second improved feature map concatenation based CNN architecture is also introduced. In this scheme, instead of the Flatten and MLP blocks in common for all of the branches, each branch is followed by its respective Flatten and MLP blocks, whose outputs are forwarded to a simple majority voting module in which the final classification is computed.
3. Numerical simulations are conducted in order to validate the proposed schemes by using the CIC-IDS2017 and CIC-DDoS2019 benchmark datasets. The results show that the proposed approaches outperform their competing techniques in terms of several performance evaluation metrics.

Fig. 1.5 presents an overview of the complete network infrastructure depicting the DDoS attack detection and the UAV localization/identification at the network and physical layer levels of a CPS, respectively. In Fig. 1.5a, the DDoS attack infrastructure is represented by three components: attackers, handlers and bots. Attackers send command and control instructions to handlers and bots in order to launch attacks. In addition, handlers are malicious programs installed on compromised hosts which are used to send instructions to the bots through instant messaging or internet relay chat, for instance. Finally, bots are compromised hosts which carry out the large scale distributed attacks [56]. Next, in Fig. 1.5b, the Local Area Network (LAN) of the victim and its internal components, such as computers, LAN servers, Demilitarized Zone (DMZ) servers, routers and firewalls, are represented. In order to efficiently detect and prevent DDoS attacks from the Internet, the proposed models for DDoS attack detection, introduced in Chapters 3 to 5, must be positioned between the router and the firewall. Furthermore, the proposed antenna array based UAV localization and identification system, detailed in Chapter 2, is represented in Fig. 1.5c. Note that such system can be controlled by the LAN administrator via command and control server, which directly communicates with the LAN through the router.

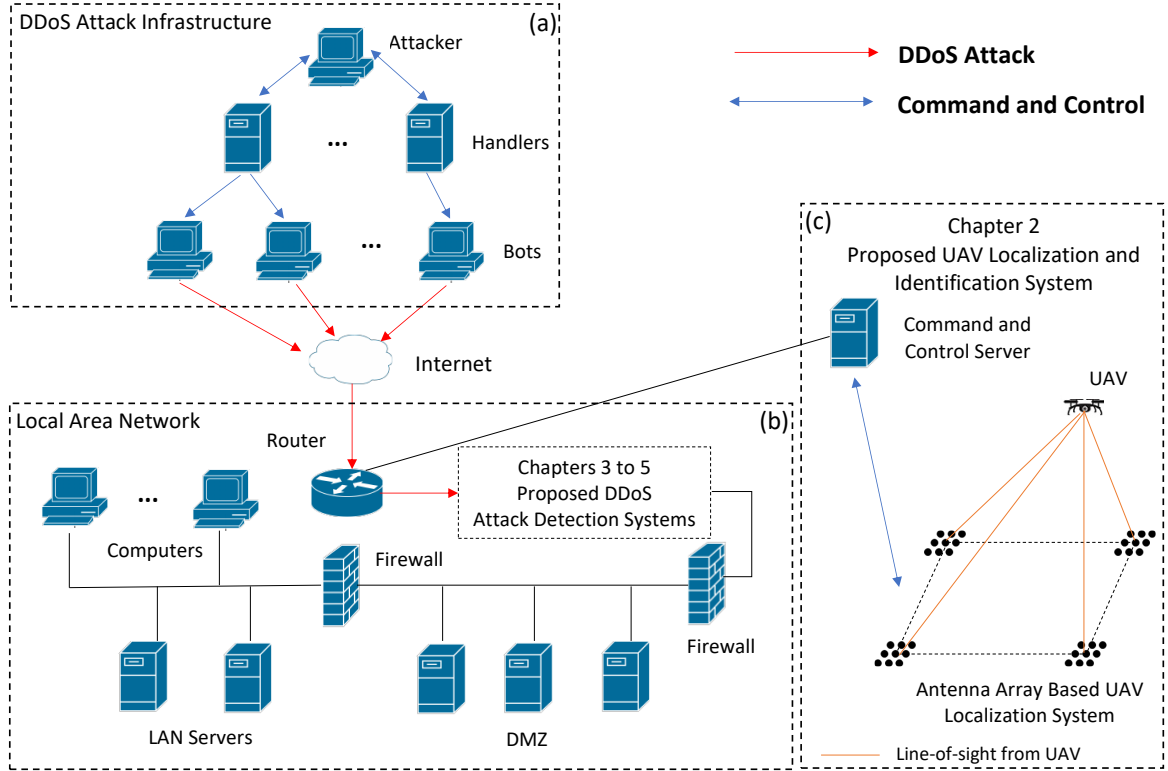


Figure 1.5 – Overview of the complete solution for intrusion detection in CPSs proposed in this thesis. (a) DDoS attack infrastructure. (b) Local Area Network (Target). (c) UAV localization and identification system.

1.4 NOTATION

In this subsection, the mathematical notation used along this thesis is presented. Italic letters (x, y, z) denote scalars, lowercase bold letters denote column vectors $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ and uppercase bold letters denote matrices $(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$. Higher order tensors are represented by uppercase bold calligraphic letters $(\mathcal{X}, \mathcal{Y}, \mathcal{Z})$. The superscript $\{\cdot\}^C$ is used for concatenation of matrices and tensors. Further, the superscripts $\{\cdot\}^T$ and $\{\cdot\}^H$ are used for transposition and Hermitian of a matrix, respectively. Moreover, the operator $\text{diag}\{\cdot\}$ transforms its argument vector into the main diagonal of a diagonal matrix, whereas $\text{vec}\{\cdot\}$ transforms its argument into a vector. The big O operator $\mathcal{O}[\cdot]$ returns the order of magnitude of the number of steps in an algorithm computation, whereas $\text{EV}\{\cdot\}$ returns the eigenvalues of a matrix. The Khatri-Rao product, outer product, Kronecker product and Hadamard product are represented by operators \diamond , \circ , \otimes and \odot , respectively. The matrix $[\mathcal{X}]_{(r)}$ corresponds to the r -th mode unfolding of the tensor \mathcal{X} and can be obtained by varying the r -th index along the rows and stacking all other indices along the columns of $[\mathcal{X}]_{(r)}$. Finally, the r -mode product between \mathcal{X} and \mathcal{Z} is given by $\mathcal{Y} = \mathcal{X} \times_r \mathcal{Z}$, which can also be expressed in a matricized fashion as $[\mathcal{Y}]_{(r)} = \mathbf{Z}[\mathcal{X}]_{(r)}$.

1.5 THESIS ORGANIZATION

The remainder of this thesis is divided as follows. Chapter 2 introduces a tensor based framework for localizing and identifying multiple UAVs randomly positioned within a region of interest considering multipath environments. Next, in Chapter 3, a novel noise-robust architecture for DDoS attack detection based on the Multiple Denoising technique when intrusion detection systems are trained with corrupted datasets is presented. Following the same idea of Chapter 3, a noise-robust multilayer perceptron scheme for DDoS attack detection once again considering corrupted datasets is proposed in Chapter 4. Then, differently from the models presented in Chapters 3 and 4, two novel tensor feature map concatenation based CNN schemes for DDoS attack detection considering noiseless datasets are introduced in Chapter 5. Finally, conclusions are drawn in Chapter 6.

2 TENSOR BASED FRAMEWORK FOR UAV LOCALIZATION AND IDENTIFICATION IN MULTIPATH ENVIRONMENTS

In this chapter, the following research question is addressed: *How to efficiently localize and identify UAVs in multipath environments by jointly applying multidimensional signal processing techniques and machine learning (ML) algorithms?*

The remainder of this chapter is organized as follows:

- **Motivation:** in this section, an introduction to UAV localization and identification in multipath environments is presented.
- **Data Model:** this section defines the data model used throughout this chapter.
- **Proposed Tensor Based Framework for UAV Localization and Identification:** in this section, the proposed tensor based framework for UAV localization and identification in multipath environments is presented and discussed. A tensor representation to better exploit the multidimensional nature of the data is adopted, and a denoising preprocessing scheme is included to increase the signal-to-noise ratio (SNR) of the received signal.
- **Computational Complexity:** this section introduces the computational complexity of the proposed approach.
- **Simulation Results:** the performance of the proposed framework is assessed through numerical simulations. In this section, the improvements introduced by the tensor approach and the multiple denoising scheme are illustrated.

The research contributions presented in this chapter are:

1. A tensor based framework for localizing multiple UAVs randomly positioned within a region of interest. Since tensors provide a more natural approach to store and manipulate multidimensional data, their UAV localization performance outperforms its matrix based counterpart.

2. The inclusion of a recent denoising preprocessing scheme known as Multiple Denoising (MuDe). Such technique drastically attenuates the noise present in the signals received by each antenna array, which improves the UAV localization performance in terms of the estimated spatial frequency and estimated position coordinates.
3. The inclusion of a machine learning based UAV identification module in order to identify the drone model and its flight mode at the time of the detection/localization. Several ML algorithms are trained and tested by using the recent DroneRF dataset for validation.
4. The performance of the proposed UAV localization and identification model is evaluated using numerical simulations. According to the obtained results, the proposed scheme outperforms state-of-the-art localization techniques.

2.1 MOTIVATION

Originally developed for military purposes, Unmanned Aerial Vehicles (UAVs), commonly known as drones, are remotely piloted aircrafts which present several applications such as photography, agriculture, surveillance, search and rescue, traffic monitoring and fire fighting [57, 58]. However, UAVs can also be used for evil purposes, such as espionage, drug cartels and terrorism-related activities [59]. For example, recently UAVs have been used to transport drugs, carry explosives, steal personal privacy and impose serious threats to airplanes taking off and landing in airport zones.

In this sense, in order to deal with such security threats, the development of accurate UAV localization and identification systems is fundamental. Such systems can be consisted of two modules: (i) localization, and (ii) identification. In the former module, a set of antenna arrays are used to estimate the position of a UAV through the application of Direction of Arrival (DoA) techniques. On the other hand, in the latter module, machine learning algorithms are applied on the estimated transmitted signals in order to identify the detected UAV. Additionally, the data captured by the antenna arrays can be processed by some state-of-the-art DoA scheme, such as the Estimation of Signal Parameters via Rotational Invariance Techniques (ESPRIT) [11, 60], which estimates the angle of arrival of the received signals. However, the accuracy of DoA techniques is severely degraded in multipath environments since the antenna arrays receive highly correlated copies of a signal [12, 16, 17]. Thus, additional preprocessing techniques should be applied to remove the effects of coherency and perform the DoA estimation [5, 6]. The spatial smoothing (SS) scheme [61], for example, is a well-known technique used to decorrelate the sources while increasing the number of available snapshots [62].

Several works in the literature propose different solutions for the problem of signal emitter localization. Marinho et al. proposed different solutions regarding localization and tracking. First, in [63], instead of relying on antenna arrays, the authors proposed an alternative to relative sensor localization by employing a crossed dipole antenna in the reception with a known polarization in the transmission. Next, in [64], they proposed the usage of DoA estimation tools to provide continuous authentication such that the user localization and tracking can be determined by using triangulation techniques, whereas the movement patterns are analyzed in order to identify potential fraud indicators. Following, Marinho et al. also proposed a localization and tracking method based on joint DoA, time delay, and range estimation using the Space Alternating Generalized Expectation Maximization (SAGE) algorithm, where no location-based data is exchanged [65]. Finally, the same authors presented a fully passive vehicle localization and tracking method based on DoA estimation which can be used to mitigate the possibility of spoofing or to provide a second independent source of position estimation [66]. In line with the ideas of Marinho et al., a high resolution framework with multipath mitigation for detecting and locating a signal emitter randomly positioned within a region of interest was proposed by Maranhão et al. in [5]. Moreover, in [6], the same authors presented an antenna array based framework for detection and localization of correlated signals, with low average localization errors and average angle of arrival errors. Finally, in [67], Gomes et al. proposed a tensor-based method for joint Direction of Departure (DoD) and Direction of Arrival (DoA) estimation in bistatic Multiple Input Multiple Output (MIMO) radar systems, providing a highly-accurate localization of multiple targets in real-world scenarios when compared to state-of-the-art tensor-based solutions.

Particularly considering researches of UAV localization, several recent works in the literature can be cited. Nam and Joshi [68] presented an approach in which image sensors measure the azimuth and elevation angles of a UAV and send such information to a collector node, where the UAV position is estimated based on the collected samples. Moreover, Chang et al. [59] proposed a Time Difference of Arrival (TDoA) estimation algorithm to improve the accuracy of UAV localization and real-time tracking. Furthermore, an anti-drone system which combines multiple passive surveillance technologies to perform drone detection, localization and radio frequency jamming is proposed by Shi et al. in [69]. In addition, in [70], Miranda et al. proposed an enhanced framework using arrays of directional antennas for DoA estimation in order to localize drones by exploiting their transmitted NTSC signal for different angular spacing between the array antennas. Besides, a similar approach was proposed by Ando et al. [71], in which drones were localized by a DoA estimation algorithm based on the Received Signal Strength (RSS) at the antenna array. At last, Oliveira et al. [72] presented a low cost antenna array based UAV tracking device for outdoor environments, composed by hardware and software parts, which exploits tensor-based techniques.

A number of recent works about UAV identification based on Radio Frequency (RF) sens-

ing approaches can be cited. In [73], Saria Allahham et al. introduced a RF based dataset of UAVs, composed by multiple RF segments collected from three different commercial drones. The recorded segments contain data of five different drone functioning modes, namely, off, on and connected, hovering, flying, and video recording. Additionally, in [25], Al-Sa'd et al. proposed a technique for identifying drones by using the frequency content of the UAV received signals. The Discrete Fourier Transform (DFT) of the RF signal with a fixed window size is used in order to generate several frequency segments, which are forwarded to a deep neural network to detect the UAV and identify its model. Moreover, Saria Allahham et al. [74] proposed an extension of [25] by introducing a deep learning based technique for detecting and identifying UAVs. The recent DroneRF dataset introduced in [73] was used to validate the proposed model. Differently from [25], a multi-channel one-dimensional convolutional neural network was used to detect the drones and recognize their models, such that data channelization combined with feature extraction techniques improved the UAV detection and identification performance. Table 2.1 summarizes the main approaches and drawbacks of each related work, separated by research area, as well as the weaknesses addressed by this chapter.

The high resolution framework proposed by [6] is applied for localizing multiple emitters randomly positioned within a region of interest. However, the authors adopted a classical matrix technique by considering correlated sources in 2-D scenarios. Since tensors provide a more natural approach to store and manipulate multidimensional data, an extension of that work is proposed in this chapter by adopting a tensor representation to localize UAVs in R -D scenarios as well as by including a recent denoising preprocessing scheme known as Multiple Denoising (MuDe) [75]. Additionally, an UAV identification module is added to the framework such that the drone types are recognized through machine learning classification algorithms. The recent DroneRF dataset is used to validate the proposed framework when identifying UAV types. Simulation results confirm that our scheme outperforms both matrix-based and tensor-based approaches, which adopted spatial smoothing preprocessing scheme to decorrelate the signals received at each antenna array. Additionally, our technique showed an outstanding performance for UAV identification when considering different ML classifiers.

Table 2.1 – Related works summary.

Research Area	Ref.	Main Approach	Drawback
Signal Emitter Localization	[5]	- Detection and localization of signal emitters with multipath mitigation.	- Lack of UAV identification module. - The proposed solution is not specifically designed for localizing UAVs.
	[6]	- Detection and localization of signal emitters considering correlated multipath components.	- Lack of UAV identification module. - The proposed solution is not specifically designed for localizing UAVs.
	[63]	- Localization and tracking of signal emitters by using crossed dipole antennas in the reception.	- Lack of UAV identification module. - The proposed solution is not specifically designed for localizing UAVs. - Multipath mitigation is not considered. - Localization of a single emitter.
	[64]	- Localization and tracking of signal emitters by using continuous authentication and triangulation.	- Lack of UAV identification module. - The proposed solution is not specifically designed for localizing UAVs. - Multipath mitigation is not considered. - Localization of a single emitter.
	[65]	- Localization and tracking of signal emitters based on joint DoA, time delay and range estimation using on the SAGE algorithm.	- Lack of UAV identification module. - The proposed solution is not specifically designed for localizing UAVs. - Localization of a single emitter.
	[66]	- Passive vehicle localization and tracking based on DoA estimation to prevent spoofing.	- Lack of UAV identification module. - The proposed solution is not specifically designed for localizing UAVs. - Multipath mitigation is not considered. - Localization of a single emitter.
	[67]	- Localization of multiple targets by using tensor based methods for joint DoD and DoA estimation in MIMO radar systems.	- Lack of UAV identification module. - The proposed solution is not specifically designed for localizing UAVs.
UAV Localization	[59]	- UAV localization and tracking by using the TDoA estimation algorithm based on Gauss priori probability density function and acoustic arrays.	- Lack of UAV identification module.
	[68]	- UAV localization based on the measurement of azimuth and elevation angles through image sensors.	- Lack of UAV identification module. - Localization of a single UAV. - Multipath mitigation is not considered.
	[69]	- UAV detection and localization based on multiple passive surveillance technologies.	- Lack of UAV identification module. - Localization of a single UAV.
	[70]	- UAV localization by exploiting the transmitted NTSC signals for different angular spacing between array antennas.	- Lack of UAV identification module. - Localization of a single UAV. - Multipath mitigation is not considered.
	[71]	- UAV localization based on DoA estimation using directional antenna arrays via RSS.	- Lack of UAV identification module. - Localization of a single UAV. - Multipath mitigation is not considered.
	[72]	- UAV localization and tracking devices based on tensor techniques for outdoor environments.	- Lack of UAV identification module. - Localization of a single UAV.
UAV Identification	[25]	- Construction of a database containing the RF signals of several UAVs under different flight modes.	- Lack of UAV localization module.
	[74]	- Extension of [25] by including a deep learning based algorithm (CNN).	- Lack of UAV localization module.
UAV Localization and Identification	This chapter	- Tensor based framework for UAV localization and identification in multipath environments.	Addressed drawbacks: - The proposed solution is specifically designed for localizing and identifying UAVs. - Localization of multiple UAVs. - Multipath mitigation is considered.

2.2 DATA MODEL

In this section, the data model used throughout this chapter is presented. Let us consider a multidimensional antenna array based localization system composed by U R -D antenna arrays with size $M_1 \times M_2 \times \dots \times M_R$ containing $M = \prod_{r=1}^R M_r$ antennas, where M_r for $r = 1, \dots, R$ is the number of antennas in the r -th spatial dimension. The goal of such

system is to estimate the coordinates $(\hat{x}_q, \hat{y}_q, \hat{z}_q)$ of the q -th UAV for $q = 1, \dots, Q$ within the system coverage region by performing, at the u -th antenna array, a triangulation of the Line-of-Sight (LOS) of the signals emitted from such drone. To accomplish this, the u -th antenna array must estimate the elevation and azimuth angles $\hat{\theta}_{u,\text{los}}^q$ and $\hat{\phi}_{u,\text{los}}^q$ from the LOS of the q -th UAV.

The data matrix $\mathbf{X}_u \in \mathbb{C}^{M \times N}$ received at the u -th antenna array for $u = 1, \dots, U$ from the superposition of Q far-field narrowband LOS signals sampled on N subsequent time instants is given by

$$\mathbf{X}_u = \mathbf{A}_u^{\text{los}} \mathbf{S}_u^{\text{los}} + \mathbf{N}_u, \quad (2.1)$$

where $\mathbf{A}_u^{\text{los}} \in \mathbb{C}^{M \times Q}$, $\mathbf{S}_u^{\text{los}} \in \mathbb{C}^{Q \times N}$ and $\mathbf{N}_u \in \mathbb{C}^{M \times N}$ are, respectively, the array steering matrix, symbols matrix and noise samples matrix. The LOS components correspond to the direct signals received at the u -th antenna array from each UAV localized within the system coverage region. In this sense, the array steering matrix $\mathbf{A}_u^{\text{los}} \in \mathbb{C}^{M \times Q}$, as well as the symbols matrix $\mathbf{S}_u^{\text{los}}$ and the noise samples matrix \mathbf{N}_u , are defined as follows

$$\begin{aligned} \mathbf{A}_u^{\text{los}} &= [k_{u1}^{\text{los}} \mathbf{a}_{u1}^{\text{los}}, \dots, k_{uQ}^{\text{los}} \mathbf{a}_{uQ}^{\text{los}}] \\ &= \left[k_{u1}^{\text{los}} \begin{pmatrix} 1 \\ e^{j \cdot \mu_{u1}^{\text{los}}} \\ e^{j \cdot 2\mu_{u1}^{\text{los}}} \\ \vdots \\ e^{j \cdot (M-1)\mu_{u1}^{\text{los}}} \end{pmatrix}, \dots, k_{uQ}^{\text{los}} \begin{pmatrix} 1 \\ e^{j \cdot \mu_{uQ}^{\text{los}}} \\ e^{j \cdot 2\mu_{uQ}^{\text{los}}} \\ \vdots \\ e^{j \cdot (M-1)\mu_{uQ}^{\text{los}}} \end{pmatrix} \right], \end{aligned} \quad (2.2)$$

$$\mathbf{S}_u^{\text{los}} = \begin{bmatrix} s_{u1}^{\text{los}}(1) & s_{u1}^{\text{los}}(2) & \cdots & s_{u1}^{\text{los}}(N) \\ s_{u2}^{\text{los}}(1) & s_{u2}^{\text{los}}(2) & \cdots & s_{u2}^{\text{los}}(N) \\ \vdots & \vdots & \ddots & \vdots \\ s_{uQ}^{\text{los}}(1) & s_{uQ}^{\text{los}}(2) & \cdots & s_{uQ}^{\text{los}}(N) \end{bmatrix}, \quad (2.3)$$

$$\mathbf{N}_u = \begin{bmatrix} n_{u1}(1) & n_{u1}(2) & \cdots & n_{u1}(N) \\ n_{u2}(1) & n_{u2}(2) & \cdots & n_{u2}(N) \\ \vdots & \vdots & \ddots & \vdots \\ n_{uM}(1) & n_{uM}(2) & \cdots & n_{uM}(N) \end{bmatrix}, \quad (2.4)$$

where μ_{uq}^{los} is the spatial frequency and k_{uq}^{los} is a constant which controls the power of the LOS signal received from the q -th UAV at the u -th antenna array.

If the multidimensional antenna array based localization system is positioned within a

multipath environment, the data matrix \mathbf{X}_u in (2.1) must be rewritten as the sum of the LOS plus Non Line-of-Sight (NLOS) terms as follows

$$\mathbf{X}_u = \mathbf{A}_u^{\text{los}} \mathbf{S}_u^{\text{los}} + \sum_{q=1}^Q \mathbf{A}_u^{q,\text{nlos}} \mathbf{S}_u^{q,\text{los,C}} + \mathbf{N}_u, \quad (2.5)$$

where $\mathbf{S}_u^{q,\text{los,C}} \in \mathbb{C}^{G \times N}$ contains G copies of the symbols vector of the q -th UAV, $\mathbf{S}_u^{\text{los}}(q, :) = [s_{uq}^{\text{los}}(1), \dots, s_{uq}^{\text{los}}(N)]$, stacked along the 1st dimension. In this chapter, it is assumed that the delay between the LOS signal and each one of the NLOS components is negligible. Therefore, $\mathbf{A}_u^{q,\text{nlos}} \in \mathbb{C}^{M \times G}$, which is the array steering matrix corresponding to the G non line-of-sight signals received from the q -th UAV at the u -th antenna array, can be denoted as

$$\begin{aligned} \mathbf{A}_u^{q,\text{nlos}} &= [k_{u1}^{q,\text{nlos}} \mathbf{a}_{u1}^{q,\text{nlos}}, \dots, k_{uG}^{q,\text{nlos}} \mathbf{a}_{uG}^{q,\text{nlos}}] \\ &= \left[k_{u1}^{q,\text{nlos}} \begin{pmatrix} 1 \\ e^{j \cdot \mu_{u1}^{q,\text{nlos}}} \\ e^{j \cdot 2\mu_{u1}^{q,\text{nlos}}} \\ \vdots \\ e^{j \cdot (M-1)\mu_{u1}^{q,\text{nlos}}} \end{pmatrix}, \dots, k_{uG}^{q,\text{nlos}} \begin{pmatrix} 1 \\ e^{j \cdot \mu_{uG}^{q,\text{nlos}}} \\ e^{j \cdot 2\mu_{uG}^{q,\text{nlos}}} \\ \vdots \\ e^{j \cdot (M-1)\mu_{uG}^{q,\text{nlos}}} \end{pmatrix} \right], \end{aligned} \quad (2.6)$$

where $\mathbf{a}_{ug}^{q,\text{nlos}} = [1, e^{j \cdot \mu_{ug}^{q,\text{nlos}}}, \dots, e^{j \cdot (M-1)\mu_{ug}^{q,\text{nlos}}}]^T$ is the array response, $\mu_{ug}^{q,\text{nlos}}$ is the spatial frequency and $k_{ug}^{q,\text{nlos}}$ is a constant which controls the power of the g -th multipath signal. The values of k_{uq}^{los} and $k_{ug}^{q,\text{nlos}}$ in (2.2) and (2.6), respectively, can be defined according to a frequency-flat Rician environment such that a wireless channel with different degrees of scattering richness can be represented [76]. Further, the symbols matrix $\mathbf{S}_u^{q,\text{los,C}} \in \mathbb{C}^{G \times N}$ in (2.5) can be expressed as

$$\mathbf{S}_u^{q,\text{los,C}} = \begin{bmatrix} s_{uq}^{\text{los}}(1) & s_{uq}^{\text{los}}(2) & \dots & s_{uq}^{\text{los}}(N) \\ s_{uq}^{\text{los}}(1) & s_{uq}^{\text{los}}(2) & \dots & s_{uq}^{\text{los}}(N) \\ \vdots & \vdots & \ddots & \vdots \\ s_{uq}^{\text{los}}(1) & s_{uq}^{\text{los}}(2) & \dots & s_{uq}^{\text{los}}(N) \end{bmatrix}. \quad (2.7)$$

Since the u -th antenna array receives one direct signal plus G multipath components from each one of the Q UAVs, the total number of received signals is given by $D = Q(G + 1)$. In this sense, the data matrix $\mathbf{X}_u \in \mathbb{C}^{M \times N}$ defined in (2.5) for multipath environments can be rewritten as a superposition of D far-field narrowband signals, i.e.,

$$\mathbf{X}_u = \mathbf{A}_u \mathbf{S}_u + \mathbf{N}_u, \quad (2.8)$$

where $\mathbf{A}_u \in \mathbb{C}^{M \times D}$ and $\mathbf{S}_u \in \mathbb{C}^{D \times N}$ are, respectively, the array steering matrix and the symbols matrix related to the D LOS plus NLOS received signals, whose structures are similar to the ones shown in (2.2) and (2.3).

Additionally, if the signal presents a multidimensional structure, the received data matrix in (2.8) can be denoted as

$$\mathbf{X}_u = (\mathbf{A}_u^{(1)} \diamond \mathbf{A}_u^{(2)} \diamond \dots \diamond \mathbf{A}_u^{(R)}) \mathbf{S}_u + \mathbf{N}_u, \quad (2.9)$$

where $\mathbf{A}_u = \mathbf{A}_u^{(1)} \diamond \mathbf{A}_u^{(2)} \diamond \dots \diamond \mathbf{A}_u^{(R)}$, and $\mathbf{A}_u^{(r)} = [\mathbf{a}_{u1}^{(r)}, \dots, \mathbf{a}_{uD}^{(r)}]$ corresponds to the array steering matrix of the r -th array dimension at the u -th antenna array. The array response $\mathbf{a}_{ud}^{(r)}$ of the d -th received signal for $d = 1, \dots, D$ presents the following structure

$$\mathbf{a}_{ud}^{(r)} = [1, e^{j \cdot \mu_{ud}^{(r)}}, e^{j \cdot 2 \mu_{ud}^{(r)}}, \dots, e^{j \cdot (M_r - 1) \mu_{ud}^{(r)}}]^\top, \quad (2.10)$$

where $\mu_{ud}^{(r)}$ is the spatial frequency of the d -th wavefront in the r -th array dimension at the u -th antenna array.

If we wish to naturally explore the multidimensional structure of the data [8], the received signal can be represented as a tensor $\mathbf{X}_u \in \mathbb{C}^{M_1 \times M_2 \times \dots \times M_R \times N}$, which is given by

$$\begin{aligned} \mathbf{X}_u &= \mathbf{X}_{u0} + \mathbf{N}_u \\ &= \mathbf{J}_{R+1, D} \times_1 \mathbf{A}_u^{(1)} \dots \times_R \mathbf{A}_u^{(R)} \times_{R+1} \mathbf{S}_u^\top + \mathbf{N}_u, \end{aligned} \quad (2.11)$$

where $\mathbf{J}_{R+1, D}$ is the identity tensor of order $R + 1$ in which each dimension has size D , and $\mathbf{N}_u \in \mathbb{C}^{M_1 \times M_2 \times \dots \times M_R \times N}$ is the noise samples tensor obtained by folding \mathbf{N}_u as a tensor of order $R + 1$ [75].

Finally, the r -th unfolding matrix $[\mathbf{X}_u]_{(r)} \in \mathbb{C}^{M_r \times \prod_{j \neq r} M_j N}$ can be expressed as

$$\begin{aligned} [\mathbf{X}_u]_{(r)} &= [\mathbf{X}_{u0}]_{(r)} + [\mathbf{N}_u]_{(r)} \\ &= \mathbf{A}_u^{(r)} (\mathbf{S}_u^\top \diamond \mathbf{A}_u^{(R)} \diamond \dots \diamond \mathbf{A}_u^{(r+1)} \diamond \mathbf{A}_u^{(r-1)} \diamond \dots \diamond \mathbf{A}_u^{(1)}) + [\mathbf{N}_u]_{(r)}. \end{aligned} \quad (2.12)$$

If the signals received by the u -th R -D antenna array are non correlated, the rank of the signal subspace is equal to the number of sources. Otherwise, if two or more sources are coherent or if not enough snapshots N are available, such condition is not fulfilled [62]. In order to solve such problem, the well-known Spatial Smoothing (SS) preprocessing scheme [61] can be applied to decorrelate the sources and increase N .

According to the SS approach, the r -th unfolding matrix $[\mathbf{X}_u]_{(r)}$ given by (2.12) is divided

into L_r subarrays of size $M_r^{(\text{sub})} = M_r - L_r + 1$ each. The r -th mode spatially smoothed matrix $\mathbf{X}_{\text{SS,ur}}^{(L_r)} \in \mathbb{C}^{M_r^{(\text{sub})} \times \prod_{j \neq r} M_j L_r K}$ for $r = 1, \dots, R$ is defined as

$$\begin{aligned} \mathbf{X}_{\text{SS,ur}}^{(L_r)} &= [[\mathbf{x}_u]_{(r)}^{(1)}, [\mathbf{x}_u]_{(r)}^{(2)}, \dots, [\mathbf{x}_u]_{(r)}^{(L_r)}] \\ &= [\mathbf{J}_1^{(N_r)} [\mathbf{x}_u]_{(r)}, \mathbf{J}_2^{(N_r)} [\mathbf{x}_u]_{(r)}, \dots, \mathbf{J}_{L_r}^{(N_r)} [\mathbf{x}_u]_{(r)}], \end{aligned} \quad (2.13)$$

where $\mathbf{J}_{l_r}^{(N_r)} = [\mathbf{0}_{M_r^{(\text{sub})} \times (l_r - 1)}, \mathbf{I}_{M_r^{(\text{sub})}}, \mathbf{0}_{M_r^{(\text{sub})} \times (L_r - l_r)}]$ is the selection matrix, whereas $[\mathbf{x}_u]_{(r)}^{(l_r)} = \mathbf{J}_{l_r}^{(N_r)} [\mathbf{x}_u]_{(r)} \in \mathbb{C}^{M_r^{(\text{sub})} \times \prod_{j \neq r} M_j N}$ for $l_r = 1, \dots, L_r$ is the output signal of the l_r -th subarray in the r -th dimension at the u -th antenna array.

2.3 PROPOSED TENSOR BASED FRAMEWORK FOR UAV LOCALIZATION AND IDENTIFICATION

This section introduces the proposed tensor based framework for UAV localization and identification in multipath environments. As mentioned in Section 2.1, the framework proposed by [6] is extended in three aspects: (i) by adopting a tensor representation to better explore the structure inherently multidimensional of the data, (ii) by including a denoising preprocessing scheme to increase the signal-to-noise ratio of the received signal, and (iii) by including a machine learning classification module for identifying the UAV type.

In this chapter, the following assumptions are made for deriving the proposed framework:

- The UAV localization system is composed by U antenna arrays and provides the position coordinates of Q drones flying within its coverage region.
- The UAV localization system is placed within a multipath environment. In this sense, the u -th antenna array for $u = 1, \dots, U$ receives D multiple delayed copies of the LOS signal emitted from the q -th UAV for $q = 1, \dots, Q$, i.e., $D > Q$.
- The UAV localization system is composed by U antenna arrays, all of them positioned along the xy -plane in a Cartesian coordinate system, i.e., with coordinates $(x_u, y_u, 0)$ for $u = 1, \dots, U$.
- The number of received signals D can be estimated by using Model Order Selection (MOS) schemes [7, 13, 14], such as the Akaike's Information Theoretic Criteria (AIC) [77], Efficient Detection Criterion (EDC) [78], Minimum Description Length (MDL) [79] or RADOI [80]. Nonetheless, the model order estimation is beyond the scope of this chapter and, consequently, D is assumed to be known.

- The spacing between the antenna array elements is given by $\lambda/2$, where λ is the wavelength of the radio signals transmitted by the detected drones.

Before introducing the proposed tensor based framework for UAV localization and identification in multipath environments, some important concepts regarding the azimuth and elevation angles at an antenna array are discussed. Fig. 2.1a illustrates an overview of the antenna array based UAV localization system used in our proposed framework. The system is composed by U two-dimensional antenna arrays used to estimate the position coordinates $(\hat{x}_q, \hat{y}_q, \hat{z}_q)$ of the q -th drone within the system coverage region, which is delimited by the black dotted lines along the xy -plane. Furthermore, the u -th antenna array for $u = 1, \dots, U$ receives a LOS signal emitted from the q -th UAV, represented by red color lines. In addition, Fig. 2.1b illustrates the azimuth and elevation angles, $\hat{\phi}_{u,\text{los}}^q$ and $\hat{\theta}_{u,\text{los}}^q$, respectively, at the u -th antenna array. The azimuth angle corresponds to the acute angle between the y -direction and the projection of the LOS onto the xy -plane. However, due to the different relative positions between each antenna array and the UAV, $\hat{\phi}_{u,\text{los}}^q$ must be written as a function of $\hat{\varphi}_{u,\text{los}}^q$, which is the angle between the y -direction and the LOS projected onto the xy -plane in the counterclockwise rotation, i.e.,

$$\hat{\phi}_{u,\text{los}}^q = \begin{cases} \hat{\varphi}_{u,\text{los}}^q, & 0^\circ < \hat{\varphi}_{u,\text{los}}^q < 90^\circ, \\ 180^\circ - \hat{\varphi}_{u,\text{los}}^q, & 90^\circ < \hat{\varphi}_{u,\text{los}}^q < 180^\circ, \\ 180^\circ + \hat{\varphi}_{u,\text{los}}^q, & 180^\circ < \hat{\varphi}_{u,\text{los}}^q < 270^\circ, \\ 360^\circ - \hat{\varphi}_{u,\text{los}}^q, & 270^\circ < \hat{\varphi}_{u,\text{los}}^q < 360^\circ, \end{cases} \quad (2.14)$$

$u = 1, \dots, U.$

This fact is detailed in Fig. 2.2, which shows four different relative positions between the u -th URA and the q -th UAV. For example, in Fig. 2.2a, $\hat{\varphi}_{u,\text{los}}^q$ is acute and, consequently, it coincides with the azimuth angle, i.e., $\hat{\phi}_{u,\text{los}}^q = \hat{\varphi}_{u,\text{los}}^q$. On the other hand, in Fig. 2.2d, $\hat{\varphi}_{u,\text{los}}^q$ is between 270° and 360° and, thus, the azimuth angle is adjusted to $\hat{\phi}_{u,\text{los}}^q = 360^\circ - \hat{\varphi}_{u,\text{los}}^q$ due to the new relative position between the UAV and URA.

After discussing the aforementioned concepts, the proposed tensor based framework for UAV localization and identification in multipath environments is introduced. The proposed approach, depicted by the diagram shown in Fig. 2.3, is composed by two main modules: (i) localization, and (ii) identification. The former module contains five blocks, whereas the latter is composed by two blocks, as detailed in Subsections 2.3.1 to 2.3.7

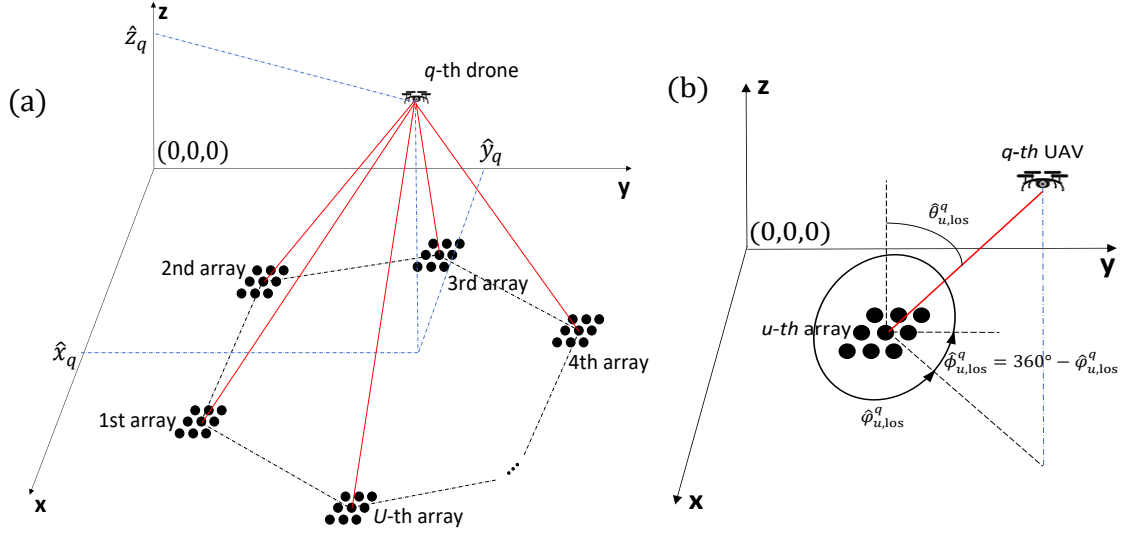


Figure 2.1 – (a) Overview of the antenna array based UAV localization system. (b) The azimuth angle at the u -th antenna array corresponds to the acute angle between the y -direction and the projection of the LOS onto the xy -plane, i.e., $\hat{\phi}_{u,\text{los}}^q = 360^\circ - \hat{\phi}_{u,\text{los}}^q$.

2.3.1 Decorrelation and Multiple Denoising Preprocessing

Block 1.1 of Fig. 2.3 corresponds to the decorrelation and denoising preprocessing. In this step, the Signal-to-Noise Ratio (SNR) of the received signals is increased by applying the recent Multiple Denoising (MuDe) scheme, which is composed of three successive phases: spatial smoothing, low-rank approximation and reconstruction [75].

Initially, the r -th mode spatially smoothed matrix $\mathbf{X}_{\text{SS},ur}^{(l_r)}$ is constructed according to (2.13) for the l_r -th subarray size in the r -th array dimension at the u -th antenna array. Next, assuming that the number of received signal components D is known, the low-rank approximation $\tilde{\mathbf{X}}_{\text{SS},ur}^{(l_r)}$ of the spatially smoothed matrix $\mathbf{X}_{\text{SS},ur}^{(l_r)}$ is computed by truncating the Singular Value Decomposition (SVD) of $\mathbf{X}_{\text{SS},ur}^{(l_r)}$ to the signal subspace with the condition $M_r^{(\text{sub})} \geq D$, i.e.,

$$\tilde{\mathbf{X}}_{\text{SS},ur}^{(l_r)} = [[\tilde{\mathbf{x}}_u]_{(r)}^{(1)}, [\tilde{\mathbf{x}}_u]_{(r)}^{(2)}, \dots, [\tilde{\mathbf{x}}_u]_{(r)}^{(l_r)}] = \mathbf{U}_s^{(l_r)} \boldsymbol{\Sigma}_s^{(l_r)} \mathbf{V}_s^{(l_r)H}, \quad (2.15)$$

where the columns of $\mathbf{U}_s^{(l_r)} \in \mathbb{R}^{N_r \times D}$ and $\mathbf{V}_s^{(l_r)} \in \mathbb{R}^{\prod_{j \neq r} M_j \times D}$ correspond to the singular vectors of $\tilde{\mathbf{X}}_{\text{SS},ur}^{(l_r)}$, whereas the diagonal of $\boldsymbol{\Sigma}_s^{(l_r)} \in \mathbb{R}^{D \times D}$ contains the singular values of $\tilde{\mathbf{X}}_{\text{SS},ur}^{(l_r)}$.

Finally, the multiple denoised unfolding matrix $[\tilde{\mathbf{x}}_u]_{(r)} \in \mathbb{C}^{M_r \times \prod_{j \neq r} M_j N}$ is obtained as

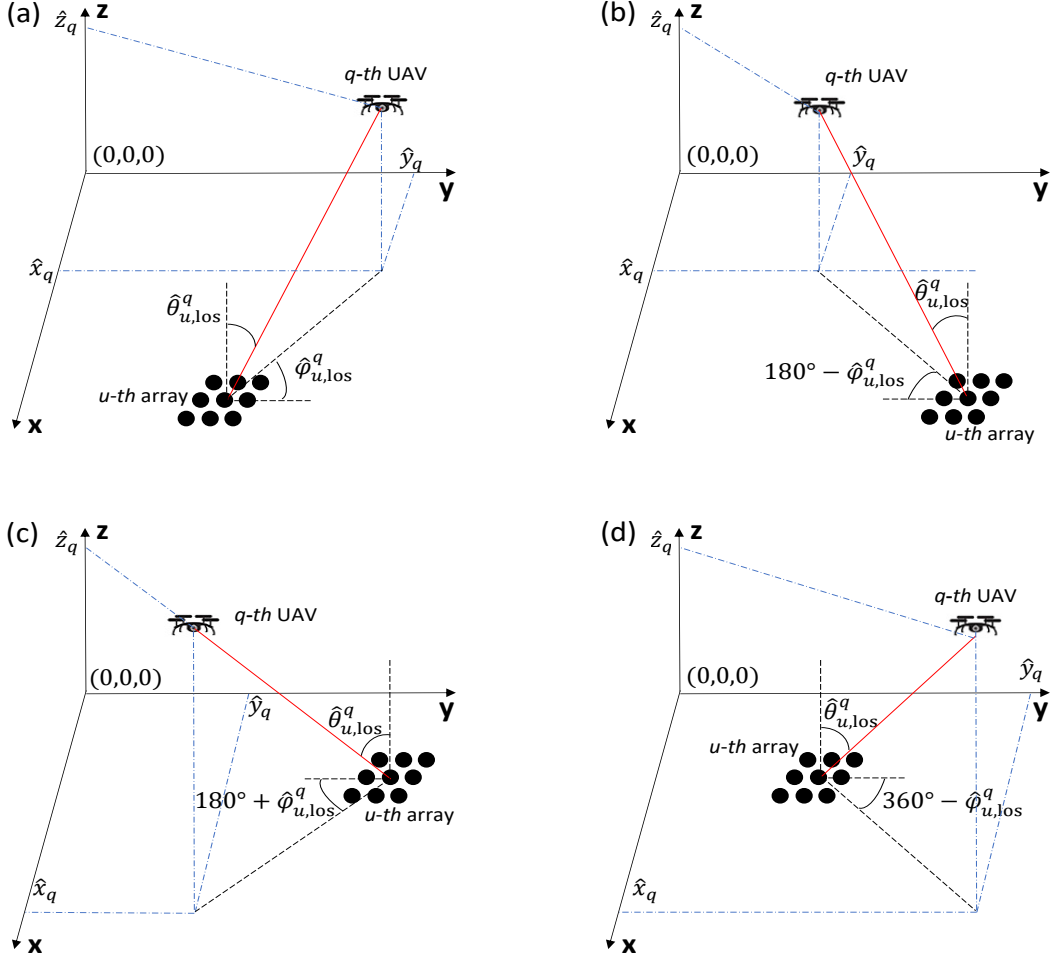


Figure 2.2 – Elevation and azimuth angles for four different relative positions between the u -th URA and the q -th UAV: (a) $0^\circ < \hat{\varphi}_{u,\text{los}}^q < 90^\circ$. (b) $90^\circ < \hat{\varphi}_{u,\text{los}}^q < 180^\circ$. (c) $180^\circ < \hat{\varphi}_{u,\text{los}}^q < 270^\circ$. (d) $270^\circ < \hat{\varphi}_{u,\text{los}}^q < 360^\circ$.

follows

$$[\tilde{\mathbf{x}}_u]_{(r)} = \begin{bmatrix} [\tilde{\mathbf{x}}_u]_{(r)}(1, :) \\ [\tilde{\mathbf{x}}_u]_{(r)}(2, :) \\ \vdots \\ [\tilde{\mathbf{x}}_u]_{(r)}(M_r, :) \end{bmatrix}, \quad (2.16)$$

$$[\tilde{\mathbf{x}}_u]_{(r)}(n, :) = \left(\frac{1}{l}\right) \sum_{i=1}^{l_r} [\tilde{\mathbf{x}}_u]_{(r)}^{(i)}(n-i+1, :), \quad (2.17)$$

where l corresponds to the number of times in which $[\tilde{\mathbf{x}}_u]_{(r)}^{(i)}(n-i+1, :)$ is a valid output in the l_r -th subarray of the r -th dimension at the u -th antenna array [75].

After covering all possible subarray sizes for each dimension, $[\tilde{\mathbf{x}}_u]_{(r)}$ is folded into the multiple denoised tensor $\tilde{\mathbf{x}}_u \in \mathbb{C}^{M_1 \times \dots \times M_R \times N}$ with order $R+1$.

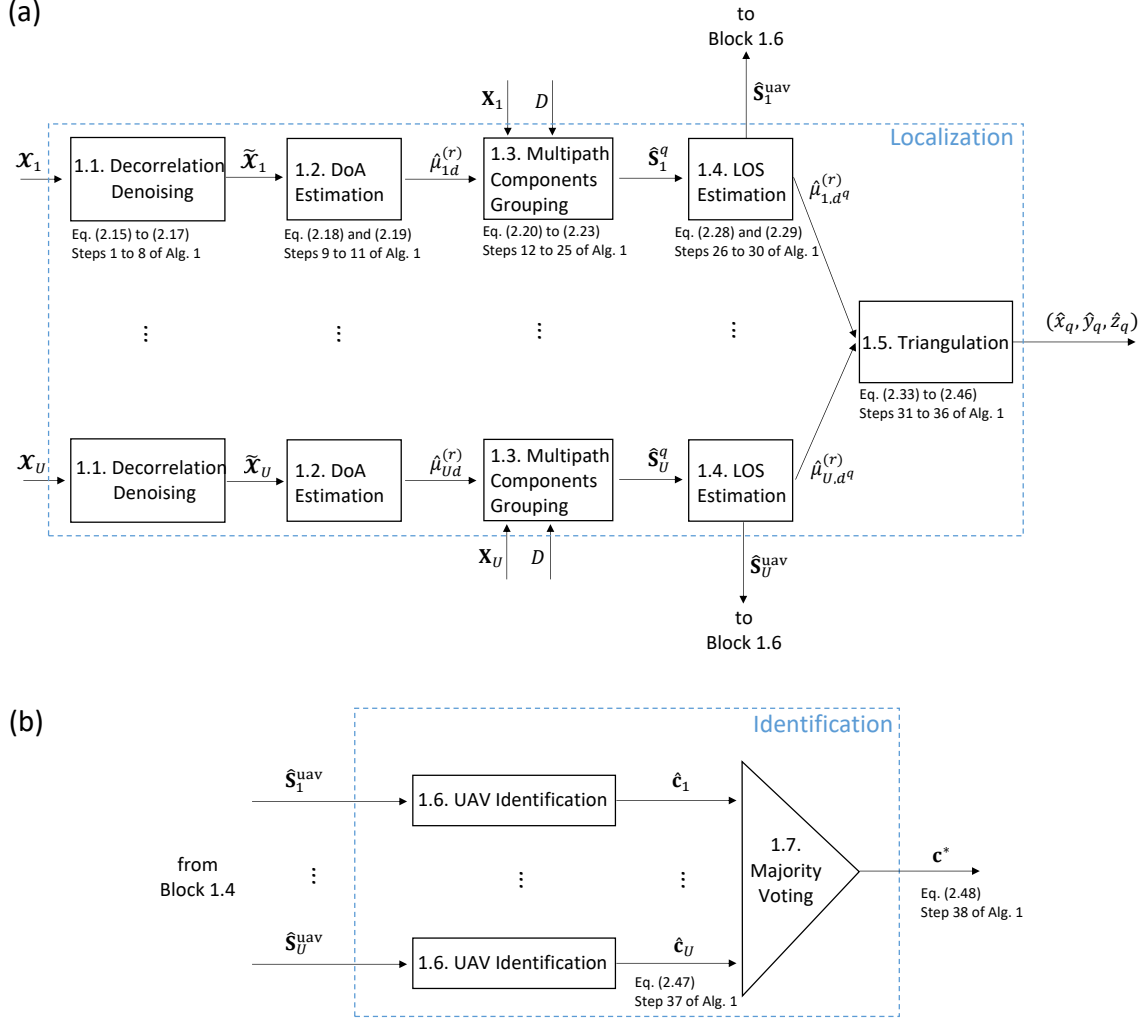


Figure 2.3 – Block diagram of the proposed framework, which is composed by two main modules: (a) UAV localization, and (b) UAV identification. Each block presents the respective references to steps of Algorithm 1 as well as equations of Subsections 2.3.1 to 2.3.7.

2.3.2 Direction of Arrival Estimation

Block 1.2 of the diagram described in Fig. 2.3 corresponds to the DoA estimation. In this chapter, the R -dimensional subspace method known as R -D Standard Tensor ESPRIT, proposed by Haardt et al. [62], is considered.

The eigenvalues of the matrices $\Psi_u^{(r)}$ are estimates of the spatial frequencies $e^{j \cdot \hat{\mu}_{ud}^{(r)}}$ for $d = 1, \dots, D$ and $r = 1, \dots, R$, i.e., $[e^{j \cdot \hat{\mu}_{u1}^{(r)}}, \dots, e^{j \cdot \hat{\mu}_{uD}^{(r)}}] = \text{EV}\{\Psi_u^{(r)}\}$. The values of $\Psi^{(r)}$ are given by the following relationship

$$\Psi_u^{(r)\top} = \left(\tilde{\mathbf{J}}_1^{[r]} \cdot [\mathbf{u}_u^{[s]}]_{(R+1)}^\top \right)^+ \cdot \tilde{\mathbf{J}}_2^{[r]} \cdot [\mathbf{u}_u^{[s]}]_{(R+1)}^\top, \quad (2.18)$$

where $\tilde{\mathbf{J}}_i^{[r]} = \mathbf{I}_{\Gamma_1^{(r)}} \otimes \mathbf{J}_i^{[r]} \otimes \mathbf{I}_{\Gamma_2^{(r)}}$ for $i = 1, 2$ are the selection matrices for the r -th mode,

$\Gamma_1^{(r)} = \prod_{q=1}^{r-1} M_q$ and $\Gamma_2^{(r)} = \prod_{q=r+1}^R M_q$. The tensor $\mathbf{u}_u^{[s]}$ is defined as follows

$$\mathbf{u}_u^{[s]} = \mathbf{g}_u^{[s]} \times_1 \mathbf{U}_{u1}^{[s]} \times_2 \mathbf{U}_{u2}^{[s]} \cdots \times_R \mathbf{U}_{uR}^{[s]}, \quad (2.19)$$

where $\mathbf{U}_{ur}^{[s]} \in \mathbb{C}^{M_r \times p_r}$ and $\mathbf{g}_u^{[s]} \in \mathbb{C}^{p_1 \times p_2 \cdots \times p_R \times D}$ for $r = 1, \dots, R$ are computed from an ‘‘economy size’’ Higher Order Singular Value Decomposition (HOSVD) of the denoised tensor $\tilde{\mathbf{x}}_u \in \mathbb{C}^{M_1 \times M_2 \times \cdots \times M_R \times N}$ obtained from Block 1 of Fig. 2.3, and $p_r = \min\{M_r, D\}$.

2.3.3 Multipath Components Grouping

Usually, the signals with the highest power received at an antenna array are considered as the line-of-sight of their corresponding sources. However, this assumption is not always valid, since NLOS signals from closer sources may be received with a higher power compared to the LOS signals from distant sources. Such fact is exemplified in Fig. 2.4, which depicts two different scenarios where the LOS plus NLOS signals from two UAVs are received at an antenna array. The power of each signal is proportional to the thickness of the arrows, such that, for a given UAV, the LOS signal has the highest power compared to the multipath components and, consequently, it is represented by a thicker arrow. In Fig. 2.4a, both UAVs are localized at similar distances from the antenna array and, consequently, the received signals with the highest power, LOS₁ and LOS₂, coincide with the line-of-sight of each UAV. Nonetheless, an exceptional situation is depicted in Fig. 2.4b. Since UAV 2 is closer to the antenna array, one of its non line-of-sight signals, NLOS_{2,2}, is received with a higher power compared to the line-of-sight signal from the UAV 1, LOS₁, which is much farther from the array. In this sense, by applying the multipath components grouping method described in this chapter, the component with the highest power from a given UAV necessarily corresponds to its line-of-sight.

Block 1.3 of Fig. 2.3 represents the multipath components grouping at the u -th R -D antenna array, in which the signals emitted from the q -th UAV are grouped into a matrix. First, the estimated array steering matrix $\hat{\mathbf{A}}_u = \hat{\mathbf{A}}_u^{(1)} \diamond \hat{\mathbf{A}}_u^{(2)} \diamond \cdots \diamond \hat{\mathbf{A}}_u^{(R)} \in \mathbb{C}^{M \times D}$ is rebuilt, where $\hat{\mathbf{A}}_u^{(r)} = [\hat{\mathbf{a}}_{u1}^{(r)}, \dots, \hat{\mathbf{a}}_{uD}^{(r)}]$ for $r = 1, \dots, R$. Moreover, each vector $\hat{\mathbf{a}}_{ud}^{(r)}$ is given by (2.10) with the spatial frequencies $\hat{\mu}_{ud}^{(r)}$ for $d = 1, \dots, D$ estimated from the previous block. Next, the SVD-based low-rank approximation is applied on the received data matrix $\mathbf{X}_u \in \mathbb{C}^{M \times N}$ truncated to D singular values,

$$\mathbf{X}_u = \mathbf{U}_u \Sigma_u \mathbf{V}_u^H, \quad (2.20)$$

where the columns of $\mathbf{U}_u \in \mathbb{C}^{M \times D}$ and $\mathbf{V}_u \in \mathbb{C}^{N \times D}$ correspond to the left singular vectors and right singular vectors of \mathbf{X}_u , respectively, and the diagonal of $\Sigma_u \in \mathbb{C}^{D \times D}$ contains the singular values of \mathbf{X}_u .

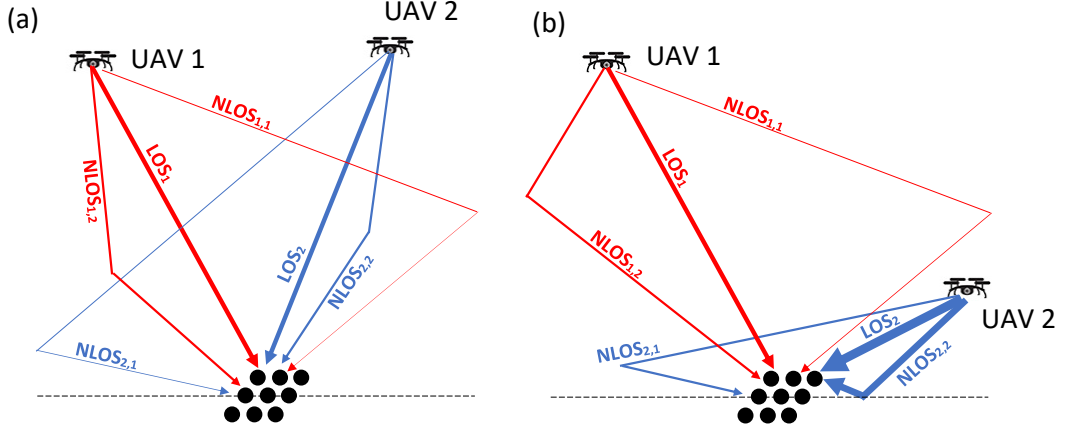


Figure 2.4 – LOS plus NLOS signals from two UAVs received at an antenna array. (a) UAVs 1 and 2 localized at similar distances from the antenna array. (b) UAV 2 is closer to the antenna array than UAV 1.

Then, the estimated symbols matrix $\hat{\mathbf{S}}_u \in \mathbb{C}^{D \times N}$ is obtained through the following relationship

$$\begin{aligned} \hat{\mathbf{S}}_u &= \hat{\mathbf{A}}_u^+ (\mathbf{U}_u \Sigma_u \mathbf{V}_u^H) \\ &= \begin{bmatrix} \hat{s}_{u1}(1) & \hat{s}_{u1}(2) & \cdots & \hat{s}_{u1}(N) \\ \hat{s}_{u2}(1) & \hat{s}_{u2}(2) & \cdots & \hat{s}_{u2}(N) \\ \vdots & \vdots & \ddots & \vdots \\ \hat{s}_{uD}(1) & \hat{s}_{uD}(2) & \cdots & \hat{s}_{uD}(N) \end{bmatrix}, \end{aligned} \quad (2.21)$$

where each row of $\hat{\mathbf{S}}_u$ is a vector $\hat{\mathbf{s}}_{ud} = [\hat{s}_{ud}(1), \hat{s}_{ud}(2), \dots, \hat{s}_{ud}(N)] \in \mathbb{C}^N$ corresponding to the estimated symbols received from the d -th direction of arrival at the u -th antenna array.

The next step is to compute the correlation coefficient ρ_{ij} between each pair of estimated symbol vectors $\hat{\mathbf{s}}_{ui}, \hat{\mathbf{s}}_{uj}$ for $i, j = 1, \dots, D$ and $i \neq j$. If such coefficient is lower than a given threshold, $\hat{\mathbf{s}}_{ui}$ and $\hat{\mathbf{s}}_{uj}$ are assumed to be emitted from different sources, otherwise we consider that both were transmitted from the same source. Typically, such threshold is considered with values around 0.6 and 0.7 [81]. If we consider the Pearson's correlation coefficient [82], then ρ_{ij} is given by

$$\rho_{ij} = \frac{\sum_{i=1}^D (\hat{\mathbf{s}}_{ui} - \bar{\hat{\mathbf{s}}}_{ui})(\hat{\mathbf{s}}_{uj} - \bar{\hat{\mathbf{s}}}_{uj})}{\sqrt{\sum_{i=1}^D (\hat{\mathbf{s}}_{ui} - \bar{\hat{\mathbf{s}}}_{ui})^2 \sum_{i=1}^D (\hat{\mathbf{s}}_{uj} - \bar{\hat{\mathbf{s}}}_{uj})^2}}, \quad (2.22)$$

where $\bar{\hat{\mathbf{s}}}_{ui}$ and $\bar{\hat{\mathbf{s}}}_{uj}$ for $i = 1, \dots, D$ and $j = 1, \dots, D$ are the average values of $\hat{\mathbf{s}}_{ui}$ and $\hat{\mathbf{s}}_{uj}$, respectively.

Since the u -th antenna array receives one direct signal plus G multipath components

from the q -th UAV, the estimated symbol vectors $\hat{\mathbf{s}}_{ug}^q$ for $g = 1, \dots, G + 1$ are stacked into a matrix $\hat{\mathbf{S}}_u^q \in \mathbb{C}^{(G+1) \times (N+1)}$ together with the index d_p^q which identifies the row of each vector in the original estimated symbols matrix $\hat{\mathbf{S}}_u \in \mathbb{C}^{D \times N}$ in (2.21), i.e.,

$$\hat{\mathbf{S}}_u^q = \begin{bmatrix} \hat{s}_{u1}^q(1) & \hat{s}_{u1}^q(2) & \cdots & \hat{s}_{u1}^q(N) & d_1^q \\ \hat{s}_{u2}^q(1) & \hat{s}_{u2}^q(2) & \cdots & \hat{s}_{u2}^q(N) & d_2^q \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \hat{s}_{u(G+1)}^q(1) & \hat{s}_{u(G+1)}^q(2) & \cdots & \hat{s}_{u(G+1)}^q(N) & d_{G+1}^q \end{bmatrix}. \quad (2.23)$$

For the sake of illustration of the multipath components grouping scheme, let us consider that a given antenna array receives six signals from two drones, labeled as UAV 1 and UAV 2, such that $N = 3$, $Q = 2$, $G = 2$ and, thus, $D = Q(G + 1) = 6$. In this example, the estimated symbols matrix $\hat{\mathbf{S}}_u \in \mathbb{C}^{6 \times 3}$ is given by

$$\hat{\mathbf{S}}_u = \begin{bmatrix} 0.4943 - 0.0059i, & 0.5050 - 0.0138i, & 0.4952 - 0.0185i \\ 0.6968 + 0.0067i, & 0.7087 - 0.0004i, & 0.7017 - 0.0093i \\ 1.3909 - 0.0685i, & 1.2143 + 0.0701i, & 1.3891 - 0.2079i \\ 1.2143 + 0.0276i, & 1.3104 + 0.0538i, & 1.3056 + 0.0143i \\ 0.6855 + 0.0147i, & 0.6989 - 0.0064i, & 0.6932 + 0.0018i \\ 1.2718 + 0.0650i, & 1.3023 - 0.0334i, & 1.2505 + 0.0862i \end{bmatrix}, \quad (2.24)$$

whose data are reorganized in Table 2.2, where d indicates the row index of each symbol vector $\hat{\mathbf{S}}_u(d, :)$ for $d = 1, \dots, 6$.

Table 2.2 – Data from the estimated symbols matrix $\hat{\mathbf{S}}_u \in \mathbb{C}^{6 \times 3}$.

d	$\hat{\mathbf{S}}_u(d, 1)$	$\hat{\mathbf{S}}_u(d, 2)$	$\hat{\mathbf{S}}_u(d, 3)$
1	0.4943 - 0.0059i	0.5050 - 0.0138i	0.4952 - 0.0185i
2	0.6968 + 0.0067i	0.7087 - 0.0004i	0.7017 - 0.0093i
3	1.3909 - 0.0685i	1.2143 + 0.0701i	1.3891 - 0.2079i
4	1.2143 + 0.0276i	1.3104 + 0.0538i	1.3056 + 0.0143i
5	0.6855 + 0.0147i	0.6989 - 0.0064i	0.6932 + 0.0018i
6	1.2718 + 0.0650i	1.3023 - 0.0334i	1.2505 + 0.0862i

The next step is to compute the correlation coefficient ρ_{ij} between each pair of estimated symbol vectors $\hat{\mathbf{S}}_u(i, :)$, $\hat{\mathbf{S}}_u(j, :)$ for $i, j = 1, \dots, 6$ and $i \neq j$, which is illustrated in Table 2.3. The fifth column indicates whether ρ_{ij} is greater than 0.7 and, if so, $\hat{\mathbf{S}}_u(i, :)$ and $\hat{\mathbf{S}}_u(j, :)$ are considered to be emitted from the same source, as reported in the last column. Therefore, from the results shown in Table 2.3, we conclude that the symbol vectors $\hat{\mathbf{S}}_u(d, :)$, with row indexes d given by (1,2,5) and (3,4,6), are emitted from the UAVs 1 and 2, respectively.

After grouping the multipath components according to their source, the matrices $\hat{\mathbf{S}}_u^1 \in$

Table 2.3 – Correlation coefficient between $\hat{S}_u(i, :)$ and $\hat{S}_u(j, :)$ for $i, j = 1, \dots, 6$ and $i \neq j$.

d	$\hat{S}_u(d, 1)$	$\hat{S}_u(d, 2)$	$\hat{S}_u(d, 3)$	$\rho > 0.7?$	Conclusion
1	0.4943 - 0.0059i	0.5050 - 0.0138i	0.4952 - 0.0185i	Y	1 and 2 from the same source
2	0.6968 + 0.0067i	0.7087 - 0.0004i	0.7017 - 0.0093i		
1	0.4943 - 0.0059i	0.5050 - 0.0138i	0.4952 - 0.0185i	N	1 and 3 from different sources
3	1.3909 - 0.0685i	1.2143 + 0.0701i	1.3891 - 0.2079i		
1	0.4943 - 0.0059i	0.5050 - 0.0138i	0.4952 - 0.0185i	N	1 and 4 from different sources
4	1.2143 + 0.0276i	1.3104 + 0.0538i	1.3056 + 0.0143i		
1	0.4943 - 0.0059i	0.5050 - 0.0138i	0.4952 - 0.0185i	Y	1 and 5 from the same source
5	0.6855 + 0.0147i	0.6989 - 0.0064i	0.6932 + 0.0018i		
1	0.4943 - 0.0059i	0.5050 - 0.0138i	0.4952 - 0.0185i	N	1 and 6 from different sources
6	1.2718 + 0.0650i	1.3023 - 0.0334i	1.2505 + 0.0862i		
d	$\hat{S}_u(d, 1)$	$\hat{S}_u(d, 2)$	$\hat{S}_u(d, 3)$	$\rho > 0.7?$	Conclusion
2	0.6968 + 0.0067i	0.7087 - 0.0004i	0.7017 - 0.0093i	N	2 and 3 from different sources
3	1.3909 - 0.0685i	1.2143 + 0.0701i	1.3891 - 0.2079i		
2	0.6968 + 0.0067i	0.7087 - 0.0004i	0.7017 - 0.0093i	N	2 and 4 from different sources
4	1.2143 + 0.0276i	1.3104 + 0.0538i	1.3056 + 0.0143i		
2	0.6968 + 0.0067i	0.7087 - 0.0004i	0.7017 - 0.0093i	Y	2 and 5 from the same source
5	0.6855 + 0.0147i	0.6989 - 0.0064i	0.6932 + 0.0018i		
2	0.6968 + 0.0067i	0.7087 - 0.0004i	0.7017 - 0.0093i	N	2 and 6 from different sources
6	1.2718 + 0.0650i	1.3023 - 0.0334i	1.2505 + 0.0862i		
d	$\hat{S}_u(d, 1)$	$\hat{S}_u(d, 2)$	$\hat{S}_u(d, 3)$	$\rho > 0.7?$	Conclusion
3	1.3909 - 0.0685i	1.2143 + 0.0701i	1.3891 - 0.2079i	Y	3 and 4 from the same source
4	1.2143 + 0.0276i	1.3104 + 0.0538i	1.3056 + 0.0143i		
3	1.3909 - 0.0685i	1.2143 + 0.0701i	1.3891 - 0.2079i	N	3 and 5 from different sources
5	0.6855 + 0.0147i	0.6989 - 0.0064i	0.6932 + 0.0018i		
3	1.3909 - 0.0685i	1.2143 + 0.0701i	1.3891 - 0.2079i	Y	3 and 6 from the same source
6	1.2718 + 0.0650i	1.3023 - 0.0334i	1.2505 + 0.0862i		
d	$\hat{S}_u(d, 1)$	$\hat{S}_u(d, 2)$	$\hat{S}_u(d, 3)$	$\rho > 0.7?$	Conclusion
4	1.2143 + 0.0276i	1.3104 + 0.0538i	1.3056 + 0.0143i	N	4 and 5 from different sources
5	0.6855 + 0.0147i	0.6989 - 0.0064i	0.6932 + 0.0018i		
4	1.2143 + 0.0276i	1.3104 + 0.0538i	1.3056 + 0.0143i	Y	4 and 6 from the same source
6	1.2718 + 0.0650i	1.3023 - 0.0334i	1.2505 + 0.0862i		
d	$\hat{S}_u(d, 1)$	$\hat{S}_u(d, 2)$	$\hat{S}_u(d, 3)$	$\rho > 0.7?$	Conclusion
5	0.6855 + 0.0147i	0.6989 - 0.0064i	0.6932 + 0.0018i	N	5 and 6 from different sources
6	1.2718 + 0.0650i	1.3023 - 0.0334i	1.2505 + 0.0862i		

$\mathbb{C}^{3 \times 4}$ and $\hat{\mathbf{S}}_u^2 \in \mathbb{C}^{3 \times 4}$ are obtained as follows

$$\hat{\mathbf{S}}_u^1 = \begin{bmatrix} 0.4943 - 0.0059i & 0.5050 - 0.0138i & 0.4952 - 0.0185i & \mathbf{1} \\ 0.6968 + 0.0067i & 0.7087 - 0.0004i & 0.7017 - 0.0093i & \mathbf{2} \\ 0.6855 + 0.0147i & 0.6989 - 0.0064i & 0.6932 + 0.0018i & \mathbf{5} \end{bmatrix}, \quad (2.25)$$

$$\hat{\mathbf{S}}_u^2 = \begin{bmatrix} 1.3909 - 0.0685i & 1.2143 + 0.0701i & 1.3891 - 0.2079i & \mathbf{3} \\ 1.2143 + 0.0276i & 1.3104 + 0.0538i & 1.3056 + 0.0143i & \mathbf{4} \\ 1.2718 + 0.0650i & 1.3023 - 0.0334i & 1.2505 + 0.0862i & \mathbf{6} \end{bmatrix}, \quad (2.26)$$

where the indexes d_g^q for $g = 1, \dots, 3$, shown in the last column of $\hat{\mathbf{S}}_u^1$ and $\hat{\mathbf{S}}_u^2$ and highlighted in red and blue colors, respectively, identify the rows of $\hat{\mathbf{S}}_u$ where each vector $\hat{\mathbf{S}}_u^q(g, :)$ can be found. Such fact is reinforced in the matrix $\hat{\mathbf{S}}_u$, reproduced below, whose rows are highlighted with the same colors of the indexes d_g^q in $\hat{\mathbf{S}}_u^1$ and $\hat{\mathbf{S}}_u^2$.

$$\hat{\mathbf{S}}_u = \begin{bmatrix} 0.4943 - 0.0059i, & 0.5050 - 0.0138i, & 0.4952 - 0.0185i \\ 0.6968 + 0.0067i, & 0.7087 - 0.0004i, & 0.7017 - 0.0093i \\ 1.3909 - 0.0685i, & 1.2143 + 0.0701i, & 1.3891 - 0.2079i \\ 1.2143 + 0.0276i, & 1.3104 + 0.0538i, & 1.3056 + 0.0143i \\ 0.6855 + 0.0147i, & 0.6989 - 0.0064i, & 0.6932 + 0.0018i \\ 1.2718 + 0.0650i, & 1.3023 - 0.0334i, & 1.2505 + 0.0862i \end{bmatrix}. \quad (2.27)$$

2.3.4 Line-of-Sight Estimation

Block 1.4 of Fig. 2.3 represents the LOS estimation. In this step, we assume that the LOS signal from the q -th UAV presents the highest power among all $\hat{\mathbf{s}}_{up}^q \in \hat{\mathbf{S}}_u^q$ for $g = 1, \dots, G+1$. As stated in the previous subsection, each vector $\hat{\mathbf{s}}_{up}^q$ is stored together with the index d_g^q which identifies its corresponding row in the original estimated symbols matrix $\hat{\mathbf{S}}_u$ in (2.21). Therefore, the index d^q which corresponds to the LOS of the q -th UAV in $\hat{\mathbf{S}}_u$ is given by

$$d^q = \underset{d_g^q}{\operatorname{argmax}} \sum_{n=1}^N |\hat{\mathbf{s}}_{ug}^q[n]|^2, \quad (2.28)$$

where $\operatorname{argmax}\{\cdot\}$ returns the argument that gives the maximum value from a target function.

Next, given d^q , the estimated spatial frequencies $\hat{\mu}_{ud^q}^{(r)}$ for $r = 1, \dots, R$ corresponding to the DoA of the UAV can be directly extracted from the respective eigenvalues of $\Psi^{(r)}$ given by (2.18).

The last step is to identify the Q rows of $\hat{\mathbf{S}}_u$ that contain the signals transmitted by the UAVs. After extracted from $\hat{\mathbf{S}}_u$, such vectors can be stacked along the 1st dimension, generating the estimated UAV symbols matrix $\hat{\mathbf{S}}_u^{\text{uav}} \in \mathbb{C}^{Q \times N}$ as follows

$$\hat{\mathbf{S}}_u^{\text{uav}} = \begin{bmatrix} \hat{s}_{u1}^{\text{uav}}(1) & \hat{s}_{u1}^{\text{uav}}(2) & \cdots & \hat{s}_{u1}^{\text{uav}}(N) \\ \hat{s}_{u2}^{\text{uav}}(1) & \hat{s}_{u2}^{\text{uav}}(2) & \cdots & \hat{s}_{u2}^{\text{uav}}(N) \\ \vdots & \vdots & \ddots & \vdots \\ \hat{s}_{uQ}^{\text{uav}}(1) & \hat{s}_{uQ}^{\text{uav}}(2) & \cdots & \hat{s}_{uQ}^{\text{uav}}(N) \end{bmatrix}. \quad (2.29)$$

Furthermore, the matrix $\hat{\mathbf{S}}_u^{\text{uav}}$, which contains the RF signatures of the Q drones, is sent to Block 1.6 where a trained machine learning classifier $\text{Cl}_u(\cdot)$, associated to the u -th URA, identifies the UAV type.

Continuing with the same numerical example shown in Subsection 2.3.3, let us reproduce the matrices $\hat{\mathbf{S}}_u^1$ and $\hat{\mathbf{S}}_u^2$ to improve readability,

$$\hat{\mathbf{S}}_u^1 = \begin{bmatrix} 0.4943 - 0.0059i, & 0.5050 - 0.0138i, & 0.4952 - 0.0185i, & \mathbf{1} \\ 0.6968 + 0.0067i, & 0.7087 - 0.0004i, & 0.7017 - 0.0093i, & \mathbf{2} \\ 0.6855 + 0.0147i, & 0.6989 - 0.0064i, & 0.6932 + 0.0018i, & \mathbf{5} \end{bmatrix}, \quad (2.30)$$

$$\hat{\mathbf{S}}_u^2 = \begin{bmatrix} 1.3909 - 0.0685i, & 1.2143 + 0.0701i, & 1.3891 - 0.2079i, & \mathbf{3} \\ 1.2143 + 0.0276i, & 1.3104 + 0.0538i, & 1.3056 + 0.0143i, & \mathbf{4} \\ 1.2718 + 0.0650i, & 1.3023 - 0.0334i, & 1.2505 + 0.0862i, & \mathbf{6} \end{bmatrix}. \quad (2.31)$$

Next, data from the estimated symbols matrices $\hat{\mathbf{S}}_u^1 \in \mathbb{C}^{3 \times 4}$ and $\hat{\mathbf{S}}_u^2 \in \mathbb{C}^{3 \times 4}$ in (2.30) and (2.31) are reorganized in Table 2.4 in order to determine the LOS index of the UAVs 1 and 2, given by d^q for $q = 1, 2$.

Table 2.4 – Data from the estimated symbols matrices $\hat{\mathbf{S}}_u^q \in \mathbb{C}^{3 \times 4}$ for $q = 1, 2$.

q	d	$\hat{\mathbf{S}}_u(d, 1)$	$\hat{\mathbf{S}}_u(d, 2)$	$\hat{\mathbf{S}}_u(d, 3)$	Power	d^q
1	1	0.4943 - 0.0059i	0.5050 - 0.0138i	0.4952 - 0.0185i	1.49	
	2	0.6968 + 0.0067i	0.7087 - 0.0004i	0.7017 - 0.0093i	2.10	2
	5	0.6855 + 0.0147i	0.6989 - 0.0064i	0.6932 + 0.0018i	2.08	
2	3	1.3909 - 0.0685i	1.2143 + 0.0701i	1.3891 - 0.2079i	4.01	
	4	1.2143 + 0.0276i	1.3104 + 0.0538i	1.3056 + 0.0143i	3.83	3
	6	1.2718 + 0.0650i	1.3023 - 0.0334i	1.2505 + 0.0862i	3.82	

Finally, the estimated UAV symbols matrix $\hat{\mathbf{S}}_u^{\text{uav}} \in \mathbb{C}^{2 \times 3}$ is obtained by stacking, along

the 1st dimension, the symbol vectors identified as LOS in Table 2.4 as follows

$$\hat{\mathbf{S}}_u^{\text{uav}} = \begin{bmatrix} 0.6968 + 0.0067i, & 0.7087 - 0.0004i, & 0.7017 - 0.0093i \\ 1.3909 - 0.0685i, & 1.2143 + 0.0701i, & 1.3891 - 0.2079i \end{bmatrix}, \quad (2.32)$$

where the estimated symbol vectors from the UAVs 1 and 2 are highlighted in red and blue colors, respectively.

2.3.5 Triangulation

Finally, the estimated position coordinates of the UAV are provided by triangulation [64], represented by Block 1.5 of Fig. 2.3. If we intend to localize Q UAVs localized within the system coverage region, the triangulation process must be repeated for the positioning of each UAV individually.

Without loss of generality, let us consider the antenna array based UAV localization system composed by U two-dimensional URAs depicted in Figure 2.1a. Since the origin of the Cartesian coordinate system $(0, 0, 0)$ is known, two antenna arrays are required to estimate the UAV position coordinates via triangulation. Fig. 2.5a to 2.5d illustrate the process of localization of the q -th UAV considering four different relative positions between the u -th and v -th URAs, as well as the detected drone. The LOS from the UAVs to the URAs are represented by lines in red color, whereas the distances between the URAs and the projection of the UAV coordinates onto the xy -plane are represented by dotted lines in black color.

The coordinates of the central element of the u -th and v -th URAs are (x_u, y_u, z_u) and (x_v, y_v, z_v) , respectively, whereas the lines-of-sight from the q -th UAV at both URAs present elevation and azimuth angles given by the pairs $(\hat{\theta}_{u,\text{los}}^q, \hat{\phi}_{u,\text{los}}^q)$ and $(\hat{\theta}_{v,\text{los}}^q, \hat{\phi}_{v,\text{los}}^q)$. Such angles can be directly extracted from the LOS spatial frequencies $\hat{\mu}_{ud^q}^{(r)}$ and $\hat{\mu}_{vd^q}^{(r)}$ for $r = 1, 2$ estimated in the previous subsection,

$$\hat{\mu}_{ud^q}^{(1)} = \frac{2\pi}{\lambda}(l_x \sin \hat{\theta}_{u,\text{los}}^q \cos \hat{\phi}_{u,\text{los}}^q), \quad \hat{\mu}_{ud^q}^{(2)} = \frac{2\pi}{\lambda}(l_y \sin \hat{\theta}_{u,\text{los}}^q \sin \hat{\phi}_{u,\text{los}}^q), \quad (2.33)$$

$$\hat{\mu}_{vd^q}^{(1)} = \frac{2\pi}{\lambda}(l_x \sin \hat{\theta}_{v,\text{los}}^q \cos \hat{\phi}_{v,\text{los}}^q), \quad \hat{\mu}_{vd^q}^{(2)} = \frac{2\pi}{\lambda}(l_y \sin \hat{\theta}_{v,\text{los}}^q \sin \hat{\phi}_{v,\text{los}}^q), \quad (2.34)$$

where l_x and l_y are the spacing between the URA elements in x and y axes, respectively, and λ is the center wavelength. In this chapter, the estimated elevation and azimuth angles $\hat{\theta}_{u,\text{los}}^q$ and $\hat{\phi}_{u,\text{los}}^q$ for $u = 1, \dots, U$, respectively, are considered as random variables with standard

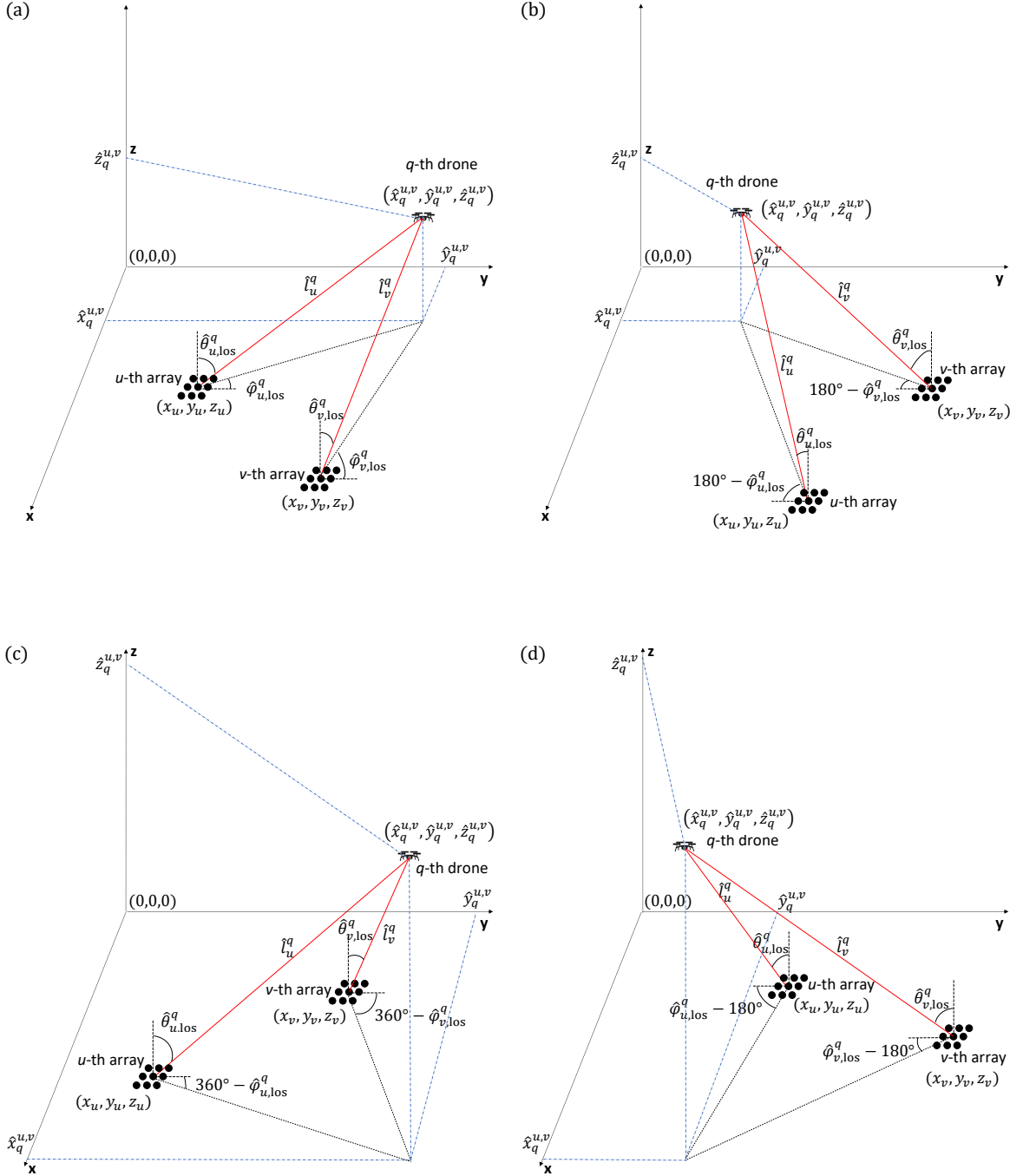


Figure 2.5 – Localization of the q -th UAV via triangulation techniques, considering four different relative positions between the u -th and v -th URAs, as well as the detected drone.

uniform distribution given by

$$\begin{aligned}\hat{\theta}_{u,\text{los}}^q &\sim \mathcal{U}(i, j), \quad i = 5^\circ, \quad j = 85^\circ, \\ \hat{\phi}_{u,\text{los}}^q &\sim \mathcal{U}(i, j), \quad i = 5^\circ, \quad j = 85^\circ,\end{aligned}\tag{2.35}$$

where $\mathcal{U}(i, j)$ stands for a uniformly distributed random variable within the range $[i, j]$.

Furthermore, the coordinates of the q -th UAV estimated by the u -th and v -th URAs are given by $(\hat{x}_q^{u,v}, \hat{y}_q^{u,v}, \hat{z}_q^{u,v})$, which correspond to the point of intersection between the DoAs $(\hat{\theta}_{u,\text{los}}^q, \hat{\phi}_{u,\text{los}}^q)$ and $(\hat{\theta}_{v,\text{los}}^q, \hat{\phi}_{v,\text{los}}^q)$. Such point can be estimated by using trigonometric relations referring to the coordinates of either the u -th or the v -th URA. Therefore, if the coordinates of the u -th URA are used as a reference, the UAV positions depicted in Fig. 2.5a to 2.5d can be computed, respectively, by either of the following systems of equations

$$\begin{cases} \hat{x}_q^{u,v} = x_u - \hat{l}_u^q \sin \hat{\theta}_{u,\text{los}}^q \sin \hat{\phi}_{u,\text{los}}^q, \\ \hat{y}_q^{u,v} = y_u + \hat{l}_u^q \sin \hat{\theta}_{u,\text{los}}^q \cos \hat{\phi}_{u,\text{los}}^q, \\ \hat{z}_q^{u,v} = \hat{l}_u^q \cos \hat{\theta}_{u,\text{los}}^q, \end{cases} \quad (2.36)$$

$$\begin{cases} \hat{x}_q^{u,v} = x_u - \hat{l}_u^q \sin \hat{\theta}_{u,\text{los}}^q \sin(180^\circ - \hat{\phi}_{u,\text{los}}^q) = x_u - \hat{l}_u^q \sin \hat{\theta}_{u,\text{los}}^q \sin \hat{\phi}_{u,\text{los}}^q, \\ \hat{y}_q^{u,v} = y_u - \hat{l}_u^q \sin \hat{\theta}_{u,\text{los}}^q \cos(180^\circ - \hat{\phi}_{u,\text{los}}^q) = y_u + \hat{l}_u^q \sin \hat{\theta}_{u,\text{los}}^q \cos \hat{\phi}_{u,\text{los}}^q, \\ \hat{z}_q^{u,v} = \hat{l}_u^q \cos \hat{\theta}_{u,\text{los}}^q, \end{cases} \quad (2.37)$$

$$\begin{cases} \hat{x}_q^{u,v} = x_u + \hat{l}_u^q \sin \hat{\theta}_{u,\text{los}}^q \sin(360^\circ - \hat{\phi}_{u,\text{los}}^q) = x_u - \hat{l}_u^q \sin \hat{\theta}_{u,\text{los}}^q \sin \hat{\phi}_{u,\text{los}}^q, \\ \hat{y}_q^{u,v} = y_u + \hat{l}_u^q \sin \hat{\theta}_{u,\text{los}}^q \cos(360^\circ - \hat{\phi}_{u,\text{los}}^q) = y_u + \hat{l}_u^q \sin \hat{\theta}_{u,\text{los}}^q \cos \hat{\phi}_{u,\text{los}}^q, \\ \hat{z}_q^{u,v} = \hat{l}_u^q \cos \hat{\theta}_{u,\text{los}}^q, \end{cases} \quad (2.38)$$

$$\begin{cases} \hat{x}_q^{u,v} = x_u + \hat{l}_u^q \sin \hat{\theta}_{u,\text{los}}^q \sin(\hat{\phi}_{u,\text{los}}^q - 180^\circ) = x_u - \hat{l}_u^q \sin \hat{\theta}_{u,\text{los}}^q \sin \hat{\phi}_{u,\text{los}}^q, \\ \hat{y}_q^{u,v} = y_u - \hat{l}_u^q \sin \hat{\theta}_{u,\text{los}}^q \cos(\hat{\phi}_{u,\text{los}}^q - 180^\circ) = y_u + \hat{l}_u^q \sin \hat{\theta}_{u,\text{los}}^q \cos \hat{\phi}_{u,\text{los}}^q, \\ \hat{z}_q^{u,v} = \hat{l}_u^q \cos \hat{\theta}_{u,\text{los}}^q, \end{cases} \quad (2.39)$$

where \hat{l}_v^q and \hat{l}_u^q are the distances between $(\hat{x}_q^{u,v}, \hat{y}_q^{u,v}, \hat{z}_q^{u,v})$ and the coordinates of each URA, (x_v, y_v, z_v) and (x_u, y_u, z_u) , respectively. Note that the azimuth angle $\hat{\phi}_{u,\text{los}}^q$ is written as a function of $\hat{\phi}_{u,\text{los}}^q$, as illustrated in Figure 2.2.

From (2.36) to (2.39), it can be seen that the coordinates of the q -th UAV obtained by trigonometric relations present identical equations, regardless of the relative position between the URAs and UAV shown in Fig. 2.5a to 2.5d. Since x_u, x_v, y_u and y_v are known, the distances \hat{l}_v^q and \hat{l}_u^q can be obtained from either of the above-mentioned systems of equations, for example, (2.36), such that

$$\hat{l}_v^q = \frac{\cos \hat{\theta}_{u,\text{los}}^q (x_v - x_u)}{a - b}, \quad (2.40)$$

$$\hat{l}_u^q = \frac{\cos \hat{\theta}_{v,\text{los}}^q (x_v - x_u)}{a - b}, \quad (2.41)$$

when the x -coordinates of both URAs are used as a reference, or

$$\hat{l}_v^q = \frac{\cos \hat{\theta}_{u,\text{los}}^q (y_v - y_u)}{c - d}, \quad (2.42)$$

$$\hat{l}_u^q = \frac{\cos \hat{\theta}_{v,\text{los}}^q (y_v - y_u)}{c - d}, \quad (2.43)$$

when, alternatively, the y -coordinates are used as a reference. In the above systems of equations, $a = \sin \hat{\theta}_{v,\text{los}}^q \sin \hat{\varphi}_{v,\text{los}}^q \cos \hat{\theta}_{u,\text{los}}^q$, $b = \sin \hat{\theta}_{u,\text{los}}^q \sin \hat{\varphi}_{u,\text{los}}^q \cos \hat{\theta}_{v,\text{los}}^q$, $c = \sin \hat{\theta}_{u,\text{los}}^q \cos \hat{\varphi}_{u,\text{los}}^q \cos \hat{\theta}_{v,\text{los}}^q$ and $d = \sin \hat{\theta}_{v,\text{los}}^q \cos \hat{\varphi}_{v,\text{los}}^q \cos \hat{\theta}_{u,\text{los}}^q$.

In this sense, the UAV coordinates $(x_q^{u,v}, y_q^{u,v}, z_q^{u,v})$ estimated by the (u,v) -th pair of URAs can be computed, for example, by substituting (2.41) into (2.36), i.e.,

$$\begin{cases} \hat{x}_q^{u,v} = \frac{ax_u - bx_v}{a - b}, \\ \hat{y}_q^{u,v} = \frac{(a - b)y_u + c(x_v - x_u)}{a - b}, \\ \hat{z}_q^{u,v} = \frac{\cos \hat{\theta}_{u,\text{los}}^q \cos \hat{\theta}_{v,\text{los}}^q (x_v - x_u)}{a - b}, \end{cases} \quad (2.44)$$

or, alternatively, by substituting (2.43) into (2.36), obtaining

$$\begin{cases} \hat{x}_q^{u,v} = \frac{(c - d)x_u - b(y_v - y_u)}{c - d}, \\ \hat{y}_q^{u,v} = \frac{cy_v - dy_u}{c - d}, \\ \hat{z}_q^{u,v} = \frac{\cos \hat{\theta}_{u,\text{los}}^q \cos \hat{\theta}_{v,\text{los}}^q (y_v - y_u)}{c - d}. \end{cases} \quad (2.45)$$

This fact reflects the displacement invariance inherent to the proposed antenna array based UAV localization system, since (2.44) and (2.45) estimate the UAV coordinates by applying triangulation techniques on the LOS signals received by any pair of antenna arrays, regardless of the relative position between such arrays and the detected drone. Appendix B presents all of the possible different relative positions between a pair of URAs (u, v) and the q -th UAV, showing that the estimated coordinates of the UAV have identical mathematical formulas.

Finally, in order to obtain more accurate results, each pair of URAs $(1,2),(2,3),\dots,(U-1,U),(U,1)$, placed along a common edge of the system border, can be used to estimate

the drone coordinates. Therefore, the final estimated position $(\hat{x}_q, \hat{y}_q, \hat{z}_q)$ of the q -th UAV corresponds to the arithmetic mean of the coordinates computed in (2.44) or (2.45) related to those pairs of URAs, i.e.,

$$\begin{cases} \hat{x}_q = \frac{1}{U} \sum_{u=1}^U \hat{x}_q^{u,v}, \\ \hat{y}_q = \frac{1}{U} \sum_{u=1}^U \hat{y}_q^{u,v}, \\ \hat{z}_q = \frac{1}{U} \sum_{u=1}^U \hat{z}_q^{u,v}, \end{cases} \quad (2.46)$$

$$u = 1, \dots, U,$$

$$v = u \bmod U + 1,$$

where mod stands for the modulo operation, such that $u \bmod U$ returns the remainder of the division of u by U .

2.3.6 UAV Identification

Following, Block 1.6 of Fig. 2.3 represents the process of UAV identification. In this block, the symbols matrices $\hat{\mathbf{S}}_u^{\text{uav}} \in \mathbb{C}^{Q \times N}$ for $u = 1, \dots, U$ estimated in Block 1.4 are forwarded to a trained machine learning classification algorithm $\text{Cl}_u(\cdot)$.

In this chapter, the DroneRF dataset [73] is used for validation of the proposed framework. More details regarding such dataset are shown in Appendix C. Table 2.5 details the number of raw samples and segments for each drone model.

Table 2.5 – Number of raw samples and segments for each drone model present in the DroneRF dataset.

Drone Model	Nr Segments	Nr Samples
Bebop	84	1.680×10^6
AR	81	1.620×10^6
Phantom	21	420×10^6

If $\text{Cl}_u(\cdot)$ represents a trained base classifier associated with the u -th antenna array, the vector $\hat{\mathbf{c}}_u \in \mathbb{R}^Q$ containing the predicted drone model \hat{c}_{uq} of the q -th UAV for $q = 1, \dots, Q$ is given by

$$\hat{\mathbf{c}}_u = \text{Cl}_u(\hat{\mathbf{S}}_u^{\text{uav}}) = [\hat{c}_{u1}, \dots, \hat{c}_{uQ}]^T. \quad (2.47)$$

2.3.7 Majority Voting

Finally, the majority voting of the drone models \hat{c}_{uq} for $q = 1, \dots, Q$ predicted by all of the U base classifiers is performed in Block 1.7. The final estimated model vector $\mathbf{c}^* \in \mathbb{R}^Q$ contains the drone models c_q^* for $q = 1, \dots, Q$ estimated for the q -th UAV, which are computed through the simple majority voting of $\hat{c}_{1q}, \dots, \hat{c}_{Uq}$, i.e.,

$$\mathbf{c}^* = \begin{bmatrix} c_1^* \\ \vdots \\ c_Q^* \end{bmatrix}, \quad (2.48)$$

$$\text{with } c_q^* = \text{mode}\{\hat{c}_{1q}, \dots, \hat{c}_{Uq}\},$$

where $\text{mode}\{\cdot\}$ is the mode function, which returns the most frequent class predicted by the U machine learning classification algorithms.

The proposed tensor based framework for UAV localization and identification in multipath environments is summarized in Algorithm 1.

Algorithm 1: The proposed algorithm for UAV localization and identification in multipath environments.

```

1 Begin
   Input:
   - Received signal tensor:  $\mathbf{X}_u \in \mathbb{R}^{M_1 \times \dots \times M_R \times N}$  for  $u = 1, \dots, U$ 
   - Number of received signals:  $D$ 
   - Number of UAVs:  $Q$ 
   - Maximum number of subarrays in each spatial dimension:  $L_r$ 
   Output:
   - Estimated position coordinates  $(\hat{x}_q, \hat{y}_q, \hat{z}_q)$  of the  $q$ -th UAV for  $q = 1, \dots, Q$ 
   - Estimated class label  $c_q^*$  of the  $q$ -th UAV for  $q = 1, \dots, Q$ 
   Algorithm Steps:
2 for  $u = 1$  to  $U$  do
3   for  $r = 1$  to  $R$  do
4     for  $l_r = 1$  to  $L_r$  do
5       Compute the smoothed matrix  $\mathbf{X}_{SS,ur}^{(l_r)} \in \mathbb{R}^{M_r^{(\text{sub})} \times \prod_{j \neq r} M_j L_r N}$  of the unfolding matrix
6          $[\mathbf{X}_u]_{(r)} \in \mathbb{R}^{M_r \times \prod_{j \neq r} M_j N}$  along the  $r$ -th mode for the  $l_r$ -th subarray size as in (2.13)
7       Compute the SVD low-rank approximation  $\tilde{\mathbf{X}}_{SS,ur}^{(l_r)} \in \mathbb{R}^{M_r^{(\text{sub})} \times \prod_{j \neq r} M_j L_r N}$  of
8          $\mathbf{X}_{SS,ur}^{(l_r)} \in \mathbb{R}^{M_r^{(\text{sub})} \times \prod_{j \neq r} M_j L_r N}$  along the  $r$ -th mode for the  $l_r$ -th subarray size as in (2.15)
9       Reconstruct the matrix  $[\tilde{\mathbf{X}}_u]_{(r)} \in \mathbb{R}^{M_r \times \prod_{j \neq r} M_j N}$  as in (2.16) and (2.17)
10      Fold the matrix  $[\tilde{\mathbf{X}}_u]_{(r)} \in \mathbb{R}^{M_r \times \prod_{j \neq r} M_j N}$  into the denoised tensor  $\tilde{\mathbf{X}}_u \in \mathbb{R}^{M_1 \times \dots \times M_R \times N}$ 
11    for  $d = 1$  to  $D$  do
12      Compute the estimated matrices  $\Psi_u^{(r)}$  as in (2.18) and (2.19)
13      Rebuild the estimated array steering matrix  $\hat{\mathbf{A}}_u = \hat{\mathbf{A}}_u^{(1)} \diamond \hat{\mathbf{A}}_u^{(2)} \diamond \dots \diamond \hat{\mathbf{A}}_u^{(R)} \in \mathbb{C}^{M \times D}$ 
14      Compute the SVD low-rank approximation of  $\mathbf{X}_u \in \mathbb{C}^{M \times N}$  truncated to  $D$  singular values as in (2.20)
15      Compute the estimated symbol matrix  $\hat{\mathbf{S}}_u \in \mathbb{C}^{D \times N}$  as in (2.21)
16     $P \leftarrow 1$ 

```

```

15
16   for  $i = 1$  to  $D$  do
17       for  $j = 1$  to  $D$  do
18           Compute the correlation coefficient  $\rho_{ij}$  between  $\hat{\mathbf{s}}_{ui}$  and  $\hat{\mathbf{s}}_{uj}$  as in (2.22)
19           if  $\rho_{ij} > \text{threshold}$  and  $i \neq j$  then
20                $\hat{\mathbf{s}}_{ui}$  and  $\hat{\mathbf{s}}_{uj}$  are emitted from the same source
21                $G \leftarrow G + 1$ 
22           else
23                $\hat{\mathbf{s}}_{ui}$  and  $\hat{\mathbf{s}}_{uj}$  are emitted from different sources
24           end
25       end
26   end
27   for  $g = 1$  to  $G$  do
28       for  $q = 1$  to  $Q$  do
29           Stack the estimated symbol vector  $\hat{\mathbf{s}}_{uq}^g$  into the matrix  $\hat{\mathbf{S}}_u^g \in \mathbb{C}^{(G+1) \times (N+1)}$  along with  $d_q^g$  as in (2.23)
30       end
31   end
32   for  $q = 1$  to  $Q$  do
33       Compute the index  $d^q$  in  $\hat{\mathbf{S}}_u$  as in (2.28)
34       for  $r = 1$  to  $R$  do
35           Compute the estimated spatial frequencies  $\hat{\mu}_{udq}^{(r)}$  from the eigenvalues of  $\Psi^{(r)}$ 
36       end
37   end
38   Compute the estimated UAV symbols matrix  $\hat{\mathbf{S}}_u^{\text{uav}} \in \mathbb{C}^{Q \times N}$  as in (2.29)
39   for  $u = 1$  to  $U$  do
40       Compute the azimuth and elevation angles  $(\hat{\theta}_u^q, \hat{\phi}_u^q)$  and  $(\hat{\theta}_v^q, \hat{\phi}_v^q)$  as in (2.33) to (2.35)
41       Compute the estimated coordinates  $(\hat{x}_q^{u,v}, \hat{y}_q^{u,v}, \hat{z}_q^{u,v})$  as in (2.36) to (2.45)
42   end
43   for  $u = 1$  to  $U$  do
44       for  $q = 1$  to  $Q$  do
45           Compute the final estimated coordinates  $(\hat{x}_q, \hat{y}_q, \hat{z}_q)$  as in (2.46)
46           Estimate the identification  $\hat{c}_{uq}$  of the  $q$ -th UAV as in (2.47)
47       end
48   end
49   Compute the estimated class label  $c_q^*$  of the  $q$ -th UAV as in (2.48)
50 end

```

2.4 COMPUTATIONAL COMPLEXITY

In this section, the computational complexity of the proposed framework is presented and discussed. The computational costs related to folding and unfolding of matrices and tensors are not considered, since such functions are about data representations. Moreover, the asymptotic time cost as a function of the largest contributions of the most important variables is considered, namely, $M = \prod_{r=1}^R M_r$, L_r , N , R , G and D . As it can be seen in Section 2.5, the proposed framework is compared with state-of-the-art UAV localization schemes, namely, matrix based ESPRIT with SS and tensor based ESPRIT with SS. Therefore, in this section, the computational complexities of such competing techniques are also introduced.

First, the computational complexity of the traditional MuDe technique is presented. Such scheme is described in steps 2 to 8 of Alg. 1. The time cost of the SVD low-rank approximation of a matrix with dimensions $(M_r - l_r + 1) \times ((\prod_{j \neq r} M_j) l_r N)$ truncated to rank D is given by $\mathcal{O}[(\prod_{j \neq r} M_j)(M_r - l_r + 1) D l_r]$ [83]. Since MuDe computes a total of L_r truncated

SVDs for each dimension $r = 1, \dots, R$, its computational complexity can be expressed as

$$\mathcal{O}[\text{MuDe}] = \mathcal{O} \left[\sum_{r=1}^R \sum_{l_r=1}^{L_r} \left(\prod_{j \neq r} M_j \right) (M_r - l_r + 1) D l_r N \right]. \quad (2.49)$$

Moreover, the direction of arrival estimation is performed by the R -D Standard Tensor ESPRIT algorithm, as shown in steps 9 to 10 of Alg. 1. According to [62], its computational complexity can be expressed as

$$\mathcal{O}[\text{T-ESPRIT}] = \mathcal{O} [k_t M N D (R + 1) + M N D R + M D^2 R], \quad (2.50)$$

where k_t is a constant that depends on the design of the algorithm.

Furthermore, the proposed framework computes the SVD low-rank approximation of \mathbf{X}_u truncated to D in step 12 of Alg. 1, with complexity given by

$$\mathcal{O}[\text{SVD}] = \mathcal{O} [k_t M N D]. \quad (2.51)$$

Besides, steps 16 to 26 of Alg. 1 compute the correlation coefficient between each pair of vectors $\hat{\mathbf{s}}_{ui}, \hat{\mathbf{s}}_{uj}$ for $i = 1, \dots, D$ and $j = 1, \dots, D$, with complexity

$$\mathcal{O}[\text{Corr}] = \mathcal{O} [D^2 N]. \quad (2.52)$$

Next, the proposed algorithm performs the line-of-sight estimation, where the index d^q is computed in step 33 of Alg. 1. Its computational complexity can be expressed as

$$\mathcal{O}[\text{LOS}] = \mathcal{O} [(G + 1) N \log^2(N)], \quad (2.53)$$

where $G + 1$ is the number of signal components received from the same UAV (one LOS plus G NLOS signals).

In addition, step 35 of Alg. 1 computes the Eigenvalue Decomposition (EVD) of $\Psi^{(r)}$ in order to obtain the estimated spatial frequencies $\hat{\mu}_{ud^q}^{(r)}$. Such process presents complexity

$$\mathcal{O}[\text{EVD}] = \mathcal{O} [R D^3]. \quad (2.54)$$

Finally, the overall computational complexity of the proposed tensor based framework for UAV localization in multipath environments corresponds to the sum of the aforementioned

complexities, i.e.,

$$\begin{aligned} \mathcal{O}[\text{Framework}] = & \mathcal{O}[\text{MuDe}] + \mathcal{O}[\text{T-ESPRIT}] + \mathcal{O}[\text{SVD}] + \\ & + \mathcal{O}[\text{Corr}] + \mathcal{O}[\text{LOS}] + \mathcal{O}[\text{EVD}]. \end{aligned} \quad (2.55)$$

The time costs of the matrix based ESPRIT and spatial smoothing techniques are given by $\mathcal{O}[\text{M-ESPRIT}] = \mathcal{O}[k_t DNM]$ and $\mathcal{O}[\text{SS}] = \mathcal{O}[(2F + 1)N + (F + 1)^3]$, respectively, where F corresponds to the degree of freedom of the antenna array [84]. Furthermore, the computational complexity of the tensor based ESPRIT is shown in (2.50). Table 2.6 summarizes the computational complexities of the proposed framework and its competing schemes. The second column of Table 2.6 illustrates the total computational cost, whereas the last column shows the final complexities, corresponding to the asymptotic dominant terms. From the results shown in Table 2.6, it can be seen that the proposed framework presents the highest computational complexity, which is composed by two dominant terms: $\mathcal{O}[k_t RDNM]$, corresponding to the standard tensor ESPRIT algorithm, plus $\mathcal{O}[LRDNM]$, related to the MuDe technique. Therefore, the mean processing times are considerably impacted by the MuDe algorithm, as shown in the numerical simulations of Section 2.5. Note that, among the competing schemes, the lowest computational complexity is achieved by the M-ESPRIT + SS technique, which is a matrix based approach.

Table 2.6 – Computational complexity of the following models: (1) proposed framework, (2) tensor based ESPRIT with SS, and (3) matrix based ESPRIT with SS.

Model	Total Complexity	Dominant Complexity
Proposed framework	$\mathcal{O} \left[\sum_{r=1}^R \sum_{l_r=1}^{L_r} (\prod_{j \neq r} M_j) (M_r - l_r + 1) D l_r N \right] +$ $+ \mathcal{O} [k_t MND(R + 1) + MNDR + MD^2 R] +$ $+ \mathcal{O} [k_t MND] + \mathcal{O} [D^2 N] + \mathcal{O} [PN \log^2(N)] + \mathcal{O} [RD^3]$	$\mathcal{O} [k_t RDNM] + \mathcal{O} [LRDNM]$
T-ESPRIT + SS [6]	$\mathcal{O} [k_t MND(R + 1) + MNDR + MD^2 R] +$ $+ \mathcal{O} [(2F + 1)N + (F + 1)^3]$	$\mathcal{O} [k_t RDNM]$
M-ESPRIT + SS [62]	$\mathcal{O} [k_t DNM] + \mathcal{O} [(2F + 1)N + (F + 1)^3]$	$\mathcal{O} [k_t DNM]$

2.5 SIMULATION RESULTS

This section is divided into two subsections. First, Subsection 2.5.1 presents the simulation results regarding UAV localization. Next, numerical simulations considering the UAV identification are shown in Subsection 2.5.2. Finally, Subsection 2.5.3 discusses the obtained results.

2.5.1 Localization Module

In this subsection, the improvements introduced by both the tensor approach and the MuDe scheme for localizing UAVs are illustrated through computer simulations. The Root Mean Square Error (RMSE) of the estimated spatial frequency and the estimated UAV position coordinates are evaluated as a function of the following parameters: Signal-to-Interference Ratio (SIR), number of signal samples (N), number of antennas per dimension (M_r) and number of UAVs within the system (Q). Moreover, three techniques are compared: (i) the classical standard matrix ESPRIT with SS, which was adopted in [6]; (ii) the R -D standard tensor ESPRIT with SS, developed by Haardt et al. in [62] and applied in [4]; and (iii) the proposed framework, which adopts the R -D standard tensor ESPRIT with MuDe. A third-order received signal tensor with two spatial dimensions and one temporal dimension is considered in simulations. In addition, the experiments were executed on a desktop computer with processor Intel Core i7-2600 3.40 GHz and 16 GB of Random Access Memory (RAM). Further, MATLAB R2018a software was used in all simulations. The final results of each metric correspond to the average value of 50 Monte Carlo simulation runs. The RMSE of a given metric γ can be expressed as

$$\text{RMSE}_{(\tau)} = \sqrt{\frac{1}{R \cdot Q} \left(\sum_{r=1}^R \sum_{q=1}^Q (\gamma_{q,r}^{(\tau)} - \hat{\gamma}_{q,r}^{(\tau)})^2 \right)}, \quad (2.56)$$

where $\gamma_{q,r}^{(\tau)}$ and $\hat{\gamma}_{q,r}^{(\tau)}$ denote, respectively, the real value and the estimated value of γ for the q -th UAV in the r -th array dimension at the τ -th Monte Carlo run.

The matrix $\mathbf{I}_u \in \mathbb{C}^{M \times N}$ is the interference matrix, which corresponds to the sum of the power of the other interfering signals. Thus, the signal-to-noise and signal-to-interference ratios are defined as

$$\text{SNR} = \frac{\sigma_{\mathbf{X}_u}^2}{\sigma_{\mathbf{N}_u}^2}, \quad (2.57)$$

$$\text{SIR} = \frac{\sigma_{\mathbf{X}_u}^2}{\sigma_{\mathbf{I}_u}^2}, \quad (2.58)$$

where $\sigma_{\mathbf{X}_u}^2$, $\sigma_{\mathbf{N}_u}^2$ and $\sigma_{\mathbf{I}_u}^2$ are the variances of \mathbf{X}_u , \mathbf{N}_u and \mathbf{I}_u , respectively. Usually, both SNR and SIR are expressed in decibels as follows

$$\text{SNR}_{\text{dB}} = 10 \log_{10} \left(\frac{\sigma_{\mathbf{X}_u}^2}{\sigma_{\mathbf{N}_u}^2} \right), \quad (2.59)$$

$$\text{SIR}_{\text{dB}} = 10 \log_{10} \left(\frac{\sigma_{\mathbf{X}_u}^2}{\sigma_{\mathbf{I}_u}^2} \right). \quad (2.60)$$

First, the performance of the proposed framework is assessed for different number of URAs. Four different configurations of the antenna array based UAV localization system, depicted in Fig. 2.6, are considered in experiments, in which the total number of URAs ranges from three (triangle-shaped) to six (hexagon-shaped). All of the URAs are symmetrically distributed along the xy -plane of the system coverage region, such that the distances between two URAs positioned along a common edge of the system border are approximately identical. For the triangle, square, pentagon and hexagon-shaped configurations, such distances are given by $\Delta_t = \Delta_s = 400$ m, $\Delta_p = 280$ m and $\Delta_h = 240$ m, respectively. Additionally, in all simulation scenarios, the UAV moves within the following distance ranges: $120 < x < 280$ meters, $140 < y < 260$ meters, and $30 < z < 130$ meters. The elevation and azimuth angles, $\hat{\theta}_{u,\text{los}}^q$ and $\hat{\phi}_{u,\text{los}}^q$ for $u = 1, \dots, U$ and $q = 1, \dots, Q$, are uniformly distributed random variables within the range $[5^\circ, 85^\circ]$. Further, the number of dimensions is $R = 2$, $\text{SIR} = \text{SNR} = 0$ dB, $N = 20$, $Q = 2$ UAVs and $\lambda = 0.125$ m. The number of antennas M_r is fixed in 50 and the subarray length $L_r = 15$ is constant for each spatial mode ($r = 1, 2$). The spacing between URA antennas are $l_x = l_y = \lambda/2$, whereas each URA receives two multipath signals from each UAV. Table 2.7 shows the RMSE of the spatial frequency, considering different system configurations, of the proposed framework and its competing schemes. In addition, in order to illustrate the benefits of adopting the MuDe algorithm on the overall UAV localization performance, simulation results from a tensor based ESPRIT model without denoising techniques are also included. From the results shown in Table 2.7, it can be observed that the proposed framework outperforms its competitor methods in all of the system configurations, as highlighted in bold. Note that the M-ESPRIT + SS scheme presents the worst performance, especially for the triangle, pentagon and hexagon-shaped scenarios. Moreover, by comparing our proposed scheme with the T-ESPRIT with no denoising scheme, we observe a considerable performance gain due to the inclusion of the MuDe algorithm in Block 1.1 of Fig. 2.3. Finally, note that the tensor based schemes present similar results for different UAV localization system configurations.

Table 2.7 – RMSE of the estimated spatial frequency, considering different configurations of the antenna array based UAV localization system.

RMSE of the estimated spatial frequency				
Model	System Configuration			
	Triangle	Square	Pentagon	Hexagon
Proposed Framework	0.0010	0.0014	0.0011	0.0012
T-ESPRIT + SS	0.0011	0.0015	0.0012	0.0015
T-ESPRIT (no denoising)	0.0015	0.0018	0.0016	0.0019
M-ESPRIT + SS	0.0452	0.0019	0.0718	0.0811

From this point on, the square-shaped UAV localization system is adopted in simulations,

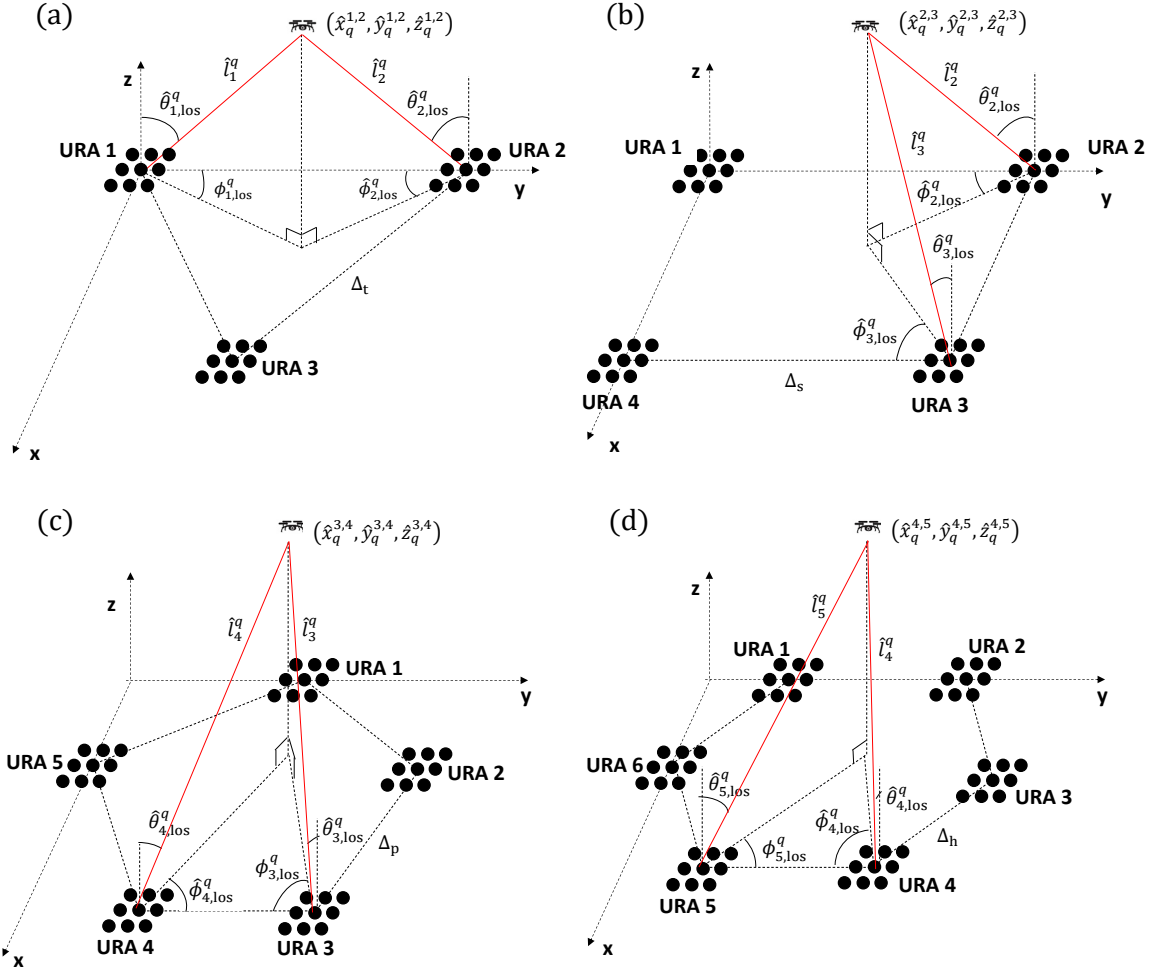


Figure 2.6 – Four different configurations of the proposed antenna array based UAV localization system used in simulations. (a) triangle. (b) square. (c) pentagon. (d) hexagon.

with edge length $\Delta_s = 400$ meters and $U = 4$ URAs, such that a single URA is deployed at each vertice. The system is the same depicted in Fig. 2.6b, in which URAs 1 to 4 are positioned at the $(0, \Delta_s, 0)$, $(\Delta_s, \Delta_s, 0)$, $(0, 0, 0)$ and $(\Delta_s, 0, 0)$ coordinates, respectively. Furthermore, as stated in Subsection 2.3.5, the UAV position is estimated by each pair of URAs placed along a common edge of the system border. The number of spatial dimensions is $R = 2$, the SIR ranges from 2 to 4 dB, N varies between 10 and 50 samples and $\lambda = 0.125$ m. The number of antennas M_r ranges from 20 to 50 and the subarray length $L_r = 15$ is constant for each spatial mode ($r = 1, 2$). The spacing between URA antennas are $l_x = l_y = \lambda/2$. Each URA receives $D = 3Q$ wavefronts (Q LOS plus $2Q$ multipath signals) originated from Q UAVs placed at different position coordinates, such that Q varies between 2 and 5.

Fig. 2.7a and 2.7b illustrate, respectively, the RMSE of the estimated spatial frequencies and estimated UAV position coordinates as a function of the SIR. In this first scenario, we assume $N = 20$, $Q = 2$, $\text{SNR} = 0$ dB and $M_1 = M_2 = 50$. It can be observed that the

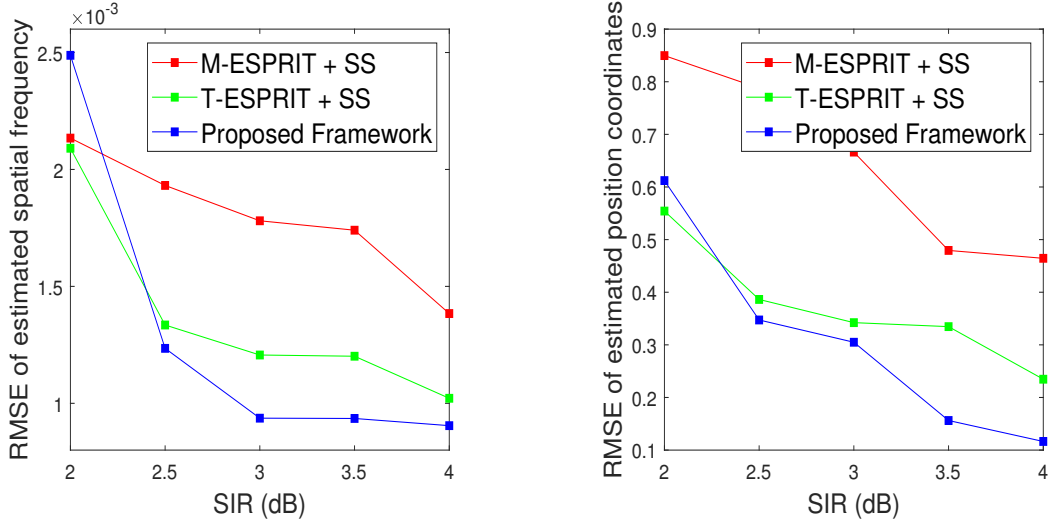


Figure 2.7 – (a) RMSE of the estimated spatial frequency versus SIR (dB). (b) RMSE of the estimated position coordinates versus SIR (dB).

proposed tensor-based framework with MuDe outperforms the tensor approach with SS when $\text{SNR} > 2$ dB. As expected, the matrix-based approach with spatial smoothing presents lower performance in most of the SIR range. Moreover, it is clear that all techniques deliver better performance as the SIR is higher.

Next, Fig. 2.8a and 2.8b show the RMSE of the estimated spatial frequencies and estimated position coordinates versus the number of antennas per dimension, respectively. Now we consider $\text{SIR} = 20$ dB, $\text{SNR} = 30$ dB, $Q = 2$, $D = 6$ and $N = 10$. From the results, it is evident that our proposed framework presents better performance compared to both matrix-based SS and tensor-based SS techniques. Furthermore, note that all of the competing schemes deliver better results when the number of antennas is larger, which improves the SNR in such case.

Following, the RMSE of the estimated spatial frequencies and estimated position coordinates as a function of the number of samples is shown in Fig. 2.9a and 2.9b. In this scenario, $\text{SIR} = 20$ dB, $\text{SNR} = 30$ dB, $M_r = 50$, $Q = 2$ and $D = 6$. Once again the proposed tensor-based framework with MuDe outperforms both matrix ESPRIT SS and tensor ESPRIT SS approaches. In addition, from the results shown in Fig. 2.9a and 2.9b, we observe that all of the competitor techniques show better performance for larger number of samples.

Additionally, Fig. 2.10a and 2.10b illustrate, respectively, the RMSE of the estimated spatial frequencies and estimated position coordinates versus the number of UAVs within the system coverage region. Now we consider $\text{SNR} = 30$ dB, $\text{SIR} = 20$ dB, $M_r = 50$, $N = 20$ and $1 \leq Q \leq 4$. As expected, our proposed framework outperforms the other approaches. Moreover, note that all of the compared techniques show worse performance when the number of UAVs is higher. In such case, since the coverage area is fixed, there is a higher number of signal emitters per area unit, implying in higher interference. In summary, from the results

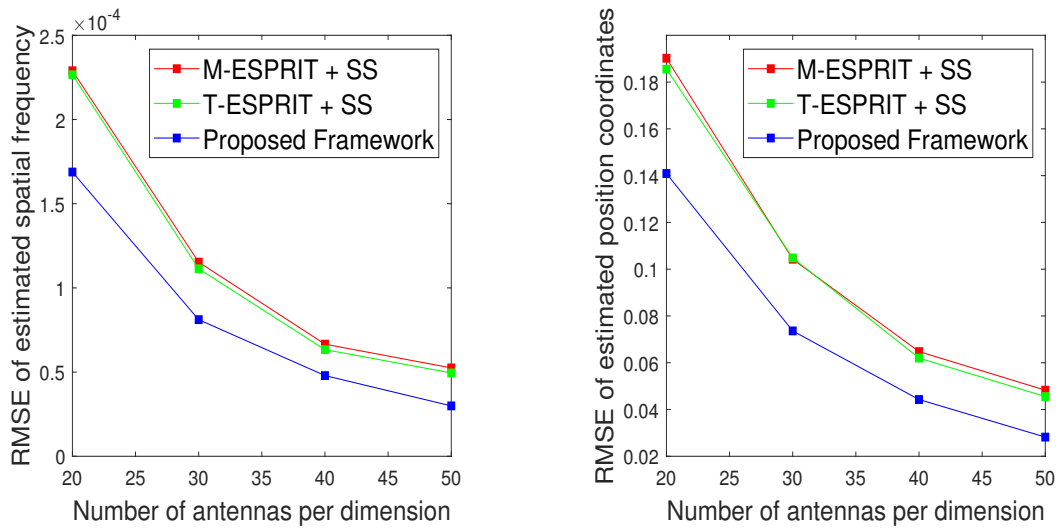


Figure 2.8 – (a) RMSE of the estimated spatial frequency versus number of antennas per dimension. (b) RMSE of the estimated position coordinates versus number of antennas per dimension.

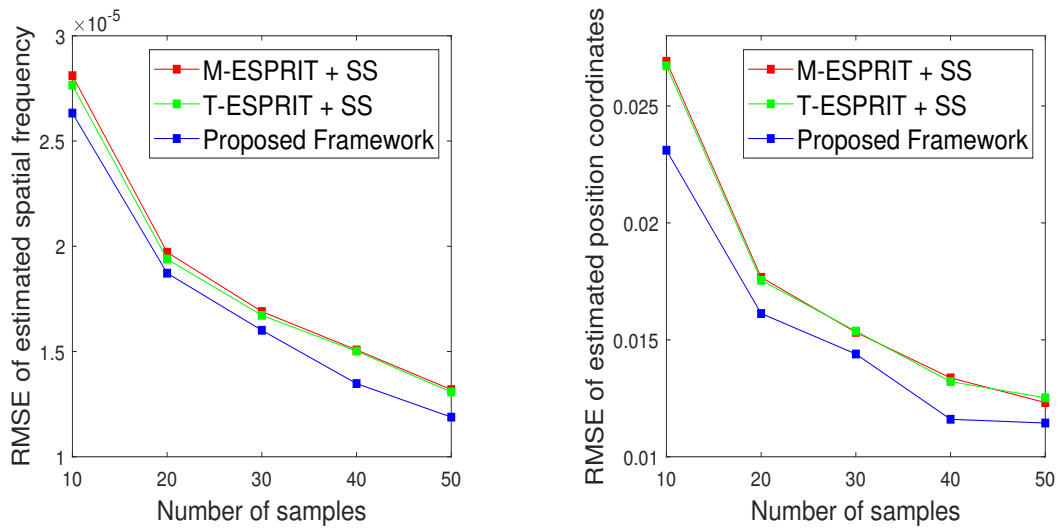


Figure 2.9 – (a) RMSE of the estimated spatial frequency versus number of samples. (b) RMSE of the estimated position coordinates versus number of samples.

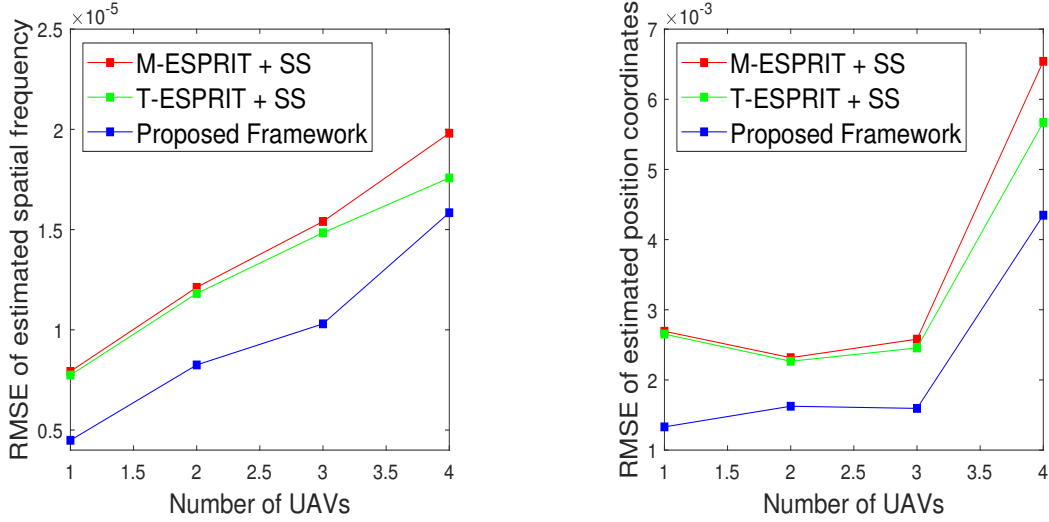


Figure 2.10 – RMSE of the estimated spatial frequency versus number of UAVs. (b) RMSE of the estimated position coordinates versus number of UAVs.

shown in Fig. 2.7 to 2.10, it is clear that our proposed framework exploits the inherent tensor structure present in the received signal and, consequently, outperforms all of the competing schemes in simulations.

Finally, the mean processing times (in seconds) of the proposed framework as well as the competing M-ESPRIT + SS and T-ESPRIT + SS schemes are evaluated. Fig. 2.11 illustrates the processing times corresponding to the simulations shown in Fig. 2.9 and 2.10. From the results shown in Fig. 2.11, it can be seen that the proposed framework is more computationally expensive than the matrix-based approach, showing an increase in the mean processing time. This fact is more evident especially for larger number of UAVs, where a higher computational processing is required. In addition, note that our proposed framework slightly outperforms the tensor based ESPRIT with SS, despite the inclusion of MuDe, which presents a higher computational complexity. In such case, MuDe yields a more efficient noise attenuation, achieving a faster algorithm convergence and, consequently, compensating its higher complexity. Moreover, as expected, all of the compared techniques present worse performance for larger number of samples and UAVs, where a higher computational processing is required. Thus, according to the results obtained in Fig. 2.11, this is the trade-off in order to achieve a more accurate UAV localization compared to the M-ESPRIT + SS.

2.5.2 Identification Module

In this subsection, the performance of the UAV identification module is assessed through numerical simulations. Accuracy (Acc), Detection Rate (DR) and False Alarm Rate (FAR), defined in Eq. (D.1), (D.3) and (D.4) of Appendix D, are adopted as performance evaluation metrics.

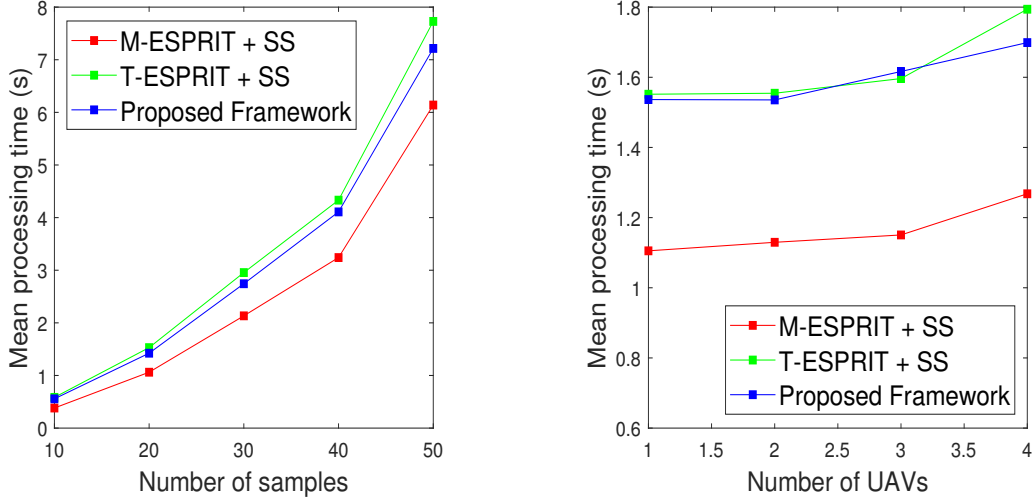


Figure 2.11 – a) Mean processing time (s) versus number of samples. (b) Mean processing time (s) versus number of UAVs.

In order to validate our scheme, we generated a dataset by using the RF signatures collected from the DroneRF database [85], detailed in Table 2.5. Despite such database contains samples from different drone functioning modes, as mentioned in Section C.1 of Appendix C, only the identification of intruding UAVs operating in the “on and connected” mode is assessed in this chapter. First, ten segments were extracted from each UAV model of the DroneRF dataset, with number of time samples N fixed in 50. In order to simulate the signatures corresponding to the Bebop, AR and Phantom drone models, the time samples extracted from the DroneRF dataset are used as the transmitted symbols $[s_{uq}(1), \dots, s_{uq}(N)]$ for $q = 1, \dots, Q$. Three drones ($Q = 3$) are randomly positioned within the system coverage region, with two multipath components per drone and SNR = 30 dB. The number of antennas M_r is fixed in 50, with $L_1 = L_2 = 5$, whereas the spacing between URA antennas is given by $l_x = l_y = \lambda/2$.

As pointed out by [73], ML classification algorithms can benefit from the latent information present in the RF signals and, consequently, show an improved performance for detecting and identifying UAVs. To accomplish this, each vector $\hat{\mathbf{s}}_{uq}^{\text{uav}} = [\hat{s}_{uq}^{\text{uav}}(1), \dots, \hat{s}_{uq}^{\text{uav}}(N)]$ for $q = 1, \dots, Q$ in $\hat{\mathbf{S}}_u^{\text{uav}}$ is converted to the frequency domain by applying the DFT as follows

$$w_{uq}(h) = \left\| \sum_{n=1}^N \hat{s}_{uq}^{\text{uav}}(n) \exp\left(\frac{-j2\pi h(n-1)}{N}\right) \right\|, \quad (2.61)$$

where $w_{uq}(h)$ is the estimated symbols vector of the q -th UAV at the u -th URA in the frequency domain, whereas h and n are the frequency and time domain indices, respectively.

After the simulation of 100 different experiments, 300 segments in the time domain, estimated by each one of the four URAs, were converted into the frequency domain. The

conversion was performed by applying the DFT in (2.61) such that each instance is composed by 2048 frequency bins, which correspond to the dataset features. The class labels “Bebop”, “AR” and “Phantom” were coded respectively as “0”, “1” and “2”. Moreover, at the u -th URA for $u = 1, \dots, U$, the dataset was split into the training and testing sets, with proportion 50:50, which were sent to the base classifier $Cl_u(\cdot)$. Additionally, in order to increase the training set size and reduce its class imbalance problem, minority samples were artificially generated by applying the Synthetic Minority Oversampling Technique (SMOTE) on each training set separately. Furthermore, seven different types of machine learning classification algorithms are evaluated in this chapter, namely, Decision Trees (DT), Extra Trees (ET), k-Nearest Neighbors (kNN), Linear Discriminant Analysis (LDA), Logistic Regression (LR), Naïve Bayes (NB) and Random Forest (RFo). The predictions resulting from each classifier are combined via simple majority voting in order to provide the final classification. The main simulation parameters adopted for each ML classifier in Keras library are summarized in Appendix E.

Table 2.8 presents the values of accuracy, detection rate and false alarm rate of the proposed framework and its competitor schemes for identifying three different drone models (Bebop, AR and Phantom) when considering the above-mentioned ML classification algorithms. The best metrics for each ML classifier and for each SIR are highlighted in bold. Here we consider macro-averaged metrics, i.e., each metric value corresponds to the average between the values computed for all classes. The SNR is fixed in 30 dB, whereas SIR ranges from 0 dB to 20 dB. From Table 2.8, we observe that all schemes present very similar performance, regardless of the SIR. Especially for SIR = 0 dB, a very poor performance is shown by all approaches due to the high interference level present in the environment, with detection rate around 33%. On the other hand, for SIR = 20 dB, all classifiers but LR and NB presented detection rate higher than 90%. Despite the UAV identification performance of the proposed framework is not the best for all SIR range, it still presents outstanding results compared to the other schemes, especially when SIR = 20 dB for ET, LDA, LR and NB classifiers.

Table 2.9 presents the values of Acc, DR and FAR of the proposed framework and its competing techniques for identifying Bebop, AR and Phantom UAVs. Since Extra Trees and Linear Discriminant Analysis showed the best performance in UAV identification, here only such algorithms are evaluated. Differently from Table 2.8, which presented the macro-averaged metrics, Table 2.9 illustrates the metric values obtained for each class label. Moreover, the best metrics for each ML classifier and for each UAV are highlighted in bold. The SNR and SIR are fixed in 30 dB and 20 dB, respectively. From Table 2.9, we observe that the AR drone model is perfectly identified when Extra Trees are applied for classification in all URAs, with Acc, DR and FAR of 100%, 100% and 0%, respectively, for all compared techniques. Moreover, the proposed framework outperforms the competitor schemes for iden-

Table 2.8 – Macro-averaged values of accuracy, detection rate and false alarm rate of the proposed framework and its competitor schemes for identifying three different types of drones (Bebop, AR and Phantom), considering SIR between 0 dB and 20 dB.

Accuracy								
SIR	Model	DT	ET	kNN	LDA	LR	NB	RFo
0 dB	Proposed Framework	0.5503	0.5507	0.5491	0.5573	0.5557	0.5557	0.5536
	T-ESPRIT + SS [62]	0.5588	0.5559	0.5520	0.5578	0.5556	0.5556	0.5527
	M-ESPRIT + SS [6]	0.5562	0.5600	0.5484	0.5552	0.5556	0.5556	0.5552
10 dB	Proposed Framework	0.8981	0.9797	0.9462	0.9444	0.7223	0.7223	0.8941
	T-ESPRIT + SS [62]	0.8973	0.9796	0.9471	0.9433	0.7222	0.7222	0.8934
	M-ESPRIT + SS [6]	0.8980	0.9809	0.9484	0.9433	0.7222	0.7222	0.8940
20 dB	Proposed Framework	0.9471	0.9913	0.9940	0.9940	0.7811	0.7779	0.9461
	T-ESPRIT + SS [62]	0.9473	0.9897	0.9949	0.9927	0.7797	0.7778	0.9462
	M-ESPRIT + SS [6]	0.9468	0.9866	0.9946	0.9936	0.7800	0.7778	0.9463

Detection Rate								
SIR	Model	DT	ET	kNN	LDA	LR	NB	RFo
0 dB	Proposed Framework	0.3255	0.3260	0.3237	0.3360	0.3334	0.3336	0.3303
	T-ESPRIT + SS [62]	0.3382	0.3338	0.3280	0.3367	0.3333	0.3333	0.3290
	M-ESPRIT + SS [6]	0.3343	0.3400	0.3227	0.3328	0.3333	0.3335	0.3328
10 dB	Proposed Framework	0.8472	0.9695	0.9193	0.9167	0.5834	0.5834	0.8412
	T-ESPRIT + SS [62]	0.8460	0.9693	0.9207	0.9150	0.5833	0.5833	0.8402
	M-ESPRIT + SS [6]	0.8470	0.9713	0.9227	0.9150	0.5833	0.5833	0.8410
20 dB	Proposed Framework	0.9207	0.9870	0.9910	0.9910	0.6717	0.6668	0.9192
	T-ESPRIT + SS [62]	0.9210	0.9845	0.9923	0.9890	0.6695	0.6667	0.9193
	M-ESPRIT + SS [6]	0.9202	0.9798	0.9918	0.9903	0.6700	0.6667	0.9195

False Alarm Rate								
SIR	Model	DT	ET	kNN	LDA	LR	NB	RFo
0 dB	Proposed Framework	0.3373	0.3370	0.3382	0.3320	0.3333	0.3333	0.3348
	T-ESPRIT + SS [62]	0.3309	0.3331	0.3360	0.3317	0.3330	0.3330	0.3355
	M-ESPRIT + SS [6]	0.3328	0.3300	0.3387	0.3336	0.3333	0.3333	0.3336
10 dB	Proposed Framework	0.0764	0.0153	0.0403	0.0417	0.2080	0.2080	0.0790
	T-ESPRIT + SS [62]	0.0770	0.0153	0.0397	0.0425	0.2083	0.2083	0.0799
	M-ESPRIT + SS [6]	0.0765	0.0143	0.0387	0.0425	0.2083	0.2083	0.0795
20 dB	Proposed Framework	0.0397	0.0065	0.0045	0.0045	0.1642	0.1660	0.0404
	T-ESPRIT + SS [62]	0.0395	0.0077	0.0038	0.0055	0.1652	0.1667	0.0404
	M-ESPRIT + SS [6]	0.0399	0.0101	0.0041	0.0048	0.1650	0.1667	0.0403

tifying the Phantom drone model with ET classifier, achieving Acc and DR of 98.70% and 96.45%, respectively. Further, the proposed approach also presents superior performance for Bebop identification in terms of accuracy and false alarm rate, obtaining 98.70% and 1.78%, respectively, when Extra Trees are used for classification. Regarding the LDA classification algorithm, we observe that all of the competing approaches show false alarm rate of 0% for

identifying the AR and Phantom UAV models. Furthermore, all techniques present the same values of Acc and DR for the Phantom identification, whereas a detection rate of 100% was achieved by all schemes for identifying the Bebop model. Additionally, note that the proposed framework outperforms its counterpart methods when identifying AR drone model in terms of Acc and DR, achieving 100% for both metrics with LDA classifier.

Table 2.9 – Values of accuracy, detection rate and false alarm rate of the proposed framework and its competitor schemes for identifying three different drone models (Bebop, AR and Phantom), considering ET and LDA classifiers.

Metric	Model	ET			LDA		
		Bebop	AR	Phantom	Bebop	AR	Phantom
Acc	Proposed Framework	0.9870	1.0000	0.9870	0.9167	1.0000	0.9167
	T-ESPRIT + SS [62]	0.9845	1.0000	0.9845	0.9150	0.9983	0.9167
	M-ESPRIT + SS [6]	0.9798	1.0000	0.9798	0.9150	0.9983	0.9167

Metric	Model	ET			LDA		
		Bebop	AR	Phantom	Bebop	AR	Phantom
DR	Proposed Framework	0.9965	1.0000	0.9645	1.0000	1.0000	0.7500
	T-ESPRIT + SS [62]	1.0000	1.0000	0.9535	1.0000	0.9950	0.7500
	M-ESPRIT + SS [6]	1.0000	1.0000	0.9395	1.0000	0.9950	0.7500

Metric	Model	ET			LDA		
		Bebop	AR	Phantom	Bebop	AR	Phantom
FAR	Proposed Framework	0.0178	0.0000	0.0178	0.1250	0.0000	0.0000
	T-ESPRIT + SS [62]	0.0232	0.0000	0.0000	0.1275	0.0000	0.0000
	M-ESPRIT + SS [6]	0.0303	0.0000	0.0000	0.1275	0.0000	0.0000

2.5.3 Discussion

In this subsection, the results shown in Subsections 2.5.1 and 2.5.2 are discussed. First, Fig. 2.7a and 2.7b illustrated the RMSE of the estimated spatial frequency and the estimated UAV coordinates as a function of the SIR. The signal-to-interference ratio reflects a relationship between the transmitted signal and the co-channel interference from other radio transmitters. As expected, it was observed that the RMSE is higher as the SIR is diminished due to the larger number of co-channel interfering transmissions, which impacts all compared approaches. In addition, the proposed framework presents a higher robustness against co-channel interference compared to M-ESPRIT + SS and T-ESPRIT + SS in most of the SIR range, especially due to the MuDe module, which compensates the higher environment interference level, outperforming the spatial smoothing denoising technique adopted by the competitor methods.

Next, Fig. 2.8a and 2.8b showed the RMSE of the estimated spatial frequency and the estimated position coordinates versus M_r for fixed values of SNR and SIR. In an antenna array,

the higher number of antennas per dimension can improve the signal-to-noise ratio by exploiting redundancy across the multiple transmit and receive channels, as well as by reusing spatial information in order to improve coverage. As observed in the above-mentioned figures, the accuracies of UAV position and spatial frequency are improved when the number of antenna elements is increased, regardless of the considered scheme. Moreover, note that the proposed framework considerably outperforms both competitor methods in all M_r range, once again due to the MuDe algorithm. As the number of antennas is larger, the maximum number of subarrays in each spatial dimension is increased and, consequently, the performance of MuDe is improved, which corroborates the outstanding results shown by our proposed scheme, despite its higher computational complexity.

Then, the RMSE of the estimated spatial frequency and the estimated position coordinates as a function of the number of samples were respectively shown in Fig. 2.9a and 2.9b. Usually, in antenna array based systems, a higher number of time samples leads to a more accurate performance, despite the trade-off between increased accuracy and processing time. Such fact is clearly observed in Fig. 2.9a and 2.9b, where all of the compared schemes show lower RMSE as the number of time samples is higher. Note that the proposed framework outperforms both M-ESPRIT + SS and T-ESPRIT + SS, since MuDe benefits from the higher values of N compared to spatial smoothing approach.

Following, the RMSE of the estimated spatial frequency and the estimated position coordinates versus the number of UAVs were presented in Fig. 2.10a and 2.10b, respectively. In communications systems, a higher number of co-channel transmitters reduces the SIR, increasing the interference level and packet losses. Consequently, all of the competitor schemes present worse performance as the number of drones per unit area is higher, since high interference levels are achieved in such situation, impacting the accuracy of the antenna array based localization system. Furthermore, note that the M-ESPRIT + SS and T-ESPRIT + SS schemes are considerably outperformed by our proposed framework in all of the Q range, presenting a lower robustness against co-channel interference in crowded environments.

Finally, Subsection 2.5.2 showed several experiments regarding the UAV identification. The estimated drone signature is sent to a trained machine learning classification algorithm at each URA, which classifies the UAV into one of three different models: Bebop, AR and Phantom. In line with the findings in Fig. 2.7a and 2.7b, all of the competitor schemes present better UAV identification performance as the SIR is higher. In such case, the drone signatures estimated in each URA are more accurate, since the number of interfering sources is lower, leading to improved results obtained by ML classification algorithms. Despite slightly outperformed by the competing schemes when $SIR = 0$ dB for DT, ET, kNN, LDA and RFo classifiers, our proposed approach showed considerable results for identifying UAVs in low interference level conditions, especially when ET, LDA, LR and NB algorithms are applied for classification, as shown in Table 2.8. Furthermore, it was observed that some

drone models are more efficiently identified compared to other ones. For instance, considering a SIR of 20 dB, the AR drone was perfectly identified by all compared techniques when Extra Trees were used for classification, achieving 100% of accuracy. Similarly, when LDA was applied as ML classifier, all of the competitor schemes achieved a detection rate of 100% for identifying the Bebop model. Moreover, still regarding the LDA algorithm, the compared approaches identified the AR and Phantom models with a false alarm rate of 0%.

2.6 SUMMARY

In this chapter, the framework proposed in [6] is extended in three aspects: by adopting a tensor representation to better explore the intrinsic multidimensional patterns present in the data; by including a multiple denoising approach to increase the SNR of the received signal; and by including a ML classification algorithm in each URA in order to identify the intruding drone model. As shown in simulations, the proposed framework outperforms the matrix-based and tensor-based ESPRIT solutions in which the spatial smoothing is adopted as denoising technique. The evaluated metrics regarding the UAV localization are the RMSE of the estimated spatial frequency and estimated UAV position as a function of several parameters. Moreover, our proposed approach presents lower processing time compared to the tensor based ESPRIT with SS scheme. However, despite the higher processing time compared to the matrix-based ESPRIT scheme, such difference is rather small, since our proposed framework provides the best performance among all of the assessed methods.

With respect to the UAV identification module, the performance of all of the competing schemes was assessed for different ML classification algorithms, namely, DT, ET, kNN, LDA, LR, NB and RFo. The DroneRF dataset was used to simulate the signatures of three well-known drone models, namely, Bebop, AR and Phantom. Further, each URA presents a ML classification algorithm, and the predictions resulting from each classifier are combined via simple majority voting such that the final estimated drone model is provided. From the numerical simulations, it was observed that the proposed scheme outperforms the competitor methods in terms of accuracy, detection rate and false alarm rate for identifying the drone model in several scenarios, for different SIR values and ML classifiers. Additionally, the AR drone model was perfectly identified by all of the compared approaches when Extra Trees were used for classification, in which accuracy and detection rate of 100% were achieved.

3 TENSOR MULTIPLE DENOISING BASED FRAMEWORK FOR DISTRIBUTED DENIAL OF SERVICE ATTACK DETECTION

In this chapter, the following research question is addressed: *How to efficiently detect DDoS attacks in a CPS network by jointly applying multidimensional signal processing techniques and machine learning algorithms, assuming that a noisy dataset is used for training and testing?*

The remainder of this chapter is organized as follows:

- **Motivation:** in this section, an introduction to DDoS attack detection models based on signal processing and machine learning techniques is presented.
- **Data Model:** this section defines the data model used throughout this chapter.
- **Proposed Tensor Multiple Denoising Based Framework for DDoS Attack Detection:** this section introduces and discusses the proposed tensor multiple denoising based framework for DDoS attack detection.
- **Computational Complexity:** the computational complexity of the proposed scheme is presented and discussed.
- **Simulation Results:** the performance of the proposed framework is evaluated through numerical simulations.

The research contributions of Chapter 3 are summarized as follows:

1. An extension of the recent MuDe algorithm in order to attenuate the noise present in the instances of DDoS attack detection datasets. Given the outstanding performance of MuDe to denoise measurement data collected in sensor arrays, such scheme shows a good potential for DDoS attack dataset denoising.
2. The inclusion of a second denoising stage performed by a LRA technique such that a higher degree of noise reduction is achieved, with significant gain on the overall DDoS attack detection performance.

3. The performance of the proposed scheme is validated through numerical simulations by using samples extracted from the CIC-DDoS2019 and NSL-KDD benchmark datasets. According to the obtained results, the proposed framework achieves satisfactory performance, with considerable values of accuracy, detection rate and false alarm rate compared with state-of-the-art low-rank approximation techniques.

3.1 MOTIVATION

Distributed Denial of Service (DDoS) attacks are one of the most harmful threats to network security. Their main goal is to deny legitimate accesses to network services by exhausting bandwidth and resources through massive volume of traffic, usually launched by multiple compromised devices known as “bots” [86]. Such bots constitute a botnet, which is remotely controlled by an attacker in order to execute coordinated attacks against a specific target.

DDoS attacks have become more hazardous and sophisticated, especially over the last years. Due to their ease of organization and execution, such malicious activities have widely been used by hackers in order to create chaos and disruption, with little sign of slowing [87]. For instance, in October, 2020, the Google Cloud Team revealed that a 2.54 Tbps DDoS attack had been mitigated by the organization in September, 2017 [52]. Performed by Chinese Internet Service Providers, the attack targeted thousands of Google’s IP addresses and lasted more than six months. It was four times larger than the 623 Gbps DDoS attack launched by the Mirai botnet against the blog of cybersecurity KrebsOnSecurity one year earlier [46]. Another recent massive DDoS attack occurred in February, 2020, when the Amazon’s AWS Shield protection service mitigated an attack of 2.3 Tbps. Such attack was based on Connectionless Lightweight Directory Access Protocol (CLDAP) reflection and caused three days of elevated threat [51].

In this sense, it is fundamental that network administrators adopt accurate and efficient schemes in order to detect and prevent DDoS attacks in their organizations. For instance, tensor based signal processing techniques have attracted an increasing attention in the last years since they allow us to better exploit the inherent multidimensional structure of large datasets [75, 88]. Furthermore, supervised Machine Learning (ML) based methods can provide an efficient way to detect DDoS attacks [22, 23]. As ML algorithms can be trained on benchmark datasets provided by cybersecurity institutes [89, 90], such schemes can be used to identify, with high reliability, malicious patterns eventually present in the input network traffic in an automated fashion.

In order to obtain higher performance, ML based NIDSs must be trained with massive amount of data. Usually, large datasets present inherent multidimensional structure, which

can be better explored by applying tensor signal processing techniques. However, a potential drawback consists of the presence of noise in such large datasets. In this case, noise can refer to uncalibrated measures occurred during the process of dataset creation [91], or due to false data injection performed by attackers on publicly available datasets [92], leading to data corruption. For example, Gaussian noise injection attacks are easy to implement in practice and aim to fool machine learning classifiers during the training and testing phases. Such facts can degrade the performance of the ML classifier and, consequently, reduce the reliability of the DDoS attack detection model.

Schemes based on traditional signal processing techniques for DDoS attack detection have attracted a great attention in the last decades. Model Order Selection (MOS) techniques for blind automatic malicious activity detection in distributed honeypots were proposed by David et al. in [93, 94], where human intervention or information about attacks were not required. In line with the ideas of [94], Da Costa et al. proposed a blind automatic scheme to detect malicious traffic in network data collected at honeypot systems [95] as well as the R-D Akaike Information Criterion and the R-D Minimum Description Length to automatically identify malicious activities in honeypots [96]. Moreover, Tenório et al. proposed a solution where malicious traffic is blindly detected for any computer connected to the network [97]. In addition, it is worth to mention the recent work of Vieira et al., which proposed a framework in order to detect the number of port scanning and flood attacks by analyzing the largest eigenvalues in time frames after applying MOS and similarity analysis on the dataset [98]. More recently, Vilaça et al. presented a semi-supervised machine learning model, named RPCA-MD, in order to identify anomalies on network traffic such that potential attacks can be detected in an automated way by using the Robust Principal Component Analysis (RPCA) and Mahalanobis Distance [99]. However, since the approaches in [94]-[99] are not tensor based solutions and do not consider automatic learning, we fill those gaps by exploiting the inherent tensor structure present in large datasets as well as by applying classic machine learning classification algorithms such that the proposed technique learns to recognize patterns in multidimensional data.

Further, machine learning based schemes have also been successfully used for DDoS attack detection. Osanaiye et al. [100] presented an ensemble based multi-filter feature selection method for DDoS attack detection in cloud computing where the output of filter methods are combined to achieve an optimum selection. Furthermore, a model based on artificial neural networks and black hole optimization algorithm to detect DDoS attacks in cloud computing was presented in [101]. Moreover, in [22], the authors proposed a hybrid framework based on data stream approach for DDoS attack detection where the computational load is divided between the client and proxy side. Finally, Wang et al. [23] proposed a method for DDoS attack detection in which feature selection is combined with multilayer perceptron such that the optimal features are selected, and also designed a feedback mech-

anism to dynamically perceive detection errors. Thus, despite the machine learning based schemes proposed by [22], [23], [100] and [101] show high performance in terms of DDoS attack detection, they did not exploit multidimensional techniques. Therefore, such research gap is also filled by adopting and extending tensor based denoising approaches, particularly the recent MuDe scheme, such that the inherent tensor structure of large datasets can be exploited more efficiently. The main approaches and drawbacks of each related work, as well as the weaknesses addressed by this chapter, are summarized in Table 3.1.

Table 3.1 – Related works summary.

Ref.	Main Approach	Drawback
[22]	- Data stream based framework for DDoS attack detection with load balancing.	- Benefits of tensor representation are not exploited. - Outdated dataset (NSL-KDD) applied for model validation. - Lack of analysis of the training and testing times.
[23]	- Feature selection based MLP for DDoS attack detection with dynamic perceiving of detection errors.	- Benefits of tensor representation are not exploited. - Lack of analysis of the training and testing times.
[94]	- MOS schemes for blind automatic malicious activity detection in distributed honeypots.	- Benefits of tensor representation are not exploited. - Automatic learning is not considered. - Not capable of handling large volumes of data.
[95]	- Modified EFT to automatically identify malicious activities in honeypot networks.	- Benefits of tensor representation are not exploited. - Automatic learning is not considered. - Not capable of handling large volumes of data.
[96]	- R-D AIC and R-D MDL to automatically identify malicious activities in honeypot networks.	- Benefits of tensor representation are not exploited. - Automatic learning is not considered. - Not capable of handling large volumes of data.
[97]	- Blind detection of malicious traffic for any device connected to the network.	- Benefits of tensor representation are not exploited. - Automatic learning is not considered.
[98]	- Detect the number of port scanning and flood attacks by analyzing the largest eigenvalues in time frames.	- Benefits of tensor representation are not exploited. - Automatic learning is not considered.
[99]	- ML based model to identify anomalies on network traffic by using RPCA and Mahalanobis distance.	- Benefits of tensor representation are not exploited. - Automatic learning is not considered.
[100]	- Ensemble based multi-filter feature selection for DDoS attack detection in cloud computing.	- Benefits of tensor representation are not exploited. - Outdated dataset (NSL-KDD) applied for model validation. - Lack of analysis of the testing times.
[101]	- Artificial neural network based model for DDoS attack detection in cloud computing.	- Benefits of tensor representation are not exploited. - Outdated dataset (NSL-KDD) applied for model validation. - Lack of analysis of the training and testing times. - Lack of important performance evaluation metrics.
This chapter	- Tensor based framework for DDoS attack detection when NIDS is trained with poisoned datasets.	Addressed drawbacks: - Benefits of tensor representation are exploited. - Recent dataset (CIC-DDoS2019) applied for model validation. - Analysis of the training and testing times.

In this chapter, we propose a novel noise-robust framework for DDoS attack detection which exploits tensor based signal processing techniques as well as ML based algorithms. The proposed architecture is composed by four steps: data preprocessing, dataset splitting, dataset denoising and machine learning classification. Moreover, in the third step, an extension of the recent Multiple Denoising (MuDe) technique is proposed, which attenuates the noise present in the dataset instances. Experiments show that the proposed framework achieves satisfactory performance, with outstanding values of accuracy, detection rate and false alarm rate compared with traditional low-rank approximation techniques as well as with related works.

3.2 DATA MODEL

This section introduces the data model used throughout this chapter. First, the dataset matrix $\mathbf{X} \in \mathbb{R}^{M \times N}$ can be defined as follows

$$\mathbf{X} = \mathbf{X}_0 + \mathbf{N}, \quad (3.1)$$

where $\mathbf{X}_0 \in \mathbb{R}^{M \times N}$ is the noise-free dataset matrix, $\mathbf{N} \in \mathbb{R}^{M \times N}$ is the noise matrix, N is the number of features and M is the number of instances. In this chapter, noise refers to data corruptions as a consequence, for instance, of false data injection attacks performed on publicly available datasets. Each column $\mathbf{X}(:, n)$ for $n = 1, \dots, N$ corresponds to the n -th dataset feature, while each row $\mathbf{X}(m, :)$ for $m = 1, \dots, M$ is the m -th dataset instance. Moreover, $\mathbf{y} = [y_1, \dots, y_M]^T \in \mathbb{R}^M$ denotes the class label vector, where y_m indicates if the m -th instance $\mathbf{X}(m, :)$ for $m = 1, \dots, M$ is legitimate traffic or DDoS attack.

The dataset matrix \mathbf{X} in (3.1) can be rewritten in tensor form. In this thesis, the vectors $\mathbf{X}(m, :) \in \mathbb{R}^N$ for $m = 1, \dots, M$ are reshaped as a tensor with dimensions $N_1 \times \dots \times N_R$, with $N = \prod_{r=1}^R N_r$, and stacked along the $(R + 1)$ -th dimension, generating the dataset tensor $\mathfrak{X} \in \mathbb{R}^{N_1 \times \dots \times N_R \times M}$ given by

$$\mathfrak{X} = \mathfrak{X}_0 + \mathfrak{N}, \quad (3.2)$$

where $\mathfrak{X}_0 \in \mathbb{R}^{N_1 \times \dots \times N_R \times M}$ is the noise-free dataset tensor and $\mathfrak{N} \in \mathbb{R}^{N_1 \times \dots \times N_R \times M}$ is the noise tensor. The r -th mode unfolding matrix of \mathfrak{X} , denoted by $[\mathfrak{X}]_{(r)} \in \mathbb{R}^{N_r \times \prod_{j \neq r} N_j M}$, can be obtained by arranging its r -mode fibers as the columns of the resulting matrix. Note that the $(R + 1)$ -th unfolding matrix $[\mathfrak{X}]_{(R+1)} \in \mathbb{R}^{M \times \prod_{r=1}^R N_r}$ corresponds to $\mathbf{X} \in \mathbb{R}^{M \times N}$ in (3.1).

An example of the process of construction of a three-dimensional dataset tensor $\mathfrak{X} \in \mathbb{R}^{N_1 \times N_2 \times M}$ from the dataset matrix $\mathbf{X} \in \mathbb{R}^{M \times N}$ is represented in Fig. 3.1. In this case, each row of the dataset matrix, $\mathbf{X}(m, :) \in \mathbb{R}^N$ for $m = 1, \dots, M$, is folded as a two-dimensional tensor (or matrix) $\mathfrak{X}(:, :, m) \in \mathbb{R}^{N_1 \times N_2}$. Next, all of the M matrices are stacked along the 3rd dimension, generating the three-dimensional tensor \mathfrak{X} . In addition, Figure 3.2 depicts another example containing five different tensor foldings of the m -th dataset instance $\mathbf{X}(m, :) \in \mathbb{R}^N$, where $N = 64$. In each configuration, the instance is folded as an R -th tensor $\mathfrak{X}(:, \dots, m) \in \mathbb{R}^{N_1 \times \dots \times N_R}$, with R varying between 2 and 6. The respective tensor sizes are given by (8×8) , $(4 \times 4 \times 4)$, $(4 \times 4 \times 2 \times 2)$, $(4 \times 2 \times 2 \times 2 \times 2)$ and $(2 \times 2 \times 2 \times 2 \times 2 \times 2)$.

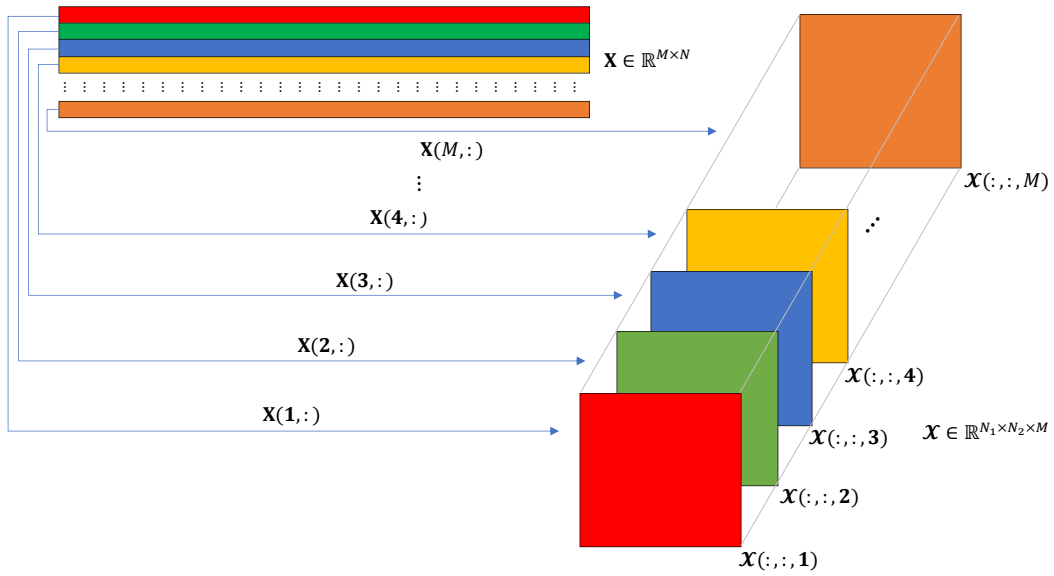


Figure 3.1 – The process of construction of a three-dimensional dataset tensor $\mathcal{X} \in \mathbb{R}^{N_1 \times N_2 \times M}$ from the dataset matrix $\mathbf{X} \in \mathbb{R}^{M \times N}$.

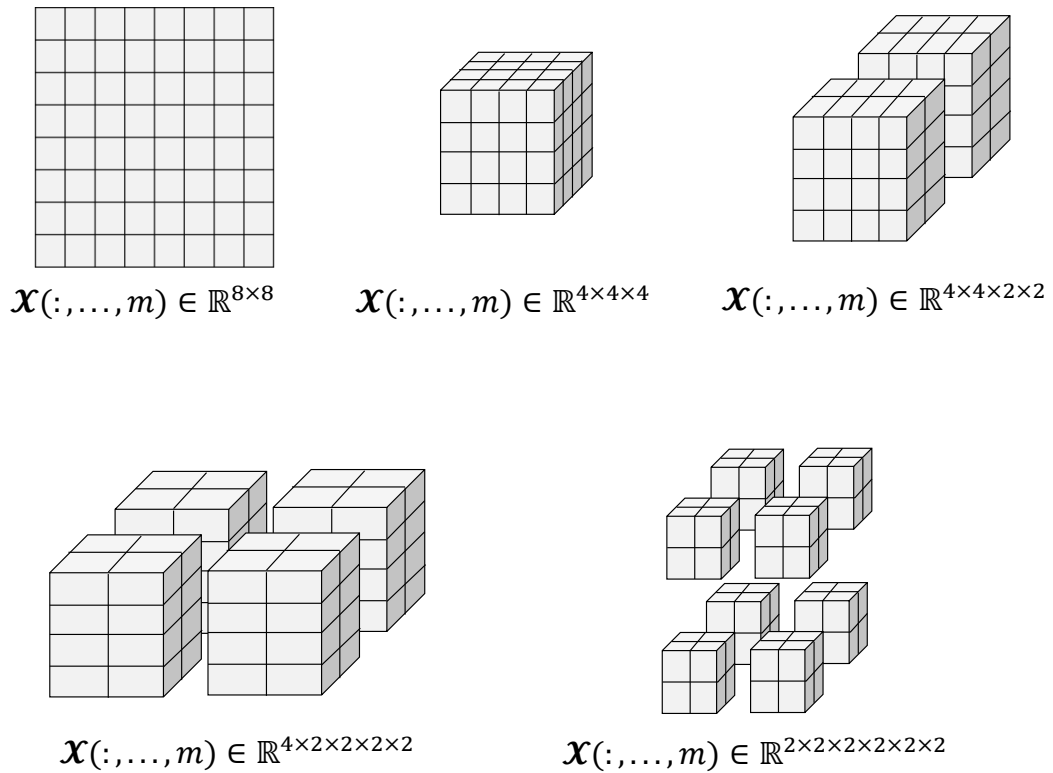


Figure 3.2 – Different tensor foldings of the m -th dataset instance $\mathcal{X}(:, \dots, m) \in \mathbb{R}^{N_1 \times \dots \times N_R}$ for $R=2, \dots, 6$.

3.3 PROPOSED TENSOR MULTIPLE DENOISING BASED FRAMEWORK FOR DDoS ATTACK DETECTION

This section introduces the proposed tensor multiple denoising based framework for DDoS attack detection, which is represented by the block diagram illustrated in Fig. 3.3. Such framework is composed by four major blocks, namely: Data Preprocessing, Dataset Splitting, Dataset Denoising and Machine Learning Supervised Classification. Particularly, in the third block, we propose an extension of the recent MuDe technique for noise attenuation. Data preprocessing and dataset splitting are detailed in Subsections 3.3.1 and 3.3.2, respectively. Next, Subsections 3.3.3 and 3.3.4 present the proposed extended MuDe technique as well as an overview of machine learning supervised classification, respectively.

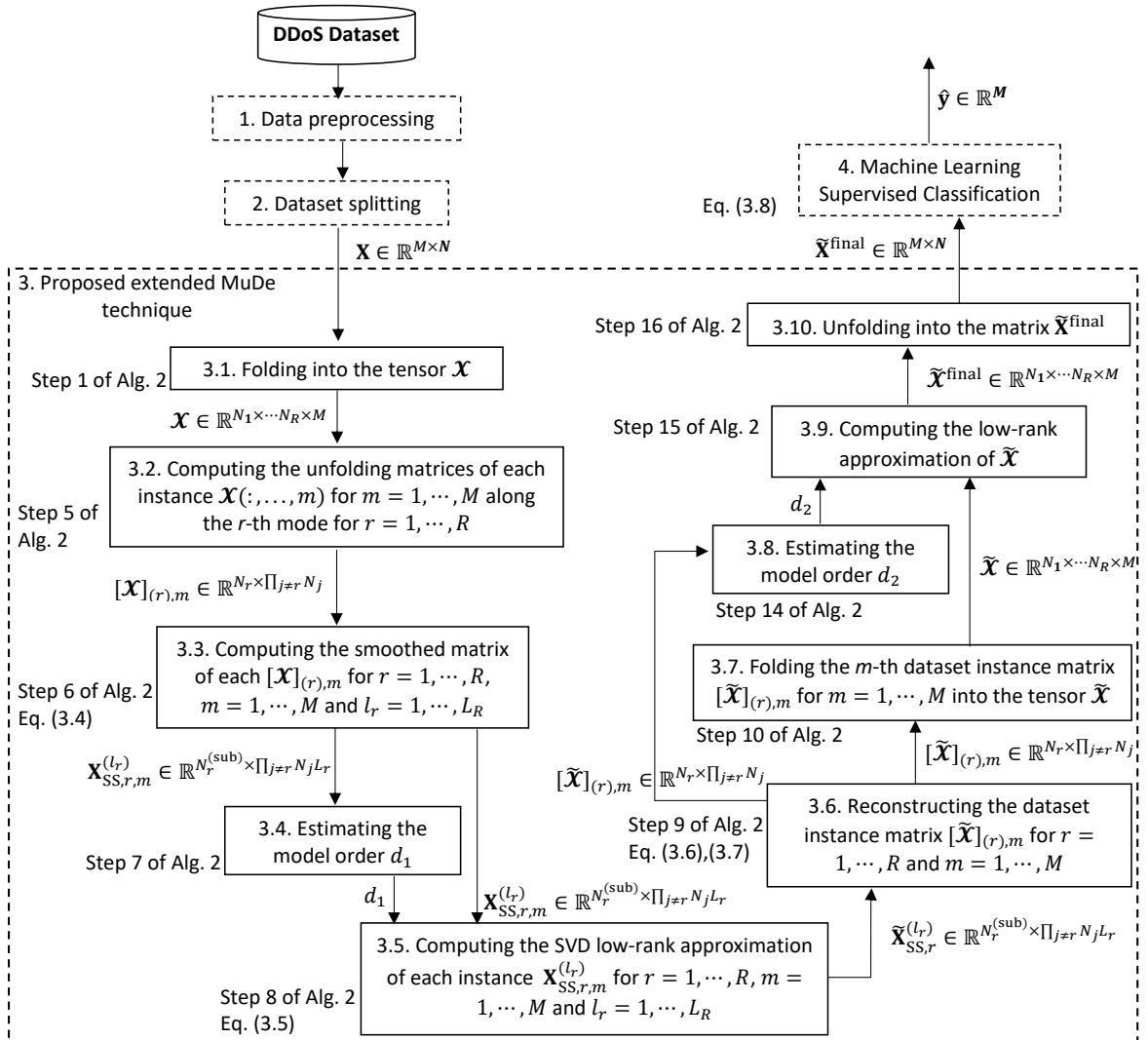


Figure 3.3 – The proposed tensor multiple denoising based framework for DDoS attack detection. The extended MuDe technique is detailed in Blocks 3.1 to 3.10 with the respective references to steps of Algorithm 2 as well as equations of Subsection 3.3.3.

3.3.1 Data Preprocessing

Initially, the DDoS attack dataset is sent to a data preprocessing unit, as depicted by Block 1 of Fig. 3.3, where some operations are performed, such as data cleansing, feature scaling and label encoding. Data instances which contain only Not a Number (NaN) values as well as zero-valued feature vectors are removed. Additionally, if the m -th instance $\mathbf{X}(m, :)$ contains some element $x(m, n)$ with a NaN value, such element is replaced by the mean value of the n -th feature vector. Such method, known as “mean imputation”, is a common missing data handling approach in the machine learning area such that missing values are eliminated from the dataset while preserving the mean of the corresponding feature values. Moreover, all features are re-scaled to the range $[0,1]$, in a process called “normalization”, given by

$$x(m, n) \leftarrow \frac{x(m, n) - \min\{\mathbf{X}(:, n)\}}{\max\{\mathbf{X}(:, n)\} - \min\{\mathbf{X}(:, n)\}}, \quad (3.3)$$

where $\min\{\cdot\}$ and $\max\{\cdot\}$ return the minimum and maximum values of the vectors $\mathbf{X}(:, n)$ for $n = 1, \dots, N$, respectively. Such feature scaling is done because a particular feature with higher order of magnitude could dominate other dataset features, generating skewed results.

Finally, since most of the machine learning algorithms can only process numerical variables, class labels must be converted from categorical to numerical values. Since our framework is developed only for binary classification, legitimate traffic is labeled as “0”, whereas DDoS attacks are labeled as “1”.

3.3.2 Dataset Splitting

After the preprocessing step, the dataset is split into training and testing sets, as depicted by Block 2 of Fig. 3.3. In this chapter, the k -fold cross validation technique is adopted, where data are randomly partitioned into k equally sized samples. In the first iteration, one sample is used for testing data, while the other $k - 1$ samples are used as training data. Such process is repeated k times so that each instance is used once for testing and $k - 1$ times for training. The final results correspond to the average of the results obtained for each round.

The value of k must be chosen such that both training and testing sets are large enough to be statistically representative of the entire dataset. Empirically, the value of k is chosen as 5 or 10. For larger values of k , the difference in size between the training set and the resampling subsets becomes smaller, which reduces the biases, despite the higher computational cost [102]. Since we are dealing with large datasets, training and testing sets which are statistically representative of the original dataset can be obtained even for smaller values of k . Consequently, throughout this chapter, $k = 5$ is adopted in order to obtain a more computationally efficient framework.

3.3.3 Proposed Extended MuDe Technique

After the dataset splitting, the training set is forwarded to the denoising module, as depicted by Block 3 of Fig. 3.3. Similar procedure is adopted for the testing set during the testing phase. For the sake of simplicity, the dataset matrix is referred simply as $\mathbf{X} \in \mathbb{R}^{M \times N}$, which can be the training or testing set depending on the respective phase.

In the context of machine learning supervised classification, datasets are considered to be noise free because we are not able to make assumptions about their base noise type and level [1, 91]. However, several factors such as data source and how the information is collected affect the data quality and, consequently, noise is introduced into the dataset. For example, instances can be incorrectly labeled due to the subjectivity during the labeling process, or dataset features can present corrupted values [103]. In this context, our idea is to apply the Multiple Denoising technique directly on each dataset instance such that better classification results can be achieved.

Since the MuDe algorithm is a fundamental part of our proposed framework, next we briefly overview some of its main characteristics. The Multiple Denoising scheme was originally proposed by Gomes et al. in [75] in order to denoise measurement data collected by multidimensional sensor arrays by applying successive Higher Order Singular Value Decomposition (HOSVD) low-rank approximations. Such algorithm achieves a higher noise reduction by using a mean based reconstruction method where the output signals of several subarrays are spatially smoothed and then averaged along the r -th spatial dimension. Given the outstanding performance presented by MuDe in [75] and [4], such scheme shows a good potential for dataset denoising in order to obtain more accurate attack detection techniques. Nonetheless, MuDe cannot be directly applied to entire datasets because instances with different class labels would be averaged each other along the dataset dimensions, leading to data corruption. Therefore, we include two more contributions on our work by extending the original MuDe algorithm in two ways: (i) by applying the traditional multiple denoising technique directly on each dataset instance, and (ii) by including a low-rank approximation based denoising module. In the former case, we intend to attenuate noise from each instance individually. Furthermore, in the latter case, we want to eliminate, from the entire dataset, noise residuals not removed by MuDe scheme. Appendix A presents the mathematical concepts regarding the state-of-the-art low-rank matrix and tensor based approximation techniques used throughout this chapter.

The proposed extended MuDe technique is composed by ten steps, as depicted by Blocks 3.1 to 3.10 of Fig. 3.3. Initially, in Block 3.1, the dataset matrix $\mathbf{X} \in \mathbb{R}^{M \times N}$ is folded into the $(R + 1)$ -dimensional tensor $\mathfrak{X} \in \mathbb{R}^{N_1 \times \dots \times N_R \times M}$. Next, Block 3.2 computes the unfolding matrix $[\mathfrak{X}]_{(r),m} \in \mathbb{R}^{N_r \times \prod_{j \neq r} N_j}$ of each dataset instance $\mathfrak{X}_{:, \dots, :, m} \in \mathbb{R}^{N_1 \times \dots \times N_R}$ for $m = 1, \dots, M$ along the r -th mode for $r = 1, \dots, R$. Then, in Block 3.3, each unfolding matrix

$[\mathcal{X}]_{(r),m}$ for $m = 1, \dots, M$ and $r = 1, \dots, R$ goes through a process known as smoothing. The r -th smoothed matrix $\mathbf{X}_{\text{SS},r,m}^{(L_r)} \in \mathbb{R}^{N_r^{(\text{sub})} \times \prod_{j \neq r} N_j L_r}$ of the m -th dataset instance can be expressed as [104]

$$\mathbf{X}_{\text{SS},r,m}^{(l_r)} = [[\mathcal{X}]_{(r),m}^{(1)}, \dots, [\mathcal{X}]_{(r),m}^{(l_r)}], \quad (3.4)$$

where $[\mathcal{X}]_{(r),m}^{(l_r)} \in \mathbb{R}^{N_r^{(\text{sub})} \times \prod_{j \neq r} N_j}$ for $l_r = 1, \dots, L_r$ corresponds to the output of the l_r -th subarray for the m -th instance in the r -th dimension, and $N_r^{(\text{sub})} = N_r - l_r + 1$ is the size of each subarray.

In the following, given $\mathbf{X}_{\text{SS},r,m}^{(l_r)} \in \mathbb{R}^{N_r^{(\text{sub})} \times \prod_{j \neq r} N_j L_r}$ from (3.4), Block 3.4 estimates the model order d_1 by using model order selection (MOS) schemes, such as the Minimum Description Length (MDL) [79, 105], Efficient Detection Criterion (EDC) [78], Akaike's Information Theoretic Criteria (AIC) [77, 105], Stein's Unbiased Risk Estimator (SURE) [106] and RADOI [80, 107]. Next, given d_1 , the Singular Value Decomposition (SVD) low-rank approximation of the smoothed matrix $\mathbf{X}_{\text{SS},r,m}^{(l_r)}$ is computed in Block 3.5 as follows

$$\tilde{\mathbf{X}}_{\text{SS},r,m}^{(l_r)} = [[\tilde{\mathcal{X}}]_{(r),m}^{(1)}, \dots, [\tilde{\mathcal{X}}]_{(r),m}^{(l_r)}] = \mathbf{U}_s^{(l_r)} \Sigma_s^{(l_r)} \mathbf{V}_s^{(l_r)\text{H}}, \quad (3.5)$$

where the columns of $\mathbf{U}_s^{(l_r)} \in \mathbb{R}^{N_r \times d_1}$ and $\mathbf{V}_s^{(l_r)} \in \mathbb{R}^{\prod_{j \neq r} N_j \times d_1}$ correspond to the singular vectors of $\tilde{\mathbf{X}}_{\text{SS},r,m}^{(l_r)}$, whereas the diagonal of $\Sigma_s^{(l_r)} \in \mathbb{R}^{d_1 \times d_1}$ contains the singular values of $\tilde{\mathbf{X}}_{\text{SS},r,m}^{(l_r)}$.

In sequence, Block 3.6 of Fig. 3.3 reconstructs each dataset instance $[\tilde{\mathcal{X}}]_{(r),m} \in \mathbb{R}^{N_r \times \prod_{j \neq r} N_j}$ for $m = 1, \dots, M$ and $r = 1, \dots, R$. The multiple denoised unfolding matrices $[\tilde{\mathcal{X}}]_{(r),m}$ as well as their n -th row are given by [75]

$$[\tilde{\mathcal{X}}]_{(r),m} = \begin{bmatrix} [\tilde{\mathcal{X}}]_{(r),m}(1, :) \\ [\tilde{\mathcal{X}}]_{(r),m}(2, :) \\ \vdots \\ [\tilde{\mathcal{X}}]_{(r),m}(N_r, :) \end{bmatrix}, \quad (3.6)$$

$$[\tilde{\mathcal{X}}]_{(r),m}(n, :) = \frac{1}{l} \sum_{i=1}^{l_r} [\tilde{\mathcal{X}}]_{(r),m}^{(i)}(n - i + 1, :), \quad (3.7)$$

where l is the number of times in which $[\tilde{\mathcal{X}}]_{(r),m}^{(i)}(n - i + 1, :)$ is a valid output in the l_r -th subarray of the r -th dimension. After exploiting all possible subarrays $l_r = 1, \dots, L_r$ in each dimension $r = 1, \dots, R$, the dataset tensor $\tilde{\mathcal{X}} \in \mathbb{R}^{N_1 \times \dots \times N_R \times M}$ is reconstructed by arranging (3.6) as a tensor of order $R + 1$. Next, in Block 3.7, each m -th dataset instance matrix $[\tilde{\mathcal{X}}]_{(r),m} \in \mathbb{R}^{N_r \times \prod_{j \neq r} N_j}$ for $m = 1, \dots, M$ and $r = 1, \dots, R$ is folded back into the tensor form $\tilde{\mathcal{X}}_{\dots, \dots, m} \in \mathbb{R}^{N_1 \times \dots \times N_R}$, generating the dataset tensor $\tilde{\mathcal{X}} \in \mathbb{R}^{N_1 \times \dots \times N_R \times M}$.

The following step of the proposed extended MuDe technique is to compute the low-rank approximation of the dataset tensor $\tilde{\mathbf{X}}$, which is depicted in Block 3.9 of Fig. 3.3. Here our main idea is to eliminate noise residuals which eventually were not removed by the successive SVD low-rank approximations performed on the dataset instances. First, the model order d_2 is estimated in Block 3.8 by applying a given MOS technique on the dataset instance $[\tilde{\mathbf{X}}]_{(r),m} \in \mathbb{R}^{N_r \times \prod_{j \neq r} N_j}$ for $m = 1, \dots, M$ and $r = 1, \dots, R$. Then, the low-rank approximation of $\tilde{\mathbf{X}}$ is performed in Block 3.9 according to some state-of-the-art technique, such as the Higher Order Orthogonal Iteration (HOOI) [108] or the Higher Order Singular Value Decomposition (HOSVD) [62, 115]. More details regarding such LRA schemes are shown in Appendix A.

Finally, according to Block 3.10 of Fig. 3.3, the denoised tensor $\tilde{\mathbf{X}}^{\text{final}} \in \mathbb{R}^{N_1 \times \dots \times N_R \times M}$ is unfolded into the matrix form $\tilde{\mathbf{X}}^{\text{final}} \in \mathbb{R}^{M \times N}$ and can be forwarded to machine learning algorithms for classification. The proposed extended MuDe denoising technique is summarized in Algorithm 2.

Algorithm 2: The proposed extended Multiple Denoising (MuDe) algorithm

Input:
- Dataset matrix $\mathbf{X} \in \mathbb{R}^{M \times N}$
- Maximum number of subarrays in each instance dimension: L_r

Output:
- Denoised dataset matrix $\tilde{\mathbf{X}}^{\text{final}} \in \mathbb{R}^{M \times N}$

Algorithm Steps:

- 1 Fold the dataset matrix $\mathbf{X} \in \mathbb{R}^{M \times N}$ into the tensor $\mathbf{X} \in \mathbb{R}^{N_1 \times \dots \times N_R \times M}$
- 2 **for** $m = 1$ to M **do**
- 3 **for** $r = 1$ to R **do**
- 4 **for** $l_r = 1$ to L_r **do**
- 5 Compute the unfolding matrix $[\mathbf{X}]_{(r),m} \in \mathbb{R}^{N_r \times \prod_{j \neq r} N_j}$ of the m -th instance $\mathbf{X}_{:, \dots, :, m} \in \mathbb{R}^{N_1 \times \dots \times N_R}$ along the r -th mode
- 6 Compute the smoothed matrix $\mathbf{X}_{\text{SS},r,m}^{(l_r)} \in \mathbb{R}^{N_r^{(\text{sub})} \times \prod_{j \neq r} N_j L_r}$ of the unfolding matrix $[\mathbf{X}]_{(r),m} \in \mathbb{R}^{N_r \times \prod_{j \neq r} N_j}$ along the r -th mode for the l_r -th subarray size as in (3.4)
- 7 Estimate the model order d_1 by using a MOS scheme
- 8 Compute the SVD low-rank approximation $\tilde{\mathbf{X}}_{\text{SS},r,m}^{(l_r)} \in \mathbb{R}^{N_r^{(\text{sub})} \times \prod_{j \neq r} N_j L_r}$ of the smoothed matrix $\mathbf{X}_{\text{SS},r,m}^{(l_r)} \in \mathbb{R}^{N_r^{(\text{sub})} \times \prod_{j \neq r} N_j L_r}$ along the r -th mode for the l_r -th subarray size as in (3.5)
- 9 Reconstruct the instance matrix $[\tilde{\mathbf{X}}]_{(r),m} \in \mathbb{R}^{N_r \times \prod_{j \neq r} N_j}$ as in (3.6) and (3.7)
- 10 Fold the instance matrix $[\tilde{\mathbf{X}}]_{(r),m} \in \mathbb{R}^{N_r \times \prod_{j \neq r} N_j}$ into the tensor $\tilde{\mathbf{X}}_{:, \dots, :, m} \in \mathbb{R}^{N_1 \times \dots \times N_R}$
- 11 **end**
- 12 **end**
- 13 **end**
- 14 Estimate the model order d_2 by using a MOS scheme
- 15 Compute the low-rank approximation $\tilde{\mathbf{X}}^{\text{final}} \in \mathbb{R}^{N_1 \times \dots \times N_R \times M}$ by using a LRA scheme
- 16 Unfold the denoised dataset tensor $\tilde{\mathbf{X}}^{\text{final}} \in \mathbb{R}^{N_1 \times \dots \times N_R \times M}$ into the matrix $\tilde{\mathbf{X}}^{\text{final}} \in \mathbb{R}^{M \times N}$

3.3.4 Machine Learning Supervised Classification

The final step of the proposed tensor based framework for DDoS attack detection corresponds to the machine learning supervised classification, which is represented in Block 4 of Fig. 3.3. In general, classification algorithms are applied on an input dataset in order to build a model that best fits a relationship between data instances and the respective class labels.

The key idea is to correctly predict the class labels of previously unknown instances, with a good generalization capability [109].

During the training phase, the denoised dataset matrix $\tilde{\mathbf{X}}^{\text{final}}$ originated from Block 3.10 of Fig. 3.3 is used by a machine learning classification algorithm in Block 4 to build a given classification model. Next, in the testing phase, the trained model is applied on an unseen data instance in order to predict whether it is a legitimate traffic or a DDoS attack. In this chapter, the following state-of-the-art classification algorithms are adopted: AdaBoost (AB), Linear Discriminant Analysis (LDA), Logistic Regression (LR) and Random Forest (RFo). For a more detailed explanation about such classifiers, we refer the reader to [109–111].

In summary, if $\text{Cl}(\cdot)$ represents a trained ML classification algorithm in Block 4 of Fig. 3.3, the vector $\hat{\mathbf{y}} \in \mathbb{R}^M$ containing the predicted class labels \hat{y}_m of the m -th dataset instance $\tilde{\mathbf{X}}^{\text{final}}(m, :)$ for $m = 1, \dots, M$ is given by

$$\hat{\mathbf{y}} = \text{Cl}(\tilde{\mathbf{X}}^{\text{final}}) = [\hat{y}_1, \hat{y}_2, \dots, \hat{y}_M]^T. \quad (3.8)$$

3.4 COMPUTATIONAL COMPLEXITY

This section discusses the computational complexity of the proposed extended MuDe technique. Here the computational costs related to folding and unfolding of matrices and tensors are not considered, since such functions are about data representations. For simplicity, the total computational complexity is analyzed for a three-dimensional dataset tensor $\mathbf{X} \in \mathbb{R}^{N_1 \times N_2 \times M}$. In addition, an analysis of the asymptotic time cost is also provided as a function of the largest contributions of the most important variables, namely, N , M , d , R and L , where N is the number of features, M is the number of instances, d is the model order, R is the tensor order and L is the number of subarrays adopted in MuDe computations. Further, the time complexity of the proposed extended MuDe technique is compared with three state-of-the-art LRA schemes, namely, Higher Order Orthogonal Iteration (HOOI) [108], Higher Order Singular Value Decomposition (HOSVD) [115] and Singular Value Decomposition (SVD) [113, 114].

First, the computational complexity of the MuDe technique when applied to M dataset instances is shown. The time cost of the SVD low-rank approximation of a matrix with dimensions $(N_r - l_r + 1) \times (\prod_{j \neq r} N_j l_r)$ truncated to d is given by $\mathcal{O}[(\prod_{j \neq r} N_j)(N_r - l_r + 1)d l_r]$ [75]. Since MuDe computes a total of L_r truncated SVDs for each dimension r , its overall

computational complexity for all of the M dataset instances is given by

$$\mathcal{O}[\text{MuDe}] = \mathcal{O} \left[M \sum_{r=1}^2 \sum_{l_r=1}^{L_r} N_j (N_r - l_r + 1) dl_r \right], j \neq r. \quad (3.9)$$

Finally, the overall computational complexity of the proposed feature extraction technique corresponds to the sum of the complexity in (3.9), plus the costs $\mathcal{O}[\text{MOS}]$ and $\mathcal{O}[\text{LRA}]$ for the adopted MOS and low-rank approximation algorithms, i.e.,

$$\mathcal{O}[\text{Final}] = \mathcal{O}[\text{MuDe}] + \mathcal{O}[\text{MOS}] + \mathcal{O}[\text{LRA}]. \quad (3.10)$$

The time cost of the SVD of $\mathbf{X} \in \mathbb{R}^{M \times N}$ is given by $\mathcal{O}[\text{SVD}] = \mathcal{O}[k_t dNM]$, where $N = N_1 \cdot N_2$ and k_t is a constant that depends on the design of the algorithm. Moreover, according to [112], the overall computational complexity of HOOI can be expressed as $\mathcal{O}[\text{HOOI}] = \mathcal{O}[N_{\max}^3 dJ] + \mathcal{O}[N_{\max}^2 d^2 J] + \mathcal{O}[N_{\max}^3 d] + \mathcal{O}[N_{\max} d^3]$, where $N_{\max} = \max\{N_1, N_2, M\}$ and J is the number of iterations of the HOOI algorithm. In addition, the computational complexity of the HOSVD low-rank approximation of $\mathbf{X} \in \mathbb{R}^{N_1 \times N_2 \times M}$ can be expressed as $\mathcal{O}[\text{HOSVD}] = \mathcal{O}[\sum_{j=1}^3 (N_j \prod_{k=1}^3 N_k) + \sum_{j=1}^3 (\prod_{k=1}^j d \prod_{k=j}^3 N_k)]$, where N_3 corresponds to M for simplicity. Table 3.2 summarizes the computational complexities of the proposed extended MuDe technique as well as its competitor schemes. The second column illustrates the total computational cost, whereas the last column shows the final complexities, corresponding to the asymptotic dominant terms. From the results shown in Table 3.2, it is observed that the proposed extended MuDe technique presents the highest computational complexity, which composed by two dominant terms: $\mathcal{O}[dLRNM]$, corresponding to the MuDe algorithm, and either $\mathcal{O}[dJRM^3]$ or $\mathcal{O}[RNM^2]$, depending on the adopted LRA scheme, i.e., HOOI or HOSVD, respectively. In this sense, the MuDe algorithm presents a considerable impact on the mean processing times, as it will be shown in the numerical simulations of Section 3.5. Such fact reinforces the trade-off between the more accurate DDoS attack detection and the computational complexity in our proposed scheme. In addition, note that, among the competing schemes, the HOOI low-rank approximation technique presents the highest computational complexity, especially due to the number of dataset instances raised to the third power.

Table 3.2 – Computational complexity of the following schemes: (1) proposed extended MuDe technique with HOOI, (2) proposed extended MuDe technique with HOSVD, (3) HOOI, (4) HOSVD, and (5) SVD.

Model	Total Complexity	Dominant Complexity
(1)	$\mathcal{O}[\text{MOS}] + \mathcal{O}\left[M \sum_{r=1}^2 \sum_{l_r=1}^{L_r} N_j(N_r - l_r + 1)dl_r\right] + \mathcal{O}(N_{\max}^3 dJ) + \mathcal{O}(N_{\max}^2 d^2 J) + \mathcal{O}(N_{\max}^3 d) + \mathcal{O}(N_{\max} d^3), j \neq r$	$\mathcal{O}[dLRNM] + \mathcal{O}[dJRM^3]$
(2)	$\mathcal{O}[\text{MOS}] + \mathcal{O}\left[M \sum_{r=1}^2 \sum_{l_r=1}^{L_r} N_j(N_r - l_r + 1)dl_r\right] + \mathcal{O}\left[\sum_{j=1}^3 \left(N_j \prod_{k=1}^3 N_k\right) + \sum_{j=1}^3 \left(\prod_{k=1}^j d \prod_{k=j}^3 N_k\right)\right]$	$\mathcal{O}[dLRNM] + \mathcal{O}[RNM^2]$
(3)	$\mathcal{O}[N_{\max}^3 dJ] + \mathcal{O}[N_{\max}^2 d^2 J] + \mathcal{O}[N_{\max}^3 d] + \mathcal{O}[N_{\max} d^3]$	$\mathcal{O}[dJRM^3]$
(4)	$\mathcal{O}\left[\sum_{j=1}^3 \left(N_j \prod_{k=1}^3 N_k\right) + \sum_{j=1}^3 \left(\prod_{k=1}^j d \prod_{k=j}^3 N_k\right)\right]$	$\mathcal{O}[RNM^2]$
(5)	$\mathcal{O}[k_t dNM]$	$\mathcal{O}[k_t dNM]$

3.5 SIMULATION RESULTS

This section is divided into three subsections. First, Subsection 3.5.1 presents details about the features and samples extracted from the DDoS benchmark datasets used in this chapter. Next, simulations results are shown and discussed in Subsections 3.5.2 and 3.5.3, respectively.

3.5.1 DDoS Attack Datasets

This subsection describes the number of instances as well as the DDoS attack types applied on numerical simulations. A subset of the CIC-DDoS2019 dataset is used to evaluate the performance of the proposed framework for several configuration scenarios. However, since CIC-DDoS2019 is a recent dataset, few related works have been found in the literature for performance comparison. Therefore, as NSL-KDD has been extensively used for NIDS validation, performance evaluation considering such dataset is also included. More details about the CIC-DDoS2019 and NSL-KDD datasets can be found in Appendix C.

In this chapter, we use a subset of the CIC-DDoS2019 composed by 32,000 instances and 64 features. Table 3.3 shows the types of DDoS attacks and their respective number of samples extracted from the CIC-DDoS2019 dataset. Since DDoS attacks are not as frequent as normal traffic, 80% of the dataset is represented by legitimate traffic. All of the DDoS attack instances are labeled as “1”, whereas legitimate traffic is coded as “0”. In addition, the 5-fold cross validation technique is applied for dataset splitting, i.e., at each round, 80% of the dataset is used for training and 20% for testing. Consequently, each data instance is used once for testing and four times for training.

Furthermore, the NSL-KDD dataset is applied only for comparison between our proposed approach and related works which used the same data in their experiments. Table 3.4 details the types of DDoS attacks and their respective number of samples extracted from

Table 3.3 – DDoS attack types used in this chapter as well as the corresponding number of instances extracted from the CIC-DDoS2019 dataset.

Traffic Type	Total
Legitimate	32,000
DNS-based DDoS	800
LDAP-based DDoS	800
MSSQL-based DDoS	800
NetBIOS-based DDoS	800
NTP-based DDoS	800
SNMP-based DDoS	800
SSDP-based DDoS	800
UDP flood	800
SYN flood	800
TFTP-based DDoS	800
Total Legitimate Traffic	32,000
Total DDoS Attack	8,000
Total Nr of Instances	40,000

the NSL-KDD dataset. Similarly to CIC-DDoS2019, all of the legitimate and DDoS attack instances are labeled as “0” and “1”, respectively.

Table 3.4 – DDoS attack types used in this chapter as well as the corresponding number of instances extracted from the NSL-KDD dataset.

Traffic Type	Training Set	Testing Set	Total
Legitimate	67,343	9,710	77,053
Neptune	41,214	4,657	45,871
Teardrop	892	12	904
Smurf	2,646	665	3,311
Pod	201	41	242
Back	956	359	1,315
Land	18	7	25
UDPStorm	0	2	2
Apache2	0	737	737
ProcessTable	0	685	685
MailBomb	0	293	293
Total Legitimate Traffic	67,343	9,710	77,053
Total DDoS Attack	45,927	7,458	53,385
Total Nr of Instances	113,270	17,168	130,438

3.5.2 Results

This subsection presents the performance evaluation of the proposed tensor multiple denoising based framework for DDoS attack detection through simulations. All experiments were executed on a desktop computer with processor Intel Core i7-2600 3.40 GHz and 16 GB of RAM. MATLAB R2018a software was used to simulate data preprocessing and ten-

sor decompositions, whereas machine learning classifier algorithms were implemented in the Python Scikit-Learn and Keras libraries. In this chapter, Accuracy (Acc), Detection Rate (DR) and False Alarm Rate (FAR), defined in Eq. (D.1), (D.3) and (D.4) of Appendix D, are used as performance evaluation metrics. Further, AdaBoost (AB), Linear Discriminant Analysis (LDA), Logistic Regression (LR) and Random Forest (RFo) are used for classification. Appendix E summarizes the main simulation parameters adopted for each ML classifier.

As described in Eq. (3.1), the dataset matrix is given by $\mathbf{X} \in \mathbb{R}^{M \times N}$, where N is the number of features and M is the number of dataset instances. For CIC-DDoS2019 dataset, all of the $N = 64$ features described in Table C.2 of Appendix C were considered. Moreover, 40,000 instances were collected from the CIC-DDoS2019 dataset, as shown in Table 3.3, such that 32,000 training instances and 8,000 testing instances are taken in each iteration of the 5-fold cross validation. Moreover, a similar procedure is adopted when the NSL-KDD dataset is applied. All of the $N = 36$ features described in Table 3.4 are used in simulations. Nonetheless, since NSL-KDD is composed by separate training and testing sets, there is no need for dataset splitting. Thus, 113,270 training instances and 17,168 testing instances were directly collected from the corresponding datasets, as shown in Table 3.4.

Additionally, the number of subarrays $L_1 = L_2 = 2$ is adopted in the MuDe computations described in (3.4). The Multiple Denoising technique presents a higher noise reduction ability, which demands higher processing times due to its L_r for $r = 1, \dots, R$ truncated SVD computations [75]. Thus, L_r must be chosen such that a good trade-off between denoising capability and computational cost is achieved, which is well accomplished when $L_1 = L_2 = 2$. Moreover, since the initial noise type and level present in the DDoS attack datasets publicly available on the web are unknown, they are assumed to be noise free [91]. Consequently, in all experiments, Signal-to-Noise Ratio (SNR) values are pre-defined by adding a zero-mean white Gaussian noise into the dataset features in a supervised manner such that the framework performance can be assessed for different noise levels. In this chapter, the SNR is defined as follows

$$\text{SNR} = \frac{\sigma_{\mathbf{x}}^2}{\sigma_{\mathbf{N}}^2}, \quad (3.11)$$

where $\sigma_{\mathbf{x}}^2$ and $\sigma_{\mathbf{N}}^2$ are the variances of the tensors \mathbf{X} and \mathbf{N} , respectively. Alternatively, the SNR can be expressed in decibels as follows

$$\text{SNR}_{\text{dB}} = 10 \log_{10} \left(\frac{\sigma_{\mathbf{x}}^2}{\sigma_{\mathbf{N}}^2} \right). \quad (3.12)$$

First, the performance evaluation of our proposed framework is shown when two state-of-the-art low-rank approximation algorithms are adopted in Block 3.9 of Fig. 3.3, namely, HOSVD and HOOI. In simulations, the SNR is fixed in 10 dB, $M = 40,000$ and the dataset

matrix $\mathbf{X} \in \mathbb{R}^{M \times N}$ is folded into a three-dimensional tensor $\mathfrak{X} \in \mathbb{R}^{N_1 \times N_2 \times M}$, with $N_1 = N_2 = 8$. Moreover, the dataset is split using 5-fold cross validation and the model order selection is performed by using the MDL [79, 105] technique. Table 3.5 illustrates the values of Acc, DR and FAR obtained by the proposed framework when HOOI and HOSVD are adopted as low-rank approximation techniques, considering different ML classification algorithms. For each SNR and ML classifier, the best metric value obtained by a given approach is highlighted in bold. From the results shown in Table 3.5, it can be observed that the proposed approach with HOSVD outperforms its HOOI based counterpart in some simulation scenarios, whereas the opposite behavior is verified in other experiments. For example, considering SNR = 5 dB, the proposed framework with HOOI shows better DR and FAR, regardless of the ML classifier. On the other hand, the HOOI based approach is outperformed by the proposed scheme with HOSVD, in terms of detection rate and false alarm rate, when the SNR is fixed in 0 dB. Additionally, when SNR = 15 dB, it can be seen that the assessed models present better or worse performances, depending on the applied ML classifier. For instance, in terms of detection rate, considering the AdaBoost and Logistic Regression classifiers, the proposed framework with HOSVD achieves 92.10% and 95.49%, respectively, against 91.95% and 94.66% obtained by the HOOI based approach. Inversely, considering the same metrics but now with LDA and RFo algorithms, the HOOI based scheme obtained 88.45% and 96.96%, respectively, whereas the values of 85.74% and 96.89% are reported when HOSVD is adopted as LRA scheme.

Table 3.5 – Performance evaluation, considering different LRA schemes, for the following models: (1) proposed framework with HOOI, and (2) proposed framework with HOSVD.

SNR	Model	Accuracy				Detection Rate				False Alarm Rate			
		AB	LDA	LR	RFo	AB	LDA	LR	RFo	AB	LDA	LR	RFo
-5 dB	(1)	0.8530	0.8676	0.8675	0.8532	0.7552	0.7328	0.7320	0.7339	0.4062	0.4897	0.4915	0.4630
	(2)	0.8588	0.8572	0.8556	0.8605	0.7509	0.7549	0.7556	0.7253	0.4271	0.4138	0.4094	0.4977
0 dB	(1)	0.9005	0.8998	0.9051	0.9038	0.7613	0.7684	0.8097	0.7669	0.4685	0.4486	0.3477	0.4590
	(2)	0.9058	0.9073	0.9008	0.9215	0.7896	0.7982	0.8363	0.8272	0.4021	0.3818	0.2701	0.3286
5 dB	(1)	0.9377	0.9340	0.9386	0.9638	0.9176	0.8797	0.9013	0.9281	0.1108	0.2101	0.1602	0.1307
	(2)	0.9202	0.9363	0.9412	0.9616	0.8955	0.8673	0.8922	0.9124	0.1452	0.2467	0.1888	0.1687
10 dB	(1)	0.9672	0.9437	0.9621	0.9769	0.9375	0.8697	0.9156	0.9472	0.1116	0.2524	0.1611	0.1017
	(2)	0.9716	0.9300	0.9563	0.9817	0.9325	0.8328	0.8976	0.9571	0.1320	0.3275	0.1993	0.0834
15 dB	(1)	0.9658	0.9446	0.9737	0.9862	0.9195	0.8845	0.9466	0.9696	0.1568	0.2145	0.0982	0.0579
	(2)	0.9612	0.9370	0.9756	0.9863	0.9210	0.8574	0.9549	0.9689	0.1452	0.2740	0.0792	0.0598

In addition, Table 3.6 shows the mean training times, in seconds, obtained by the experiments whose results are shown in Table 3.5. Note that, for a given SNR and ML classifier, the mean training times of the proposed framework when HOOI is chosen as LRA scheme are higher compared to those of the HOSVD based configuration. For instance, when SNR = 15 dB, the proposed algorithm with HOSVD and AdaBoost classifier achieves 37.18 s, against 137.13 s obtained when HOOI is used for dataset low-rank approximation. The re-

sults shown in Table 3.5 are in line with the computational complexities reported in Table 3.2, in which HOOI presents higher asymptotic complexity compared to HOSVD. The Higher Order Orthogonal Iteration is an iterative alternating least-squares algorithm which estimates the best rank- (d_1, \dots, d_{R+1}) approximation of the dataset tensor $\mathcal{X} \in \mathbb{R}^{N_1 \times \dots \times N_R \times M}$. On the other hand, the Higher Order Singular Value Decomposition computes the low-rank approximation of \mathcal{X} through the singular value decompositions of each r -th unfolding matrix $[\mathcal{X}]_{(r)} \in \mathbb{R}^{N_r \times \prod_{j \neq r} N_j M}$, truncated to the rank d_r for $r = 1, \dots, R + 1$. In this sense, from this point on the HOSVD algorithm is adopted as low-rank approximation technique in our proposed framework, since low processing times are required in network intrusion detection problems.

Table 3.6 – Mean training times (in seconds), considering different LRA schemes, for the following models: (1) proposed framework with HOOI, and (2) proposed framework with HOSVD.

SNR	Mean Training Time (s)							
	(1)				(2)			
	AB	LDA	LR	RFo	AB	LDA	LR	RFo
-5 dB	98.16	82.14	81.94	105.22	36.69	21.24	21.07	41.41
0 dB	103.31	87.79	87.69	110.32	36.54	21.06	21.03	43.46
5 dB	113.59	97.12	97.02	122.53	36.71	21.14	21.10	43.99
10 dB	120.08	104.16	104.10	127.07	37.02	21.38	21.38	44.84
15 dB	137.13	120.31	120.43	141.80	37.18	21.79	21.88	44.93

Next, the performance evaluation of our proposed framework is shown for different tensor sizes. In each configuration, the instance is folded as an R -th tensor $\mathcal{X}(:, \dots, m) \in \mathbb{R}^{N_1 \times \dots \times N_R}$ for $R = 2, \dots, 6$, with sizes given by (8×8) , $(4 \times 4 \times 4)$, $(4 \times 4 \times 2 \times 2)$, $(4 \times 2 \times 2 \times 2 \times 2)$ and $(2 \times 2 \times 2 \times 2 \times 2)$, respectively. The different tensor foldings of the m -th dataset instance $\mathbf{X}(m, :) \in \mathbb{R}^N$ are illustrated in Figure 3.2. In all simulated scenarios, SNR = 10 dB and $M = 40,000$. Further, the dataset is split using 5-fold cross validation and the MDL [79, 105] is used for model order selection. Table 3.7 shows the values of Acc, DR and FAR obtained for each tensor folding considering the AB, LDA, LR and RFo classifiers. For each classifier, the best metric value obtained by a given tensor folding is highlighted in bold. From Table 3.7, it can be observed that all classifiers showed different behaviors, depending on the adopted tensor configuration. For example, in terms of detection rate, the four-dimensional configuration outperforms the other schemes when LDA and LR are adopted, showing values of 91.69% and 93.91%, respectively. Moreover, the accuracy of the 6D configuration is also superior compared to its competitors when AdaBoost is used as classifier, with a value of 97.41%. On the other hand, for RFo, the configuration 8×8 outperforms the competing schemes in terms of Acc, DR and FAR, achieving 98.17%, 95.71% and 8.30%, respectively. Furthermore, the mean training times (in seconds) of the proposed framework considering the above-mentioned tensor foldings are shown in Table 3.8. Note that the processing times are higher as the tensor order is larger, regardless of the classifier. In this sense, despite larger tensor order configurations present better performance for some

ML classification algorithms, from this point on the two-dimensional instance configuration, 8×8 , is adopted in simulations due to its lower processing time, which is a fundamental requirement in intrusion detection problems. Thus, the dataset matrix $\mathbf{X} \in \mathbb{R}^{M \times N}$ is folded into a three-dimensional tensor $\mathfrak{X} \in \mathbb{R}^{N_1 \times N_2 \times M}$, where $N_1 = N_2 = 8$. Similarly, when NSL-KDD is used for model validation, the configuration $N_1 = N_2 = 6$ is adopted.

Table 3.7 – Performance evaluation of the proposed framework for different instance configurations.

Tensor Size	Accuracy				Detection Rate				False Alarm Rate			
	AB	LDA	LR	RFo	AB	LDA	LR	RFo	AB	LDA	LR	RFo
8×8	0.9716	0.9300	0.9563	0.9817	0.9325	0.8328	0.8976	0.9571	0.1320	0.3275	0.1993	0.0830
$4 \times 4 \times 4$	0.9715	0.9380	0.9579	0.9782	0.9362	0.8564	0.9037	0.9483	0.1220	0.2782	0.1857	0.1012
$4 \times 4 \times 2 \times 2$	0.9197	0.9410	0.9527	0.9594	0.9226	0.9169	0.9391	0.9388	0.0726	0.1230	0.0834	0.0952
$4 \times 2 \times 2 \times 2 \times 2$	0.9533	0.9285	0.9554	0.9648	0.9114	0.9159	0.9224	0.9272	0.1578	0.1050	0.1320	0.1348
$2 \times 2 \times 2 \times 2 \times 2 \times 2$	0.9741	0.9383	0.9618	0.9808	0.9562	0.8571	0.9188	0.9567	0.0734	0.2769	0.1522	0.0835

Table 3.8 – Mean training times (in seconds) of the proposed framework for different instance configurations.

Tensor Size	Mean Training Time (s)			
	AB	LDA	LR	RFo
8×8	37.13	21.50	21.49	43.95
$4 \times 4 \times 4$	71.45	54.41	54.49	78.63
$4 \times 4 \times 2 \times 2$	89.95	61.85	62.14	101.79
$4 \times 2 \times 2 \times 2 \times 2$	110.37	94.67	94.84	124.21
$2 \times 2 \times 2 \times 2 \times 2 \times 2$	131.36	115.38	115.42	136.96

Then, the performance evaluation of the proposed framework is presented for different MOS schemes, namely, AIC [77, 105], EDC [78], MDL [79, 105], RADOI [80] and Stein’s Unbiased Risk Estimator (SURE) [106]. Table 3.9 presents the experiment results for the AB, LDA, LR and RFo classification algorithms, with SNR fixed in 10 dB. From the values shown in Table 3.9, it can be observed that all of the MOS schemes present better performance when random forest algorithm is applied for classification. Random forests are composed by multiple combined decision trees, which can handle datasets with higher dimensionality, obtaining more stable and robust predictions. In addition, from the results highlighted in bold in Table 3.9, it can be seen that RADOI outperforms the other MOS techniques in terms of detection rate and false alarm rate. For example, considering the RFo classification algorithm, RADOI achieves DR and FAR of 96.77% and 5.21%, respectively, against 90.83% and 18.11% obtained by EDC. In this sense, from this point on, the RADOI algorithm is adopted as the MOS scheme in our proposed framework, regardless of the ML classifier.

Following, the proposed framework is compared to NIDS in which state-of-the-art low-rank approximation techniques are previously applied to the dataset, namely, Higher Order Orthogonal Iteration (HOOI) [108], Higher Order Singular Value Decomposition (HOSVD) [62, 115] and Singular Value Decomposition (SVD) [113, 114]. In simulations, the SNR

Table 3.9 – Performance evaluation of the proposed framework considering different MOS techniques.

MOS	Accuracy				Detection Rate				False Alarm Rate			
	AB	LDA	LR	RFo	AB	LDA	LR	RFo	AB	LDA	LR	RFo
AIC [77, 105]	0.9679	0.9513	0.9676	0.9790	0.9364	0.9005	0.9364	0.9518	0.1155	0.1834	0.1149	0.0932
EDC [78]	0.9493	0.9261	0.9572	0.9624	0.8797	0.8301	0.9025	0.9083	0.2352	0.3283	0.1878	0.1811
MDL [79, 105]	0.9716	0.9300	0.9563	0.9822	0.9325	0.8328	0.8976	0.9571	0.1320	0.3275	0.1993	0.0830
RADOI [80]	0.9494	0.9511	0.9667	0.9817	0.9488	0.9108	0.9506	0.9677	0.0561	0.1557	0.0759	0.0521
SURE [106]	0.9691	0.9452	0.9674	0.9801	0.9466	0.8863	0.9433	0.9548	0.0906	0.2110	0.0967	0.0868

ranges from -5 dB to 15 dB. Additionally, since SVD is a matrix based denoising technique, in this case the dataset is considered in its matrix form, $\mathbf{X} \in \mathbb{R}^{M \times N}$, with $N = 64$ features. The accuracy, detection rate and false alarm rate as a function of the SNR are assessed in Fig. 3.4 to 3.6 for the proposed framework, as well as the SVD, HOSVD and HOOI schemes considering the AB, LDA, LR and RFo classifiers. As expected, all of the compared techniques show better performance as the SNR is higher. Furthermore, especially for the Linear Discriminant Analysis, Logistic Regression and Random Forest classifiers, the proposed scheme outperforms its competitor methods in terms of DR and FAR for low SNR values. Moreover, considering accuracy, our proposed approach with AB and RFo classifiers deliver better results compared to the SVD, HOSVD and HOOI algorithms.

Then, the proposed framework is assessed against the SVD, HOSVD and HOOI low-rank approximation techniques for different training dataset size proportions. The dataset is split into training and testing sets, where the proportion of the training data ranges from 20% to 70% of the original dataset, with SNR fixed in 10 dB. Fig. 3.7 to 3.9 show the accuracy, detection rate and false alarm rate as a function of the Training Size Proportion (TSP) considering the AB, LDA, LR and RFo classification algorithms. Note that all of the compared techniques present better performance as the training dataset size proportion grows. Furthermore, it can be seen that, in terms of DR and FAR, our proposed framework outperforms the competitor methods in most of the training size proportion range, especially when AB, LR and RFo are used as ML classifiers. In addition, the proposed scheme delivers better accuracy, compared to the HOOI, HOSVD and SVD techniques, when adopting Logistic Regression and Random Forest algorithms.

Next, experiment results obtained from the comparison between our proposed approach and related works are introduced. Since CIC-DDoS2019 is a novel dataset, few related works applying such data for NIDS validation have been found in the literature. In this sense, as NSL-KDD has been widely applied to validate intrusion detection systems, performance evaluation considering such dataset is also included. Furthermore, since the related works consider noise-free datasets, the proposed scheme is simulated considering a very low noise level, with SNR fixed in 40 dB. Table 3.10 shows the classification method and the values of accuracy, detection rate and false alarm rate obtained by all of the compared ap-

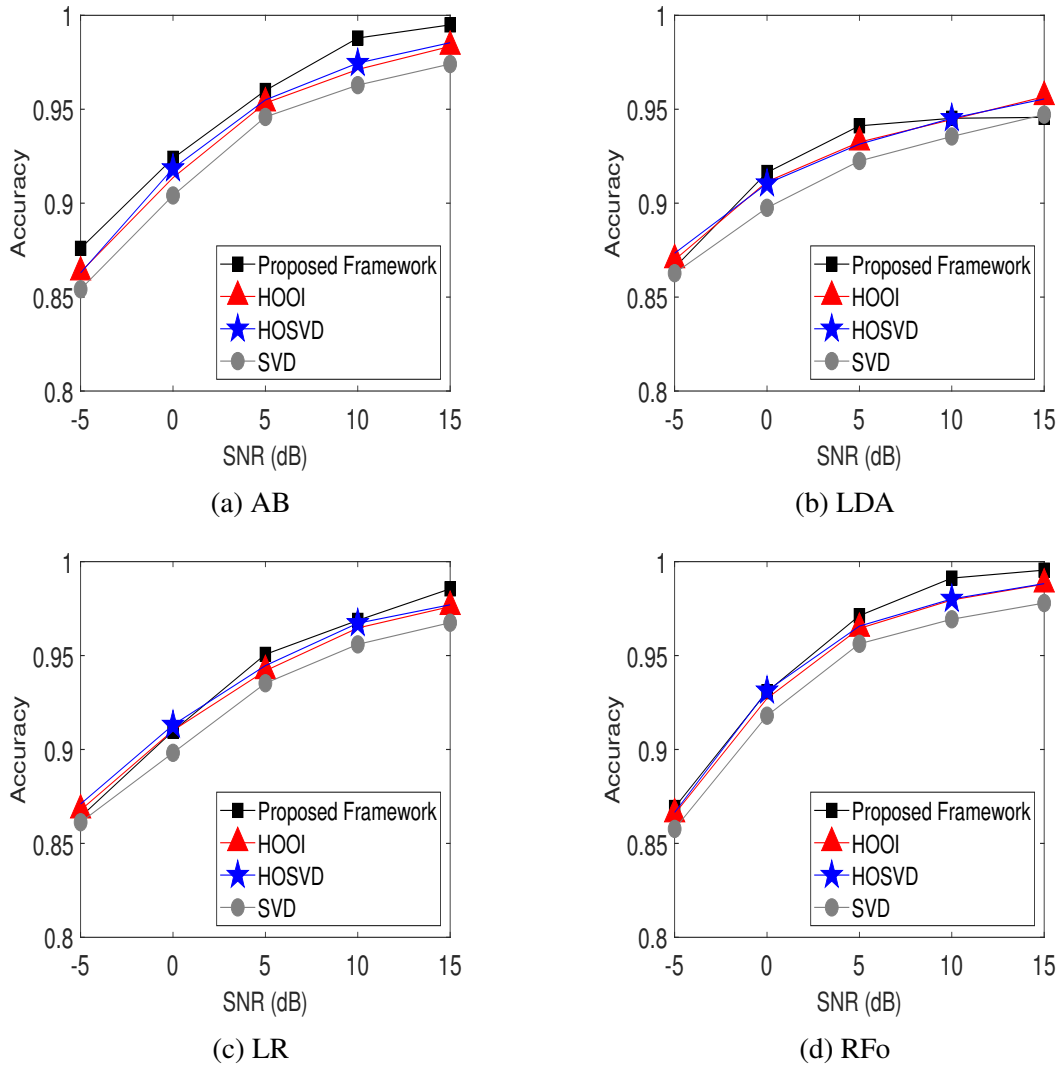


Figure 3.4 – Plots of accuracy, as a function of the signal-to-noise ratio (dB), considering: (a) AB, (b) LDA, (c) LR, and (d) RFo classification algorithms.

proaches. Some metrics are represented as Not Available (N/A) because the corresponding works did not present such values. Note that, when NSL-KDD is applied for validation, the proposed framework presents outstanding results, outperforming the related works, in terms of accuracy, when all of the ML algorithms but LDA are used for classification. However, our approach with AdaBoost is outperformed by Gogoi et al. [116] in terms of DR, and by Wang et al. [23] when considering FAR. Similarly, when CIC-DDoS2019 is applied on simulations, the proposed scheme achieves considerable results, outperforming most of the compared techniques, in terms of detection rate, when AdaBoost is adopted as ML classifier.

Finally, Fig. 3.10 illustrates the mean training times (in seconds) of the proposed framework and the competing schemes as a function of the training size proportion, considering the AB, LDA, LR and RFo classifiers. Note that the proposed framework is more computationally expensive than the competing approaches, showing a considerable increase in the

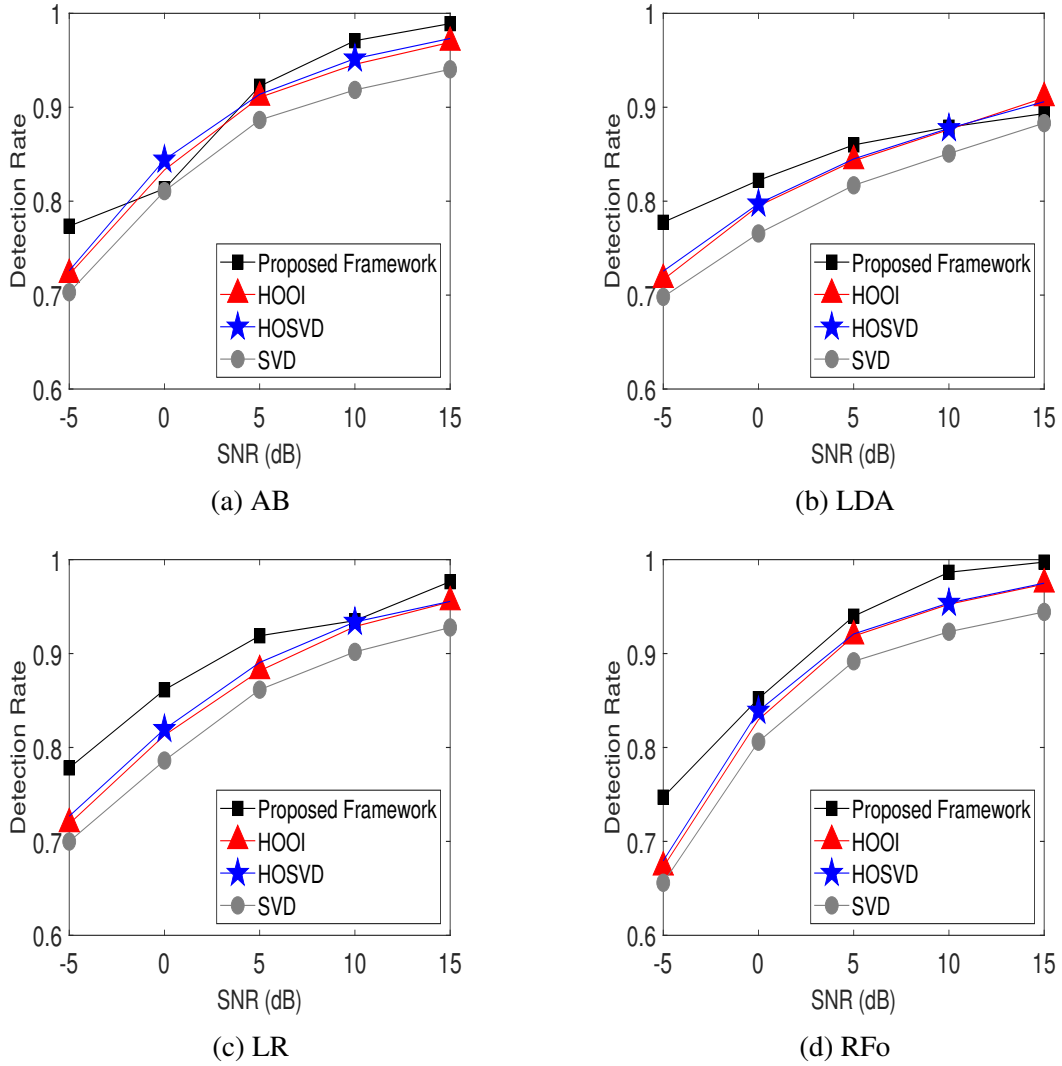


Figure 3.5 – Plots of detection rate, as a function of the signal-to-noise ratio (dB), considering: (a) AB, (b) LDA, (c) LR, and (d) RFo classification algorithms.

mean training times.

3.5.3 Discussion

In this subsection, the results presented in Subsection 3.5.2 are discussed. First, Fig. 3.4 to 3.6 illustrate the values of Acc, DR and FAR as a function of the SNR for all competing schemes considering the AB, LDA, LR and RFo classifiers. Since data corruption in datasets can degrade the performance of ML classification algorithms, all compared approaches present better results for higher values of SNR. Furthermore, we observe that the proposed framework delivers better detection rate and false alarm rate compared to its competing schemes for almost all SNR range, for LDA, LR and RFo classification algorithms, especially for low SNR values. In such cases, the benefits of our proposed extended MuDe algorithm are more evident because it provides a higher noise reduction due to the multiple

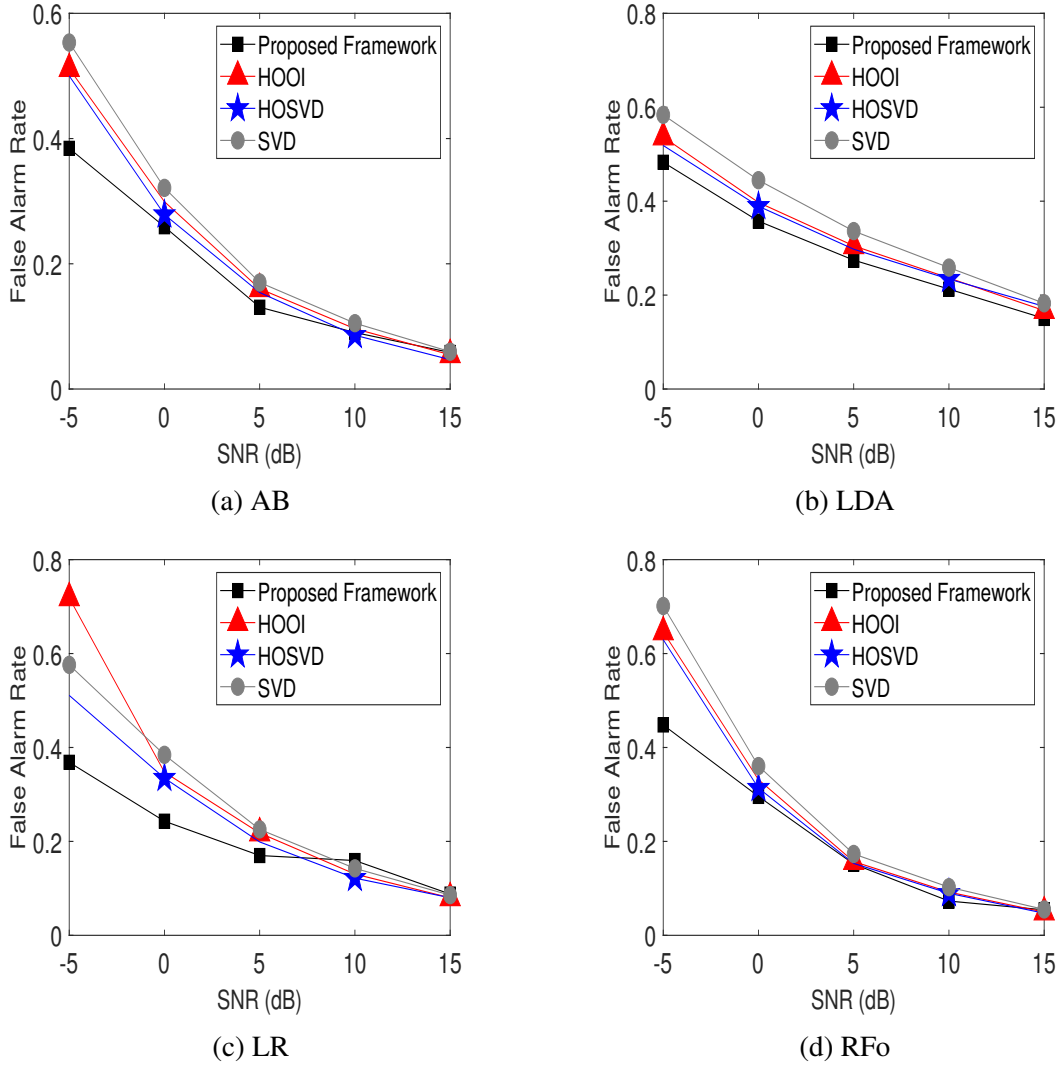


Figure 3.6 – Plots of false alarm rate, as a function of the signal-to-noise ratio (dB), considering: (a) AB, (b) LDA, (c) LR, and (d) RFo classification algorithms.

denoising performed along the r -th dimension of the m -th dataset instance for $r = 1, \dots, R$ and $m = 1, \dots, M$. However, the proposed framework is outperformed by the competitor methods in terms of accuracy, for low values of SNR, when the aforementioned classifiers are used. In this situation, despite the superior performance in terms of detection rate and false alarm rate, the proposed framework achieves a higher number of false negatives, i.e., true attack traffic is wrongly classified as legitimate. In addition, note that the competing techniques present slightly different performances, regardless of the classifier, for all SNR range. Since the third dimension (corresponding to the number of instances M) of the dataset tensor \mathcal{X} is much larger than the first and second dimensions (corresponding to the number of features N_1 and N_2), the competing multidimensional techniques, HOSVD and HOOI, do not provide significant gain compared to the matrix based scheme, SVD. Therefore, the results shown in Fig. 3.4 to 3.6 highlight the higher robustness of our proposed framework against data corruptions. Further, the multiple denoising introduced by our approach along different

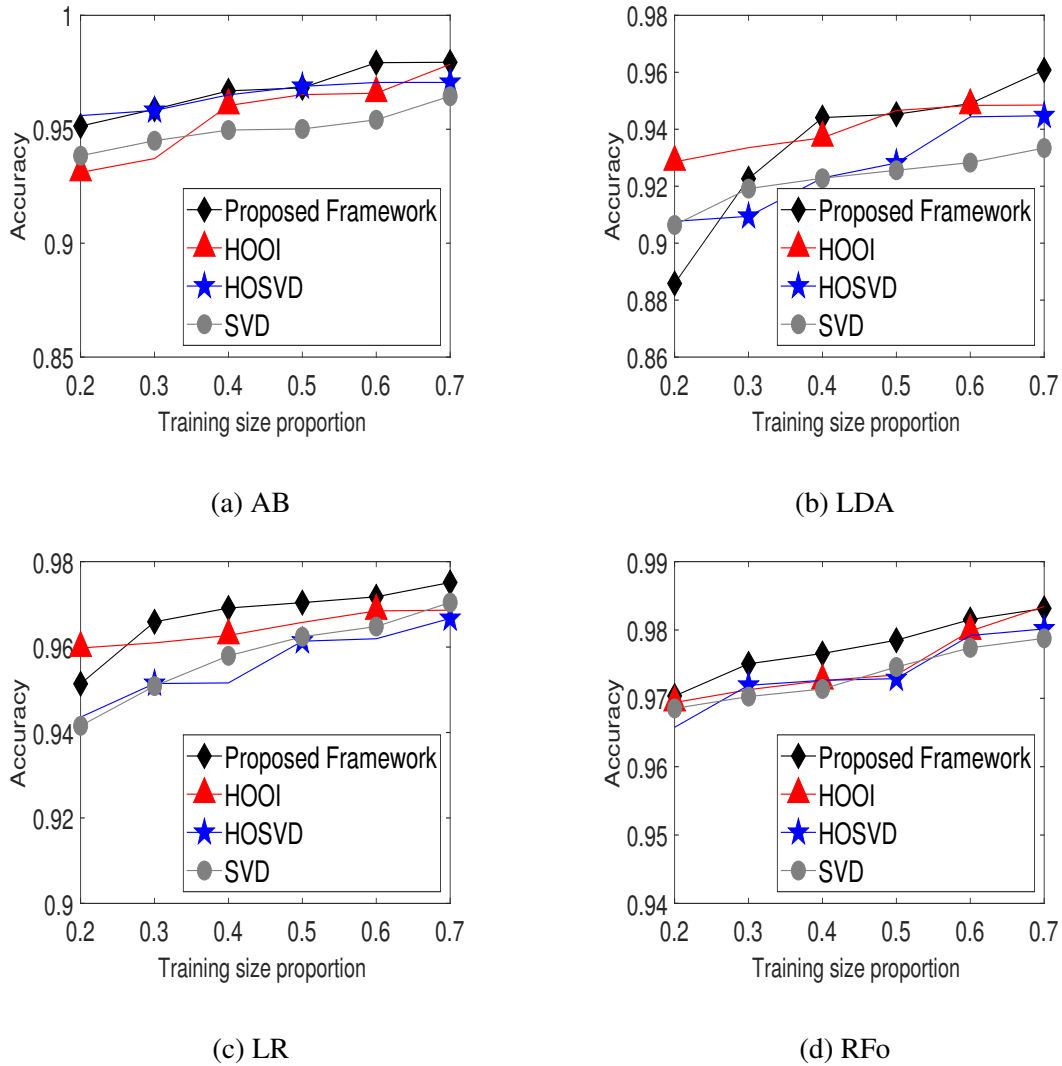
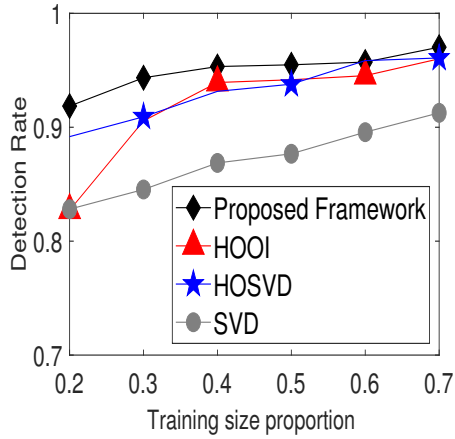


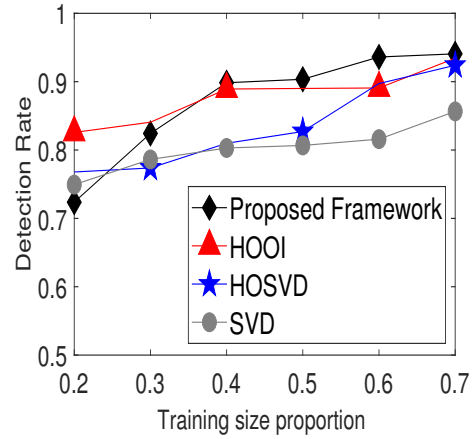
Figure 3.7 – Plots of accuracy, as a function of the training dataset size proportion, considering: (a) AB, (b) LDA, (c) LR, and (d) RFo classification algorithms.

modes of the dataset instances provides a better classification performance compared to the state-of-the-art low-rank approximation techniques.

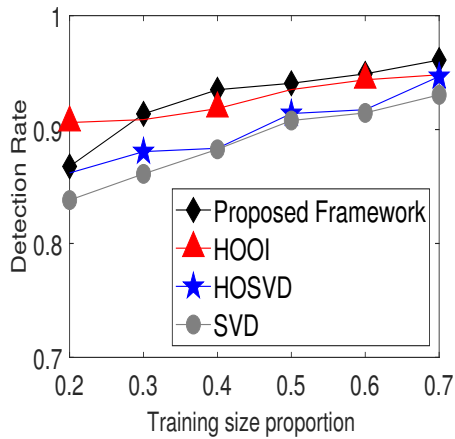
Next, Fig. 3.7 to 3.9 illustrate the values of accuracy, detection rate and false alarm rate as a function of the TSP for the AB, LDA, LR and RFo classification algorithms. Note that all compared techniques present better performance as the training dataset size proportion grows, since ML classifiers need more training samples when the dataset has a large number of features. In this case, the classification model is better fit to the training data after adjusting several parameters such that the errors between the actual and predicted classes are minimal, despite the higher risk of overfitting. Moreover, the difference in performance between SVD and its multidimensional counterparts, HOSVD and HOOI, is more noticeable in terms of detection rate, as shown in Fig. 3.8. Consequently, we conclude that matrix based denoising techniques are more sensitive to the variation of training dataset size. On the



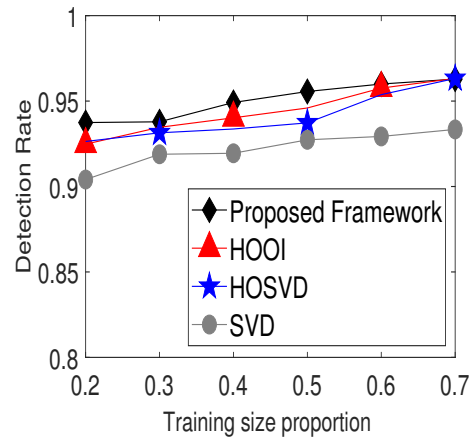
(a) AB



(b) LDA



(c) LR



(d) RFo

Figure 3.8 – Plots of detection rate, as a function of the training dataset size proportion, considering: (a) AB, (b) LDA, (c) LR, and (d) RFo classification algorithms.

other hand, tensor based schemes perform a more efficient noise attenuation through multiple SVDs along the dataset dimensions, which partially compensates the lack of training data in machine learning classifiers. Finally, note that all of the compared techniques are not so efficient in terms of network attack detection because we consider a high noise level scenario, with SNR fixed in 10 dB. Our intent is to simulate an environment where dataset is partially corrupted and, consequently, the performance of the machine learning classification algorithm is degraded. Hence, from the results shown in Fig. 3.7 to 3.9, we conclude that our proposed framework is very robust against the variation of the training dataset size proportion, which once again highlights the benefits of the multidimensional denoising introduced on the dataset by our proposed extended MuDe algorithm.

Following, in Table 3.10 the values of accuracy, detection rate and false alarm rate obtained by the proposed framework and related works were shown. Despite the proposed

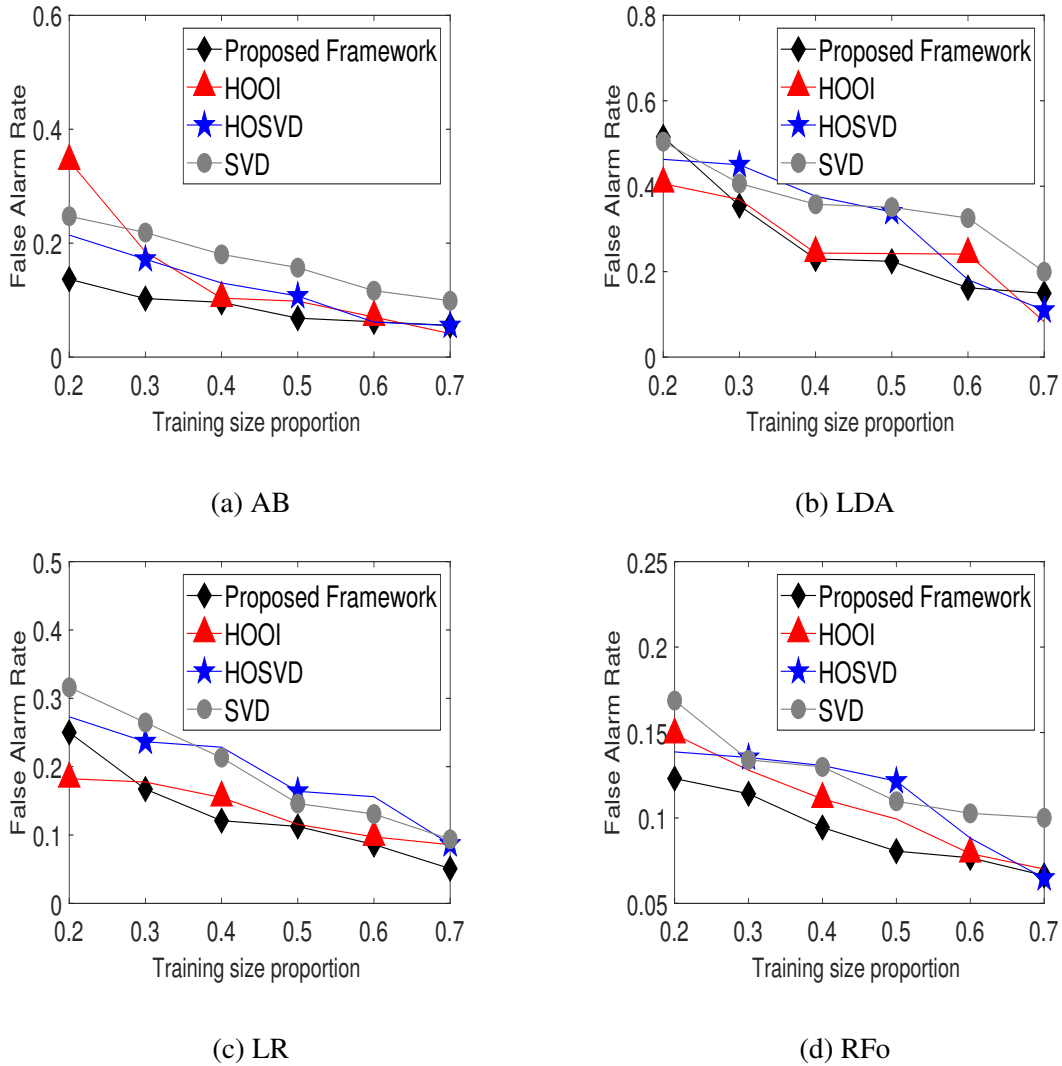


Figure 3.9 – Plots of false alarm rate, as a function of the training dataset size proportion, considering: (a) AB, (b) LDA, (c) LR, and (d) RFo classification algorithms.

approach does not outperform some works in terms of accuracy, detection rate and false alarm rate, it still presents an outstanding DDoS attack detection performance. For instance, considering the NSL-KDD dataset, despite the models of Wang et al. [23] presented a higher Acc compared to the proposed scheme with LDA classifier, our approach is far superior in terms of DR and FAR. The method proposed by Wang et al. reconstructs the detection model by dynamically perceiving the occurrence of detection errors through a feedback mechanism, which can explain its better accuracy. Nonetheless, its best detection rate was 94.88%, whereas this current proposal obtained 99.05% with LR and RFo classifiers. Another example is the work of Gogoi et al. [116], which provided a detailed analysis of the NSL-KDD dataset and proposed two real life network intrusion datasets. Such method outperforms the proposed framework with LDA classifier in terms of detection rate, but our approach shows lower FAR, regardless of the classification algorithm. While the worst FAR presented by our approach was 0.94% with AdaBoost, the model proposed by Gogoi et al. showed a false

Table 3.10 – Performance comparison with related works, considering the NSL-KDD and CIC-DDoS2019 datasets.

Dataset	Work	Classification Method	Acc	DR	FAR
NSL-KDD	Proposed Tensor MuDe Based Framework	AB	0.9859	0.9872	0.0094
	Proposed Tensor MuDe Based Framework	LDA	0.9694	0.9773	0.0004
	Proposed Tensor MuDe Based Framework	LR	0.9873	0.9905	0.0006
	Proposed Tensor MuDe Based Framework	RFo	0.9878	0.9905	0.0018
	Hosseini and Azizi [22]	NB	0.9310	N/A	N/A
	Hosseini and Azizi [22]	DT	0.9820	N/A	N/A
	Hosseini and Azizi [22]	MLP	0.9610	N/A	N/A
	Hosseini and Azizi [22]	kNN	0.9770	N/A	N/A
	Wang et al. [23]	SBS-MLP	0.9766	0.9488	0.0062
	Wang et al. [23]	SFS-MLP	0.9761	0.9471	0.0060
	Wang et al. [23]	CTSBS-MLP	0.9761	0.9478	0.0063
	Kushwah and Ali [101]	MLP	0.9630	0.9791	0.0500
	Gogoi et al. [116]	TUIDS	0.9655	0.9888	0.0112
	Yusof et al. [117]	ELM	0.9170	N/A	N/A
CIC-DDoS2019	Proposed Tensor MuDe Based Framework	AB	0.9817	0.9779	0.0286
	Proposed Tensor MuDe Based Framework	LDA	0.9588	0.9398	0.0916
	Proposed Tensor MuDe Based Framework	LR	0.9826	0.9617	0.0727
	Proposed Tensor MuDe Based Framework	RFo	0.9776	0.9435	0.1129
	Maranhão et al. [2]	MLP	0.9875	0.9746	0.0452
	Maranhão et al. [3]	DT	0.9754	0.9509	N/A
	Shurman et al. [119]	LSTM 1 layer	0.9154	N/A	N/A
	Shurman et al. [119]	LSTM 2 layer	0.9674	N/A	N/A
	Shurman et al. [119]	LSTM 3 layer	0.9919	N/A	N/A
	Sharafaldin et al. [120]	ID3	N/A	0.6500	N/A
	Sharafaldin et al. [120]	RFo	N/A	0.5600	N/A
	Sharafaldin et al. [120]	NB	N/A	0.1100	N/A
	Sharafaldin et al. [120]	LR	N/A	0.0200	N/A
	Hussain et al. [121]	ResNet	N/A	0.8600	N/A
	Aytac et al. [122]	RFo	0.9840	N/A	N/A
	Aytac et al. [122]	DT (Gini)	0.9934	N/A	N/A
	Aytac et al. [122]	DT (Entropy)	0.9912	N/A	N/A
Aytac et al. [122]	Multinomial NB	0.9910	N/A	N/A	
Aytac et al. [122]	Gaussian NB	0.9870	N/A	N/A	

alarm rate of 1.12%.

Finally, in Fig. 3.10 the mean training times as a function of the training size proportion are illustrated for all of the competitor methods. It can be observed that HOOI shows higher training times compared to HOSVD since it is an iterative alternating least squares based method. Additionally, as expected, the SVD scheme presents the lowest training times due to its low time costly matrix based decompositions. Further, note that the proposed framework presents higher training times compared to the HOSVD, HOOI and SVD schemes in all of the TSP range, especially due to the MuDe technique. Thus, according to the results obtained in Fig. 3.4 to 3.9, this is the trade-off in order to achieve a more accurate DDoS attack detection.

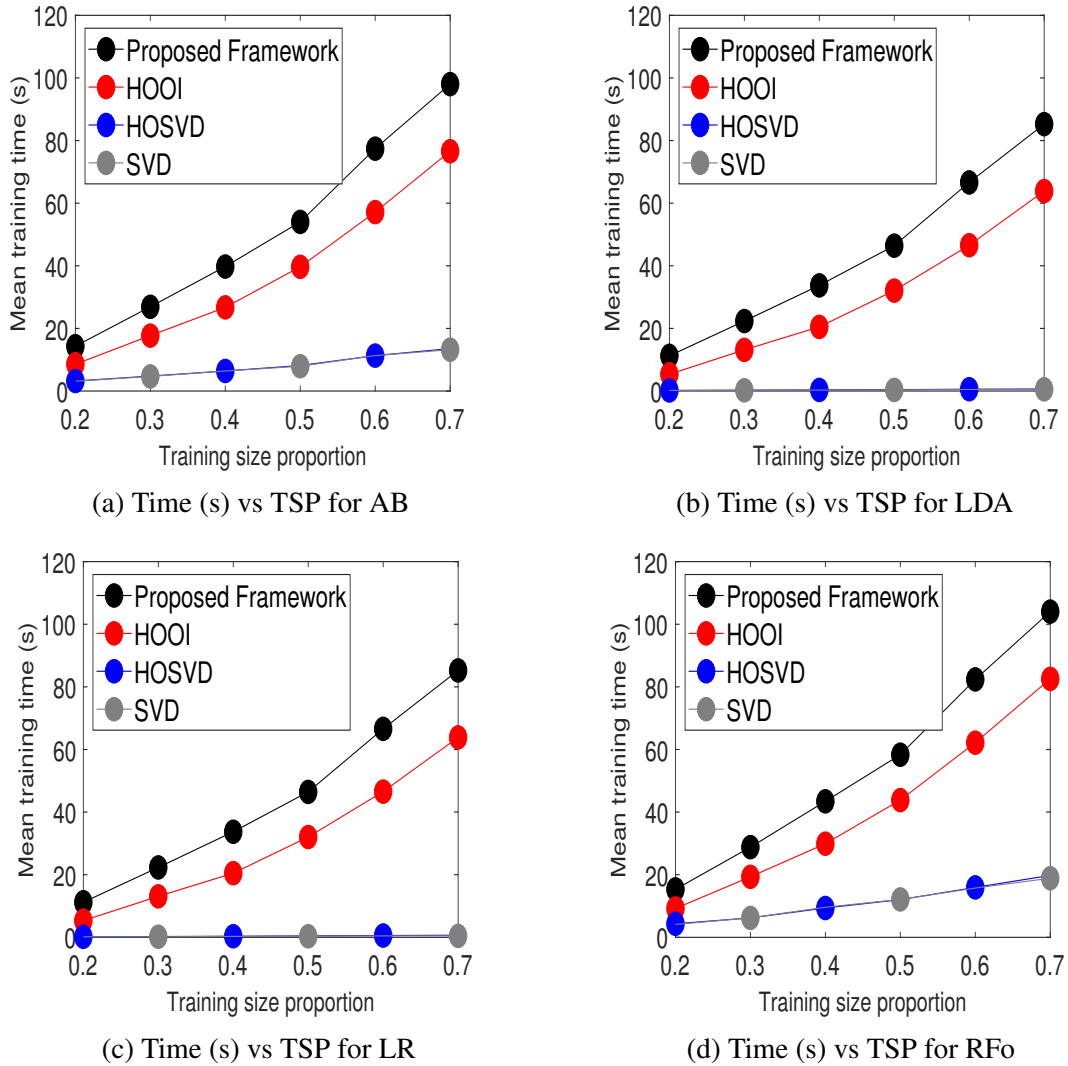


Figure 3.10 – Plots of mean training time (s), as a function of the training dataset size proportion, for AB, LDA, LR and RFo classification algorithms.

3.6 SUMMARY

In this chapter, we propose a novel tensor multiple denoising based framework for DDoS attack detection which exploits both multidimensional signal processing techniques and machine learning based classification algorithms. The proposed framework is composed by four main blocks: data preprocessing, dataset splitting, dataset denoising, and machine learning supervised classification. Particularly considering the third block, an extension of the recent MuDe technique is proposed, which was originally applied for denoising of measurement data collected by multidimensional sensors arrays.

The proposed framework was validated in simulations through comparison with state-of-the-art low-rank approximation techniques, namely, SVD, HOSVD and HOOI, by using the CIC-DDoS2019 benchmark dataset. Nonetheless, since CIC-DDoS2019 is a very recent

dataset, few works applying such dataset for NIDS validation have been found in the literature. Consequently, the well-known NSL-KDD dataset was also applied to validate the proposed technique. According to the obtained results, our proposed scheme considerably outperforms the competing techniques, showing outstanding values of accuracy, detection rate and false alarm rate. Therefore, it can be concluded that the proposed extended MuDe technique provides a considerable performance gain in terms of DDoS attack detection efficiency.

4 NOISE-ROBUST MULTILAYER PERCEPTRON ARCHITECTURE FOR DISTRIBUTED DENIAL OF SERVICE ATTACK DETECTION

In this chapter, the following research question is addressed: *Following the same idea of the previous question, how to efficiently detect DDoS attacks in a CPS network, still applying multidimensional signal processing techniques on noisy datasets, but now using deep learning algorithms instead of ML techniques?*

The remainder of this chapter is organized as follows:

- **Motivation:** in this section, an introduction to DDoS attack detection models based on multilayer perceptron based NIDSs is presented.
- **Proposed Noise-Robust MLP Architecture for DDoS Attack Detection:** this section introduces and discusses the proposed noise-robust MLP architecture for DDoS attack detection.
- **Computational Complexity:** the computational complexity of the proposed MLP is presented and discussed.
- **Simulation Results:** the performance of the proposed MLP is evaluated through numerical simulations.

The research contributions of Chapter 4 are summarized as follows:

1. A novel feature extraction method applied on data classification is proposed such that the average value of the common features among dataset instances is iteratively filtered out via HOSVD algorithm, providing to the NIDS more robustness against data corruption.
2. A noise-robust MLP architecture for DDoS attack detection which applies the technique cited in the previous item is also introduced. The best parameters used for dataset filtering are dynamically computed in order to minimize the errors between the expected and predicted classifications.

3. The performance of the proposed MLP is validated through numerical simulations by using samples extracted from the CIC-DDoS2019, CIC-IDS2018 and CIC-IDS2017 benchmark datasets. According to the simulation results, the proposed scheme outperforms state-of-the-art low-rank approximation techniques in terms of accuracy, detection rate and false alarm rate.

4.1 MOTIVATION

As pointed out in Chapter 3, Distributed Denial of Service (DDoS) attacks are one of the most challenging threats to network security, especially after the emergence of new computing paradigms such as Cloud Computing and Internet of Things, presenting very sophisticated and damaging attacks [123]. Typically DDoS attacks are launched from several compromised machines known as “zombies”, which compose remotely controlled networks called botnets to attack a specific victim. As a consequence of DDoS attacks, the target resources are exhausted, affecting servers, network devices, computer systems and web applications, and preventing legitimate accesses [123, 124].

DDoS attacks can be efficiently detected by using supervised machine learning (ML) techniques trained with large datasets such that malicious patterns present in the incoming network traffic can be detected with high reliability. Nonetheless, the classifier performance can be severely degraded if corrupted datasets are considered during the training and testing phases and, consequently, it is fundamental to develop DDoS attack detection models robust against noise present in data. Similarly to Chapter 3, uncalibrated measures during the process of dataset creation or false data injection attacks performed on publicly available datasets are considered as the main causes of data corruption in this chapter.

Deep learning based schemes have been successfully used for DDoS attack detection, achieving great success in recent years. A lightweight deep learning based approach for DDoS attack detection in online resource-constrained environments was presented in [125]. Furthermore, Roopak et al. [126] proposed several deep learning models for DDoS attack detection in IoT networks, including MLP, CNN and LSTM based classifiers, which outperformed state-of-the-art ML algorithms. Additionally, Haider et al. [127] proposed a deep CNN ensemble framework for efficient DDoS attack detection in Software Defined Networks (SDNs), presenting higher training and testing times, which was compensated by a satisfactory performance detection. Moreover, a multi-channel CNN based detection and early warning framework for DDoS attacks was proposed by Chen et al. [128], whose results were based on real-time traffic and network package information. Finally, an intrusion detection system against DDoS attacks in SDN environments was proposed in [118], in which Recurrent Neural Networks are combined with autoencoders.

In addition, multilayer perceptron based methods for DDoS attack detection are also very popular nowadays. Saied et al. [129] proposed a framework where specific features are used to separate DDoS attacks from legitimate traffic by using an artificial neural network algorithm. Furthermore, Singh et al. [130] presented a novel technique where application-layer DDoS attacks can be detected by computing the entropy of HTTP GET request count per connection, the variance of the entropy per Internet Protocol (IP) address, and the number of HTTP GET request counts. Besides, a model to detect DDoS attacks in cloud computing based on artificial neural networks and black hole optimization algorithm was proposed by Kushwah and Ali [101]. Moreover, Wang et al. [23] proposed a neural network based scheme to detect DDoS attacks where the optimal features are selected during the training phase and a feedback mechanism dynamically updates the detector according to detection errors.

Nonetheless, since the aforementioned proposed models are matrix based solutions and do not consider corrupted training data, we fill those gaps by exploiting the benefits of multidimensional datasets as well as by applying noise-robust techniques. The main approaches and drawbacks of each related work, as well as the weaknesses addressed by this chapter, are summarized in Table 4.1.

Table 4.1 – Related works summary.

Ref.	Main Approach	Drawback
[23]	- Feature selection based MLP for DDoS attack detection with dynamic perceiving of detection errors.	- Benefits of tensor representation are not exploited. - Lack of analysis of the training and testing times.
[101]	- Artificial neural network based model for DDoS attack detection in cloud computing.	- Benefits of tensor representation are not exploited. - Outdated dataset (NSL-KDD) applied for model validation. - Lack of analysis of the training and testing times. - Lack of important performance evaluation metrics.
[118]	- IDS for detecting DDoS attacks in SDN environments.	- Benefits of tensor representation are not exploited. - Lack of analysis of the training and testing times. - Lack of important performance evaluation metrics.
[125]	- Lightweight DL based approach for DDoS attack detection in online resource-constrained environments.	- Benefits of tensor representation are not exploited. - Lack of analysis of the testing times.
[126]	- DL based models for DDoS attack detection in IoT networks.	- Benefits of tensor representation are not exploited. - Lack of analysis of the training and testing times.
[127]	- Deep CNN ensemble framework for efficient DDoS attack detection in SDNs.	- Benefits of tensor representation are not exploited. - Higher training and testing times.
[128]	- Multi-channel CNN based detection and early warning framework for DDoS attacks.	- Benefits of tensor representation are not exploited. - Outdated dataset (KDDCUP99) applied for model validation. - Focus only on the overall accuracy.
[129]	- Artificial neural network based scheme for detecting DDoS attacks by using specific data features.	- Benefits of tensor representation are not exploited. - Lack of analysis of the training and testing times.
[130]	- Application-layer DDoS attacks detected by computing entropy-related parameters.	- Benefits of tensor representation are not exploited.
This chapter	- Tensor DL based framework for DDoS attack detection trained with poisoned datasets.	Addressed drawbacks: - Benefits of tensor representation are exploited. - Recent dataset (CIC-DDoS2019) applied for model validation. - Analysis of the training and testing times.

In this chapter, we propose a novel feature extraction method applied on data classification in which the average value of the common features among dataset instances is iteratively filtered out via HOSVD algorithm, providing more robustness against data corruption. Addi-

tionally, a novel multilayer perceptron (MLP) architecture for DDoS attack detection which applies such feature extraction technique is also introduced. Extensive experiments conducted on a customized dataset containing samples extracted from the CIC-IDS2017 [131], CIC-IDS2018 [132] and CIC-DDoS2019 [89] benchmark datasets showed that the proposed scheme outperforms state-of-the-art low-rank approximation based MLPs in terms of accuracy, detection rate and false alarm rate.

4.2 PROPOSED NOISE-ROBUST MLP ARCHITECTURE FOR DDoS ATTACK DETECTION

This section is divided into two subsections. First, Subsection 4.2.1 describes the proposed feature extraction technique applied on data classification. Next, in Subsection 4.2.2 the proposed multilayer perceptron architecture for DDoS attack detection is introduced.

In this chapter, the proposed scheme is developed under two main assumptions:

- Similarly to Chapter 3, datasets are considered to be noise free because assumptions about their base noise type and level cannot be made. In this sense, zero-mean white Gaussian noise is added into the dataset features in a supervised manner such that the DDoS attack detection performance of the proposed architecture can be assessed for different noise levels.
- The Higher Order Singular Value Decomposition (HOSVD) technique is adopted in our proposed scheme due to its lower processing time compared to state-of-the-art tensor decomposition approaches, such as the Higher Order Orthogonal Iteration (HOOI). This is a crucial requirement in real time network attack detection systems.

4.2.1 Proposed Feature Extraction Method Applied on Data Classification

Throughout this chapter, the dataset matrix $\mathbf{X} \in \mathbb{R}^{M \times N}$ and its tensor counterpart $\mathcal{X} \in \mathbb{R}^{N_1 \times \dots \times N_R \times M}$ are defined by the data model shown in Section 3.2 of Chapter 3, particularly in Eq. (3.1) and (3.2), respectively.

First, the concept of common and individual features of a given dataset is introduced. Such concept is well-known in image classification problems, in which data share some common variables while exhibiting their own features simultaneously [133]. Fig. 4.1 illustrates the intuition behind the idea of common features. The original tensor $\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times 5}$, at the left of Fig. 4.1, is composed of the slices $\mathcal{Y}(:, :, s)$ for $s = 1, \dots, 5$ with the bright blue, bright green, gray, orange and pink colors, respectively. Each slice $\mathcal{Y}(:, :, s)$ is equivalent to a combination of the three base colors, namely, green, red and blue, represented by the ma-

trices $\mathbf{B}_G \in \mathbb{R}^{I_1 \times I_2}$, $\mathbf{B}_R \in \mathbb{R}^{I_1 \times I_2}$ and $\mathbf{B}_B \in \mathbb{R}^{I_1 \times I_2}$ shown at the center of Fig. 4.1. Usually, the base colors are obtained through tensor decompositions, such as the LL1 decomposition with non-negativity constraint, such that $\mathbf{y} = (\mathbf{B}_G \times_3 \mathbf{c}_G) + (\mathbf{B}_R \times_3 \mathbf{c}_R) + (\mathbf{B}_B \times_3 \mathbf{c}_B)$, where $\mathbf{c}_G \in \mathbb{R}^5$, $\mathbf{c}_R \in \mathbb{R}^5$ and $\mathbf{c}_B \in \mathbb{R}^5$ contain the intensity values of the red, green and blue colors, respectively [134]. Moreover, note that the original tensor presents rank three, which corresponds to the number of base colors. Alternatively, the base colors can be stacked along the 3rd dimension, generating $\mathbf{B} \in \mathbb{R}^{I_1 \times I_2 \times 3}$, whereas the vectors \mathbf{c}_G , \mathbf{c}_R and \mathbf{c}_B can be grouped into the matrix $\mathbf{C} \in \mathbb{R}^{5 \times 3}$, as depicted at the right of Fig. 4.1. The tensor \mathbf{B} , known as the common feature tensor, can also be represented as $\tilde{\mathbf{y}}$, as a reference to the original dataset \mathbf{y} .

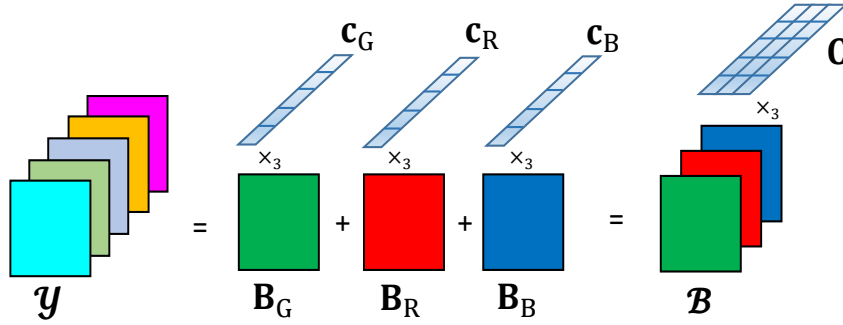


Figure 4.1 – The concept of common features for multidimensional data. The original tensor $\mathbf{y} \in \mathbb{R}^{I_1 \times I_2 \times 5}$ consists of slices with the bright blue, bright green, gray, orange and pink colors. Each slice $\mathbf{y}(:, :, s) \in \mathbb{R}^{I_1 \times I_2}$ for $s = 1, \dots, 5$ corresponds to a combination of the base colors green, red and blue, given by $\mathbf{B}_G \in \mathbb{R}^{I_1 \times I_2}$, $\mathbf{B}_R \in \mathbb{R}^{I_1 \times I_2}$ and $\mathbf{B}_B \in \mathbb{R}^{I_1 \times I_2}$, respectively. Such base colors can be stacked along the 3rd dimension, generating the common feature tensor $\mathbf{B} \in \mathbb{R}^{I_1 \times I_2 \times 3}$.

Applying such concepts to the DDoS attack detection problem, the following intuition can be drawn. Let us consider a 3D network traffic dataset composed by legitimate and DDoS attack sample matrices as its frontal slices. Although from different classes, these samples may present some common features, such as source and destination IPs. Therefore, the common features across all the samples can be removed and then the more discriminative individual features can be applied for classification. The tensors $\hat{\mathbf{X}} \in \mathbb{R}^{N_1 \times \dots \times N_R \times d_{R+1}}$ and $\check{\mathbf{X}} \in \mathbb{R}^{N_1 \times \dots \times N_R \times M}$ are called the common and individual feature tensors, respectively. The tensor $\hat{\mathbf{X}}$ can be obtained after applying some tensor decomposition technique on \mathbf{X} , such as the Higher Order Singular Value Decomposition (HOSVD), which is a generalization of the matrix SVD to tensors.

The HOSVD of \mathbf{X} can be expressed as $\mathbf{X} = \mathcal{G} \times_1 \mathbf{A}_1 \cdots \times_R \mathbf{A}_R \times_{R+1} \mathbf{A}_{R+1}$, where $\mathcal{G} \in \mathbb{R}^{d_1 \times \dots \times d_{R+1}}$ is the truncated core tensor, $\mathbf{A}_r \in \mathbb{R}^{N_r \times d_r}$ for $r = 1, \dots, R+1$ are the truncated singular matrices and (d_1, \dots, d_{R+1}) is the multilinear rank of \mathbf{X} . The number of common features among dataset instances, given by d_{R+1} , can be obtained empirically. In addition, $\hat{\mathbf{X}} \in \mathbb{R}^{N_1 \times \dots \times N_R \times d_{R+1}}$ is defined as the r -mode product between the core tensor \mathcal{G} and the first R factor matrices, i.e., $\hat{\mathbf{X}} = \mathcal{G} \times_1 \mathbf{A}_1 \cdots \times_R \mathbf{A}_R$. Then, the individual

feature tensor $\check{\mathbf{X}}$ is computed by subtracting, from \mathbf{X} , the weighted common features, i.e., $\check{\mathbf{X}}(:, \dots, m) = \mathbf{X}(:, \dots, m) - \sum_{k \in K_n} \alpha_k \cdot \hat{\mathbf{X}}(:, \dots, k)$ for $m = 1, \dots, M$, where K_n is a subset of common features for \mathbf{X} related to the values in the n -th row of \mathbf{A}_r [134].

The proposed approach improves the DDoS attack detection performance due to two factors: noise-robustness and dataset filtering. Noise-robustness is achieved through the HOSVD of the dataset tensor. In HOSVD, the singular value decomposition (SVD) is applied to each r -th unfolding matrix $[\mathbf{X}]_{(r)} = [\mathbf{X}_0]_{(r)} + [\mathbf{N}]_{(r)}$ and is given by $[\mathbf{X}]_{(r)} = \mathbf{U}_r \Sigma_r \mathbf{V}_r^H$, where $\mathbf{U}_r \in \mathbb{R}^{N_r \times d_r}$ and $\mathbf{V}_r \in \mathbb{R}^{\prod_{j \neq r} N_j M \times d_r}$ are the left and right singular matrices, respectively, and $\Sigma_r \in \mathbb{R}^{d_r \times d_r}$ is the singular values matrix. On the other hand, after dataset filtering, ML algorithms can exploit the filtered features in order to identify each dataset instance. In this approach, instead of obtained empirically, the number of common features d_{R+1} can be approximated as the model order, which is a parameter computed from classic model order selection (MOS) techniques. Thus, a more discriminative information is present in each sample, which can be exploited by ML classifiers during the training phase.

Initially, three steps are necessary, namely, dataset splitting, dataset pre-processing and minibatch splitting. First, the dataset \mathbf{X} is split into the training and testing tensors $\mathbf{X}^{\text{tr}} \in \mathbb{R}^{N_1 \times \dots \times N_R \times M^{\text{tr}}}$ and $\mathbf{X}^{\text{te}} \in \mathbb{R}^{N_1 \times \dots \times N_R \times M^{\text{te}}}$, where M^{tr} and M^{te} correspond to the total number of training and testing instances, respectively, with $M = M^{\text{tr}} + M^{\text{te}}$. Next, a preprocessing step is applied on each tensor as well, including data cleansing, feature scaling and label encoding. Finally, \mathbf{X}^{tr} is split into S minibatches $\mathbf{X}^s \in \mathbb{R}^{N_1 \times \dots \times N_R \times M^{\text{mb}}}$ for $s = 1, \dots, S$ containing M^{mb} instances each, i.e., $M^{\text{tr}} = S \cdot M^{\text{mb}}$. If M^{tr} is not a multiple of M^{mb} , then random instances from \mathbf{X}^{tr} are added into the last minibatch such that the condition $M^{\text{tr}} = S \cdot M^{\text{mb}}$ is satisfied.

Fig. 4.2 depicts the block diagram of the proposed feature extraction method applied on the training and testing phases of the ML classification, which are described in the following subsections.

4.2.1.1 Training Phase

First, given $\mathbf{X}^s \in \mathbb{R}^{N_1 \times \dots \times N_R \times M^{\text{mb}}}$, the model orders d_r^s for $r = 1, \dots, R + 1$ and $s = 1, \dots, S$ are estimated in Block 1.1 of Fig. 4.2a through multidimensional MOS schemes, such as the R -D Minimum Description Length [107]. Further, Block 1.2 computes the HOSVD of \mathbf{X}^s as follows

$$\mathbf{X}^s = \mathbf{G}^s \times_1 \mathbf{A}_1^s \cdots \times_R \mathbf{A}_R^s \times_{R+1} \mathbf{A}_{R+1}^s, \quad (4.1)$$

where $\mathbf{G}^s \in \mathbb{R}^{d_1^s \times \dots \times d_{R+1}^s}$ is the core tensor, $\mathbf{A}_r^s \in \mathbb{R}^{N_r \times d_r^s}$ for $r = 1, \dots, R+1$ are the singular matrices, and $(d_1^s, \dots, d_{R+1}^s)$ correspond to the multilinear rank of \mathbf{X}^s , i.e., the r -tuple of the

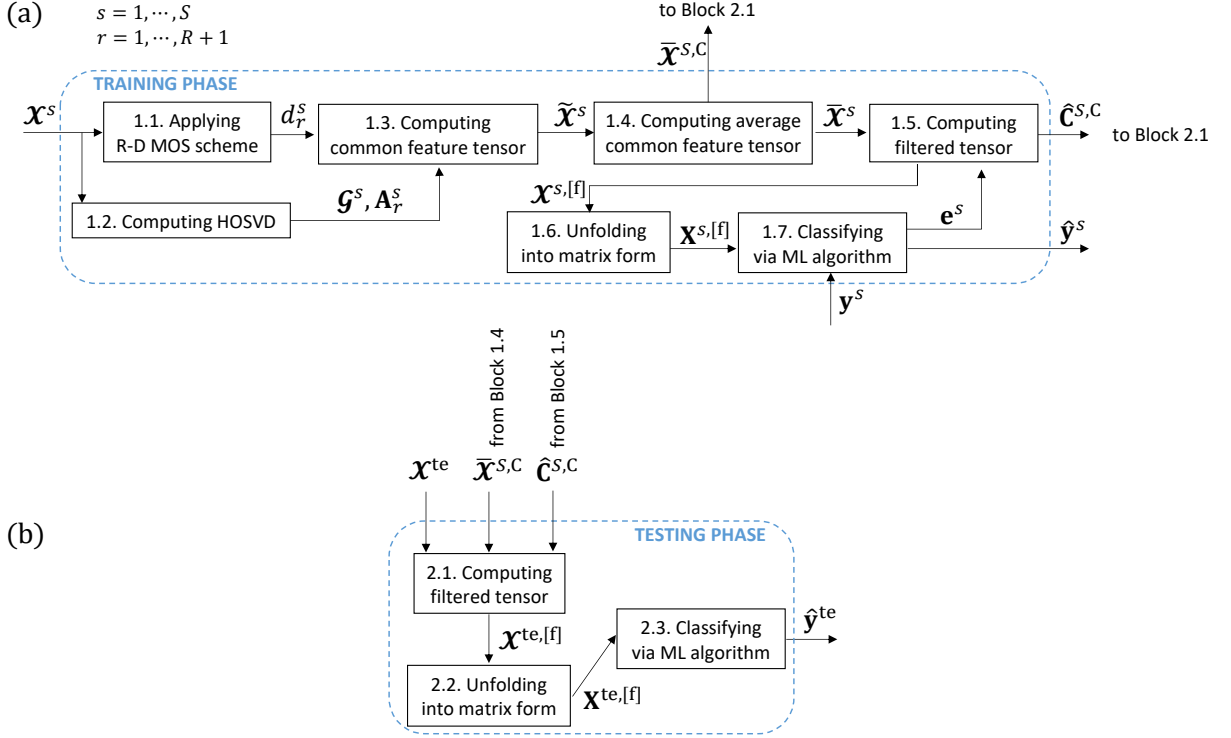


Figure 4.2 – Block diagram of the proposed feature extraction method applied on ML classification. (a) Training phase. (b) Testing phase.

mode- r ranks. Moreover, d_{R+1}^s is interpreted as the number of common features among the dataset instances.

Following, Block 1.3 computes $\tilde{\mathbf{X}}^s \in \mathbb{R}^{N_1 \times \dots \times N_R \times d_{R+1}^s}$, which contains the common features among the instances $\mathbf{X}^s(:, \dots, m) \in \mathbb{R}^{N_1 \times \dots \times N_R}$ for $m = 1, \dots, M^{\text{mb}}$. The tensor $\tilde{\mathbf{X}}^s$ is defined as the r -mode product between the core tensor \mathcal{G}^s and the first R factor matrices [135],

$$\tilde{\mathbf{X}}^s = \mathcal{G}^s \times_1 \mathbf{A}_1^s \times_2 \mathbf{A}_2^s \cdots \times_R \mathbf{A}_R^s. \quad (4.2)$$

Next, $\bar{\mathbf{X}}^s \in \mathbb{R}^{N_1 \times \dots \times N_R}$ is obtained in Block 1.4 of Fig. 4.2a and corresponds to $\tilde{\mathbf{X}}^s$ averaged along the $(R + 1)$ -th dimension, i.e.,

$$\bar{\mathbf{X}}^s = \frac{1}{d_{R+1}^s} \sum_{d=1}^{d_{R+1}^s} \tilde{\mathbf{X}}^s(:, \dots, d). \quad (4.3)$$

Before subtracting the average common features from \mathbf{X}^s in Block 1.5, we have to multiply each one of the elements of $\bar{\mathbf{X}}^s$ by a positive number smaller than 1. This can be done by computing the Hadamard product between $\bar{\mathbf{X}}^s$ and a weight tensor $\mathbf{C}^s \in \mathbb{R}^{N_1 \times \dots \times N_R}$. The tensor \mathbf{C}^s is obtained such that the errors between the expected and predicted classifications during the training phase are minimized. Thus, M^{mb} copies of \mathbf{C}^s are concatenated along the

$(R + 1)$ -th dimension, generating the tensor $\mathbf{C}^{s,C} \in \mathbb{R}^{N_1 \times \dots \times N_R \times M^{\text{mb}}}$. The same procedure is adopted for $\bar{\mathbf{X}}^s$ in order to obtain $\bar{\mathbf{X}}^{s,C} \in \mathbb{R}^{N_1 \times \dots \times N_R \times M^{\text{mb}}}$. Both computations can be expressed as

$$\bar{\mathbf{X}}^{s,C} = [\bar{\mathbf{X}}^s \mid \dots \mid \bar{\mathbf{X}}^s]_{R+1}. \quad (4.4)$$

$$\mathbf{C}^{s,C} = [\mathbf{C}^s \mid \dots \mid \mathbf{C}^s]_{R+1}, \quad (4.5)$$

Thus, we compute the Hadamard product between $\mathbf{C}^{s,C}$ and $\bar{\mathbf{X}}^{s,C}$ such that the weights are applied to each element of the average common feature tensor, i.e.,

$$\mathbf{W}^s = \mathbf{C}^{s,C} \odot \bar{\mathbf{X}}^{s,C}. \quad (4.6)$$

Following, the filtered tensor $\mathbf{X}^{s,[f]}$ computed in Block 1.5 is given by

$$\mathbf{X}^{s,[f]} = \mathbf{X}^s - \mathbf{W}^s. \quad (4.7)$$

Alternatively, the filtered tensor $\mathbf{X}^{s,[f]}(:, \dots, m)$ for $m = 1, \dots, M^{\text{mb}}$ can be computed as a function of the m -th dataset instance as follows

$$\mathbf{X}^{s,[f]}(:, \dots, m) = \mathbf{X}^s(:, \dots, m) - \mathbf{C}^s \odot \bar{\mathbf{X}}^s, \quad \text{for } m = 1, \dots, M^{\text{mb}}. \quad (4.8)$$

Furthermore, in Block 1.6 of Fig. 4.2a we compute the $(R + 1)$ -th mode unfolding matrix of $\mathbf{X}^{s,[f]}$, given by $[\mathbf{X}^{s,[f]}]_{(R+1)} \in \mathbb{R}^{M^{\text{mb}} \times N}$. In general, the r -th unfolding matrix $[\mathbf{X}^{s,[f]}]_r$ is obtained after each element $(x_1^{s,[f]}, \dots, x_{R+1}^{s,[f]})$ in $\mathbf{X}^{s,[f]}$ is mapped to the element $(x_r^{s,[f]}, j)$ in $[\mathbf{X}^{s,[f]}]_r$ as follows

$$j = 1 + \sum_{\substack{k=1 \\ k \neq r}}^{R+1} (x_k^{s,[f]} - 1) J_k, \quad \text{with } J_k = \prod_{\substack{m=1 \\ m \neq r}}^{k-1} N_m. \quad (4.9)$$

Next, $[\mathbf{X}^{s,[f]}]_{(R+1)}$ is sent to Block 1.7 for machine learning classification. In summary, if $\text{Cl}(\cdot)$ represents a trained ML classification algorithm, the vector $\hat{\mathbf{y}}^s \in \mathbb{R}^{M^{\text{mb}}}$ containing the predicted class labels \hat{y}_m^s of the m -th dataset instance $\mathbf{X}^{s,[f]}(m, :)$ for $m = 1, \dots, M^{\text{mb}}$ is given by

$$\hat{\mathbf{y}}^s = \text{Cl}(\mathbf{X}^{s,[f]}) = [\hat{y}_1^s, \hat{y}_2^s, \dots, \hat{y}_{M^{\text{mb}}}^s]^\top. \quad (4.10)$$

At last, the error e^s between the expected and predicted class label vectors \mathbf{y}^s and $\hat{\mathbf{y}}^s$, respectively, is computed by using the binary cross-entropy cost function, given by

$$e^s = -\frac{1}{M^{\text{h}}} \sum_{m=1}^{M^{\text{mb}}} [(y_m^s \log(P(\hat{y}_m^s))) + (1 - y_m^s) \log(1 - P(\hat{y}_m^s))], \quad (4.11)$$

where y_m^s and \hat{y}_m^s are the true and predicted classes of the m -th instance in \mathbf{y}^s and $\hat{\mathbf{y}}^s$, respectively. In sequel, e^s is sent to Block 1.5, where \mathbf{C}^s is updated such that the error is minimized for the next minibatch training.

The training phase of the proposed average common feature extraction technique applied on ML classification is summarized in Algorithm 3 and depicted in Fig. 4.3 for a three-dimensional dataset tensor $\mathcal{X}^s \in \mathbb{R}^{N_1 \times N_2 \times M}$.

Algorithm 3: Proposed feature extraction technique - training phase

Input:

- Training minibatch tensor $\mathcal{X}^s \in \mathbb{R}^{N_1 \times \dots \times N_R \times M^{\text{mb}}}$
- Expected class label vector $\mathbf{y}^s \in \mathbb{R}^{M^{\text{mb}}}$

Output:

- Predicted training class label vector $\hat{\mathbf{y}}^s \in \mathbb{R}^{M^{\text{mb}}}$

Algorithm Steps:

- 1 Estimate the model order d_r^s for $r = 1, \dots, R$ by applying an R -D model order selection scheme on \mathcal{X}^s
 - 2 Compute the HOSVD of $\mathcal{X}^s \in \mathbb{R}^{N_1 \times \dots \times N_R \times M^{\text{mb}}}$ as in (4.1)
 - 3 Compute the common feature tensor $\tilde{\mathcal{X}}^s \in \mathbb{R}^{N_1 \times \dots \times N_R \times d_{R+1}^s}$ as in (4.2)
 - 4 Compute the average common feature tensor $\bar{\mathcal{X}}^s \in \mathbb{R}^{N_1 \times \dots \times N_R}$ as in (4.3)
 - 5 Compute the concatenated dataset tensor $\tilde{\mathcal{X}}^{s,C} \in \mathbb{R}^{N_1 \times \dots \times N_R \times M^{\text{mb}}}$ as in (4.4)
 - 6 Compute the concatenated weight tensor $\mathbf{C}^{s,C} \in \mathbb{R}^{N_1 \times \dots \times N_R \times M^{\text{mb}}}$ as in (4.5)
 - 7 Compute the Hadamard product between $\mathbf{C}^{s,C}$ and $\tilde{\mathcal{X}}^{s,C}$ as in (4.6)
 - 8 Compute the filtered dataset tensor $\mathcal{X}^{s,[f]} \in \mathbb{R}^{N_1 \times \dots \times N_R \times M^{\text{mb}}}$ as in (4.7)
 - 9 Unfold $\mathcal{X}^{s,[f]}$ into the matrix form $[\mathcal{X}^{s,[f]}]_{(R+1)} \in \mathbb{R}^{M^{\text{mb}} \times N}$ as in (4.9)
 - 10 Compute the predicted training class label vector $\hat{\mathbf{y}}^s \in \mathbb{R}^{M^{\text{mb}}}$ as in (4.10)
 - 11 Compute the error e^s between $\mathbf{y}^s \in \mathbb{R}^{M^{\text{mb}}}$ and $\hat{\mathbf{y}}^s \in \mathbb{R}^{M^{\text{mb}}}$ as in (4.11)
 - 12 Update the weight tensor \mathbf{C}^s such that the error is minimized for the next minibatch training
-

4.2.1.2 Testing Phase

The testing phase is composed by three steps, as depicted in Boxes 2.1 to 2.3 of Fig. 4.2b. Instead of applying a scheme similar to the one described for the training set, we simply apply a transformation to each testing instance by using information extracted from the training phase. Since testing and training instances are extracted from the same dataset, they have similarities and, consequently, we consider that the testing dataset presents the same average common features of the training dataset.

First, the filtered testing dataset $\mathcal{X}^{\text{te},[f]}$ is computed in Box 2.1 by subtracting, from \mathcal{X}^{te} , the weighted common feature tensor obtained from the training phase, i.e.,

$$\mathcal{X}^{\text{te},[f]} = \mathcal{X}^{\text{te}} - \hat{\mathbf{C}}^{S,C} \odot \bar{\mathcal{X}}^{S,C}, \quad (4.12)$$

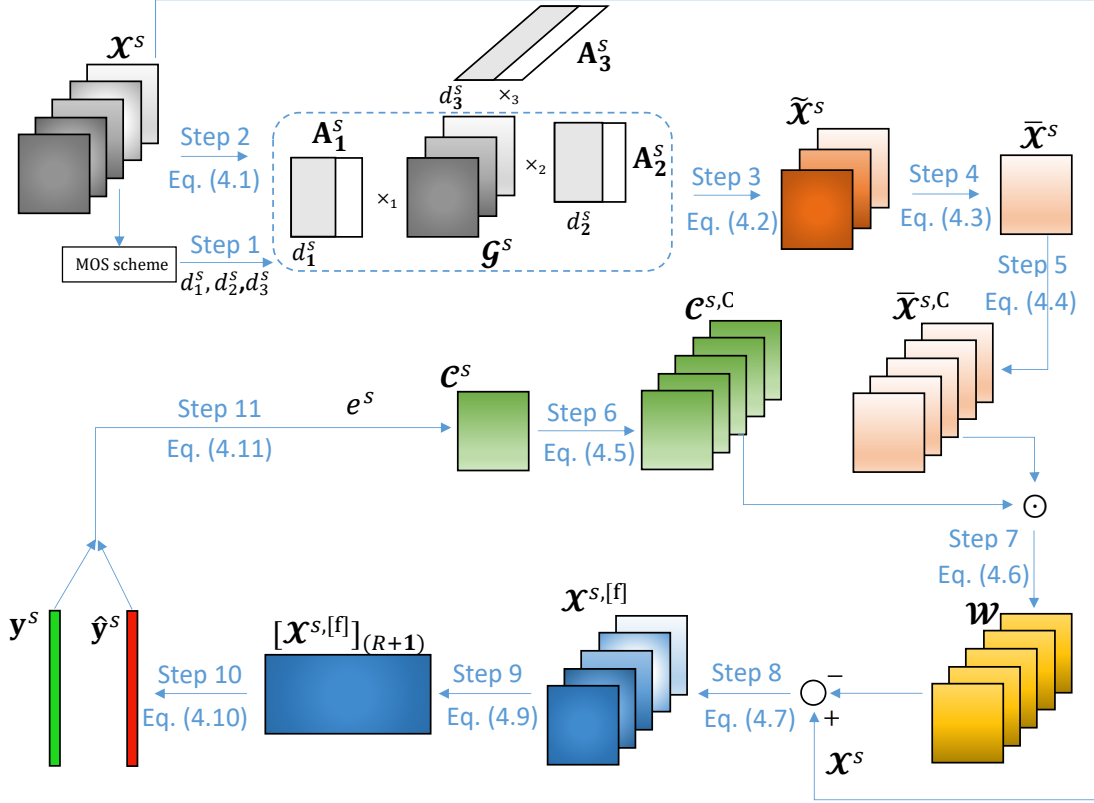


Figure 4.3 – Block diagram of the training phase of the proposed feature extraction technique considering a three-dimensional dataset tensor $\mathbf{X}^s \in \mathbb{R}^{N_1 \times N_2 \times M}$. The diagram presents references to steps of Algorithm 3 as well as equations of Subsection 4.2.1.1.

where $\hat{\mathbf{e}}^{s,C}$ and $\bar{\mathbf{X}}^{s,C}$ are, respectively, the weight tensor and the average common feature tensor, both obtained after the training of the S -th minibatch. Next, $\mathbf{X}^{\text{te},[f]}$ is converted to the $(R + 1)$ -th mode unfolding matrix $[\mathbf{X}^{\text{te},[f]}]_{(R+1)} \in \mathbb{R}^{M^{\text{te}} \times N}$ in Box 2.2 of Fig. 4.2b. The r -th unfolding matrix $[\mathbf{X}^{\text{te},[f]}]_r$ is obtained after each element $(x_1^{\text{te},[f]}, \dots, x_{R+1}^{\text{te},[f]})$ in $\mathbf{X}^{\text{te},[f]}$ is mapped to the element $(x_r^{\text{te},[f]}, j)$ in $[\mathbf{X}^{\text{te},[f]}]_r$ as follows

$$j = 1 + \sum_{\substack{k=1 \\ k \neq r}}^{R+1} (x_k^{\text{te},[f]} - 1) J_k, \quad \text{with } J_k = \prod_{\substack{m=1 \\ m \neq r}}^{k-1} N_m. \quad (4.13)$$

Finally, $[\mathbf{X}^{\text{te},[f]}]_{(R+1)}$ is applied on the trained machine learning classifier $\text{Cl}(\cdot)$ in Box 2.3 for testing purposes. The vector $\hat{\mathbf{y}}^{\text{te}}$ containing the predicted class labels \hat{y}_m^{te} of the m -th testing dataset instance $\mathbf{X}^{\text{te},[f]}(m, :)$ for $m = 1, \dots, M^{\text{te}}$ is given by

$$\hat{\mathbf{y}}^{\text{te}} = \text{Cl}(\mathbf{X}^{\text{te},[f]}) = [\hat{y}_1^{\text{te}}, \hat{y}_2^{\text{te}}, \dots, \hat{y}_{M^{\text{te}}}^{\text{te}}]^{\top}. \quad (4.14)$$

Algorithm 4 describes the testing phase of the proposed technique.

Algorithm 4: Proposed feature extraction technique - testing phase

Input:

- Testing dataset tensor $\mathfrak{X}^{\text{te}} \in \mathbb{R}^{N_1 \times \dots \times N_R \times M^{\text{te}}}$
- Weight tensor $\mathbf{e}^{S,C} \in \mathbb{R}^{N_1 \times \dots \times N_R}$
- Average common feature tensor $\tilde{\mathfrak{X}}^{S,C} \in \mathbb{R}^{N_1 \times \dots \times N_R}$

Output:

- Predicted testing class label vector $\hat{\mathbf{y}}^{\text{te}} \in \mathbb{R}^{M^{\text{te}}}$

Algorithm Steps:

- 1 Compute the filtered testing dataset tensor $\mathfrak{X}^{\text{te},[f]} \in \mathbb{R}^{N_1 \times \dots \times N_R \times M^{\text{te}}}$ as in (4.12)
 - 2 Unfold $\mathfrak{X}^{\text{te},[f]}$ into the matrix form $[\mathfrak{X}^{\text{te},[f]}]_{(R+1)} \in \mathbb{R}^{M^{\text{te}} \times N}$ as in (4.13)
 - 3 Compute the predicted testing class label vector $\hat{\mathbf{y}}^{\text{te}} \in \mathbb{R}^{M^{\text{te}}}$ as in (4.14).
-

4.2.2 Proposed MLP Architecture

The training and testing phases of the proposed MLP architecture for DDoS attack detection using the method presented in Subsection 4.2.1 are described as follows.

4.2.2.1 Training Phase

Multilayer perceptrons are fully connected feedforward neural networks where the output of one layer is the input of the subsequent layer. If the MLP presents J layers, the output vector $\mathbf{z}^{[j]} \in \mathbb{R}^{N^{[j]}}$ of the j -th layer after receiving the input vector $\mathbf{z}^{[j-1]} \in \mathbb{R}^{N^{[j-1]}}$ from the $(j-1)$ -th layer is given by

$$\mathbf{z}^{[j]} = f^{[j]}(\mathbf{W}^{[j,j-1]} \cdot \mathbf{z}^{[j-1]} + \mathbf{b}^{[j]}), \quad (4.15)$$

where $\mathbf{W}^{[j,j-1]} \in \mathbb{R}^{N^{[j]} \times N^{[j-1]}}$ is the weight matrix, $\mathbf{b}^{[j]} \in \mathbb{R}^{N^{[j]}}$ is the bias vector, $f^{[j]}(\cdot)$ is the activation function and $N^{[j]}$ is the number of neurons for $j = 1, \dots, J$.

The minibatch gradient descent is a well-known technique applied on multilayer perceptron optimization [2]. According to such approach, when the training of the s -th minibatch for $s = 1, \dots, S$ is finished, each weight w^s between any two neurons of the MLP is updated by a learning rate β in order to minimize the error function $e(w^s)$ as follows [109]

$$w^{s+1} = w^s - \beta \cdot \frac{\partial e(w^s)}{\partial w^s}. \quad (4.16)$$

Since the input data of MLPs is one-dimensional,, we have to matricize the input dataset tensor such that each instance can be directly forwarded to the neural network. In this sense, first we compute the $(R+1)$ -th mode unfolding of (4.7) as follows

$$[\mathfrak{X}^{s,[f]}]_{(R+1)} = [\mathfrak{X}^s]_{(R+1)} - [\mathbf{e}^{s,C}]_{(R+1)} \odot [\tilde{\mathfrak{X}}^{s,C}]_{(R+1)}, \quad \text{for } s = 1, \dots, S. \quad (4.17)$$

The weight tensor $\mathbf{C}^{s,C}$ is composed by M^{mb} identical tensors \mathbf{C}^s stacked along the $(R + 1)$ -th dimension. Consequently, the unfolded matrix $[\mathbf{C}^{s,C}]_{(R+1)} \in \mathbb{R}^{M^{\text{mb}} \times N}$ presents M^{mb} identical row vectors $\mathbf{c}^s = [c_1^s, \dots, c_N^s]^\top = \text{vec}\{\mathbf{C}^s(:, \dots, m)\}$ for $m = 1, \dots, M^{\text{mb}}$. Alternatively, \mathbf{c}^s can also be written as a diagonal matrix $\mathbf{C}^{s,\text{diag}} = \text{diag}\{[c_1^s, \dots, c_N^s]\} \in \mathbb{R}^{N \times N}$.

Similarly, $[\bar{\mathbf{x}}^{s,C}]_{(R+1)}$ is composed by M^{mb} identical row vectors $\bar{\mathbf{x}}^s = [\bar{x}_1^s, \dots, \bar{x}_N^s]^\top = \text{vec}\{\bar{\mathbf{x}}^s(:, \dots, m)\}$ for $m = 1, \dots, M^{\text{mb}}$. If we represent the $(R + 1)$ -th mode unfolding matrices in (4.17) as a function of its rows and replace the Hadamard product by the dot product, such equation can be rewritten as

$$[\mathbf{x}^{s,[f]}]_{(R+1)}(m, :) = -\mathbf{C}^{s,\text{diag}} \cdot \bar{\mathbf{x}}^s + [\mathbf{x}^s]_{(R+1)}(m, :), \quad \text{for } m = 1, \dots, M^{\text{mb}}. \quad (4.18)$$

Note the similarity between (4.18) and the argument of the activation function $f^{[j]}(\cdot)$ in (4.15). In this sense, a new input layer is included on the traditional MLP architecture where the computation described in (4.18) can be directly performed. Fig. 4.4 illustrates the proposed MLP architecture for DDoS attack detection with the new input layer, labeled as “layer 0”, in red color, placed at the left of the conventional input layer, referred as “layer 1”.

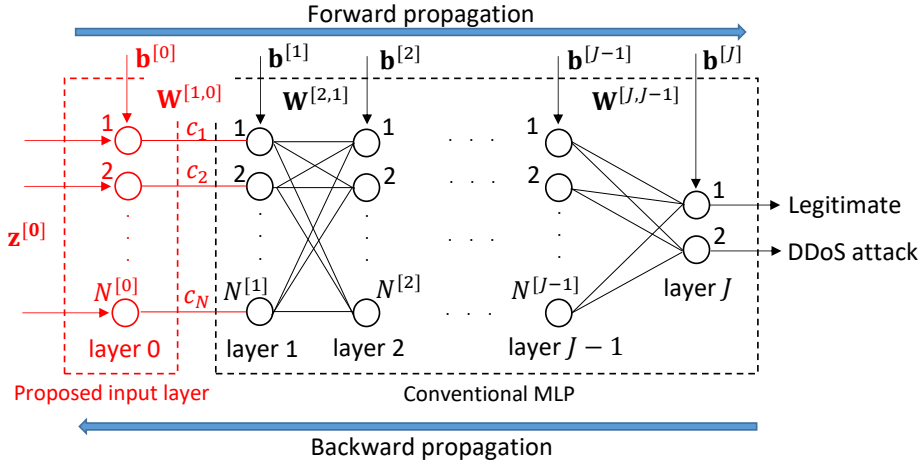


Figure 4.4 – The proposed MLP architecture for DDoS attack detection.

In summary, the main contribution relative to the conventional MLP corresponds to the inclusion of a new input layer, with the following parameters obtained from the comparison between (4.15) and (4.18):

- the number of nodes of the proposed input layer is given by $N^{[0]} = N$, i.e., the number of dataset features;
- the weight matrix $\mathbf{W}^{[1,0]}$ between the conventional and the proposed input layers is

given by $-\mathbf{C}^{s,\text{diag}}$;

- the input data vector $\mathbf{z}^{[0]}$ at the proposed input layer corresponds to the average principal components vector $\bar{\mathbf{x}}^s$; and
- the bias vector $\mathbf{b}^{[0]}$ at the proposed input layer corresponds to the instance vector $[\mathcal{X}^s]_{(R+1)}(m, :)$ for $m = 1, \dots, M^{\text{mb}}$.

Note that the weights in $\mathbf{C}^{s,\text{diag}} \in \mathbb{R}^{N \times N}$ are iteratively updated during the MLP forward and backward propagation in the training phase and, consequently, more accurate results can be achieved. From (4.16), when the $(s+1)$ -th minibatch training begins, each weight c_n for $n = 1, \dots, N$ is updated as follows

$$c_n^{s+1} = c_n^s - \beta \cdot \frac{\partial e(c_n^s)}{\partial c_n^s}. \quad (4.19)$$

4.2.2.2 Testing Phase

In this phase, each testing instance is feedforwarded through the trained MLP for classification. Consequently, the filtering operation on the testing vectors is given by

$$[\mathcal{X}^{\text{te},[\text{f}]}]_{(R+1)}(m, :) = -\hat{\mathbf{C}}^{S,\text{diag}} \cdot \bar{\mathbf{x}}^S + [\mathcal{X}^{\text{te}}]_{(R+1)}(m, :), \quad \text{for } m = 1, \dots, M^{\text{te}}, \quad (4.20)$$

where $\hat{\mathbf{C}}^{S,\text{diag}}$ is the weight diagonal matrix and $\bar{\mathbf{x}}^S$ is the average common feature vector, both obtained after the training of the S -th minibatch.

4.2.2.3 Summary

As shown in the numerical simulations of Section 4.4, the proposed MLP is compared with conventional MLPs in which state-of-the-art low-rank approximation techniques are previously applied to the dataset, namely, HOSVD [62, 115] and HOOI [108]. Fig. 4.5 illustrates the block diagram corresponding to the training and testing phases of the competing LRA based MLPs. After the multilinear rank (d_1, \dots, d_{R+1}) is obtained in Block 1.1 by some R -D MOS scheme, the dataset tensor $\mathcal{X} \in \mathbb{R}^{N_1 \times \dots \times N_R \times M}$ is denoised by state-of-the-art LRA techniques, such as HOSVD or HOOI, generating the tensor $\tilde{\mathcal{X}} \in \mathbb{R}^{N_1 \times \dots \times N_R \times M}$. Next, after unfolding $\tilde{\mathcal{X}}$ into the matrix form in Block 1.3, the unfolded matrix $\tilde{\mathbf{X}} \in \mathbb{R}^{M \times N}$ is split, in Block 1.4, into S minibatches $\tilde{\mathbf{X}}^s \in \mathbb{R}^{M^{\text{mb}} \times N}$ for $s = 1, \dots, S$, which are then forwarded to a conventional MLP for classification in Block 1.5. Further, Figs. 4.6 and 4.7 depicts the training and testing phases of the proposed common feature extraction based MLP, respectively. In Fig. 4.6, initially Block 2.1 splits the training dataset tensor \mathcal{X}^{tr} into S minibatches

$\mathbf{X}^s \in \mathbb{R}^{N_1 \times \dots \times N_R \times M^{\text{mb}}}$ for $s = 1, \dots, S$. Next, Block 2.2 estimates the multilinear rank of \mathbf{X}^s , (d_1, \dots, d_{R+1}) , by using an R -D MOS scheme, whereas Block 2.3 performs the low-rank approximation of $\tilde{\mathbf{X}}^s$, obtaining $\mathbf{G}^s \in \mathbb{R}^{d_1^s \times \dots \times d_{R+1}^s}$ and $\mathbf{A}_r^s \in \mathbb{R}^{N_r \times d_r^s}$ for $r = 1, \dots, R + 1$. Following, after computing the common feature tensor $\tilde{\mathbf{X}}^s \in \mathbb{R}^{N_1 \times \dots \times N_R \times d_{R+1}^s}$ in Block 2.4, Block 2.5 computes the average common feature tensor $\bar{\mathbf{X}}^{s,C} \in \mathbb{R}^{N_1 \times \dots \times N_R \times M^{\text{mb}}}$. Finally, the unfolded dataset matrices $\mathbf{X}^s \in \mathbb{R}^{M^{\text{mb}} \times N}$ and $\bar{\mathbf{X}}^{s,C} \in \mathbb{R}^{M^{\text{mb}} \times N}$, obtained in Blocks 2.6 and 2.7, respectively, are sent to the proposed MLP for classification in Block 2.8. Finally, the testing phase of the proposed common feature extraction based MLP is illustrated in Fig. 4.7. First, the testing dataset tensor $\mathbf{X}^{\text{te}} \in \mathbb{R}^{N_1 \times \dots \times N_R \times M^{\text{te}}}$ is unfolded into the matrix $\mathbf{X}^{\text{te}} \in M^{\text{te}} \times N$ in Block 3.1. Next, the unfolded matrices \mathbf{X}^{te} and $\bar{\mathbf{X}}^{s,C} \in \mathbb{R}^{M^{\text{mb}} \times N}$, where the latter is computed in Block 2.7 of Fig. 4.6, are forwarded to the proposed MLP in Block 3.2 for classification tasks.

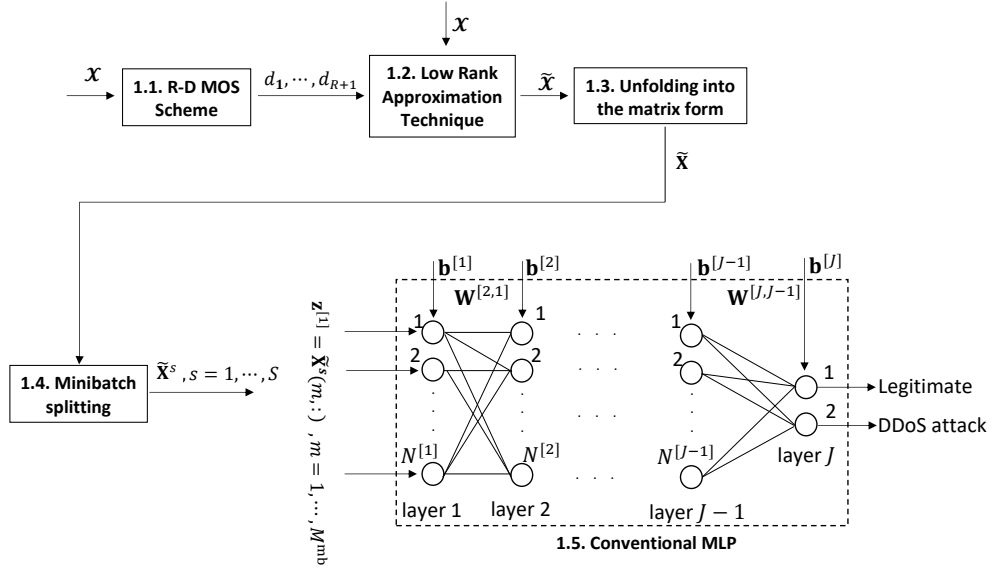


Figure 4.5 – Block diagram of the low-rank approximation based MLP (training and testing phases).

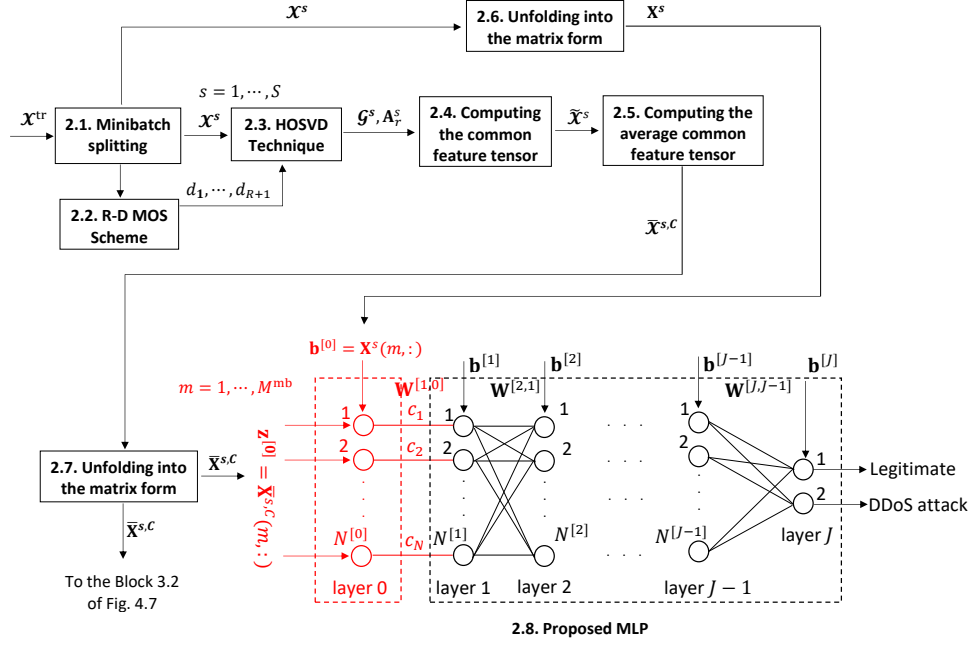


Figure 4.6 – Block diagram of the proposed common feature extraction based MLP (training phase).

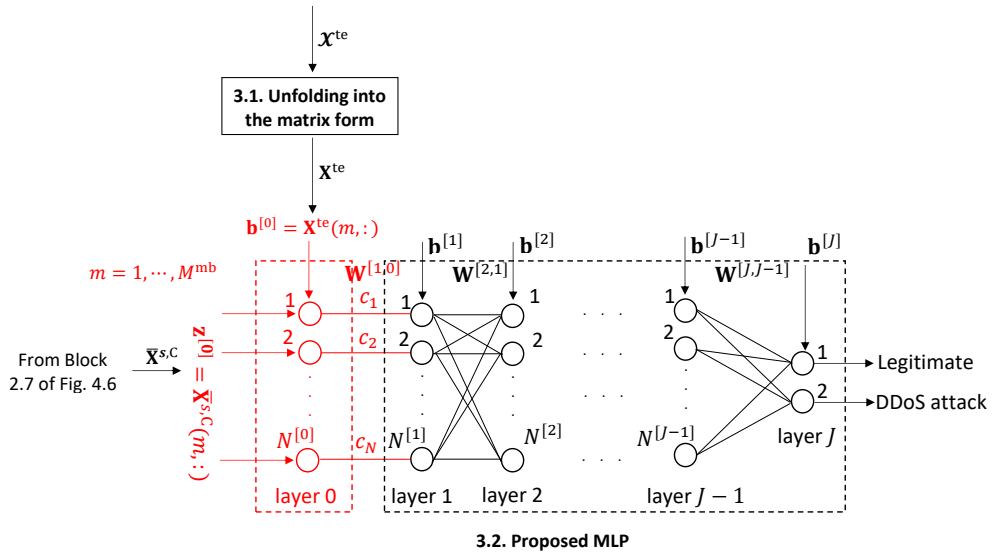


Figure 4.7 – Block diagram of the proposed common feature extraction based MLP (testing phase).

4.3 COMPUTATIONAL COMPLEXITY

This section discusses the computational complexity of the training phase of the proposed feature extraction technique presented in Subsection 4.2.1. The computational cost related to folding and unfolding of matrices and tensors is not considered, since such functions are about data representations. For simplicity, the computational complexity is analyzed for a three-dimensional dataset tensor $\mathcal{X}^s \in \mathbb{R}^{N_1 \times N_2 \times M^{mb}}$. We provide an analysis of the most

costly calculations as a function of the largest contributions of the most important variables, namely, N_1 , N_2 , S , M^{mb} , as well as the multilinear rank, (d_1, d_2, d_3) . Further, as it can be seen in Section 4.4, the proposed framework is compared with state-of-the-art low-rank approximation techniques, namely, Higher Order Orthogonal Iteration (HOOI) [108] and Higher Order Singular Value Decomposition (HOSVD) [62, 115]. Thus, in this section, the computational complexities of the competing schemes are also introduced.

In our proposed approach, the HOSVD is applied S times, one for each minibatch \mathcal{X}^s for $s = 1, \dots, S$. Thus, the computational complexity of the proposed scheme related to the HOSVD low-rank approximation technique can be expressed as [136]

$$\mathfrak{O}[\text{Proposed}_{\text{HOSVD}}] = \mathfrak{O} \left[S \left(\sum_{j=1}^3 \left(N_j \prod_{k=1}^3 N_k \right) + \sum_{j=1}^3 \left(\prod_{k=1}^j d_k \prod_{k=j}^3 N_k \right) \right) \right], \quad (4.21)$$

where, for simplicity of notation, N_3 corresponds to M^{mb} .

The computation of the common feature tensor as well as its average along the 3rd dimension require two tensor times matrix products plus the average calculation. The time cost of this step can be denoted as

$$\mathfrak{O}[\text{CF-AC}] = \mathfrak{O}[N_1^2 N_2 d_3] + \mathfrak{O}[N_1 N_2^2 d_3] + \mathfrak{O}[N_1 N_2 d_3]. \quad (4.22)$$

Finally, the overall computational complexity of the proposed feature extraction technique corresponds to the sum of the above mentioned complexities, plus the cost $\mathfrak{O}[\text{MOS}]$ related to the adopted MOS scheme, i.e.,

$$\mathfrak{O}[\text{Proposed}] = \mathfrak{O}[\text{MOS}] + \mathfrak{O}[\text{Proposed}_{\text{HOSVD}}] + \mathfrak{O}[\text{CF-AC}]. \quad (4.23)$$

Table 4.2 summarizes the computational complexities of the proposed feature extraction technique as well as its competing schemes. The second column illustrates the total computational cost, whereas the last column shows the final complexities, corresponding to the asymptotic dominant terms. The total complexity of the HOOI low-rank approximation technique is given by $\mathfrak{O}[\text{HOOI}] = \mathfrak{O}[N_{\text{max}}^3 d J] + \mathfrak{O}[N_{\text{max}}^2 d^2 J] + \mathfrak{O}[N_{\text{max}}^3 d] + \mathfrak{O}[N_{\text{max}} d^3]$, where $N_{\text{max}} = \max\{N_1, N_2, M\}$ and J is the number of algorithm iterations [112]. Furthermore, the total complexity of the HOSVD low-rank approximation scheme can be expressed as $\mathfrak{O}[\text{HOSVD}] = \mathfrak{O} \left[\left(\sum_{j=1}^3 (N_j \prod_{k=1}^3 N_k) + \sum_{j=1}^3 \left(\prod_{k=1}^j d_k \prod_{k=j}^3 N_k \right) \right) \right]$. In addition, from the asymptotic complexities shown in Table 2.6, it is observed that the proposed feature extraction technique presents the highest dominant complexity, given by $\mathfrak{O}[S R N^2 M^2]$, since tensor decompositions are applied S times, once for each minibatch. The numeri-

cal simulations presented in Section 4.4 reinforces the trade-off between the more accurate DDoS attack detection and the computational complexity. Moreover, note that the HOOI approach also presents a high computational cost, especially due to the number of instances raised to the cubic power. Finally, from Table 2.6, it can be seen that HOSVD shows the lowest time cost, bounded by the square of the number of dataset instances.

Table 4.2 – Computational complexity of the following schemes: (1) proposed feature extraction technique, (2) HOSVD, and (3) HOOI.

Model	Total Complexity	Dominant Complexity
(1)	$\mathcal{O}[\text{MOS}] + \mathcal{O}[N_1^2 N_2 d_3] + \mathcal{O}[N_1 N_2^2 d_3] + \mathcal{O}[N_1 N_2 d_3] +$ $+ \mathcal{O} \left[S \left(\sum_{j=1}^3 \left(N_j \prod_{k=1}^3 N_k \right) + \sum_{j=1}^3 \left(\prod_{k=1}^j d_k \prod_{k=j}^3 N_k \right) \right) \right]$	$\mathcal{O}[SRN^2M^2]$
(2)	$\mathcal{O} \left[\sum_{j=1}^3 \left(N_j \prod_{k=1}^3 N_k \right) + \sum_{j=1}^3 \left(\prod_{k=1}^j d_k \prod_{k=j}^3 N_k \right) \right]$	$\mathcal{O}[RN^2M^2]$
(3)	$\mathcal{O}[N_{\max}^3 dJ] + \mathcal{O}[N_{\max}^2 d^2 J] + \mathcal{O}[N_{\max}^3 d] + \mathcal{O}[N_{\max} d^3]$	$\mathcal{O}[dJRM^3]$

4.4 SIMULATION RESULTS

This section is divided into three subsections. First, details about the features and samples extracted from the DDoS benchmark datasets used in this chapter are shown in Subsection 4.4.1. Next, Subsections 4.4.2 and 4.4.3 present and discuss the simulation results, respectively.

4.4.1 DDoS Attack Datasets

In this chapter, we customized a single dataset composed by legitimate and DDoS attack samples extracted from the CIC-DDoS2019, CIC-IDS2018 and CIC-IDS2017 benchmark datasets, which are detailed in Appendix C. Next, the customized dataset is split into training and testing sets. Following, a 3-fold cross validation is performed on the training dataset such that each fold contains samples from a given benchmark dataset. At each iteration, the classifier is trained on samples from two benchmark datasets and validated on the third one. Finally, the trained classifier is evaluated on the testing dataset. Table 4.3 details the number of instances collected from the CIC-DDoS2019, CIC-IDS2018 and CIC-IDS2017 datasets and their respective types. Note that $M = 40,000$ instances were extracted from each dataset, of which 20% correspond to DDoS attacks.

4.4.2 Results

This subsection presents the performance and noise-robustness evaluation of the proposed MLP architecture for DDoS attack detection through numerical simulations. The per-

Table 4.3 – DDoS attack types used in this chapter, as well as the corresponding number of instances extracted from each dataset.

Dataset	Traffic Type	Total
CICDDoS2019	Legitimate	32,000
	DNS-based DDoS	800
	LDAP-based DDoS	800
	MSSQL-based DDoS	800
	NetBIOS-based DDoS	800
	NTP-based DDoS	800
	SNMP-based DDoS	800
	SSDP-based DDoS	800
	UDP flood	800
	TCP SYN flood	800
TFTP-based DDoS	800	
CICIDS2018	Legitimate	32,000
	HTTP and UDP-based DDoS	8,000
CICIDS2017	Legitimate	32,000
	HTTP and UDP-based DDoS	8,000
Total	Legitimate	96,000
	DDoS attacks	24,000

formance is assessed through the accuracy (Acc), detection rate (DR) and false alarm rate (FAR) metrics, whereas the Relative Loss of Accuracy (RLA) and Relative Loss of Detection Rate (RLDR) are adopted as noise-robustness evaluation metrics. Both performance and noise-robustness evaluation metrics are defined in Appendix D.

All experiments were executed on a desktop computer with processor Intel Core i7-2600 3.40 GHz and 16 GB of RAM. Data pre-processing and machine learning classifier algorithms were implemented in the Python libraries Scikit-Learn and Keras, whereas Tensorly [137] and HOTTBOX [135] were used to implement tensor computations. Additionally, in order to simulate false data injection, noise is added to each dataset prior to the pre-processing phase. As pointed out in [138], Gaussian noise is easy to implement and more difficult to be detected in practice, successfully fooling machine learning classifiers during the training and testing phases. Thus, $x\%$ of the instances of each feature $\mathbf{X}(:, n)$ for $n = 1, \dots, N$ are corrupted with Gaussian noise with mean zero and standard deviation $(\max(\mathbf{X}(:, n)) - \min(\mathbf{X}(:, n)))/5$. A total of 50 different experiments were simulated, with a MLP containing number of hidden layers varying from 2 to 5 and noise level between 5% and 30%. Additionally, the training dataset size ranges from 40% to 70% of all available instances. The tensor multilinear rank is estimated by applying the R -D MDL scheme and each experiment was trained for 100 epochs. The Rectified Linear Unit (ReLU) was selected as the activation function for all hidden layers, whereas Softmax function was applied on the output layer for classification tasks.

Initially, the proposed MLP is assessed for five different tensor foldings of the m -th

dataset instance $\mathbf{X}(m, :) \in \mathbb{R}^N$, as depicted in Figure 3.2 of Chapter 3. Since $N = 64$, each instance is folded as an R -th tensor $\mathfrak{X}(:, \dots, m) \in \mathbb{R}^{N_1 \times \dots \times N_R}$ for $R = 2, \dots, 6$, with sizes given by (8×8) , $(4 \times 4 \times 4)$, $(4 \times 4 \times 2 \times 2)$, $(4 \times 2 \times 2 \times 2 \times 2)$ and $(2 \times 2 \times 2 \times 2 \times 2 \times 2)$, respectively. Moreover, the noise level is fixed in 10%. The values of Acc, DR and FAR, as well as the training times (in seconds) obtained for each instance configuration are shown in Table 4.4. Further, the best metric value obtained by a given tensor folding is highlighted in bold. In terms of accuracy and detection rate, the two-dimensional configuration outperforms the other schemes, showing values of 96.92% and 97.42%, respectively. On the other hand, the lowest false alarm rate was obtained by the configuration $4 \times 2 \times 2 \times 2 \times 2$, achieving 1.38%. Note that the mean training times are higher as the tensor order is larger. In this sense, the two-dimensional instance configuration is adopted in simulations due to two main reasons: the lower processing time, which is a fundamental requirement in intrusion detection problems; and the higher performance regarding Acc and DR, despite its FAR is not the best compared to the other tensor foldings. Therefore, from this point on, the dataset matrix $\mathbf{X} \in \mathbb{R}^{M \times N}$ is folded into a three-dimensional tensor $\mathfrak{X} \in \mathbb{R}^{N_1 \times N_2 \times M}$, where $N_1 = N_2 = 8$.

Table 4.4 – Performance evaluation and mean training times (in seconds) of the proposed technique for different instance configurations.

Tensor Size	Accuracy	Detection Rate	False Alarm Rate	Training time (s)
8×8	0.9692	0.9742	0.0155	461.9
$4 \times 4 \times 4$	0.9659	0.9719	0.0157	474.0
$4 \times 4 \times 2 \times 2$	0.9684	0.9737	0.0153	487.2
$4 \times 2 \times 2 \times 2 \times 2$	0.9641	0.9712	0.0138	504.1
$2 \times 2 \times 2 \times 2 \times 2 \times 2$	0.9639	0.9724	0.0097	520.8

Fig. 4.8a and 4.8b show the training and validation accuracies, as well as the training and validation costs, as a function of the number of training epochs, for the proposed MLP architecture. Such curves are very useful to diagnose problems, as well as to check the learning and generalization behavior of the model in order to avoid overfitting issues. From the results, we observe that our approach obtained accuracy of 90% and cost of 6.4% just after 20 epochs, for both training and validation. Besides, after 60 epochs, the proposed scheme achieves accuracy and cost of 99.5% and 3.4%, respectively. Therefore, these findings confirm that our proposed MLP architecture for DDoS attack detection presents good learning and generalization behaviors, which can potentially reflect in good testing results.

Next, in Fig. 4.9a, we assess the mean value of the weights c_1, \dots, c_N , given by $\bar{c} = (1/N) \sum_{n=1}^N c_n$, as a function of the number of epochs. The mean weight \bar{c} corresponds to the average value for 50 experiments. Our idea is to verify the average behavior of the weights as the machine learning classification algorithm is trained after a given number of training iterations. From the results shown in Fig. 4.9a, we observe that \bar{c} presents an exponential growth until it reaches an approximately constant value, in which the model is theoretically

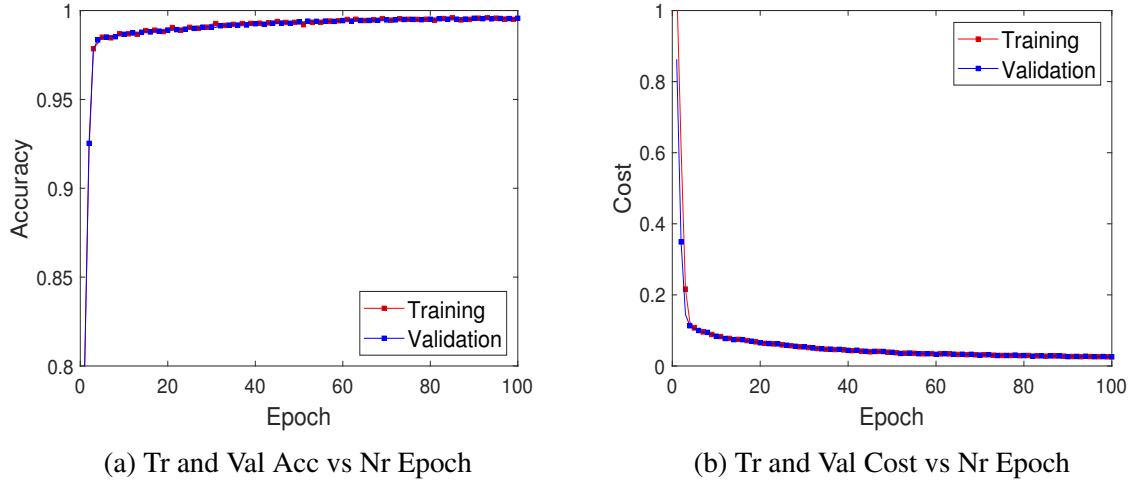


Figure 4.8 – Plots of: (a) training and validation accuracy, and (b) training and validation cost, as a function of the number of epochs, for the proposed MLP architecture.

fit to the training data. In Fig. 4.9b, we present a section of the proposed MLP architecture, in which the weights are highlighted within the dotted red rectangle.

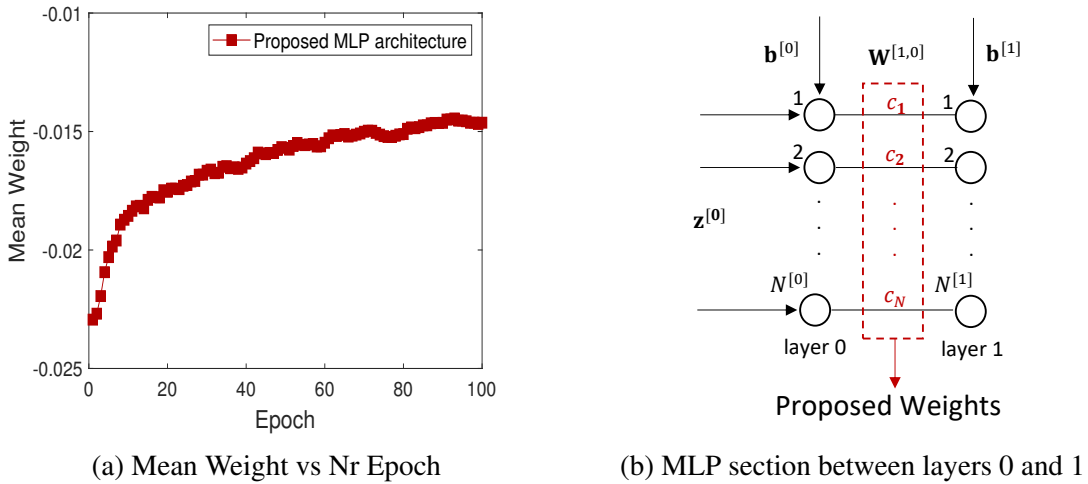


Figure 4.9 – (a) Mean weight as a function of the number of epochs. (b) Section of the MLP representing the proposed weights within the dotted red rectangle.

Throughout this section, the proposed approach is compared with conventional MLPs in which the state-of-the-art HOSVD and HOOI low-rank approximation techniques are previously applied to the dataset. Fig. 4.10 shows the Acc, DR and FAR as a function of the Noise Level (NL) and the number of Hidden Layers (HL). The learning rate β is 0.1, whereas the number of hidden layers is fixed in 5, with layer configuration 100/80/60/40/20 neurons and $M^{\text{mb}} = 512$. Moreover, in Fig. 4.10d to 4.10f, the NL is fixed in 10%. Note that the proposed MLP outperforms its competitor methods, especially under high noise levels and larger number of hidden layers. Such results confirm that our scheme leads to better performance due to the iterative updating of filtering weights.

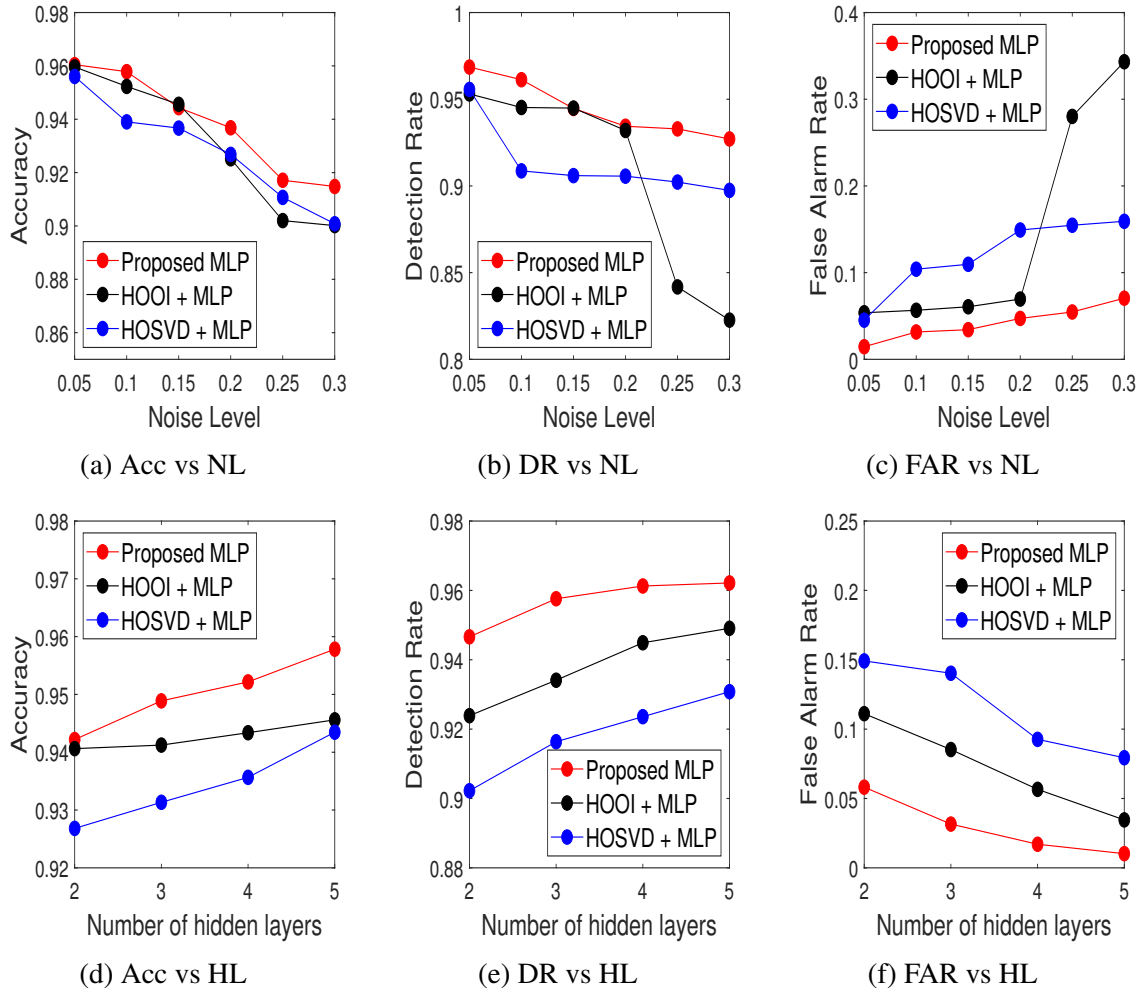


Figure 4.10 – Acc, DR and FAR as a function of the noise level and number of hidden layers.

Fig. 4.11 depicts the values of Acc, DR and FAR obtained for the noise-free case. The proposed scheme is compared with a conventional MLP, with no denoising technique, for different HL and TSP. The simulations present HL varying between 2 and 5, and $M^{mb} = 512$. The layer configurations for each scenario are, respectively: 100, 100/80, 100/80/60, 100/80/60/40 and 100/80/60/40/20. From the results shown in Fig. 4.11, we observe that all techniques deliver significantly better performance for higher number of hidden layers and higher training size proportions. Note that, considering Acc and DR, the proposed scheme outperforms its competitors for all of the HL and TSP ranges, despite presenting a worse performance of FAR for some metric values.

Following, the proposed MLP is assessed for detecting real time DDoS attacks. Three IDSs trained and tested under noise levels between 10% and 30% are compared: the proposed scheme, as well as the HOSVD and HOOI based MLPs. To simulate a small scale DDoS attack, a virtual network was implemented, as depicted in Fig. 4.12. DDoS traffic included TCP, UDP and HTTP GET flooding attacks generated by three attackers, whereas the victim is a web server. The attacks were launched during a period of 60 minutes by using the Low

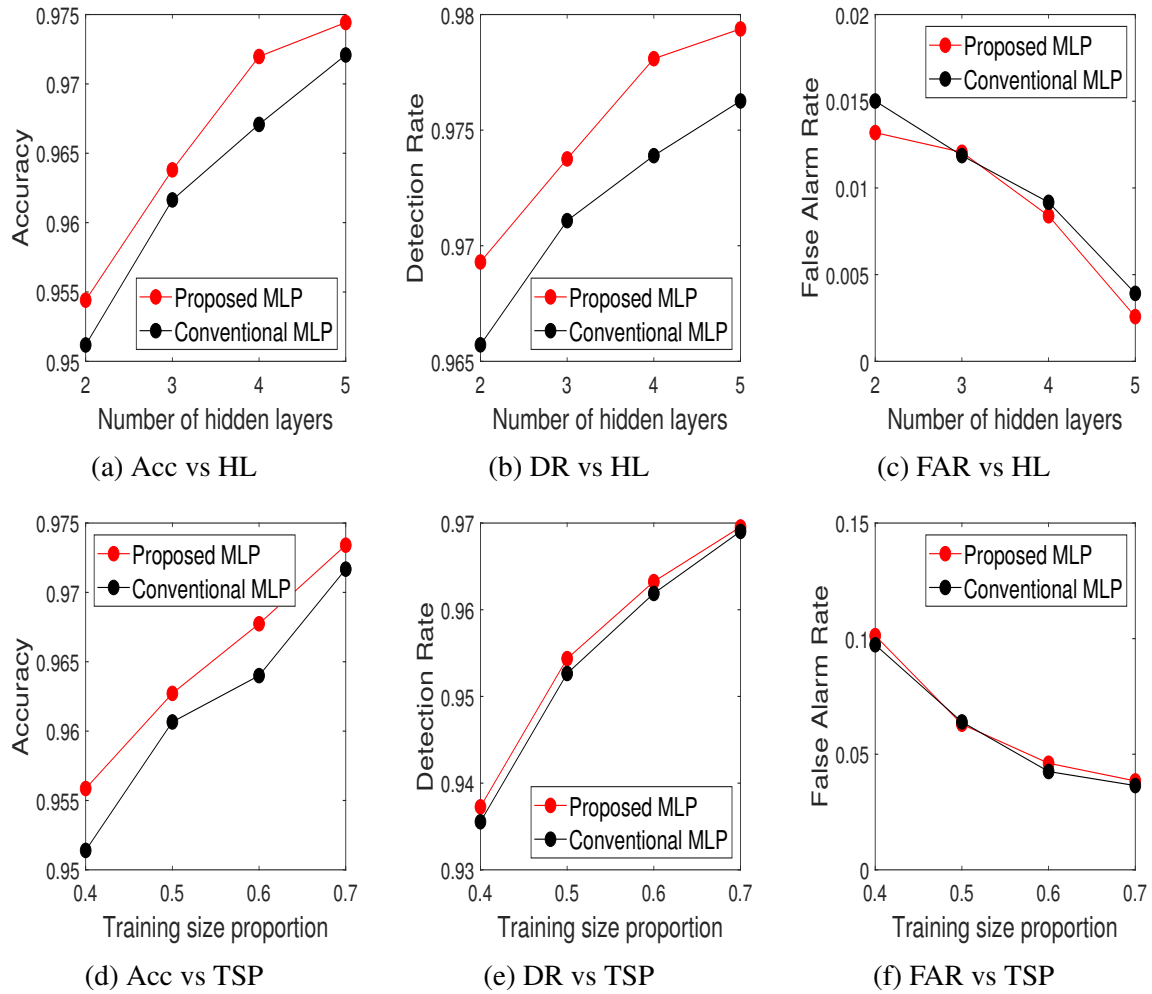


Figure 4.11 – Acc, DR and FAR as a function of the number of hidden layers and training size proportion under noise-free conditions.

Orbit Ion Cannon (LOIC) tool. Additionally, legitimate traffic was generated by two users accessing the web server, and the IDS is positioned between the router and the victim. The network traffic captured by the IDS is converted into Comma-Separated Values (CSV) files for further processing. Table 4.5 shows the values of DR and FAR obtained for real time detection. Note that the proposed scheme outperforms the compared approaches when NL is higher than 20%, presenting considerable detection results.

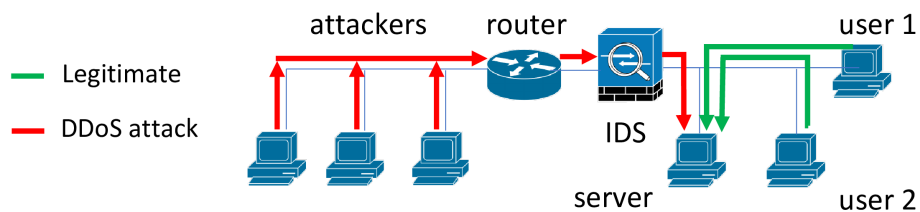


Figure 4.12 – Network topology for simulating real time attacks.

Further, Fig. 4.13 presents the mean training and testing times (in seconds), considering

Table 4.5 – Performance evaluation under real time attacks.

Model	10%		20%		30%	
	DR	FAR	DR	FAR	DR	FAR
Proposed MLP	0.9967	0.0001	0.9661	0.0447	0.8373	0.1792
HOSVD + MLP	0.9413	0.1173	0.8485	0.1256	0.7697	0.3894
HOOI + MLP	0.9999	0.0000	0.8850	0.0659	0.7736	0.3088

different number of hidden layers. Three techniques are compared: the proposed scheme, as well as the HOSVD and HOOI based MLPs. Since denoising and filtering are performed through the MLP layers in our proposed scheme, its total time correspond to the MLP processing time. On the other hand, since HOSVD and HOOI are preprocessing steps in their respective MLPs, the total time correspond to the sum between the corresponding low-rank approximation technique time and the MLP processing time. The training times refer to a period of 100 epochs and NL is fixed in 10%. Note that, from Fig. 4.13a, the proposed technique is more computationally expensive than the competing approaches, showing higher training times, which reflects the trade-off to achieve a more accurate detection. However, this is compensated considering the testing times, as shown in Fig. 4.13b.

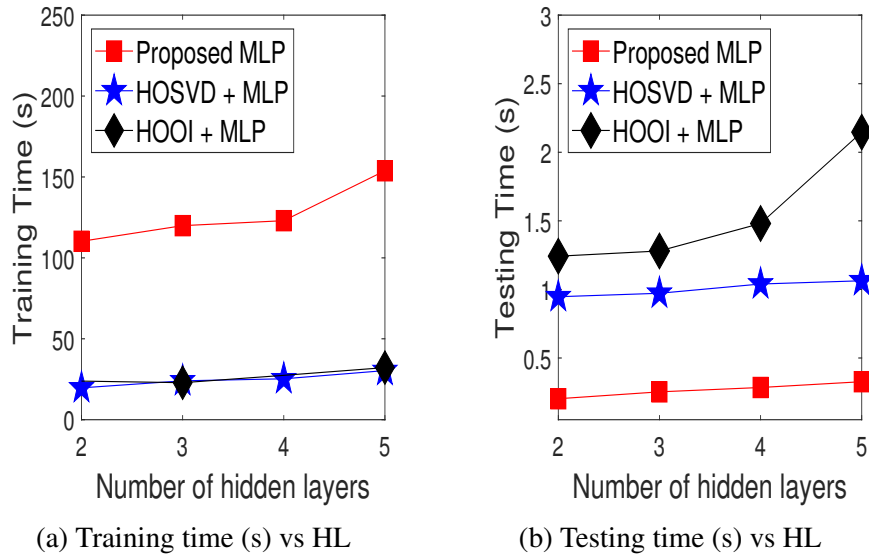


Figure 4.13 – (a) Training time (s) versus number of hidden layers. (b) Testing time (s) versus number of hidden layers.

Then, Table 4.6 shows the noise-robustness results of each compared technique. In this case, the same simulation parameters adopted in the previous experiments (those whose results are shown in Fig. 4.10) are considered. The best values for each compared technique and noise level are highlighted in bold letters. It can be observed that the proposed MLP outperforms all competitor methods when $NL \geq 20\%$, showing an outstanding noise-robustness.

Next, simulation results obtained from the comparison between the proposed MLP and related works are shown in Table 4.7. Since CIC-DDoS2019 has been recently developed,

Table 4.6 – Noise-robustness evaluation results.

NL	RLA (%)			RLDR (%)		
	Proposed	HOOI + MLP	HOSVD + MLP	Proposed	HOOI + MLP	HOSVD + MLP
10%	0.68	0.38	0.33	0.72	0.53	0.82
20%	0.86	1.17	2.44	0.86	2.16	3.21
30%	1.03	2.62	2.99	1.73	4.40	5.29
40%	2.10	8.03	3.29	1.86	6.20	6.67
50%	2.76	8.60	7.46	4.09	6.29	8.63

few works applying such dataset for NIDS validation have been found in the literature. Consequently, the CIC-IDS2017 benchmark dataset was also applied to validate the proposed technique. Additionally, in Table 4.7, NL = 0 was adopted by our proposed scheme. Despite the detection performance of the proposed MLP is not the best when considering CIC-IDS2017, it still presents outstanding results, with Acc = 98.95%, DR = 98.31% and FAR = 0.15%. Note that our scheme is outperformed by Doriguzzi-Corin et al. [125], in which a CNN based IDS is proposed. In [125], DDoS attacks and legitimate traffic patterns are learned by CNN through convolutional filters sliding over packet flow inputs to identify anomalous characteristics, which may explain its higher Acc and DR results compared to our MLP based solution. On the other hand, when CIC-DDoS2019 is considered for model validation, the proposed MLP outperforms the competitor works in terms of accuracy, achieving 98.75% against 98.70% obtained by Aytac et al. [122], which is the second best model among the compared ones when considering accuracy values. Therefore, it can be observed that our proposed scheme also presents considerable results with CIC-DDoS2019, although it is not the best model in terms of detection rate, being outperformed by Elsayed et al. [118] in such aspect.

4.4.3 Discussion

In this subsection, the simulation results presented in Subsection 4.4.2 are discussed. From the results shown in Fig. 4.10, we observe that, when noisy datasets are considered, the proposed approach is more efficient for detecting DDoS attacks compared to HOSVD and HOOI based MLPs in most of the simulations. However, the accuracy of our scheme is matched by HOOI based MLP in scenarios with low NL. In this case, the more accurate core tensor and singular matrices generated through orthogonal iterations in HOOI lead to a better dataset denoising. Moreover, the HOSVD based MLP presents the worst performance in terms of accuracy, since a single tensor decomposition is performed on the whole dataset during the preprocessing phase. For instance, when NL = 5%, the proposed scheme and the HOOI based MLP present accuracy of 96.05% and 95.96%, respectively, against 95.60% achieved by the HOSVD based MLP. On the other hand, under larger number of hidden layers and higher noise level conditions, the proposed approach is far superior compared to the HOOI and HOSVD based MLPs. For example, when NL = 30% and HL = 5, our

Table 4.7 – Performance comparison with related works, considering the CIC-IDS2017 and CIC-DDoS2019 datasets.

Dataset	Work	Classification Method	Acc	DR	FAR
CIC-IDS2017	Proposed Scheme	MLP	0.9895	0.9831	0.0015
	Doriguzzi-Corin et al. [125]	LUCID	0.9967	0.9994	0.0059
	Roopak et al. [126]	MLP	0.8634	0.8625	N/A
	Roopak et al. [126]	1D-CNN	0.9514	0.9017	N/A
	Roopak et al. [126]	LSTM	0.9624	0.8989	N/A
	Roopak et al. [126]	1D-CNN+LSTM	0.9716	0.9910	N/A
	Aamir and Ali Zaidi [139]	kNN	0.9500	N/A	N/A
	Aamir and Ali Zaidi [139]	SVM	0.9200	N/A	N/A
	Aamir and Ali Zaidi [139]	RFo	0.9666	N/A	N/A
	Lima Filho et al. [140]	RFo	N/A	0.8000	0.0020
	Haider et al. [127]	Ensemble CNN	0.9945	0.9964	N/A
	Aksu et al. [141]	kNN	0.9572	0.9589	N/A
	Aksu et al. [141]	SVM	0.6069	0.7142	N/A
	Aksu et al. [141]	DT	0.9900	0.9900	N/A
	Chen et al. [128]	CNN	0.9887	N/A	N/A
	Ustebay et al. [142]	MLP	0.9100	N/A	N/A
	Yulianto et al. [143]	AdaBoost+PCA+SMOTE	0.8147	0.9576	N/A
	Zhu et al. [144]	AMF-LSTM	N/A	0.9800	N/A
	Yao et al. [145]	DeepGFL	N/A	0.3024	N/A
	CIC-DDoS2019	Proposed Scheme	MLP	0.9875	0.9746
Maranhão et al. [3]		DT	0.9754	0.9509	N/A
Elsayed et al. [118]		RNN-Autoencoder	N/A	0.9900	N/A
Shurman et al. [119]		LSTM 1 layer	0.9154	N/A	N/A
Shurman et al. [119]		LSTM 2 layer	0.9674	N/A	N/A
Sharafaldin et al. [120]		ID3	N/A	0.6500	N/A
Sharafaldin et al. [120]		RFo	N/A	0.5600	N/A
Sharafaldin et al. [120]		NB	N/A	0.1100	N/A
Sharafaldin et al. [120]		LR	N/A	0.0200	N/A
Hussain et al. [121]		ResNet	N/A	0.8600	N/A
Aytac et al. [122]		RFo	0.9840	N/A	N/A
Aytac et al. [122]		Gaussian NB	0.9870	N/A	N/A

scheme achieves detection rates equal to 92.71% and 96.13%, respectively. Considering the same NL and HL values, the HOSVD based MLP presents DR of 89.74% and 93.08%, respectively, whereas detection rates of 82.26% and 94.49% were achieved by the HOOI based MLP. Note that the orthogonal iterations performed by HOOI during the process of tensor decomposition are more sensitive to higher noise levels compared to HOSVD.

Next, Fig. 4.11 illustrates the comparison between the proposed scheme and a conventional MLP under noise-free conditions. From the results shown in Fig. 4.11, it is observed that the proposed approach outperforms the conventional MLP in terms of Acc and DR for multiple HL and TSP configurations. For instance, considering HL = 5 and TSP = 0.4, the proposed scheme achieves detection rates of 97.94% and 95.44%, respectively. On the other hand, the conventional MLP presents DR equal to 97.63% and 95.26% for the same HL and TSP configurations, respectively. However, our scheme is more prone to false positives and, hence, legitimate traffic is wrongly classified as malicious activity. Therefore, the proposed MLP presents a higher noise-robustness and efficiency due to two factors: the noise attenuation provided by HOSVD and the more discriminative individual information resulting from

dataset filtering.

Furthermore, a drawback of our approach is its higher training time, especially for larger number of hidden layers, caused by multiple HOSVDs performed over training data batches. From Fig. 4.13a, it can be observed that, when $HL = 5$, a training time of 153.82 s was shown by our scheme, whereas the HOSVD and HOOI based MLPs presented 30.34 s and 32.19 s, respectively. On the other hand, the proposed MLP shows lower testing times, since low cost computations are performed during the testing phase, in contrast to the tensor decompositions executed in HOSVD and HOOI. Still regarding the same number of hidden layers, our approach showed a testing time of 0.3272 s, whereas 1.04 s and 1.48 s were achieved by the HOSVD and HOOI based schemes, respectively.

Moreover, the proposed MLP was assessed for detecting DDoS attacks under real time conditions. From the results shown in Table 4.5, it is clear that our proposed scheme is more accurate for detecting real time DDoS attacks in comparison with the HOOI and HOSVD based MLPs when $NL \geq 20\%$. For instance, when the noise level is fixed in 30%, a detection rate of 83.73% was achieved by the proposed MLP, whereas detection rates of 76.97% and 77.36% were shown by the HOSVD and HOOI based approaches, respectively. Such results reflect the superiority of our scheme due to a more efficient dataset filtering during the training phase. Nonetheless, the best performance for $NL = 10\%$ was shown by the HOOI based MLP, in which orthogonal iterations provided a more accurate separation between signal and noise subspaces and, consequently, better classification results. In this case, the HOOI based MLP presented a detection rate of 99.99%, against 99.67% and 94.13% achieved by the proposed scheme and the HOSVD based MLP, respectively.

Finally, the above-mentioned schemes were compared in terms of noise-robustness evaluation metrics, namely, relative loss of accuracy and relative loss of detection rate. From the results shown in Table 4.6, it can be observed that our proposed scheme presents higher noise-robustness compared to the competitor methods, especially under high noise level conditions. For instance, when $NL = 50\%$, the proposed MLP presents $RLA = 2.76\%$, whereas 8.60% and 7.46% were shown by the HOOI and HOSVD based MLPs, respectively. Similarly, for a noise level of 30%, our approach outperforms both HOOI and HOSVD based schemes in terms of $RLDR$, achieving 1.73% against 4.40% and 5.29%, respectively. In this sense, the results discussed so far reinforce the higher noise-robustness showed by our proposed scheme, which is due to the more efficient HOSVD noise attenuation as well as the more discriminative individual information resulting from dataset filtering.

4.5 SUMMARY

Data corruption or “noise” present in datasets can lead to a performance degradation of machine learning classification algorithms, depending on their degree of sensitiveness. Such corrupted data can be generated, for example, after false data injection attacks performed on publicly available datasets on the web. One of the easiest attacks to be implemented in practice is the Gaussian noise injection, which aims to fool machine learning classifiers during the training phase and, consequently, can lead to misclassification results when applied on real time detection.

In this chapter, concepts regarding common and individual feature extraction were borrowed from image classification problems. Usually, ML algorithms present better performance after extracting the common features, since the more discriminative individual information is applied during the training phase. In this sense, we propose a novel technique in which the average value of the common features among dataset instances is iteratively filtered out via the Higher Order Singular Value Decomposition (HOSVD) algorithm. Moreover, we also propose a novel MLP architecture for DDoS attack detection in which the cited technique is directly applied. The best MLP parameters used for dataset filtering are dynamically computed such that the errors between the expected and predicted classifications are minimized. According to the simulation results performed by using a customized dataset containing samples extracted from the CIC-IDS2017, CIC-IDS2018 and CIC-DDoS2019 benchmark datasets, the proposed scheme outperforms the state-of-the-art HOSVD and HOOI based MLPs in terms of accuracy, detection rate and false alarm rate. Thus, our proposed MLP shows a considerable potential for detecting DDoS attacks when NIDSs are trained with corrupted data.

5 TENSOR FEATURE MAP CONCATENATION BASED CONVOLUTIONAL NEURAL NETWORK SCHEMES FOR DISTRIBUTED DENIAL OF SERVICE ATTACK DETECTION

In this chapter, the following research question is addressed: *How to efficiently detect DDoS attacks in a CPS network by jointly applying multidimensional feature map concatenation based techniques and deep learning algorithms, but now assuming that a noiseless dataset is used for training and testing?*

The remainder of this chapter is organized as follows:

- **Motivation:** in this section, an introduction to DDoS attack detection models based on deep learning techniques is presented.
- **Proposed Tensor Feature Map Concatenation Based CNN Schemes for DDoS Attack Detection:** this section introduces and discusses the proposed tensor feature map concatenation based CNN architectures for DDoS attack detection.
- **Computational Complexity:** the computational complexities of the proposed schemes are presented.
- **Simulation Results:** the performance of the proposed approaches is evaluated through numerical simulations.

The research contributions of Chapter 5 are summarized as follows:

1. A tensor feature map concatenation based CNN architecture for DDoS attack detection is proposed. In this approach, multiple branches, composed by CNN basic building blocks alternately positioned with concatenation modules, are placed in parallel such that the outputs from CNNs of consecutive branches are concatenated and sent to the following CNN within each branch. Finally, the outputs from all branches are concatenated and forwarded to the same flattening and multilayer perceptron blocks, where samples are classified as legitimate traffic or DDoS attack.

2. A second improved feature map concatenation based CNN architecture is also introduced. In this scheme, instead of the Flatten and MLP blocks in common for all of the branches, each branch is followed by its respective Flatten and MLP blocks, whose outputs are forwarded to a simple majority voting module in which the final classification is computed.
3. Numerical simulations are conducted in order to validate the proposed schemes by using the CIC-IDS2017 and CIC-DDoS2019 benchmark datasets. The results show that the proposed approaches outperform their competing techniques in terms of several performance evaluation metrics.

5.1 MOTIVATION

Distributed Denial of Service (DDoS) attacks are bandwidth-saturating cyberthreats which continuously trouble network operators and service providers in organizations over the world [146]. Their main goal is to prevent legitimate users from accessing network services by exhausting bandwidth and resources through huge volume of traffic, usually launched by thousands or even millions compromised devices of unsuspecting users, known as “bots” [86, 99, 147]. DDoS attacks can be classified as volumetric or flooding, and commonly use layer 3 or layer 4 protocols, such as ICMP, UDP and TCP, to generate extensive volume of traffic. In addition, application layer protocols, such as HTTP and DNS, can be used to generate more sophisticated DDoS attacks in which lower network bandwidth is used for starting. In this case, network resources are slowly depleted, since specific applications or services are targeted. Moreover, by sending the requested data with a very small packet window, attackers are able to keep the connection open as long as possible [118].

Several massive DDoS attacks have been launched on big organization’s networks in the recent years. For example, in October 2016, an extensive DDoS attack of 1.5 Tbps was executed against the Dyn DNS service provider. Multiple high-profile Dyn’s client websites, such as GitHub, HBO, Twitter, PayPal, Netflix and others were rendered after Dyn’s DNS infrastructure was knocked offline for several hours [148]. Furthermore, in October, 2020, the Google Cloud Team revealed that a 2.54 Tbps DDoS attack had been mitigated by the organization in September, 2017 [52]. Performed by Chinese Internet Service Providers, the attack targeted thousands of Google’s IP addresses and lasted more than six months. It was four times larger than the 623 Gbps attack launched by the Mirai botnet against the blog of cybersecurity KrebsOnSecurity one year earlier [46]. Moreover, the most recent massive DDoS attack occurred in February, 2020, when the Amazon’s AWS Shield protection service mitigated an extreme attack of 2.3 Tbps, targeted against one of its customers [51].

Recently, DDoS attacks became harder to detect and prevent due to the evolution of attack

approaches, which cannot be faced by traditional signature-based Network Intrusion Detection Systems (NIDS) [98]. In this sense, deep learning (DL) based solutions have been proposed such that high level features can be derived from traffic data, improving the detection performance [149]. Among the DL algorithms, one of the most popular is the Convolutional Neural Network (CNN), which presents outstanding results in multiple research areas, such as image classification [150], electroencephalography [151] and sentiment analysis [152]. In addition, CNNs have been successfully applied on the field of information security, including network intrusion detection problems [153, 154], but the literature in which they are specifically designed for detecting DDoS attacks is still limited.

In [118], Elsayed et al. proposed an intrusion detection system against DDoS attacks in Software-Defined Networking environments in which recurrent neural networks (RNN) and autoencoders are combined. Despite the recently released CIC-DDoS2019 dataset was applied in simulations, further considerations regarding training and testing times were not reported. In addition, Yuan et al. [149] proposed a DL based scheme to detect DDoS attacks from legitimate network traffic at the victim end, alleviating several issues, such as the vulnerability to slow attack rates. However, the model was validated by using an obsolete dataset (ISCX2012) and presented a huge number of trainable parameters. This fact may indicate higher processing times, which were not addressed by the authors. Furthermore, in [154], a feature fusion based parallel cross convolutional neural network for detecting abnormal network traffic was proposed, in which the CIC-IDS2017 dataset was used in simulations. However, the authors did not mention the training times obtained during the experiments. Moreover, in [155], Koay et al. proposed a DDoS attack detection approach using multiple entropy-based features and machine learning (ML) classifiers. The solution consisted of a voting system in which classification results provided by multiple ML algorithms were compared, and an alternating decision tree was used as an arbiter. Nevertheless, outdated datasets, such as DARPA98 and ISCX2012, were applied for validation. Additionally, the proposed solution relies on a feature construction phase, in which raw features are extracted from packet headers in order to build entropy variation features. Such fact may denote higher preprocessing times, which were not reported in the research. Chen et al. [128] proposed a multi-channel CNN based detection and early warning framework for DDoS attacks, whose results were based on real-time traffic and network package information. The authors applied the recent CIC-IDS2017 dataset in experiments, despite the obsolete KDDCUP99 was also used for model validation. Further, in [125], Doriguzzi-Corin et al. proposed a lightweight deep learning based approach for DDoS attack detection in online resource-constrained environments. In simulations, the authors used a customized dataset, composed by samples extracted from outdated and recent datasets, such as ISCX2012, CIC-IDS2017 and CIC-IDS2018. In [127], Haider et al. proposed a deep CNN ensemble framework for efficient DDoS attack detection in SDNs validated on the CIC-IDS2017 dataset. Nonetheless, higher training and testing times were reported in simulations, which was compensated by a sat-

isfactory performance detection. At last, in [2], Maranhão et al. proposed a multilayer perceptron (MLP) architecture for DDoS attack detection trained with corrupted data, which can be resulting, for example, of false data injection attacks performed on publicly available datasets. In simulations, the authors performed cross validation between samples collected from the recently released CIC-IDS2017, CIC-IDS2018 and CIC-DDoS2019 datasets. Still, a mentioned drawback is the higher training time due to the tensor decompositions performed during the feedforward propagation.

Despite such related works showed satisfactory performance results in DDoS attack detection, many of them [125, 128, 149, 155] applied obsolete datasets, often criticized and considered by many researchers as outdated. Due to the increasing number of attack scenarios, as well as the more complex software and network infrastructures, datasets must contain the most up-to-date traffic samples such that a more efficient attack detection can be achieved [156]. Additionally, some of the obsolete datasets present drawbacks, such as a large amount of redundancy (DARPA98) or low number of features (KDDCUP99), which can hinder the NIDS performance. Moreover, most of the related works ignore key performance evaluation metrics, such as precision [2], false positive rate [118, 149, 154, 155], false negative rate [2, 118, 125, 149, 154, 155] and F-score [2], whereas one of them focus only on the final overall accuracy [128]. Table 5.1 summarizes the main approaches and drawbacks of each related work, as well as the weaknesses addressed by this chapter.

Table 5.1 – Related works summary.

Ref.	Main Approach	Drawback
[2]	- Noise-robust MLP architecture for DDoS attack detection.	- Higher training times. - Lack of important performance evaluation metrics.
[118]	- Deep learning based model for DDoS attack detection.	- Lack of important performance evaluation metrics. - Lack of analysis of the training and testing times.
[125]	- Lightweight deep learning system for DDoS attack detection.	- Solution with fixed parameters.
[127]	- Deep CNN ensemble framework for DDoS attack detection in SDNs.	- Solution with fixed parameters. - Higher training and testing times.
[128]	- Multi-channel CNN based framework for DDoS attack detection.	- Outdated dataset (KDDCUP99) applied for model validation. - Solution with fixed parameters. - Focus only on the overall accuracy.
[149]	- Deep learning based approach for DDoS attack detection.	- Outdated dataset (ISCX2012) applied for model validation. - Lack of important performance evaluation metrics - Lack of analysis of the training and testing times.
[154]	- Deep learning based architecture for network intrusion detection.	- Solution with fixed parameters - Lack of important performance evaluation metrics. - Lack of analysis of the training time.
[155]	- Multi-classifier system using entropy-based features for DDoS attack detection.	- Outdated datasets (DARPA98 and ISCX2012) applied for model validation. - Lack of important performance evaluation metrics. - Lack of analysis of the preprocessing, training and testing times.
This chapter	- Tensor feature map concatenation based CNN schemes for DDoS attack detection.	Addressed drawbacks: - Recent datasets (CIC-IDS2017 and CIC-DDoS2019) applied for model validation. - Parameter-tuning based solution. - Complete set of performance evaluation metrics. - Analysis of the training and testing times.

To address the aforementioned weaknesses, the recently released CIC-DDoS2019 and

CIC-IDS2017 datasets are applied for model validation in this work. Furthermore, several important performance metrics are assessed, as it will be shown in Subsection 5.4, such that a whole description of the model performance is shown to the reader. In addition, this chapter proposes a DDoS attack detection system in which several model parameters, for instance, number of parallel branches, number of CNNs and number of MLP dense layers, can be tuned by the network administrator. In this sense, a better attack detection performance can be achieved by tuning the model settings, in contrast to several deep learning NIDS with fixed parameters usually found in the literature [125, 127, 128, 154].

In this chapter, two novel CNN architectures for DDoS attack detection based on multi-dimensional feature map concatenation are proposed. In the first one, called Tensor Feature Map Concatenation Based CNN (TFMCB-CNN), multiple branches, composed by CNN basic building blocks alternately positioned with concatenation modules, are placed in parallel such that the outputs from CNNs of consecutive branches are concatenated and sent to the following CNN within each branch. Finally, the branch outputs are concatenated and forwarded to the Flatten and Multilayer Perceptron (MLP) blocks, in which samples are classified as legitimate traffic or DDoS attack. Additionally, a refined version of the first scheme, known as Tensor Feature Map Concatenation Based CNN Improved via Majority Voting (TFMCB-CNN-MV), is also introduced. In this second approach, instead of the flattening and multilayer perceptron blocks in common for all of the branches, each branch is followed by its respective Flatten and MLP blocks, whose outputs are sent to a Simple Majority Voting (SMV) module, in which the final classification is computed. To evaluate the effectiveness of the proposed TTFMCB-CNN-MV and TFMCB-CNN schemes, extensive experiments were conducted by using the CIC-IDS2017 and CIC-DDoS2019 benchmark datasets for validation. The results confirm that the proposed schemes outperform state-of-the-art techniques in terms of several performance evaluation metrics.

5.2 PROPOSED TENSOR FEATURE MAP CONCATENATION BASED CNN SCHEME FOR DDoS ATTACK DETECTION

This section is divided into three subsections. First, Subsection 5.2.1 details the data preprocessing steps. Next, the proposed TFMCB-CNN and TFMCB-CNN-MV schemes for DDoS attack detection are shown in Subsections 5.2.2 and 5.2.3, respectively.

Throughout this chapter, the dataset can be represented as a matrix, denoted as $\mathbf{X} \in \mathbb{R}^{M \times N}$, where N is the number of features and M is the number of instances. Each column $\mathbf{X}_{:,n}$ for $n = 1, \dots, N$ corresponds to the n -th dataset feature, whereas each row $\mathbf{X}_{m,:}$ for $m = 1, \dots, M$ is the m -th dataset instance. Moreover, $\mathbf{y} = [y_1, \dots, y_M]^T \in \mathbb{R}^M$ denotes the class label vector, where y_m indicates if the m -th instance $\mathbf{X}_{m,:}$ for $m = 1, \dots, M$ is

legitimate traffic or DDoS attack.

Furthermore, the dataset matrix \mathbf{X} can also be represented in the tensor form. Each $\mathbf{X}_{m,:} \in \mathbb{R}^N$ for $m = 1, \dots, M$ can be reshaped as an R -dimensional tensor with size $N_1 \times \dots \times N_R$, with $N = \prod_{r=1}^R N_r$, and stacked along the $(R + 1)$ -th dimension, generating the dataset tensor $\mathfrak{X} \in \mathbb{R}^{N_1 \times \dots \times N_R \times M}$. Thus, the m -th dataset instance can be denoted as $\mathfrak{X}_{:, :, m} \in \mathbb{R}^{N_1 \times \dots \times N_R}$.

5.2.1 Data Preprocessing

Previously to the proposed tensor feature map concatenation based CNN approaches for DDoS attack detection, the dataset must be preprocessed so that it is suitable for using by ML classification algorithms. The data preprocessing is composed by three steps: dataset splitting, dataset preprocessing and minibatch splitting.

- **Dataset Splitting:** in this step, the dataset $\mathfrak{X} \in \mathbb{R}^{N_1 \times \dots \times N_R \times M}$ is initially split into training and testing data. Next, we perform the k -fold cross validation, in which the training dataset is randomly partitioned into k equally sized sets such that, at each one of its k iterations, one set is used for validation, whereas the other $k - 1$ sets are used for training. Finally, the trained model performance is assessed on the testing set, which is used only once. Thus, the validation set is used for tuning model hyperparameters during the training phase, whereas the testing set provides an unbiased evaluation of the final trained model.
- **Dataset Preprocessing:** next, the training, validation and testing datasets are submitted to a preprocessing step, which includes data cleansing, feature scaling and label encoding. In data cleansing, instances with missing/corrupted values (NaN) and infinity values (Inf) are deleted from each dataset. Then, in feature scaling, dataset variables are normalized to the range between 0 and 1 such that features with lower values are not dominated by variables with higher order of magnitude. Finally, in label encoding, “DDoS attack” and “legitimate” labels are converted from categorical variables to the codes “1” and “0”, respectively.
- **Minibatch Splitting:** finally, the training dataset is split into H minibatches $\mathfrak{X}^h \in \mathbb{R}^{N_1 \times \dots \times N_R \times M^h}$ for $h = 1, \dots, H$ containing M^h instances each. If the training dataset size is not a multiple of M^h , random instances are selected from the training dataset and added into the last minibatch until such condition is valid.

5.2.2 Proposed TFMCB-CNN Scheme

This section presents the proposed Tensor Feature Map Concatenation Based CNN scheme for DDoS attack detection, which is depicted by the block diagram shown in Fig. 5.1. The architecture is composed by four blocks, namely, Proposed Tensor Feature Map Concatenation Block, Final Concatenation, Flatten, and MLP. Such blocks are detailed in the following subsections.

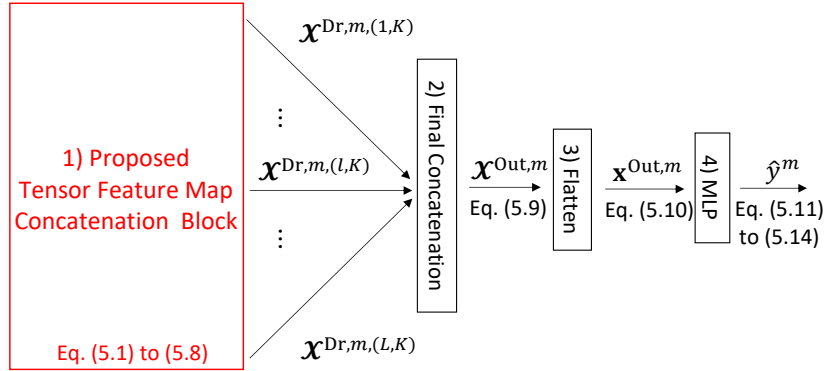


Figure 5.1 – Block diagram of the proposed TFMCB-CNN scheme.

5.2.2.1 Proposed Tensor Feature Map Concatenation Block

The proposed Tensor Feature Map Concatenation block is represented by Box 1 of Fig. 5.1, highlighted in red color. In addition, Fig. 5.2 details the inner components of the proposed block, which consists of multiple parallel branches B_l for $l = 1, \dots, L$. Each branch is composed of K CNN basic building blocks, labeled as $CNN_{l,k}$ for $l = 1, \dots, L$ and $k = 1, \dots, K$, alternately positioned with $K - 1$ feature map concatenation blocks, denoted as $conc_{l,k}$ for $l = 1, \dots, L$ and $k = 1, \dots, K - 1$. The CNN blocks are represented by Boxes a.1, \dots , a. K , whereas the concatenation blocks are depicted in Boxes b.1, \dots , b.($K - 1$) of Fig. 5.2. Additionally, a zoomed view of the inputs and outputs of the $CNN_{l,k}$ and $conc_{l,k}$ blocks is also shown at the bottom of Fig. 5.2. In our proposed Tensor Feature Map Concatenation block, the outputs from CNNs of consecutive branches are concatenated and sent to the following CNN within each branch. At the end, the branch outputs are sent to Block 2 of Fig. 5.1 for concatenation. More details regarding the operations performed on the CNN and concatenation blocks are shown in items a) and b) as follows.

a) CNN Basic Building Block, $CNN_{l,k}$

The CNN basic building blocks, $CNN_{l,k}$ for $l = 1, \dots, L$ and $k = 1, \dots, K$, are represented by Boxes a.1 to a. K of Fig. 5.2. Additionally, Fig. 5.3 illustrates the inner components of the $CNN_{l,k}$ block, including their inputs and outputs. Such components, namely, Convolution, Dropout, Batch Normalization, Activation and Pooling, are represented by

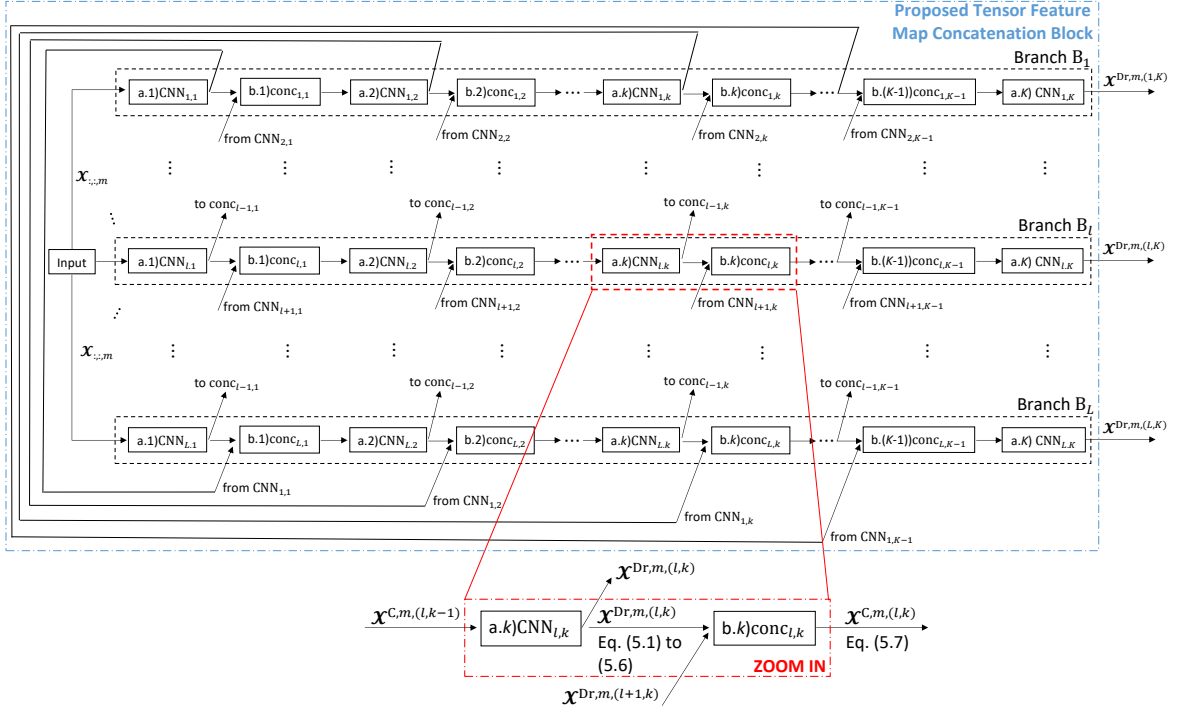


Figure 5.2 – Detailed view of the proposed Tensor Feature Map Concatenation block, including a zoomed view of the inputs and outputs of the $\text{CNN}_{l,k}$ and $\text{conc}_{l,k}$ blocks at the bottom.

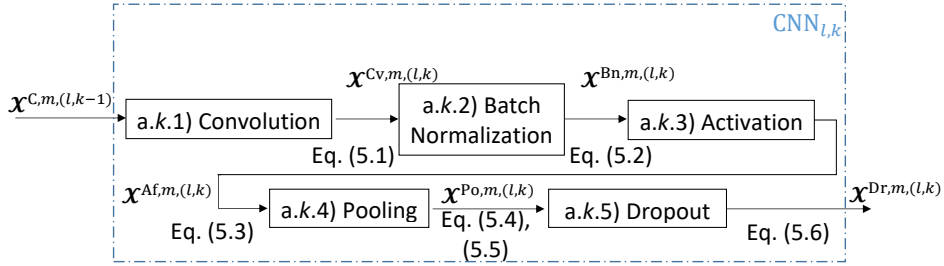


Figure 5.3 – Inputs and outputs of the Convolution, Batch Normalization, Activation, Pooling and Dropout blocks within $\text{CNN}_{l,k}$.

Boxes $a.k.1$ to $a.k.5$ of Fig. 5.3 and described in the following items.

- **Convolution:** This operation is represented by Box $a.k.1$ of Fig. 5.3. Before detailing the convolution operation performed on the input data, the concepts of padding and stride should be introduced. In deep learning architectures, the usage of small convolution filters lead to the lose of several data features, especially when a large number of convolutional layers is applied. To avoid this, padding is applied such that extra features are filled around each dataset matrix. Padding is represented by the parameter $p^{(l,k)}$, which denotes the number of elements added on each of the four sides of the data matrix at $\text{CNN}_{l,k}$. On the other hand, stride, represented as $s^{(l,k)}$, is the step of the convolutional product, i.e., the amount of shifts performed by the convolution window over the input matrix.

In this chapter, two-dimensional convolutional layers are considered due to the lower processing time compared to higher order convolutional computations, which is a fundamental requirement for network intrusion attack detection. In this sense, the input data at each branch B_l corresponds to the m -th dataset sample, i.e., $\mathbf{X}_{:,:,m} \in \mathbb{R}^{N_1 \times N_2}$ for $m = 1, \dots, M$, where $N_1 \cdot N_2 = N$. In addition, the input tensor at $\text{CNN}_{l,k}$ is given by $\mathbf{X}^{\text{C},m,(l,k-1)} \in \mathbb{R}^{N_1^{(l,k-1)} \times N_2^{(l,k-1)} \times N_c^{(l,k-1)}}$, where $N_1^{(l,k-1)}$ and $N_2^{(l,k-1)}$ are the feature map dimensions, and $N_c^{(l,k-1)}$ is the number of channels. Furthermore, if $\mathfrak{F}^{(l,k)} \in \mathbb{R}^{F_1^{(l,k)} \times F_2^{(l,k)} \times N_c^{(l,k-1)}}$ is the convolution kernel tensor, the (q, r, n_c) -th element x_{q,r,n_c}^{Cv} of the output tensor $\mathbf{X}^{\text{Cv},m,(l,k)} \in \mathbb{R}^{N_1^{(l,k)} \times N_2^{(l,k)} \times N_c^{(l,k)}}$ is given by

$$x_{q,r,n_c}^{\text{Cv},m,(l,k)} = \sum_{i=1}^{N_1^{(l,k-1)}} \sum_{j=1}^{N_2^{(l,k-1)}} \sum_{g=1}^{N_c^{(l,k-1)}} f_{i,j,g}^{(l,k)} x_{q+i-1,r+j-1,g}^{\text{C},m,(l,k-1)} + b_{n_c}^{(l,k)}, n_c = 1, \dots, N_c^{(l,k)}, \quad (5.1)$$

where $b_{n_c}^{(l,k)}$ is the bias, and $N_1^{\text{Cv},(l,k)} = \lfloor (N_1^{(l,k-1)} + 2p^{(l,k)} - F_1^{(l,k)})/s^{(l,k)} + 1 \rfloor$ and $N_2^{\text{Cv},(l,k)} = \lfloor (N_2^{(l,k-1)} + 2p^{(l,k)} - F_2^{(l,k)})/s^{(l,k)} + 1 \rfloor$ are the height and width of the feature map $\mathbf{X}_{:,:,n_c}^{\text{Cv},m,(l,k)}$ for $n_c = 1, \dots, N_c^{(l,k)}$. Moreover, $f_{i,j,g}^{(l,k)}$ is the (i, j, g) -th element of $\mathfrak{F}^{(l,k)}$ and is learned during the process of forward and backward propagation. Particularly at the first CNN of each branch, i.e., $\text{CNN}_{l,1}$ for $l = 1, \dots, L$, the input data corresponds to the m -th dataset instance, i.e., $\mathbf{X}^{\text{C},m,(l,1)} = \mathbf{X}_{:,:,m} \in \mathbb{R}^{N_1 \times N_2}$ for $m = 1, \dots, M^h$.

- **Batch Normalization:** this operation, represented by Box a.k.2 of Fig. 5.3, acts as a regularizer and allows much higher learning rates without the risk of divergence [157]. If $\mathbf{X}^{\text{Cv},(l,k),[h]} = [\mathbf{X}^{\text{Cv},1,(l,k)} | \dots | \mathbf{X}^{\text{Cv},M^h,(l,k)}]_4$ is the h -th minibatch tensor composed by M^h tensors $\mathbf{X}^{\text{Cv},m,(l,k)}$ stacked along the 4th dimension, its mean and standard deviation tensors can be computed as $\hat{\boldsymbol{\mu}}_h = (1/M^h) \sum_{m=1}^{M^h} \mathbf{X}^{\text{Cv},m,(l,k)}$ and $\hat{\boldsymbol{\sigma}}_h = \sqrt{(1/M^h) \sum_{m=1}^{M^h} (\mathbf{X}^{\text{Cv},m,(l,k)} - \hat{\boldsymbol{\mu}}_h)^2}$, respectively. Thus, the batch-normalized tensor $\mathbf{X}^{\text{Bn},(l,k),[h]} \in \mathbb{R}^{N_1^{\text{Cv},(l,k)} \times N_2^{\text{Cv},(l,k)} \times N_c^{(l,k)} \times M^h}$ can be expressed as

$$\begin{aligned} \mathbf{X}^{\text{Bn},(l,k),[h]} &= \gamma^{(l,k),[h]} \left(\frac{\mathbf{X}^{\text{Cv},(l,k),[h]} - \hat{\boldsymbol{\mu}}_h}{\sqrt{\hat{\boldsymbol{\sigma}}_h^2 + \epsilon}} \right) + \beta^{(l,k),[h]} \\ &= [\mathbf{X}^{\text{Bn},1,(l,k)} | \dots | \mathbf{X}^{\text{Bn},M^h,(l,k)}]_4, \end{aligned} \quad (5.2)$$

where $\gamma^{(l,k),[h]}$ and $\beta^{(l,k),[h]}$ are the scaling and shifting parameters at $\text{CNN}_{l,k}$ for the h -th minibatch, respectively, and ϵ is a constant included for numerical stability [157]. The variables $\gamma^{(l,k),[h]}$ and $\beta^{(l,k),[h]}$ are learned during the forward and backward prop-

agation, jointly with the other parameters of the model.

- **Activation:** after the batch normalization, $\mathbf{X}^{\text{Bn},m,(l,k)} \in \mathbb{R}^{N_1^{\text{Cv},(l,k)} \times N_2^{\text{Cv},(l,k)} \times N_c^{(l,k)}}$ for $m = 1, \dots, M^h$ is forwarded to the Activation block, represented by Box a.k.3 of Fig. 5.3. In this block, an activation function is attached to each neuron of the network. Such function determines whether the neuron should be activated or not, based on the relevance of each neuron input for the model prediction. In this chapter, the Rectified Linear Unit (ReLU), defined as $\psi(\cdot) = \max(0; \cdot)$, is chosen as activation function, since it is computationally efficient and presents a derivative function, allowing back-propagation.

Thus, the tensor $\mathbf{X}^{\text{Af},m,(l,k)} \in \mathbb{R}^{N_1^{\text{Cv},(l,k)} \times N_2^{\text{Cv},(l,k)} \times N_c^{(l,k)}}$ obtained by applying ReLU on $\mathbf{X}_{:::,n_c}^{\text{Bn},m,(l,k)}$ for $n_c = 1, \dots, N_c^{(l,k)}$ is given by

$$\mathbf{X}^{\text{Af},m,(l,k)} = [\psi(\mathbf{X}_{:::,1}^{\text{Bn},m,(l,k)}) | \dots | \psi(\mathbf{X}_{:::,N_c^{(l,k)}}^{\text{Bn},m,(l,k)})]_3. \quad (5.3)$$

- **Pooling:** this block, represented by Box a.k.4 of Fig. 5.3, consists of a pooling window slid over the input tensor $\mathbf{X}^{\text{Af},m,(l,k)}$ such that a single output is computed from the elements within that window. Differently from the Convolution block, there are no parameters in the pooling layer, and some specific function, for example the average pooling (Avg Pool) or max pooling (Max Pool), is applied on the elements ranged by the pooling window. The main advantage of pooling is to prevent overfitting by reducing the spatial size of the CNN.

The tensor $\mathbf{X}^{\text{Po},m,(l,k)} \in \mathbb{R}^{N_1^{\text{Po},(l,k)} \times N_2^{\text{Po},(l,k)} \times N_c^{(l,k)}}$ obtained by applying the pooling function $\phi(\cdot)$ on $\mathbf{X}_{:::,n_c}^{\text{Af},m,(l,k)}$ for $n_c = 1, \dots, N_c^{(l,k)}$ is given by

$$\mathbf{X}^{\text{Po},m,(l,k)} = [\phi(\mathbf{X}_{:::,1}^{\text{Af},m,(l,k)}) | \dots | \phi(\mathbf{X}_{:::,N_c^{(l,k)}}^{\text{Af},m,(l,k)})]_3, \quad (5.4)$$

where $N_1^{\text{Po},(l,k)} = \lfloor (N_1^{\text{Cv},(l,k-1)} + 2p^{(l,k)} - E_1^{(l,k)}) / s^{(l,k)} + 1 \rfloor$ and $N_2^{\text{Po},(l,k)} = \lfloor (N_2^{\text{Cv},(l,k-1)} + 2p^{(l,k)} - E_2^{(l,k)}) / s^{(l,k)} + 1 \rfloor$ are the height and width of $\mathbf{X}_{:::,n_c}^{\text{Po},m,(l,k)}$ for $n_c = 1, \dots, N_c^{(l,k)}$, and $E_1^{(l,k)}$ and $E_2^{(l,k)}$ denote the height and width of the pooling window.

If Avg Pool is considered as pooling function, the (q, r, n_c) -th element of $\mathbf{X}^{\text{Po},m,(l,k)} \in \mathbb{R}^{N_1^{\text{Po},(l,k)} \times N_2^{\text{Po},(l,k)} \times N_c^{(l,k)}}$ can be expressed as

$$x_{q,r,n_c}^{\text{Po},m,(l,k)} = \frac{1}{E_1^{(l,k)} E_2^{(l,k)}} \sum_{i=1}^{E_1^{(l,k)}} \sum_{j=1}^{E_2^{(l,k)}} x_{q+i-1,r+j-1,n_c}^{\text{Af},m,(l,k)}, \quad (5.5)$$

$$q = 1, \dots, N_1^{\text{Cv},(l,k)}, r = 1, \dots, N_2^{\text{Cv},(l,k)},$$

$$n_c = 1, \dots, N_c^{(l,k)},$$

where $x_{q+i-1, r+j-1, n_c}^{\text{Af}, m, (l, k)}$ is the $(q+i-1, r+j-1, n_c)$ -th element of the tensor $\mathfrak{X}^{\text{Af}, m, (l, k)}$. On the other hand, if Max Pool is adopted, the (q, r, n_c) -th element of $\mathfrak{X}^{\text{Po}, m, (l, k)}$ is given by

$$\begin{aligned} x_{q, r, n_c}^{\text{Po}, m, (l, k)} &= \max_{i=1, j=1}^{E_1^{(l, k)}, E_2^{(l, k)}} (x_{q+i-1, r+j-1, n_c}^{\text{Af}, m, (l, k)}), \\ q &= 1, \dots, N_1^{\text{Cv}, (l, k)}, r = 1, \dots, N_2^{\text{Cv}, (l, k)}, \\ n_c &= 1, \dots, N_c^{(l, k)}. \end{aligned} \quad (5.6)$$

- **Dropout:** represented by Box a.k.5 of Fig. 5.3, it is a technique applied to each element of the input tensor $\mathfrak{X}^{\text{Po}, m, (l, k)}$ to prevent overfitting. To accomplish this, some elements are randomly set to zero, or dropped, with rate p . In order to keep the sum over all of the components of $\mathfrak{X}^{\text{Po}, m, (l, k)}$ unchanged, elements not set to zero are scaled up by $1/(1-p)$. In addition, dropout is applied only during the training phase.

The tensor $\mathfrak{X}^{\text{Dr}, m, (l, k)} \in \mathbb{R}^{N_1^{\text{Po}, (l, k)} \times N_2^{\text{Po}, (l, k)} \times N_c^{(l, k)}}$ obtained after applying dropout on the elements of $\mathfrak{X}^{\text{Po}, m, (l, k)}$ can be expressed as

$$\mathfrak{X}^{\text{Dr}, m, (l, k)} = \mathfrak{V}^{(l, k)} \odot \mathfrak{X}^{\text{Po}, m, (l, k)}, \quad (5.7)$$

where $\mathfrak{V}^{(l, k)}$ is a tensor composed by independent Bernoulli random variables which take the value 1 with probability p [158].

b) Concatenation Block, $\text{conc}_{l, k}$

The concatenation blocks, $\text{conc}_{l, k}$ for $l = 1, \dots, L$ and $k = 1, \dots, K-1$, are represented by Boxes b.1 to b.($K-1$) of Fig. 5.2. Each $\text{conc}_{l, k}$ concatenates the output tensors from the $\text{CNN}_{l, k}$ and $\text{CNN}_{l+1, k}$ blocks for $l = 1, \dots, L-1$ along the 3rd dimension. Particularly when $l = L$, $\text{conc}_{L, k}$ concatenates the outputs from $\text{CNN}_{1, k}$ and $\text{CNN}_{L, k}$ such that the network forms a concatenation loop between consecutive branches. As pointed out by [150], the main idea behind the concatenation process is to enrich feature diversity such that the classifier can achieve better recognition ability. The concatenation process is adopted along the channel dimension and, consequently, the feature maps generated by possibly different filter sizes are concatenated such that the selection of an effective filter size is not needed.

The output $\mathfrak{X}^{\text{C}, (l, k)} \in \mathbb{R}^{N_1^{\text{Po}, (l, k)} \times N_2^{\text{Po}, (l, k)} \times (N_c^{(l, k)} + N_c^{(l+1, k)})}$ from $\text{conc}_{l, k}$ can be expressed as

$$\begin{aligned} \mathfrak{X}^{\text{C}, m, (l, k)} &= [\mathfrak{X}^{\text{Dr}, m, (l, k)} \mathfrak{X}^{\text{Dr}, m, (v, k)}]_3, \\ l &= 1, \dots, L, \quad v = l \bmod L + 1, \end{aligned} \quad (5.8)$$

where mod stands for the modulo operation.

5.2.2.2 Final Concatenation

The final concatenation process is represented by Box 2 of Fig. 5.1. In this block, the output tensors from each parallel branch, given by $\mathcal{X}^{\text{Dr},m,(l,K)}$ for $l = 1, \dots, L$, are concatenated along the 3rd dimension into the tensor $\mathcal{X}^{\text{Out},m} \in \mathbb{R}^{N_1^{\text{Po},(L,K)} \times N_2^{\text{Po},(L,K)} \times \sum_{l=1}^L N_c^{(l,K)}}$ as follows

$$\mathcal{X}^{\text{Out},m} = [\mathcal{X}^{\text{Dr},m,(1,K)} \mid \dots \mid \mathcal{X}^{\text{Dr},m,(L,K)}]_3. \quad (5.9)$$

5.2.2.3 Flatten

Since the input data of multilayer perceptrons (MLP) is one-dimensional, the concatenated tensor $\mathcal{X}^{\text{Out},m}$ in (5.9) must be flattened into the vector form. Therefore, the flattening operation, represented by Box 3 of Fig. 5.1, converts $\mathcal{X}^{\text{Out},m}$ into the vector $\mathbf{x}^{\text{Out},m}$ as follows

$$\mathbf{x}^{\text{Out},m} = \text{vec}(\mathcal{X}^{\text{Out},m}). \quad (5.10)$$

5.2.2.4 MLP

The multilayer perceptron, represented by Box 4 of Fig. 5.1, is the final step of the feature map extraction and classification phase. Fig. 5.4 illustrates the inner components of the MLP, namely, $D + 1$ fully connected dense layers and Softmax function, represented by Boxes 4.1 to 4.($D + 2$). In an MLP, the output of the d -th layer, $\mathbf{z}^{[d]} \in \mathbb{R}^{N^{[d]}}$, can be expressed as

$$\mathbf{z}^{[d]} = \psi^{[d]}(\mathbf{W}^{[d,d-1]} \cdot \mathbf{z}^{[d-1]} + \mathbf{b}^{[d]}), \quad (5.11)$$

where $\mathbf{W}^{[d,d-1]} \in \mathbb{R}^{N^{[d]} \times N^{[d-1]}}$ is the weight matrix between the $(d - 1)$ -th and d -th dense layers. Furthermore, $\mathbf{b}^{[d]} \in \mathbb{R}^{N^{[d]}}$, $\psi^{[d]}(\cdot) = \max(0; \cdot)$ and $N^{[d]}$ are the bias vector, the activation function and the number of neurons of the d -th dense layer, respectively.

The first dense layer corresponds to the MLP input layer, which receives the output vector from Block 3, i.e., $\mathbf{z}^{[1]} = \mathbf{x}^{\text{Out},m}$. Since we are dealing with binary classification, the number of neurons of the $(D + 1)$ -th layer is $N^{[D+1]} = 2$ and, consequently, $\mathbf{z}^{[D+1]} = [z_1^{[D+1]}, z_2^{[D+1]}]^\top$, as it can be seen in Box 4.($D + 1$) of Fig. 5.4. In addition, the dataset instance $\mathcal{X}_{:,:,m}$ is classified as legitimate traffic or DDoS attack by applying the Softmax function, represented in Box 4.($D + 2$) of Fig. 5.4, on the input vector $\mathbf{z}^{[D+1]}$.

Therefore, given $\mathbf{z}^{[D+1]}$, the probability that $\mathcal{X}_{:,:,m}$ is classified as $\hat{y}^m = u$ can be ex-

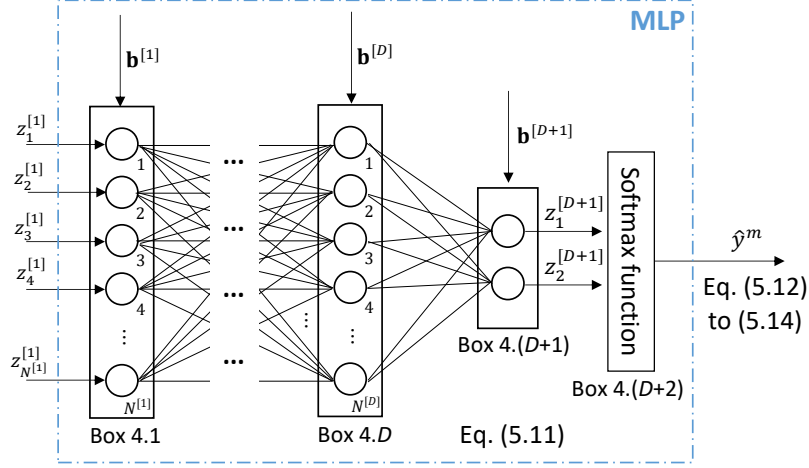


Figure 5.4 – Dense layers and Softmax function within the MLP block.

pressed as

$$P(\hat{y}^m = u | \mathbf{z}^{[D+1]}) = P(\hat{y}^m) = \frac{e^{z_{u+1}^{[D+1]}}}{e^{z_1^{[D+1]}} + e^{z_2^{[D+1]}}}, \quad (5.12)$$

$$u = 0, 1.$$

Finally, the predicted class \hat{y}^m corresponds to the class u for which $P(\hat{y}^m = u | \mathbf{z}^{[D+1]})$ is maximum, i.e.,

$$\hat{y}^m = \underset{u}{\operatorname{argmax}} \{P(\hat{y}^m = u | \mathbf{z}^{[D+1]})\}. \quad (5.13)$$

During the training phase, the aim is to minimize a cost function J by applying the gradient descent method. At each iteration, the m -th training instance is feedforwarded through the layers of the model and the error between the expected and predicted classes computed over a batch of M^h instances is backpropagated through the network. The weights and biases of the model are updated by the backpropagation algorithm according to the partial derivatives of J [110]. Such process goes on until a predetermined number of epochs is reached, or the computed error is below a given threshold [125]. Such error can be calculated by using the well-known binary cross-entropy cost function, given by

$$J = -\frac{1}{M^h} \sum_{m=1}^{M^h} [(y^m \log(P(\hat{y}^m))) + (1 - y^m) \log(1 - P(\hat{y}^m))], \quad (5.14)$$

where y^m is the true class of the m -th instance.

On the other hand, during the testing phase, the m -th testing instance is feedforwarded

through the layers of the trained model such that its predicted label \hat{y}^m is computed as in (5.13).

5.2.3 Proposed TFMCB-CNN Scheme Improved Via Majority Voting

This section introduces a refined version of the technique proposed in Subsection 5.2.2, called Tensor Feature Map Concatenation Based CNN scheme Improved via Majority Voting (TFMCB-CNN-MV), as depicted in Fig. 5.5. Differently from TFMCB-CNN, which presented a final concatenation module followed by Flatten and MLP blocks in common for all of the parallel branches, each branch B_l is followed by its respective flattening and multilayer perceptron blocks, Flatten_l and MLP_l for $l = 1, \dots, L$. At the end, the outputs from each MLP_l are sent to a Simple Majority Voting module for final classification.

Fig. 5.5 illustrates the block diagram of the proposed TFMCB-CNN-MV scheme for DDoS attack detection. The architecture is composed by four blocks, namely, Proposed Tensor Feature Map Concatenation Block, Flatten, MLP, and SMV, represented by Boxes 1 to 4. Since Box 1 is identical to the one described in Subsection 5.2.2.1, only the Flatten and MLP blocks of each branch B_l for $l = 1, \dots, L$, as well as the SMV module, are detailed in the following subsections.

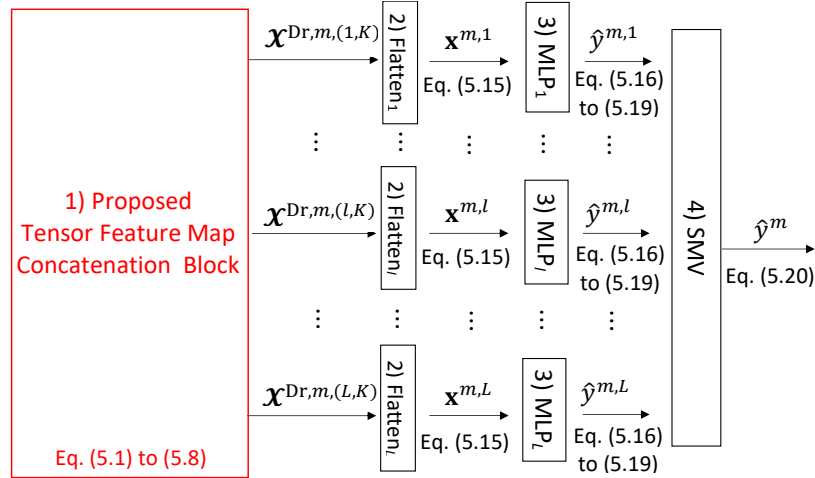


Figure 5.5 – Block diagram of the proposed TFMCB-CNN-MV scheme.

5.2.3.1 Flatten

The flattening operation is represented by Box 2 of Fig. 5.5. The output tensors from each branch, denoted as $\mathcal{X}^{\text{Dr},m,(l,K)}$, are sent to their respective flattening blocks Flatten_l for $l = 1, \dots, L$. In this sense, the outputs from each Flatten_l block correspond to the tensor

$\mathbf{x}^{\text{Dr},m,(l,K)}$ in the vector form, i.e.,

$$\begin{aligned}\mathbf{x}^{m,l} &= \text{vec}(\mathbf{x}^{\text{Dr},m,(l,K)}), \\ l &= 1, \dots, L.\end{aligned}\tag{5.15}$$

5.2.3.2 MLP

The MLP is represented by Box 3 of Fig. 5.5. Initially, the vectors $\mathbf{x}^{m,l}$ from the flattening blocks are forwarded to each MLP_l for $l = 1, \dots, L$. Similarly to the MLP block described in Subsection 5.2.2.4, each MLP_l contains $D + 1$ fully connected dense layers, plus the Softmax function for classification. Therefore, the output of the d -th layer, $\mathbf{z}^{[d],l} \in \mathbb{R}^{N^{[d],l}}$, can be denoted as

$$\mathbf{z}^{[d],l} = \psi^{[d],l}(\mathbf{W}^{[d,d-1],l} \cdot \mathbf{z}^{[d-1],l} + \mathbf{b}^{[d],l}),\tag{5.16}$$

where $\mathbf{W}^{[d,d-1],l} \in \mathbb{R}^{N^{[d],l} \times N^{[d-1],l}}$ is the weight matrix between the $(d - 1)$ -th and d -th dense layers of MLP_l . In addition, $\mathbf{b}^{[d],l} \in \mathbb{R}^{N^{[d],l}}$, $\psi^{[d],l}(\cdot) = \max(0; \cdot)$ and $N^{[d],l}$ are the bias vector, the activation function and the number of neurons of the d -th dense layer of MLP_l , respectively.

Analogously to (5.12), the probability that $\mathbf{x}_{:,:,m}$ is classified as $\hat{y}^{m,l} = u$ at the MLP_l block can be denoted as

$$\begin{aligned}P(\hat{y}^{m,l} = u | \mathbf{z}^{[D+1],l}) &= P(\hat{y}^{m,l}) = \frac{e^{z_{u+1}^{[D+1],l}}}{e^{z_1^{[D+1],l}} + e^{z_2^{[D+1],l}}}, \\ u &= 0, 1.\end{aligned}\tag{5.17}$$

Further, the label $\hat{y}^{m,l}$ predicted at MLP_l corresponds to the class u for which $P(\hat{y}^{m,l} = u | \mathbf{z}^{[D+1],l})$ is maximum, i.e.,

$$\hat{y}^{m,l} = \underset{u}{\text{argmax}}\{P(\hat{y}^{m,l} = u | \mathbf{z}^{[D+1],l})\}.\tag{5.18}$$

The training and testing phases are similar to the ones described for the proposed TFMCB-CNN scheme in Subsection 5.2.2.4. However, each MLP_l for $l = 1, \dots, L$ presents its

respective binary cross-entropy cost function, given by

$$J^l = -\frac{1}{M^h} \sum_{m=1}^{M^h} [(y^m \log(P(\hat{y}^{m,l})) + (1 - y^m) \log(1 - P(\hat{y}^{m,l}))], \quad (5.19)$$

such that the forward and backward propagations are independently performed through each MLP_{*l*}.

5.2.3.3 SMV Block

Finally, the predicted classes $\hat{y}^{m,l}$ for $l = 1, \dots, L$ are sent to the Simple Majority Voting block, which is represented by Box 4 of Fig. 5.5. The final class label \hat{y}^m for $m = 1, \dots, M^h$ corresponds to the majority of the class labels predicted by all of the L MLPs, i.e.,

$$\hat{y}^m = \text{mode}\{\hat{y}^{m,1}, \dots, \hat{y}^{m,L}\}, \quad (5.20)$$

where $\text{mode}\{\cdot\}$ is the mode function, which returns the most frequently predicted class among the $\hat{y}^{m,l}$ for $l = 1, \dots, L$.

Therefore, from the proposed schemes presented so far, it can be highlighted that a better performance for DDoS attack detection can be achieved by tuning several model settings, for example, number of parallel branches, number of MLP dense layers and convolution filter size, in contrast to several deep learning NIDS with fixed parameters commonly found in the literature.

5.3 COMPUTATIONAL COMPLEXITY

This section discusses the computational complexity of the proposed TFMCB-CNN and TFMCB-CNN-MV schemes presented in Section 5.2. We provide an analysis of the most costly calculations as a function of the largest contributions of the most important variables of the models. Furthermore, for the L multiple parallel branches, it is considered a parallel computation, where n_p is the number of processors of the NIDS host machine. The computational complexity related to the most time costly layers are shown in the following items, namely, Convolution and Dense layers. Despite it presents low computational cost, the Simple Majority Voting block is also evaluated.

- **Convolution Layer:** in this layer, a group of n_f convolution filters of dimension $f \times f$ are slid over n_c feature maps with size $n \times n$. If p and s are the padding and stride, respectively, the dimensions of the output feature map are given by $n_{\text{out}} \times n_{\text{out}}$, where

$n_{\text{out}} = (n - f + 2p)/s + 1$. To calculate the input to the activation function for each element in the output feature maps, $n_c f^2$ multiply-accumulate operations are required [159]. Since there are K Convolution blocks within each branch, and considering a batch size of M^h , the total computational complexity related to the Convolution layer for the TFMCB-CNN and TFMCB-CNN-MV schemes are given by

$$\mathcal{O}[\text{TFMCB-CNN}_{\text{Cv}}] = \mathcal{O} [K M^h n_c f^2 n_f n_{\text{out}}^2], \quad (5.21)$$

$$\mathcal{O}[\text{TFMCB-CNN-MV}_{\text{Cv}}] = \mathcal{O} [(L/n_p) K M^h n_c f^2 n_f n_{\text{out}}^2], \quad (5.22)$$

where (L/n_p) is the term corresponding to the parallel computation, defined as the ratio between the number of branches and the number of processors.

- **Dense Layers:** each MLP is composed by $D+1$ dense layers, with complexity given by $\mathcal{O}[\sum_{d=1}^D N^{[d]} N^{[d+1]}]$, which is determined by the weight matrix $\mathbf{W}^{[d,d-1],l} \in \mathbb{R}^{N^{[d],l} \times N^{[d-1],l}}$ between the d -th and $(d+1)$ -th layers. Thus, the total time complexity related to the Dense Layers for the TFMCB-CNN and TFMCB-CNN-MV schemes can be denoted as

$$\mathcal{O}[\text{TFMCB-CNN}_{\text{MLP}}] = \mathcal{O} \left[M^h \left(\sum_{d=1}^D N^{[d]} N^{[d+1]} \right) \right], \quad (5.23)$$

$$\mathcal{O}[\text{TFMCB-CNN-MV}_{\text{MLP}}] = \mathcal{O} \left[(L/n_p) M^h \left(\sum_{d=1}^D N^{[d]} N^{[d+1]} \right) \right]. \quad (5.24)$$

- **Simple Majority Voting:** in this block, the $\text{mode}\{\cdot\}$ function is applied on the predicted class labels $\hat{y}^{m,l}$ for $l = 1, \dots, L$ forwarded by all of the MLP blocks. Thus, the total time complexity related to the SMV block in the TFMCB-CNN-MV scheme is given by

$$\mathcal{O}[\text{TFMCB-CNN-MV}_{\text{SMV}}] = \mathcal{O} [L M^h]. \quad (5.25)$$

In Section 5.4, the proposed schemes are compared with three different CNN based models, namely, Parallel-CNN-MV, Parallel-CNN and Serial-CNN. The two first models are similar to the TFMCB-CNN-MV and TFMCB-CNN schemes, but without the $\text{conC}_{1,k}$ blocks such that there is no concatenation between CNN outputs from consecutive branches. Additionally, the third model, Serial-CNN, is equivalent to the Parallel-CNN scheme, but composed by a single branch. More details regarding such competitor techniques are shown in Section 5.4.

Table 5.2 summarizes the computational complexity for each layer of the compared approaches. It can be observed that the parameter L/n_p is fundamental in schemes in which parallel computations are performed. If $L \gg n_p$, the computational complexity of the model is increased, which reflects the importance of a higher number of processors in the IDS host machine. Furthermore, the convolution filter size, as well as the number of CNNs, present a direct impact on the time cost of the models. In this sense, a trade-off between the more accurate DDoS attack detection provided by larger filter sizes and the computational complexity must be observed, as shown in Section 5.4. Moreover, the time cost also depends on the number of input feature maps n_c . In the proposed TFMCB-CNN-MV and TFMCB-CNN schemes, n_c is increased after each concatenation operation, since the number of input feature maps at the $\text{CNN}_{l,k+1}$ block corresponds to the sum between the number of output feature maps from the $\text{CNN}_{l,k}$ and $\text{CNN}_{l+1,k}$ blocks. In this sense, the number of feature maps in TFMCB-CNN-MV and TFMCB-CNN is represented by n_c^* . On the other hand, in the competitor methods, the number of input feature maps at the $\text{CNN}_{l,k+1}$ block is equal to the number of convolution filters of the previous CNN block, $\text{CNN}_{l,k}$. Therefore, the proposed TFMCB-CNN-MV and TFMCB-CNN schemes present higher computational complexity compared to the other approaches especially due to the concatenation process, which increases the number of feature maps throughout each branch B_l for $l = 1, \dots, L$. Finally, from Table 5.2, it is clear that the Single-CNN scheme shows the lowest computational complexity among all of the compared architectures, since it is composed by a single branch, without concatenation operations. Nonetheless, its lower time cost is accompanied by a worse attack detection performance, which is reinforced in Section 5.4.

Table 5.2 – Computational complexity of the following schemes: (1) Proposed TFMCB-CNN-MV, (2) Proposed TFMCB-CNN, (3) Parallel-CNN-MV, (4) Parallel-CNN, and (5) Serial-CNN.

Model	Convolution	Dense Layers	Simple Majority Voting
Proposed TFMCB-CNN-MV	$\mathcal{O}[(\frac{L}{n_p})KM^hn_c^*f^2n_f n_{\text{out}}^2]$	$\mathcal{O}[(\frac{L}{n_p})M^h(\sum_{d=1}^D N^{[d]}N^{[d+1]})]$	$\mathcal{O}[LM^h]$
Proposed TFMCB-CNN	$\mathcal{O}[(\frac{L}{n_p})KM^hn_c f^2n_f n_{\text{out}}^2]$	$\mathcal{O}[M^h(\sum_{d=1}^D N^{[d]}N^{[d+1]})]$	–
Parallel-CNN-MV	$\mathcal{O}[(\frac{L}{n_p})KM^hn_c f^2n_f n_{\text{out}}^2]$	$\mathcal{O}[(\frac{L}{n_p})M^h(\sum_{d=1}^D N^{[d]}N^{[d+1]})]$	$\mathcal{O}[LM^h]$
Parallel-CNN	$\mathcal{O}[(\frac{L}{n_p})KM^hn_c f^2n_f n_{\text{out}}^2]$	$\mathcal{O}[M^h(\sum_{d=1}^D N^{[d]}N^{[d+1]})]$	–
Serial-CNN	$\mathcal{O}[KM^hn_c f^2n_f n_{\text{out}}^2]$	$\mathcal{O}[M^h(\sum_{d=1}^D N^{[d]}N^{[d+1]})]$	–

5.4 SIMULATION RESULTS

This section is divided into four subsections. First, details about the features and samples extracted from the DDoS benchmark datasets used in this chapter are shown in Subsection 5.4.1. Next, simulation results are introduced in Subsection 5.4.2. Following, Subsection

5.4.3 presents considerations regarding the processing times of all of the compared schemes. Finally, Subsection 5.4.4 discusses the obtained results.

5.4.1 DDoS Attack Datasets

Two subsets composed by legitimate and DDoS attack samples extracted from the CIC-DDoS2019 and CIC-IDS2017 benchmark datasets are used in this chapter. Moreover, theoretical details about both datasets are shown in Appendix C. First, each subset is split into training and testing sets, with proportion 80:20. Next, a 5-fold cross validation is performed on the training dataset, as explained in Subsection 5.2.1. Finally, the trained classifier is evaluated on the testing set. The data customized from the CIC-DDoS2019 dataset is composed by $N = 64$ features and $M = 40,000$ instances, of which 32,000 are legitimate and 8,000 are DDoS attacks, namely, DNS, LDAP, MSSQL, NetBIOS, NTP, SNMP, SSDP and TFTP based attacks, as well as TCP SYN flood and UDP flood. Similarly, the data customized from the CIC-IDS2017 dataset is composed by the same number of instances and features as the subset created from the CIC-DDoS2019 dataset. Further, the features evaluated in this chapter are the same as those used in Chapters 3 and 4, which are shown in Table C.2 of Appendix C. In addition, details about the DDoS attack types and the corresponding number of instances extracted from the CIC-DDoS2019 and CIC-IDS2017 datasets are presented in Table 5.3. Finally, each customized dataset is folded as a three-dimensional tensor with size $N_1 \times N_2 \times M$, and divided into minibatches with size M^h , where $M^h = 128, \dots, 1024$. For simplicity, we set $N_1 = N_2 = 8$ such that the number of features is given by $N_1 \cdot N_2 = N = 64$.

Table 5.3 – DDoS attack types used in this chapter, as well as the corresponding number of instances extracted from the CIC-DDoS2019 and CIC-IDS2017 datasets.

Dataset	Traffic Type	Total
CICDDoS2019	Legitimate	32,000
	DNS-based DDoS	800
	LDAP-based DDoS	800
	MSSQL-based DDoS	800
	NetBIOS-based DDoS	800
	NTP-based DDoS	800
	SNMP-based DDoS	800
	SSDP-based DDoS	800
	UDP flood	800
	TCP SYN flood	800
	TFTP-based DDoS	800
CICIDS2017	Legitimate	32,000
	HTTP and UDP-based DDoS	8000

5.4.2 Results

This subsection presents the performance evaluation of the proposed TFMCB-CNN-MV and TFMCB-CNN schemes for DDoS attack detection through numerical simulations. Accuracy (Acc), Precision (Prec), Recall (Rec), False Positive Rate (FPR), False Negative Rate (FNR), F-Score and Matthews Correlation Coefficient (MCC) are the performance evaluation metrics used in this chapter, which are defined in Appendix D.

Simulations were executed on a desktop computer with processor Intel Core i7-2600 3.40 GHz, composed by 8 logic processors and 16 GB of RAM. Data preprocessing and deep learning algorithms were implemented by using the Scikit-Learn and Keras Python libraries. In addition, since the CIC-IDS2017 dataset is widely applied for NIDS validation in several state-of-the-art researches in the literature, it was used only for comparison with related works. On the other hand, the CIC-DDoS2019 dataset is extensively applied for model validation throughout this subsection, since it is a recent dataset and contains the most up-to-date DDoS attacks.

Tables 5.4 and 5.5 show the layer types, with their respective output shapes and simulation parameters, at each block of the proposed TFMCB-CNN-MV and TFMCB-CNN schemes, respectively. The convolution filter size ranges from 2×2 to 5×5 , the number of branches ranges from 2 to 5, the pooling filter size is 2×2 and the dropout rate is fixed in 0.2. Further, the number of dense layers varies from 1 to 5, the minibatch size varies between 128 and 1,024 and Avg Pool is adopted as the pooling function. Note that, in Tables 5.4 and 5.5, the tensor output shape of the $\text{conc}_{l,1}$ block, given by $4 \times 4 \times 32$, presents the double of the number of channels of the Dropout layer output. In this case, the outputs from $\text{conc}_{l,1}$ and $\text{conc}_{l+1,1}$, each of which with 16 channels, are concatenated along the 3rd dimension, generating a tensor with 32 channels. In addition, considering the $\text{CNN}_{l,1}$ block, the tensor output shape of the Pooling layer is $4 \times 4 \times 16$, i.e., each tensor frontal slice presents $4 \cdot 4 = 16$ features, which is one fourth of the number of features after Activation, with shape $8 \times 8 \times 16$, whose frontal slice has $8 \cdot 8 = 64$ features. Such fact is due to the Pooling Filter Size of 2×2 , which implies a feature reduction of $2 \cdot 2 = 4$ on the Activation layer output. In this sense, since each Pooling layer performs a feature reduction of $2 \cdot 2 = 4$ and the number of features of the input tensor is $8 \cdot 8 = 64$, we conclude that the maximum number of CNNs allowed for both models is 3. In this situation, the tensor output shape after the last CNN is given by $1 \times 1 \times 64$, i.e., the number of features of the tensor frontal slice is 1. In simulations, the number of CNNs is fixed in 3, whereas the number of convolution filters of the $\text{CNN}_{l,1}$, $\text{CNN}_{l,2}$ and $\text{CNN}_{l,3}$ blocks is given by 16, 32 and 64, respectively, as observed in Tables 5.4 and 5.5. Finally, all of the compared schemes are trained during a period of 100 epochs.

As stated in Section 5.3, the proposed schemes are compared with three competing ap-

Table 5.4 – Layer types, output shapes and simulation parameters at each block of the proposed TFMCB-CNN-MV scheme for $l = 1, \dots, L$.

Block	Layer Type	Output Shape	Simulation Parameters
Input	Input	$8 \times 8 \times 1$	–
$CNN_{l,1}$	Convolution	$8 \times 8 \times 16$	Nr Filters: 16 Convolution Filter Size: from 2×2 to 5×5 Padding: “same” Strides: 1
	Batch Normalization	$8 \times 8 \times 16$	–
	Activation	$8 \times 8 \times 16$	Function: ReLU
	Pooling	$4 \times 4 \times 16$	Functions: Avg Pool Pooling Filter Size: 2×2
	Dropout	$4 \times 4 \times 16$	Droprate: 0.2
$conc_{l,1}$	Concatenation	$4 \times 4 \times 32$	–
$CNN_{l,2}$	Convolution	$4 \times 4 \times 32$	Nr Filters: 32 Convolution Filter Size: from 2×2 to 5×5 Padding: “same” Strides: 1
	Batch Normalization	$4 \times 4 \times 32$	–
	Activation	$4 \times 4 \times 32$	Function: ReLU
	Pooling	$2 \times 2 \times 32$	Functions: Avg Pool Pooling Filter Size: 2×2
	Dropout	$2 \times 2 \times 32$	Droprate: 0.2
$conc_{l,2}$	Concatenation	$2 \times 2 \times 64$	–
$CNN_{l,3}$	Convolution	$2 \times 2 \times 64$	Nr Filters: 64 Convolution Filter Size: from 2×2 to 5×5 Padding: “same” Strides: 1
	Batch Normalization	$2 \times 2 \times 64$	–
	Activation	$2 \times 2 \times 64$	Function: ReLU
	Pooling	$1 \times 1 \times 64$	Functions: Avg Pool Pooling Filter Size: 2×2
	Dropout	$1 \times 1 \times 64$	Droprate: 0.2
Flatten _l	Flattening	64	–
MLP_l	Dense _{l,1}	70	Nr Neurons: 70
	Dense _{l,2}	50	Nr Neurons: 50
	Dense _{l,3}	30	Nr Neurons: 30
	Dense _{l,4}	10	Nr Neurons: 10
	Dense _{l,5}	2	Nr Neurons: 2 Classification Function: Softmax

proaches, namely, Parallel-CNN-MV, Parallel-CNN and Serial-CNN. The first two techniques are similar to the TFMCB-CNN-MV and TFMCB-CNN schemes, respectively, but without the concatenation blocks within each branch. Thus, by comparing each proposed scheme with the Parallel-CNN-MV and Parallel-CNN approaches, we aim to show the benefits of adopting feature map concatenation over the DDoS attack detection performance. Fig. 5.6a to 5.6d illustrate an example of the TFMCB-CNN-MV, TFMCB-CNN, Parallel-CNN-MV and Parallel-CNN schemes for detecting DDoS attacks, with $L = 3$ branches, $K = 3$ CNN basic building blocks and $D + 1 = 5$ MLP dense layers. By comparing Fig. 5.6a and 5.6c, as well as Fig. 5.6b and 5.6d, it is clear the absence of the concatenation blocks

Table 5.5 – Layer types, output shapes and simulation parameters at each block of the proposed TFMCB-CNN scheme for $l = 1, \dots, L$.

Block	Layer Type	Output Shape	Simulation Parameters
Input	Input	$8 \times 8 \times 1$	–
CNN _{<i>l</i>,1}	Convolution	$8 \times 8 \times 16$	Nr Filters: 16 Convolution Filter Size: from 2×2 to 5×5 Padding: “same” Strides: 1
	Batch Normalization	$8 \times 8 \times 16$	–
	Activation	$8 \times 8 \times 16$	Function: ReLU
	Pooling	$4 \times 4 \times 16$	Functions: Avg Pool Pooling Filter Size: 2×2
	Dropout	$4 \times 4 \times 16$	Droprate: 0.2
conc _{<i>l</i>,1}	Concatenation	$4 \times 4 \times 32$	–
CNN _{<i>l</i>,2}	Convolution	$4 \times 4 \times 32$	Nr Filters: 32 Convolution Filter Size: from 2×2 to 5×5 Padding: “same” Strides: 1
	Batch Normalization	$4 \times 4 \times 32$	–
	Activation	$4 \times 4 \times 32$	Function: ReLU
	Pooling	$2 \times 2 \times 32$	Functions: Avg Pool Pooling Filter Size: 2×2
	Dropout	$2 \times 2 \times 32$	Droprate: 0.2
conc _{<i>l</i>,2}	Concatenation	$2 \times 2 \times 64$	–
CNN _{<i>l</i>,3}	Convolution	$2 \times 2 \times 64$	Nr Filters: 64 Convolution Filter Size: from 2×2 to 5×5 Padding: “same” Strides: 1
	Batch Normalization	$2 \times 2 \times 64$	–
	Activation	$2 \times 2 \times 64$	Function: ReLU
	Pooling	$1 \times 1 \times 64$	Functions: Avg Pool Pooling Filter Size: 2×2
	Dropout	$1 \times 1 \times 64$	Droprate: 0.2
conc _{Out}	Concatenation	$1 \times 1 \times 320$	–
Flatten	Flattening	320	–
MLP	Dense ₁	70	Nr Neurons: 70
	Dense ₂	50	Nr Neurons: 50
	Dense ₃	30	Nr Neurons: 30
	Dense ₄	10	Nr Neurons: 10
	Dense ₅	2	Nr Neurons: 2 Classification Function: Softmax

in the Parallel-CNN and Parallel-CNN-MV techniques. At last, the Serial-CNN approach, represented in Fig. 5.6e, corresponds to the Parallel-CNN scheme with a single branch. By confronting Serial-CNN with the proposed schemes, our idea is to illustrate the benefits of adopting multiple parallel CNN branches. Throughout this section, all of the schemes are compared for different simulation parameters, namely, number of branches, number of dense layers, batch size and convolution filter size, as shown in Tables 5.6 to 5.9. Moreover, Tables 5.10 and 5.11 present the simulation results obtained from the performance evaluation of state-of-the-art machine learning and deep learning algorithms, as well as related works in

the literature, respectively. In each table, the best metric values for a fixed simulation parameter are highlighted in bold. Additionally, the performance evaluation metrics corresponding to the second best results are underlined. Note that Tables 5.6 to 5.10 also include the training and testing times recorded for all of the compared schemes during the experiments. Such processing times are detailed and discussed in Subsection 5.4.3.

Table 5.6 shows the simulation results of the proposed TFMCB-CNN-MV and TFMCB-CNN approaches against their competitor techniques when the number of branches is varied between 1 and 5. Furthermore, we consider a pooling filter size of 2×2 , batch size of 128, convolution filter size of 3×3 and single MLP dense layer. Particularly, when the number of branches is $L = 1$, only the Serial-CNN scheme is shown, since the other four approaches boil down to the single branch model. Note that the Serial-CNN is outperformed by almost all of the competing techniques, which shows the benefits of adopting multiple parallel CNN branches. From the results shown in Table 5.6, it is clear that the proposed TFMCB-CNN-MV scheme presents outstanding results, outperforming its competitors in all of the assessed metrics when the number of branches is larger than or equal to 4. For instance, when $L = 4$, the values of Acc, Prec and MCC of the TFMCB-CNN-MV scheme are given by 99.67%, 99.43% and 0.9896, respectively, whereas its concurrent, Parallel-CNN-MV, presented 99.55%, 99.21% and 0.9856 for the same metrics. On the other hand, the proposed TFMCB-CNN scheme presents considerable results when $2 \leq L \leq 3$, outperforming the Parallel-CNN-MV and Parallel-CNN approaches in several metrics. For example, for $L = 3$, the TFMCB-CNN model showed Prec, FNR, F-score and MCC of 99.36%, 0.26%, 99.39% and 0.9878, against 99.25%, 0.33%, 99.38% and 0.9876 achieved by the Parallel-CNN.

Next, the values of Acc, Prec, Rec, FPR, FNR, F-score and MCC considering different number of dense layers are shown in Table 5.7 for all of the compared approaches. The number of dense layers ranges from 1 to 5 and the parallel CNN based architectures are composed by 3 branches. Moreover, the pooling filter size is 2×2 , the convolution filter size is fixed in 3×3 and the batch size is set to 128. Once again, it is observed that the Serial-CNN approach is outperformed by all of the compared techniques in most of the evaluated metrics. Further, note that the proposed TFMCB-CNN-MV architecture outperforms the other schemes in most of the assessed metrics, regardless of the number of MLP dense layers. For example, when such model is composed by 5 dense layers, the TFMCB-CNN-MV approach showed Acc, Prec, Rec, F-score and MCC of 99.63%, 99.34%, 99.53%, 99.42% and 0.9883, respectively, against 99.48%, 99.04%, 99.32%, 99.18% and 0.9835 obtained when Parallel-CNN-MV is applied. Additionally, for 3 dense layers, the proposed TFMCB-CNN achieved Acc = 99.63%, Prec = 99.35%, Rec = 99.49%, F-score = 99.42% and MCC = 0.9885, whereas its concurrent, Parallel-CNN, presented 99.35%, 98.68%, 99.31%, 98.98% and 0.9798 for the same metrics, respectively.

Following, Table 5.8 shows the metric values of the proposed TFMCB-CNN-MV and

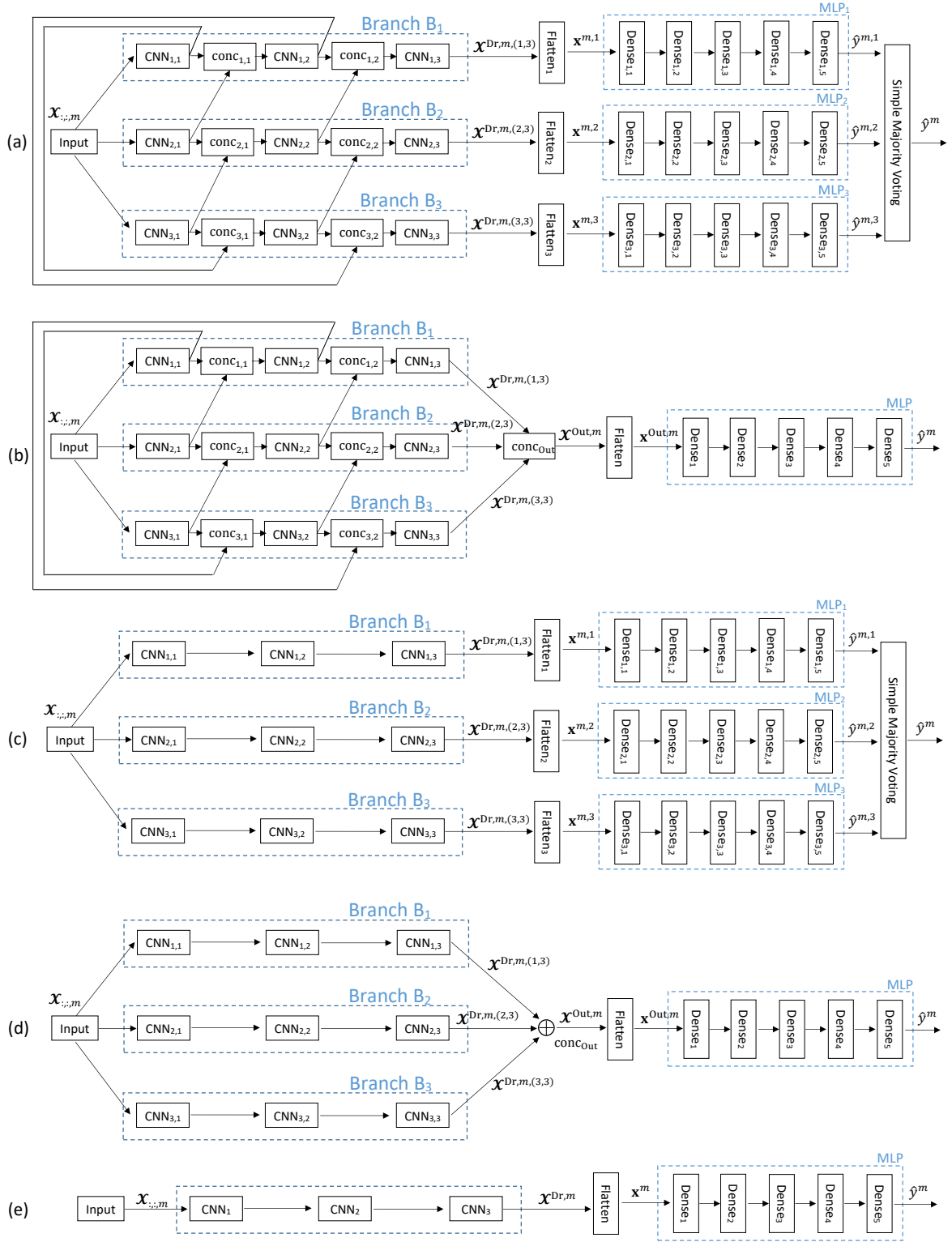


Figure 5.6 – Examples of block diagrams of the following models: (a) Proposed TFMCB-CNN-MV, (b) Proposed TFMCB-CNN, (c) Parallel-CNN-MV, (d) Parallel-CNN, and (e) Serial-CNN.

TFMCB-CNN schemes against their competitor techniques when the batch size ranges from 128 to 1,024. Furthermore, in simulations, we consider $L = 3$, pooling filter size of 2×2 , convolution filter size of 3×3 and single dense layer. It can be seen that the proposed

Table 5.6 – Performance evaluation metrics of the proposed TFMCB-CNN-MV and TFMCB-CNN schemes, as well as its competitor approaches, when the number of branches ranges from 1 to 5.

Nr Branch	Model	Acc	Prec	DR	FPR	FNR	F-Score	MCC	Tr T (s)	Te T (s)
1	Serial-CNN	0.9939	0.9911	0.9898	0.0170	0.0034	0.9904	0.9808	270.38	0.4664
	Proposed TFMCB-CNN-MV	0.9967	0.9951	<u>0.9949</u>	0.0094	0.0018	0.9947	0.9894	417.22	<u>0.3670</u>
2	Proposed TFMCB-CNN	<u>0.9961</u>	0.9926	0.9944	<u>0.0070</u>	0.0032	<u>0.9937</u>	<u>0.9875</u>	454.18	0.6812
	Parallel-CNN-MV	0.9954	<u>0.9936</u>	0.9920	0.0138	<u>0.0023</u>	0.9928	0.9860	310.05	0.3472
	Parallel-CNN	0.9960	0.9921	0.9952	0.0061	0.0035	0.9936	0.9873	<u>414.33</u>	0.6753
	Proposed TFMCB-CNN-MV	0.9973	0.9958	0.9956	0.0073	0.0016	0.9957	0.9914	<u>683.57</u>	0.6331
3	Proposed TFMCB-CNN	<u>0.9961</u>	<u>0.9936</u>	0.9940	0.0095	<u>0.0026</u>	<u>0.9939</u>	<u>0.9878</u>	866.43	1.2772
	Parallel-CNN-MV	0.9956	0.9910	<u>0.9952</u>	0.0055	0.0041	0.9931	0.9861	502.77	<u>0.6653</u>
	Parallel-CNN	0.9960	0.9925	0.9950	<u>0.0066</u>	0.0033	0.9938	0.9876	811.30	1.3991
	Proposed TFMCB-CNN-MV	0.9967	0.9943	0.9954	0.0069	0.0024	0.9948	0.9896	<u>1191.00</u>	1.1868
4	Proposed TFMCB-CNN	0.9953	0.9908	<u>0.9946</u>	<u>0.0071</u>	0.0041	0.9926	0.9852	1444.14	2.1366
	Parallel-CNN-MV	0.9955	0.9921	0.9936	0.0094	0.0034	0.9928	0.9856	761.46	<u>1.3391</u>
	Parallel-CNN	<u>0.9960</u>	<u>0.9928</u>	0.9944	0.0081	<u>0.0030</u>	<u>0.9936</u>	<u>0.9872</u>	1414.06	2.6483
	Proposed TFMCB-CNN-MV	0.9970	0.9952	0.9955	0.0071	0.0019	0.9953	0.9906	<u>1207.06</u>	1.4001
5	Proposed TFMCB-CNN	0.9955	0.9929	0.9929	0.0113	0.0028	0.9929	0.9858	2175.12	3.1885
	Parallel-CNN-MV	<u>0.9962</u>	<u>0.9935</u>	0.9944	<u>0.0085</u>	<u>0.0027</u>	<u>0.9940</u>	<u>0.9879</u>	966.35	<u>1.6664</u>
	Parallel-CNN	0.9954	0.9928	0.9926	0.0120	0.0028	0.9927	0.9854	2133.57	3.9620
	Proposed TFMCB-CNN-MV	0.9970	0.9952	0.9955	0.0071	0.0019	0.9953	0.9906	<u>1207.06</u>	1.4001

TFMCB-CNN-MV scheme delivers better results compared to the other techniques in most of the assessed metrics, regardless of the batch size. For instance, when the batch size is 256, the values of Acc, Prec, Rec, F-score and MCC of the TFMCB-CNN-MV scheme are, respectively, 99.66%, 99.43%, 99.50%, 99.47% and 0.9893, whereas the Parallel-CNN-MV shows 99.44%, 98.92%, 99.32%, 99.12% and 0.9824 for the same metrics. Furthermore, it is noted that the TFMCB-CNN approach shows outstanding results compared to Parallel-CNN when the same batch size is considered. For instance, the TFMCB-CNN model showed Acc, Rec, FNR, F-score and MCC of 99.55%, 99.48%, 0.40%, 99.30% and 0.9860, against 99.36%, 99.41%, 0.68%, 98.99% and 0.9801 achieved by the Parallel-CNN. Additionally, an important finding is related to the Serial-CNN performance for larger batch sizes, especially 512 and 1,024. For example, when the batch size is 1,024, the Serial-CNN is outperformed only by the proposed TFMCB-CNN-MV scheme, except for the false positive rate. The values of Acc, Prec, Rec and MCC of the Serial-CNN are given by 98.98%, 97.89%, 98.98% and 0.9686, respectively, whereas the proposed TFMCB-CNN scheme achieved 98.58%, 96.86%, 98.88% and 0.9571 when the same metrics are considered.

Then, the values of Acc, Prec, Rec, FPR, FNR, F-score and MCC when the convolution filter size ranges from 2×2 to 5×5 are shown in Table 5.9 for all of the compared approaches. The number of branches is fixed in 3, the pooling filter size is 2×2 and the batch size is set to 128, with a single dense layer. From the results shown in Table 5.9, it is clear that the proposed TFMCB-CNN-MV scheme outperforms the other techniques in most

Table 5.7 – Performance evaluation metrics of the proposed TFMCB-CNN-MV and TFMCB-CNN schemes, as well as its competitor approaches, when the number of dense layers ranges from 1 to 5.

Nr DeL	Model	Acc	Prec	Rec	FPR	FNR	F-Score	MCC	Tr T (s)	Te T (s)
1	Proposed TFMCB-CNN-MV	0.9973	0.9958	0.9956	0.0073	0.0016	0.9957	0.9914	683.57	<u>0.6331</u>
	Proposed TFMCB-CNN	<u>0.9961</u>	<u>0.9936</u>	0.9940	0.0095	<u>0.0026</u>	<u>0.9939</u>	<u>0.9878</u>	866.43	1.2772
	Parallel-CNN-MV	0.9956	0.9910	<u>0.9952</u>	0.0055	0.0041	0.9931	0.9861	<u>502.77</u>	0.6653
	Parallel-CNN	0.9960	0.9925	0.9950	<u>0.0066</u>	0.0033	0.9938	0.9876	811.30	1.3991
	Serial-CNN	0.9939	0.9911	0.9898	0.0170	0.0034	0.9904	0.9808	270.38	0.4664
2	Proposed TFMCB-CNN-MV	0.9964	0.9946	0.9944	<u>0.0089</u>	<u>0.0023</u>	0.9944	0.9887	688.12	<u>0.6813</u>
	Proposed TFMCB-CNN	<u>0.9961</u>	<u>0.9940</u>	<u>0.9942</u>	0.0096	0.0021	<u>0.9943</u>	<u>0.9885</u>	858.00	1.2693
	Parallel-CNN-MV	0.9947	0.9906	0.9928	0.0104	0.0040	0.9917	0.9833	<u>508.12</u>	0.6936
	Parallel-CNN	0.9956	0.9917	0.9940	0.0074	0.0037	0.9930	0.9861	820.92	1.4283
	Serial-CNN	0.9937	0.9886	0.9917	0.0115	0.0050	0.9901	0.9802	273.64	0.4761
3	Proposed TFMCB-CNN-MV	0.9967	0.9940	0.9955	<u>0.0065</u>	0.0026	0.9947	0.9894	707.27	<u>0.6809</u>
	Proposed TFMCB-CNN	<u>0.9963</u>	<u>0.9935</u>	<u>0.9949</u>	0.0074	<u>0.0027</u>	<u>0.9942</u>	<u>0.9885</u>	895.12	1.2703
	Parallel-CNN-MV	0.9960	0.9926	0.9946	0.0075	0.0032	0.9936	0.9872	<u>552.99</u>	0.6935
	Parallel-CNN	0.9935	0.9868	0.9931	0.0076	0.0062	0.9898	0.9798	828.45	1.4550
	Serial-CNN	0.9941	0.9876	0.9942	0.0058	0.0059	0.9908	0.9817	276.15	0.4850
4	Proposed TFMCB-CNN-MV	0.9961	0.9940	<u>0.9939</u>	<u>0.0099</u>	0.0024	0.9939	0.9878	710.01	<u>0.7140</u>
	Proposed TFMCB-CNN	<u>0.9957</u>	<u>0.9931</u>	0.9932	0.0108	<u>0.0028</u>	<u>0.9931</u>	<u>0.9863</u>	898.88	1.3077
	Parallel-CNN-MV	0.9948	0.9906	0.9929	0.0102	0.0040	0.9917	0.9835	<u>519.02</u>	0.7287
	Parallel-CNN	0.9956	0.9916	0.9944	0.0075	0.0037	0.9930	0.9860	853.81	1.4905
	Serial-CNN	0.9915	0.9831	0.9909	0.0102	0.0081	0.9868	0.9739	284.60	0.4968
5	Proposed TFMCB-CNN-MV	0.9963	0.9934	0.9953	0.0064	<u>0.0030</u>	0.9942	0.9883	717.30	<u>0.6819</u>
	Proposed TFMCB-CNN	<u>0.9960</u>	<u>0.9931</u>	0.9935	0.0104	0.0026	<u>0.9935</u>	<u>0.9870</u>	922.17	1.2932
	Parallel-CNN-MV	0.9948	0.9904	0.9932	0.0093	0.0042	0.9918	0.9835	<u>532.36</u>	0.7370
	Parallel-CNN	0.9959	0.9922	<u>0.9947</u>	<u>0.0071</u>	0.0034	0.9934	0.9869	858.33	1.5131
	Serial-CNN	0.9953	0.9919	0.9934	0.0099	0.0034	0.9926	0.9853	286.11	0.5044

of the assessed metrics, especially for larger convolution filter sizes. For instance, when the convolution filter size is 5×5 , the proposed TFMCB-CNN-MV scheme achieves Acc = 99.68%, Prec = 99.47%, Rec = 99.56%, F-score = 99.49% and MCC = 0.9898, whereas its concurrent, Parallel-CNN-MV, presents 99.57%, 98.41%, 99.24%, 99.32% and 0.9864 considering the same metrics, respectively. Another important finding is related to the Serial-CNN scheme, which showed considerable results when the convolution filter size is 2×2 , even outperforming the TFMCB-CNN-MV approach in terms of recall, presenting 99.29% against 99.17% achieved by our proposed architecture.

Next, the proposed TFMCB-CNN-MV and TFMCB-CNN schemes are compared with state-of-the-art machine learning and deep learning algorithms commonly used for DDoS attack detection, namely, Convolution Neural Networks (CNNs), Gated Recurrent Unit (GRU), Long Short-Term Memory (LSTM), Simple Recurrent Neural Network (SRNN), Multilayer Perceptrons (MLPs), AdaBoost (AB), Decision Trees (DTs), Extra Trees (ETs), Linear Discriminant Analysis (LDA), Logistic Regression (LR) and Support Vector Machines (SVM).

Table 5.8 – Performance evaluation metrics of the proposed TFMCB-CNN-MV and TFMCB-CNN schemes, as well as its competitor approaches, when the batch size ranges from 128 to 1024.

Batch Size	Model	Acc	Prec	Rec	FPR	FNR	F-Score	MCC	Tr T (s)	Te T (s)
128	Proposed TFMCB-CNN-MV	0.9973	0.9958	0.9956	0.0073	0.0016	0.9957	0.9914	683.57	<u>0.6331</u>
	Proposed TFMCB-CNN	<u>0.9961</u>	<u>0.9936</u>	0.9940	0.0095	<u>0.0026</u>	<u>0.9939</u>	<u>0.9878</u>	866.43	1.2772
	Parallel-CNN-MV	0.9956	0.9910	<u>0.9952</u>	0.0055	0.0041	0.9931	0.9861	<u>502.77</u>	0.6653
	Parallel-CNN	0.9960	0.9925	0.9950	<u>0.0066</u>	0.0033	0.9938	0.9876	811.30	1.3991
	Serial-CNN	0.9939	0.9911	0.9898	0.0170	0.0034	0.9904	0.9808	270.38	0.4664
256	Proposed TFMCB-CNN-MV	0.9966	0.9943	0.9950	0.0076	0.0023	0.9947	0.9893	613.53	<u>0.6516</u>
	Proposed TFMCB-CNN	<u>0.9955</u>	<u>0.9912</u>	<u>0.9948</u>	<u>0.0065</u>	<u>0.0040</u>	<u>0.9930</u>	<u>0.9860</u>	794.62	1.2896
	Parallel-CNN-MV	0.9944	0.9892	0.9932	0.0086	0.0049	0.9912	0.9824	<u>424.17</u>	0.6783
	Parallel-CNN	0.9936	0.9860	0.9941	0.0049	0.0068	0.9899	0.9801	755.83	1.4479
	Serial-CNN	0.9917	0.9825	0.9920	0.0075	0.0085	0.9870	0.9744	251.94	0.4826
512	Proposed TFMCB-CNN-MV	0.9953	0.9917	0.9936	0.0093	0.0036	0.9926	0.9853	590.26	<u>0.6604</u>
	Proposed TFMCB-CNN	0.9929	0.9852	<u>0.9925</u>	<u>0.0079</u>	0.0071	0.9888	0.9776	768.29	1.2565
	Parallel-CNN-MV	0.9928	0.9857	0.9919	0.0096	0.0066	0.9887	0.9775	<u>385.79</u>	0.6692
	Parallel-CNN	0.9910	0.9809	0.9917	0.0070	0.0095	0.9860	0.9725	724.82	1.4611
	Serial-CNN	<u>0.9934</u>	<u>0.9883</u>	0.9911	0.0128	<u>0.0051</u>	<u>0.9896</u>	<u>0.9793</u>	241.60	0.4870
1,024	Proposed TFMCB-CNN-MV	0.9947	0.9891	0.9944	<u>0.0060</u>	0.0051	0.9917	0.9834	581.80	<u>0.6520</u>
	Proposed TFMCB-CNN	0.9858	0.9686	0.9888	0.0061	0.0162	0.9781	0.9571	768.68	1.2975
	Parallel-CNN-MV	0.9854	0.9710	0.9851	0.0153	0.0144	0.9775	0.9558	<u>370.46</u>	0.6691
	Parallel-CNN	0.9755	0.9460	0.9835	0.0034	0.0297	0.9630	0.9286	729.20	1.4725
	Serial-CNN	<u>0.9898</u>	<u>0.9789</u>	<u>0.9898</u>	0.0100	<u>0.0103</u>	<u>0.9841</u>	<u>0.9686</u>	243.07	0.4908

The parallel CNN based architectures are composed by 3 branches, pooling filter size of 2×2 , convolution filter size of 3×3 , batch size set to 128 and single dense layer. Table 5.10 shows the values of Acc, Prec, Rec, FPR, FNR, F-score and MCC of the proposed TFMCB-CNN-MV and TFMCB-CNN schemes, as well as their competitor classifiers. Note that our proposed techniques outperform the competing schemes for all of the performance evaluation metrics. For example, the proposed TFMCB-CNN-MV achieved Acc = 99.73%, Prec = 99.58%, Rec = 99.56% and MCC = 0.9914. On the other hand, AdaBoost, which is the third best compared classifier, showed the values of 99.58%, 99.22%, 99.47% and 0.9869 when considering the same metrics, respectively.

Finally, Table 5.11 shows the comparison between the proposed schemes and several state-of-the-art related works in the literature regarding DDoS attack detection. The evaluated metrics are accuracy, recall and F-score. The proposed TFMCB-CNN-MV and TFMCB-CNN approaches are configured with 3 branches, convolution filter size of 3×3 , batch size of 128, pooling filter size of 2×2 and single dense layer. Additionally, since the CIC-IDS2017 dataset has been extensively applied for NIDS validation by several works in the literature, the performance evaluation considering such data is also included. From the results shown in Table 5.11, we observe that the proposed TFMCB-CNN-MV scheme outperforms the compared techniques, regardless of the evaluated metric, when the CIC-IDS2017

Table 5.9 – Performance evaluation metrics of the proposed TFMCB-CNN-MV and TFMCB-CNN schemes, as well as its competitor approaches, when the convolution filter size ranges from 2×2 to 5×5 .

Filter Size	Model	Acc	Prec	Rec	FPR	FNR	F-Score	MCC	Tr T (s)	Te T (s)
2×2	Proposed TFMCB-CNN-MV	0.9945	0.9910	0.9917	0.0130	0.0037	0.9913	0.9827	640.86	0.6220
	Proposed TFMCB-CNN	0.9924	0.9841	<u>0.9925</u>	<u>0.0074</u>	0.0077	0.9881	0.9764	754.14	1.0342
	Parallel-CNN-MV	0.9900	0.9773	0.9920	0.0046	0.0113	0.9845	0.9692	<u>469.97</u>	<u>0.5814</u>
	Parallel-CNN	0.9856	0.9694	0.9876	0.0090	0.0157	0.9779	0.9568	761.26	1.2671
	Serial-CNN	<u>0.9932</u>	<u>0.9861</u>	0.9929	0.0076	<u>0.0066</u>	<u>0.9894</u>	<u>0.9789</u>	253.75	0.4224
3×3	Proposed TFMCB-CNN-MV	0.9973	0.9958	0.9956	0.0073	0.0016	0.9957	0.9914	683.57	<u>0.6331</u>
	Proposed TFMCB-CNN	<u>0.9961</u>	<u>0.9936</u>	0.9940	0.0095	<u>0.0026</u>	<u>0.9939</u>	<u>0.9878</u>	866.43	1.2772
	Parallel-CNN-MV	0.9956	0.9910	<u>0.9952</u>	0.0055	0.0041	0.9931	0.9861	<u>502.77</u>	0.6653
	Parallel-CNN	0.9960	0.9925	0.9950	<u>0.0066</u>	0.0033	0.9938	0.9876	811.30	1.3991
	Serial-CNN	0.9939	0.9911	0.9898	0.0170	0.0034	0.9904	0.9808	270.38	0.4664
4×4	Proposed TFMCB-CNN-MV	0.9970	0.9959	0.9946	0.0093	0.0014	0.9953	0.9905	752.00	0.7918
	Proposed TFMCB-CNN	<u>0.9962</u>	<u>0.9951</u>	0.9929	0.0124	0.0018	<u>0.9939</u>	<u>0.9878</u>	1002.84	1.6330
	Parallel-CNN-MV	0.9961	0.9949	0.9927	0.0128	<u>0.0017</u>	0.9938	0.9876	<u>531.62</u>	<u>0.7142</u>
	Parallel-CNN	0.9959	0.9929	0.9943	<u>0.0084</u>	0.0030	0.9936	0.9872	902.02	1.5869
	Serial-CNN	0.9958	0.9922	<u>0.9944</u>	0.0078	0.0034	0.9933	0.9866	300.67	0.5290
5×5	Proposed TFMCB-CNN-MV	0.9968	0.9947	0.9956	0.0064	0.0025	0.9949	0.9898	848.40	0.8621
	Proposed TFMCB-CNN	0.9960	0.9927	0.9942	0.0074	0.0032	0.9937	0.9874	1158.18	1.8037
	Parallel-CNN-MV	0.9957	0.9941	0.9924	0.0132	0.0020	0.9932	0.9864	<u>577.13</u>	<u>0.7248</u>
	Parallel-CNN	<u>0.9964</u>	<u>0.9945</u>	0.9941	0.0098	<u>0.0021</u>	<u>0.9943</u>	<u>0.9885</u>	1018.33	1.9036
	Serial-CNN	0.9957	0.9917	<u>0.9946</u>	<u>0.0070</u>	0.0037	0.9932	0.9863	339.44	0.6345

Table 5.10 – Performance evaluation metrics of the proposed TFMCB-CNN-MV and TFMCB-CNN schemes against state-of-the-art ML and DL classification algorithms.

Alg. Type	Model	Acc	Prec	Rec	FPR	FNR	F-score	MCC	Tr T (s)	Te T (s)
DL	Proposed TFMCB-CNN-MV	0.9973	0.9958	0.9956	0.0073	0.0016	0.9957	0.9914	698.57	0.75
	Proposed TFMCB-CNN	<u>0.9961</u>	<u>0.9936</u>	<u>0.9940</u>	<u>0.0095</u>	<u>0.0026</u>	<u>0.9939</u>	<u>0.9876</u>	889.98	1.31
	CNN-1D 1 layer	0.9908	0.9902	0.9805	0.0365	0.0026	0.9852	0.9706	254.07	0.39
	CNN-2D 1 layer	0.9868	0.9873	0.9708	0.0556	0.0028	0.9787	0.9579	540.53	0.63
	CNN-1D 2 layer	0.9937	0.9898	0.9904	0.0151	0.0041	0.9901	0.9802	450.01	0.59
	CNN-2D 2 layer	0.9935	0.9893	0.9903	0.0150	0.0044	0.9898	0.9796	780.18	0.74
	GRU 2 layer	0.9749	0.9651	0.9550	0.0778	0.0122	0.9598	0.9200	2159.35	4.15
	LSTM 2 layer	0.9550	0.9461	0.9094	0.1659	0.0153	0.9256	0.8543	2403.04	5.20
	SRNN 2 layer	0.9781	0.9794	0.9511	0.0934	0.0044	0.9629	0.9291	895.22	2.23
	MLP 5 layer	0.9943	0.9935	0.9879	0.0226	0.0018	0.9909	0.9819	153.82	0.13
ML	AB	0.9958	0.9922	0.9947	0.0071	0.0034	0.9934	0.9869	59.71	3.07
	DT	0.9717	0.9769	0.9331	0.1306	0.0032	0.9472	0.9049	0.04	0.01
	ET	0.9902	0.9898	0.9790	0.0393	0.0026	0.9843	0.9688	7.26	0.79
	LDA	0.9179	0.8662	0.8196	0.3427	0.0181	0.8380	0.6927	<u>0.46</u>	<u>0.01</u>
	LR	0.9854	0.9834	0.9701	0.0550	0.0047	0.9766	0.9534	0.65	0.01
	SVM	0.9843	0.9813	0.9687	0.0571	0.0056	0.9748	0.9499	70.21	3.56

dataset was used for validation. For example, TFMCB-CNN-MV presented Acc, Rec, and F-score of 99.79%, 99.74% and 99.67%, respectively, whereas the best competitor technique,

Haider et al. [127], achieved 99.45%, 99.64% and 99.61% for the same metrics. However, the proposed TFMCB-CNN approach was outperformed by the referred competing scheme in some evaluated metrics. While 99.41% and 99.56% were achieved by TFMCB-CNN in terms of Rec and F-score, respectively, that related work showed considerable values of 99.64% and 99.61% when the same metrics were considered. On the other hand, when the CIC-DDoS2019 dataset was applied for validation, both proposed schemes outperformed all of the competing techniques, regardless of the assessed metric. For instance, the best competitor scheme in terms of accuracy, Maranhão et al. [3], achieved 99.55%, against 99.73% and 99.61% showed by TFMCB-CNN-MV and TFMCB-CNN, respectively. In addition, the proposed TFMCB-CNN-MV and TFMCB-CNN schemes presented 99.56% and 99.40% for Rec, and 99.57% and 99.39% for F-score, respectively. The best competitor approach in terms of recall and F-score, Elsayed et al. [118], achieved 99.00% for both metrics.

Table 5.11 – Performance comparison with related works, considering the CIC-IDS2017 and CIC-DDoS2019 datasets.

Dataset	Work	Method	Acc	Rec	F-score
CIC-IDS2017	Proposed TFMCB-CNN-MV	CNN	0.9979	0.9974	0.9967
	Proposed TFMCB-CNN	CNN	0.9972	0.9941	0.9956
	Maranhão et al. [2]	MLP	0.9895	0.9831	N/A
	Roopak et al. [126]	MLP	0.8634	0.8625	0.8735
	Roopak et al. [126]	1D-CNN	0.9514	0.9017	0.9399
	Roopak et al. [126]	LSTM	0.9624	0.8989	0.8959
	Roopak et al. [126]	1D-CNN+LSTM	0.9716	0.9910	0.9825
	Haider et al. [127]	Ensemble CNN	0.9945	0.9964	0.9961
	Chen et al. [128]	CNN	0.9887	N/A	N/A
	Aamir and Ali Zaidi [139]	kNN	0.9500	N/A	N/A
	Aamir and Ali Zaidi [139]	SVM	0.9200	N/A	N/A
	Aamir and Ali Zaidi [139]	RFo	0.9666	N/A	N/A
	Aksu et al. [141]	kNN	0.9572	0.9589	0.9577
	Aksu et al. [141]	SVM	0.6069	0.7142	0.6463
	Aksu et al. [141]	DT	0.9900	0.9900	0.9900
	Ustebay et al. [142]	MLP	0.9100	N/A	N/A
	Yulianto et al. [143]	AB+PCA+SMOTE	0.8147	0.9576	0.8817
	Zhu et al. [144]	AMF-LSTM	N/A	0.9800	0.9000
	Yao et al. [145]	DeepGFL	N/A	0.3024	0.4321
	CIC-DDoS2019	Proposed TFMCB-CNN-MV	CNN	0.9973	0.9956
Proposed TFMCB-CNN		CNN	0.9961	0.9940	0.9939
Maranhão et al. [3]		DT	0.9754	0.9509	N/A
Maranhão et al. [3]		RFo	0.9955	0.9896	N/A
Elsayed et al. [118]		RNN-Autoencoder	N/A	0.9900	0.9900
Shurman et al. [119]		LSTM 1 layer	0.9154	N/A	N/A
Shurman et al. [119]		LSTM 2 layer	0.9674	N/A	N/A
Shurman et al. [119]		LSTM 3 layer	0.9919	N/A	N/A
Sharafaldin et al. [120]		ID3	N/A	0.6500	0.6900
Sharafaldin et al. [120]		RFo	N/A	0.5600	0.6200
Sharafaldin et al. [120]		NB	N/A	0.1100	0.0500
Sharafaldin et al. [120]		LR	N/A	0.0200	0.0400
Hussain et al. [121]		ResNet	N/A	0.8600	0.8600
Aytac et al. [122]		RFo	0.9840	N/A	N/A
Aytac et al. [122]		DT (Gini)	0.9934	N/A	N/A
Aytac et al. [122]		DT (Entropy)	0.9912	N/A	N/A
Aytac et al. [122]		Multinomial NB	0.9910	N/A	N/A
Aytac et al. [122]	Gaussian NB	0.9870	N/A	N/A	

5.4.3 Processing time

This subsection introduces and discusses the performance evaluation of the proposed TFMCB-CNN-MV and TFMCB-CNN schemes in terms of training times (Tr T) and testing times (Te T), in seconds (s), which are also included in Tables 5.6 to 5.10. In DL based models, the training times refer to a period of 100 epochs. As expected, the Serial-CNN configuration presents the lowest processing times, since it is composed by a single branch, with no concatenating operations. First, the processing times are assessed for different number of parallel branches, as shown in Table 5.6. All of the compared schemes present higher training and testing times as the number of branches is larger. Such fact is more evident when SMV and non SMV based schemes are compared each other. For instance, when $L = 2$, the proposed TFMCB-CNN-MV and TFMCB-CNN schemes showed training times of 417.22 s and 454.18 s, respectively. Nevertheless, when the number of branches is 5, the Tr T obtained by each scheme are, respectively, 1207.06 s and 2175.12 s. In this sense, the difference in training time between TFMCB-CNN-MV and TFMCB-CNN is increased from 37 s to 968 s, approximately, when L goes from 2 to 5. Moreover, note that the Parallel-CNN-MV scheme presents the lowest training times for a fixed number of branches, whereas the TFMCB-CNN approach delivers the worst performance. For example, when $L = 4$, the Parallel-CNN-MV showed Tr T = 761.46 s, against 1444.14 s achieved by the TFMCB-CNN. Additionally, the Parallel-CNN-MV and Parallel-CNN schemes present lower training times compared to their respective counterparts, TFMCB-CNN-MV and TFMCB-CNN. Such results reflect the higher computational cost imposed by feature map concatenation processes. On the other hand, considering testing times, the proposed TFMCB-CNN-MV scheme delivers the best performance when $L \geq 3$. Such fact is crucial in real time network attack detection problems, in which lower testing times are required for an efficient NIDS performance.

Next, Table 5.7 illustrated the processing times for different number of MLP dense layers. From the results, it can be seen that all of the compared techniques present slightly higher training and testing times when the number of dense layers is larger. Since the MLPs are placed after the proposed tensor feature map concatenation block, the increasing in the number of dense layers has no significant impact on the model processing times. Following, for a fixed number of dense layers, the Parallel-CNN-MV and TFMCB-CNN schemes achieved, respectively, the highest and the lowest training times among the parallel CNN based techniques. For instance, when the number of dense layers is 5, TFMCB-CNN presents a Tr T of 922.17 s, whereas 532.36 s was achieved by the Parallel-CNN-MV. On the other hand, for testing times, once again our TFMCB-CNN-MV technique delivers the best performance when excluding the Serial-CNN technique. Such fact is very important for DDoS attack detection under real time conditions. For example, for 5 dense layers, the TFMCB-CNN-MV and Parallel-CNN-MV schemes achieved testing times of 0.68 s and 0.73 s, respectively.

Following, the training and testing times when the batch size is varied from 128 to 1024

are shown in Table 5.8. In terms of training time, all of the compared schemes deliver better performance as the batch size is larger. For example, the proposed TFMCB-CNN-MV and TFMCB-CNN techniques achieve Tr T of 683.57 s and 866.43 s, respectively, for a batch size of 128. Nonetheless, when such parameter is increased to 1024, lower training times (581.80 s and 768.68 s) are observed in both techniques. Lower batch sizes imply a higher number of batches and, consequently, the NIDS weights are updated more frequently during the process of forward and backward propagation in the training phase, which increases the processing time. In addition, for a fixed batch size, majority voting based schemes also achieve the best performance when comparing parallel CNN based techniques. When the batch size is 256, for instance, the Parallel-CNN-MV and TFMCB-CNN-MV schemes present training times of 424.17 s and 613.53 s, respectively, against 755.83 s and 794.62 s achieved by their non SMV based counterparts. Nevertheless, such fact is compensated considering testing times, since the proposed TFMCB-CNN-MV delivers the best performance when Serial-CNN is not taken in account. For example, for a batch size of 128, the Parallel-CNN-MV scheme achieved a Te T of 0.66 s, against 0.63 s obtained by TFMCB-CNN-MV.

Then, Table 5.9 introduces the processing times for different convolution filter sizes. It can be observed that the training and testing times are higher as the filter size grows for all of the compared schemes. For instance, for a filter size of 2×2 , the Tr T and Te T of the proposed TFMCB-CNN-MV approach are given by 640.86 s and 0.62 s, respectively. However, when the filter size is increased to 5×5 , the respective processing times are also increased to 848.40 s and 0.86 s. For higher filter sizes, more convolution operations are performed between the filter and the data window and, consequently, higher processing times are noticed. Furthermore, in line with the observed in the previous experiments, SMV based techniques deliver the best performance for a fixed convolution filter size. For example, for a filter size of 4×4 , the TFMCB-CNN-MV and Parallel-CNN-MV schemes achieve training times of 752.00 s and 531.62 s, respectively, against 1002.84 s and 902.02 s obtained by TFMCB-CNN and Parallel-CNN. Additionally, considering testing times, the TFMCB-CNN-MV scheme presents an outstanding performance and is outperformed only by the Parallel-CNN-MV technique. For instance, for the convolution filter sizes of 4×4 and 5×5 , Te T of 0.79 s and 0.86 s were achieved by the TFMCB-CNN-MV, whereas 0.71 s and 0.72 s were reported by the Parallel-CNN-MV scheme.

Finally, the proposed TFMCB-CNN-MV and TFMCB-CNN schemes are compared with several state-of-the-art ML and DL techniques in terms of processing time. From Table 5.10, we observe that ML algorithms take much less training time compared to DL schemes, in which a large number of parameters are learned during the forward and backward propagation. In our experiments, Decision Trees and Linear Discriminant Analysis showed the best results in terms of training and testing times, since both algorithms have a smaller number of parameters compared to the other assessed ML algorithms. In LDA, high-dimensional data is

projected onto a lower dimensional space and, consequently, the separation of samples from different classes is maximized, whereas the dispersion of samples within the same class is minimized [1]. In addition, DT presents low computational cost and ease of understanding and interpretation [3], such that a top-down divide-and-conquer approach is implemented to supervised classification where a condition on the attribute value is used to divide the data [160]. Among the compared DL algorithms, the five-layered MLP achieved the lowest processing times, whereas the worst performance was delivered by the Long-Short Term Memory (LSTM) algorithm. As detailed in Subsection 5.2.2.4, MLP is a feedforward neural network composed by input, hidden, and output layers in which several parameters are learned during the training process. On the other hand, LSTMs use sequential processing over time and are difficult to train since they require memory-bandwidth-bound computation. An LSTM unit is composed of a cell and three types of gate (input, output, and forget gates), which regulate the input and output information flow in the cell. Moreover, note that the proposed TFMCB-CNN-MV scheme delivers lower training times compared to the two-layered CNN-2D, GRU, LSTM and SRNN algorithms, despite it is outperformed by the other DL schemes.

5.4.4 Discussion

In this subsection, the simulation results presented in Subsection 5.4.2 are discussed in detail. Table 5.6 shows the experiment results for the proposed TFMCB-CNN-MV and TFMCB-CNN schemes, as well as their competing techniques, for different number of branches. The benefits of adopting multiple parallel CNN branches are evident, since the single-branch Serial-CNN approach is outperformed by most of the competing techniques. Furthermore, the tensor feature map concatenation reduces the performance of the proposed TFMCB-CNN scheme as the number of parallel branches is higher. However, the jointly usage of tensor feature map concatenation and majority voting techniques improves the DDoS attack detection performance compared to the approaches in which only feature map concatenation is applied, especially for higher number of parallel branches. In addition, it is observed that the inclusion of majority voting is advantageous, even for non feature map concatenation based models, when the number of branches is higher. Such fact is evident when the number of branches is 5, where both TFMCB-CNN-MV and Parallel-CNN-MV schemes showed the best performance among all of the compared techniques.

Table 5.7 shows the simulation results for different number of MLP dense layers. It can be observed that the proposed TFMCB-CNN-MV and TFMCB-CNN schemes outperform the competitor methods in most of the assessed metrics, regardless of the number of dense layers. Since the tensor feature map concatenation technique is performed between parallel branches, i.e., before the MLP block, the proposed TFMCB-CNN approach is not very sensitive to the variation of the number of dense layers. On the other hand, as the number of dense

layers is lower, most of the TFMCB-CNN-MV evaluated metrics show better performance, which reflects a higher sensitivity to the variation of the number of dense layers after the inclusion of the SMV block. In addition, from the results shown in Table 5.7, we observe that the Serial-CNN approach is still outperformed by all of the parallel based schemes, which once again reflects the benefits of adopting multiple parallel CNN based branches in DDoS attack detection models.

Next, Table 5.8 shows the simulation results obtained for different batch sizes. It can be seen that, regardless of the batch size, the proposed TFMCB-CNN-MV scheme outperforms the other techniques in most of the assessed metrics. On the other hand, the proposed TFMCB-CNN scheme is also superior compared to the competing techniques, but only for lower batch sizes. In this sense, we conclude that the tensor feature map concatenation technique is sensitive to the batch size increasing, but this fact is compensated by the inclusion of the simple majority voting block. In addition, it was observed that the Serial-CNN approach benefits from larger batch sizes, outperforming the TFMCB-CNN scheme in most of the evaluated metrics when the batch size is 512 or 1024. Moreover, from the results shown in Table 5.8, it is clear that all of the compared schemes deliver better performance for lower batch sizes. In this situation, the model weights are more frequently updated during the forward and backward propagation in the training phase.

Following, Table 5.9 illustrates the experiment results for different values of convolution filter size. Note that the proposed TFMCB-CNN-MV scheme presents a considerable performance for larger convolution filter sizes. In this case, the tensor feature map concatenation and majority voting techniques in TFMCB-CNN-MV benefit from a higher number of neighboring features convoluted across the filter. Moreover, we observe that the proposed TFMCB-CNN scheme is outperformed by the Serial-CNN approach for the 2×2 convolution filter size in several metrics. In this sense, we conclude that single-branch CNN based schemes present better performance compared to tensor feature map concatenation based techniques when smaller convolution filter sizes are considered.

Then, Table 5.10 summarizes the comparison between the TFMCB-CNN-MV and TFMCB-CNN schemes with state-of-the-art ML and DL algorithms. The best competing scheme, AdaBoost, is a strong classifier generated from a combination of several weak learners and, depending on the dataset as well as some adopted parameters, can yield more accurate results compared to classic machine learning algorithms. In addition, the MLP 5 layer, CNN-1D 2 layer and CNN-2D 2 layer schemes, despite are not the best among the compared techniques, showed considerable DDoS attack detection performance. The CNN-1D 2 layer scheme is composed by two 1D CNN basic building blocks in series, whereas CNN-2D 2 layer presents identical configuration, but with two 2D CNN blocks. Moreover, the MLP 5 layer is simply a neural network composed by five hidden layers in which data feature extraction is performed. In this sense, from the results shown in Table 5.10, we observe an outstanding performance

gain provided by jointly applying the multidimensional feature map concatenation and majority voting techniques on the TFMCB-CNN-MV scheme, compared to the MLP 5 layer, CNN-1D 2 layer and CNN-2D 2 layer approaches.

Further, a comparison between the proposed TFMCB-CNN-MV and TFMCB-CNN with related works is shown in Table 5.11. When the CIC-IDS2017 dataset is used for validation, it can be seen that all of the compared schemes are outperformed by our proposed TFMCB-CNN-MV technique. Nonetheless, the TFMCB-CNN scheme is outperformed by Haider et al. [127] considering recall and F-score. In [127], the authors proposed a deep CNN ensemble framework for DDoS attack detection in Software Defined Networks (SDN). Such model is composed by two parallel branches, each of which containing three 2D CNNs, two max pooling layers, one flatten layer and two dense fully-connected layers. At the end, the outputs of each branch are concatenated along the 3rd dimension, in a process similar to the one adopted in our TFMCB-CNN scheme. In this sense, from the metric values presented in Table 5.11, we observe that the simple majority voting in TFMCB-CNN-MV provides a considerable gain in terms of DDoS attack detection performance. In addition, when the CIC-DDoS2019 dataset is applied for validation, all of the competing approaches are outperformed by both TFMCB-CNN-MV and TFMCB-CNN schemes, regardless of the assessed metric. Therefore, we conclude that the jointly applied tensor feature map concatenation and majority voting techniques provided a significant performance gain compared to the related works when the CIC-DDoS2019 dataset is used for validation.

Finally, Tables 5.6 to 5.10 also include the training and testing times for the proposed TFMCB-CNN-MV and TFMCB-CNN schemes, as well as their competitor approaches, when several simulation parameters are varied. It can be seen that feature map concatenation based approaches (TFMCB-CNN-MV and TFMCB-CNN) have higher processing times compared to their counterparts with no concatenating operations (Parallel-CNN-MV and Parallel-CNN), since data resulting from concatenation processes present higher number of channels. In addition, non majority voting based schemes (TFMCB-CNN and Parallel-CNN) are more time costly compared to the SMV based techniques (TFMCB-CNN-MV and Parallel-CNN-MV). In the former, the outputs from each branch are concatenated into a tensor whose number of channels corresponds to the sum of the number of channels of the referred outputs. Consequently, a large vector is obtained after flattening, which increases the processing time. On the other hand, in SMV based schemes, the feature map concatenation takes places only between consecutive branches, whose outputs are forwarded to their respective flattening and MLP blocks. In this sense, despite the higher number of Flatten/MLP blocks, parallel computing allows a more efficient time processing compared to the non SVM based techniques.

5.5 SUMMARY

One of the most harmful threats to network security, Distributed Denial of Service attacks aim to deny legitimate accesses to network services by exhausting bandwidth and resources through huge volume of traffic launched by attackers. To face such challenge, deep learning based NIDSs, especially those using Convolutional Neural Networks, have been widely proposed in order to extract high level features from input traffic data and, consequently, improve the DDoS attack detection performance. In this chapter, two novel CNN architectures for DDoS attack detection based on feature map concatenation between parallel CNNs have been proposed, namely, TFMCB-CNN and TFMCB-CNN-MV. The second scheme is a refined version of the first one, in which output data are sent to a simple majority voting module for final classification. Extensive simulations were executed by using the CIC-IDS2017 and CIC-DDoS2019 benchmark datasets for validation, in which several performance evaluation metrics were assessed. According to the experiment results, the proposed schemes outperform the state-of-the-art approaches in terms of several evaluation metrics, providing an outstanding performance gain compared to those techniques. In addition, it can be concluded that a better DDoS attack detection performance can be achieved by tuning several model settings, such as number of parallel branches, number of MLP dense layers and convolution filter size, in contrast to deep learning NIDS with fixed parameters commonly found in the literature.

6 CONCLUSION

Cyber-Physical Systems are composed by networked components including sensors, control processing units and communication devices, which are used for monitoring and management of critical physical infrastructures. In this sense, it is crucial to develop high accurate intrusion detection systems in order to protect CPSs and, consequently, offer high security level to the protected structures. In this thesis, we focused on the security of the physical and network layers of Cyber-Physical Systems. To face the challenges imposed by the former case, we proposed a tensor based framework for UAV localization and identification in multipath environments. Additionally, to overcome the issues imposed by the latter case, three different network intrusion detection systems for detecting Distributed Denial of Service attacks were proposed.

The following research questions were tackled in this thesis:

1. How to efficiently localize and identify UAVs in multipath environments by jointly applying multidimensional signal processing techniques and machine learning (ML) algorithms?
2. How to efficiently detect DDoS attacks in a CPS network by jointly applying multidimensional signal processing techniques and machine learning algorithms, assuming that a noisy dataset is used for training and testing?
3. Following the same idea of the previous question, how to efficiently detect DDoS attacks in a CPS network, still applying multidimensional signal processing techniques on noisy datasets, but now using deep learning algorithms instead of ML techniques?
4. How to efficiently detect DDoS attacks in a CPS network by jointly applying multidimensional feature map concatenation based techniques and deep learning algorithms, but now assuming that a noiseless dataset is used for training and testing?

This chapter is organized as follows. In Section 6.1, the conclusion remarks are presented in detail, whereas in Section 6.2 recommendations for future research are introduced.

6.1 CONCLUSIONS

The first research question was addressed in Chapter 2, which introduced a tensor based framework for UAV localization and identification in multipath environments. The proposed scheme was an extension of the technique presented in [6] in three aspects: by adopting a tensor representation to better explore the structure inherently multidimensional of the data; by including a denoising preprocessing scheme to increase the signal-to-noise ratio of the received signal; and by including a machine learning classification module for identifying the UAV type. The proposed framework was compared with state-of-the-art localization techniques, namely, M-ESPRIT + SS and T-ESPRIT + SS.

Initially, the performance evaluation of the compared schemes was made considering the UAV localization. Extensive simulations were performed for different values of SIR, number of URA antennas, number of samples and number of intruding drones. The root mean square error of the estimated position coordinates and estimated spatial frequencies were the assessed metrics. From the simulation results, it was observed that the proposed framework presents a higher robustness against co-channel interference, especially due to the MuDe module, which compensates the high interference level. In addition, when the number of antennas was larger, the maximum number of subarrays in each spatial dimension was increased and, consequently, the MuDe performance was improved, which corroborated the considerable results shown by our proposed scheme, despite its higher computational complexity. Moreover, regardless of the compared technique, it was also observed that a higher number of time samples leads to a more accurate performance, despite the trade-off between increased accuracy and higher processing time. Furthermore, all of the competitor schemes presented worse performance as the number of intruding drones was higher, since high interference levels were achieved in such situation, which impacted the accuracy of the sensor array based localization system.

Next, in the context of drone model identification, the compared techniques were evaluated for different values of SIR. Several ML algorithms were analyzed, such as Decision Trees, Extra Trees, k-Nearest Neighbors, Linear Discriminant Analysis, Logistic Regression, Naïve Bayes and Random Forest. Further, three different drone models were considered: Be-bop, AR and Phantom. From the results obtained in simulations, it could be seen that the competitor techniques presented better performance for identifying UAVs as the SIR was higher. Since the number of interfering sources was lower in this case, the drone signatures estimated by each URA were more accurate, which led to improved classification results. In addition, the proposed framework showed considerable results for identifying intruding drones under low interference level conditions, especially when ET, LDA, LR and NB algorithms were applied for classification. Furthermore, it was also observed that some drone types were more efficiently identified by specific ML algorithms, such as the AR drone

model, which was recognized by the Extra Tress classifier with an accuracy of 100%.

The second research question was addressed in Chapter 3, which proposed a novel noise-robust framework for DDoS attack detection which exploited tensor based signal processing techniques as well as ML based algorithms. In addition, an extension of the recent Multiple Denoising (MuDe) technique was presented, which attenuated the noise present in the dataset instances. The proposed approach was compared with traditional low-rank approximation techniques, namely, SVD, HOSVD and HOOI. Several ML algorithms were analyzed, such as AdaBoost, Linear Discriminant Analysis, Logistic Regression and Random Forest. Extensive simulations were performed for different values of SNR and TSP, whereas accuracy, detection rate and false alarm rate were the assessed metrics. The NSL-KDD and CIC-DDoS2019 benchmark datasets were applied for model validation.

It was observed that the proposed framework outperformed its competing schemes in terms of detection rate and false alarm rate for low SNR values, especially for the LDA, LR and RFo classification algorithms. The benefits of our proposed extended MuDe algorithm were more evident in low SNRs because it provided a higher noise reduction due to the multiple denoising performed along the r -th dimension of the m -th dataset instance for $r = 1, \dots, R$ and $m = 1, \dots, M$. Furthermore, all of the compared techniques delivered better performance as the training dataset size proportion increased, since ML classifiers need more training samples when the dataset has a large number of features. Additionally, the proposed scheme presented higher processing times compared to the competing techniques, especially due to the MuDe algorithm. This is the trade-off in order to achieve a more accurate DDoS attack detection when considering the multiple denoising technique.

The third research question was addressed in Chapter 4, which proposed a novel noise-robust multilayer perceptron (MLP) architecture for DDoS attack detection. The average value of the common features among dataset instances was dynamically filtered out through the forward and backward propagation of the MLP, improving the model performance. The proposed approach was compared with state-of-the-art low-rank approximation techniques, namely, HOSVD and HOOI. Extensive experiments were conducted on a customized dataset containing samples extracted from the CIC-IDS2017, CIC-IDS2018 and CIC-DDoS2019 benchmark datasets, considering different noise levels and number of hidden layers. Moreover, accuracy, detection rate and false alarm rate were the evaluated metrics.

The proposed approach was more efficient for detecting DDoS attacks compared to the HOSVD and HOOI based MLPs when corrupted datasets were considered. Nonetheless, the accuracy of our proposed technique was matched by the HOOI based MLP in scenarios with low noise levels, since the more accurate core tensor and singular matrices generated through orthogonal iterations in HOOI lead to a better dataset denoising. In addition, under larger number of hidden layers and higher noise level conditions, the proposed scheme was far superior compared to the HOOI and HOSVD based MLPs. Following, the proposed

scheme was compared with conventional MLPs under noise-free conditions. Our proposed approach outperformed the conventional MLP in terms of Acc and DR for multiple HL and TSP configurations, despite it was more prone to false positives. In this sense, we conclude that the noise attenuation provided by HOSVD and the more discriminative individual information resulting from dataset filtering reflected in a higher noise-robustness and efficiency of our proposed scheme.

However, an important drawback of our scheme in Chapter 4 was its higher training time, especially for larger number of hidden layers, as a consequence of multiple HOSVDs performed over training data batches. On the other hand, lower testing times were shown by our approach, since low cost computations were performed during the testing phase, in contrast to the tensor decompositions executed in HOSVD and HOOI. Additionally, the proposed MLP outperformed the competing schemes for detecting DDoS attacks under real time conditions, which reflected the superiority of our technique due to a more efficient dataset filtering during the training phase. Finally, it was observed that our proposed MLP presented higher noise-robustness compared to the competitor methods, especially under high noise level conditions.

The last research question was addressed in Chapter 5, which proposed two novel tensor feature map concatenation based CNN architectures for DDoS attack detection. The first scheme, called TFMCB-CNN, has multiple parallel branches, each of which composed by CNNs alternately positioned with feature map concatenation blocks. The outputs from CNNs of consecutive branches are concatenated and sent to the next CNN within each branch. At last, the branch outputs are concatenated and forwarded to a common flattening and multilayer perceptron blocks for final classification. The second scheme, known as TFMCB-CNN-MV, is an improved version of the first one. Each one of its L parallel branches has its respective flattening and multilayer perceptron blocks, whose outputs are sent to a simple majority voting module for final classification. The proposed schemes were compared with state-of-the-art models, namely, Parallel-CNN-MV, Parallel-CNN and Serial-CNN. Extensive experiments were performed, considering different number of branches, number of dense layers, convolution filter sizes and batch sizes. Moreover, accuracy, precision, recall, false positive rate, false negative rate, f-score and Matthews correlation coefficient were the evaluated metrics. The CIC-IDS2017 and CIC-DDoS2019 benchmark datasets were applied for model validation in simulations.

The jointly usage of feature map concatenation and majority voting techniques improved the DDoS attack detection performance compared to the approaches in which only feature map concatenation was applied, especially for higher number of parallel branches. Further, since the feature map concatenation technique is performed before the MLP block, the proposed TFMCB-CNN approach was not very sensitive to the variation of the number of dense layers. On the other hand, the TFMCB-CNN-MV scheme presented better performance as

the number of dense layers was lower. Additionally, it was observed that the feature map concatenation technique was sensitive to the batch size increasing, but this fact was compensated by the inclusion of the simple majority voting block. As expected, all of the compared schemes presented better performance for lower batch sizes, since the model weights were more frequently updated during the forward and backward propagation in the training phase. In addition, the proposed TFMCB-CNN-MV scheme presented an outstanding performance for larger convolution filter sizes, where both feature map concatenation and majority voting techniques benefited from a higher number of neighboring features convoluted across the filter.

In summary, this thesis aimed to bring multidimensional signal processing techniques and machine learning/deep learning algorithms closer to real life problems, such as the intrusion detection in the physical and network layers of Cyber-Physical Systems. Both approaches can be jointly applied to solve the problem of localization and identification of intruding UAVs in multipath environments, as well as the DDoS attack detection when either noisy or noiseless benchmark datasets are used for NIDS training and testing.

6.2 FUTURE WORKS

This thesis addresses several problems, but some of them are still open, whereas others are emerging from current results. Thus, the following issues should be investigated as future works:

- Chapter 2:
 - to incorporate a UAV tracking module into the proposed framework such that the intruding UAVs can be tracked in real time.
 - to incorporate a UAV detection module into the proposed framework such that the intruding UAVs can be detected as soon as they invade the controlled air space.
 - to incorporate a preprocessing multidimensional pre-whitening module into the proposed framework in order to evaluate the UAV localization performance in spatial colored noise environments.
 - to improve the UAV identification module such that different flight modes present in the DroneRF dataset can be identified, for instance, “off”, “on and connected”, “hovering”, “flying” and “video recording”.
 - to implement, in hardware, the proposed framework by using several devices, such as multichannel Software Defined Radio (SDR) platforms and omnidirectional antenna arrays. The objective is to collect, analyze and record raw RF

signals emitted from different commercial UAV models, operating under several flight modes, such that a novel drone RF sensing based dataset can be built up.

- to integrate the database developed in the previous item with UAV detection systems based on alternative technologies, such as acoustic recordings, radar echoes and camera images, in order to improve the UAV identification performance.
- Chapter 3:
 - to evaluate the performance of the proposed framework for detecting DDoS attacks when the NIDS is trained and tested with datasets corrupted by different types of false data injection attacks.
 - to extend the proposed framework for detecting different types of network attacks, such as brute force, web attack, infiltration attack and port scan.
 - to implement distributed or parallel processing in order to analyze the scalability and processing capacity of the proposed framework for monitoring high throughput network traffic.
 - Chapter 4:
 - to evaluate the performance of the proposed MLP architecture for detecting DDoS attacks when the NIDS is trained and tested with datasets corrupted by different types of false data injection attacks.
 - to extend the proposed MLP architecture for detecting different types of network attacks, such as brute force, web attack, infiltration attack and port scan.
 - to implement distributed or parallel processing in order to analyze the scalability and processing capacity of the proposed MLP architecture for monitoring high throughput network traffic.
 - to apply the proposed feature extraction technique on alternative deep learning classification algorithms, such as CNNs and Autoencoders.
 - Chapter 5:
 - to evaluate the performance of the proposed schemes for detecting DDoS attacks when deeper network configurations are adopted, such that zero-day attacks can be detected in real world traffic data.
 - to extend the proposed schemes for detecting different types of network attacks, such as brute force, web attack, infiltration attack and port scan.
 - to extend the proposed schemes by using alternative feature map concatenation based configurations for detecting DDoS attacks, including different deep learning classification algorithms, such as GRU, LSTM, SRNN and Autoencoders.

APPENDICES

A LOW-RANK MATRIX AND TENSOR BASED APPROXIMATION TECHNIQUES

In this appendix, we present the mathematical concepts regarding the low-rank matrix and tensor based approximation techniques considered throughout this thesis, namely, SVD, HOSVD and HOOI low-rank approximations. First, Section A.1 recapitulates the concepts and properties of the SVD. Next, HOSVD is discussed in detail in Section A.2. Finally, Section A.3 describes the HOOI low-rank approximation technique.

A.1 SINGULAR VALUE DECOMPOSITION (SVD)

Before introducing how the SVD low-rank approximation is applied for noise attenuation in data matrices, let us first present some basic concepts regarding the Singular Value Decomposition. The dataset matrix can be expressed as $\mathbf{X} \in \mathbb{R}^{M \times N}$, where M is the number of instances and N is the number of features, with $N < M$. According to the SVD technique, \mathbf{X} can be decomposed into the matrices $\mathbf{U} \in \mathbb{R}^{M \times M}$, $\mathbf{\Sigma} \in \mathbb{R}^{M \times N}$ and $\mathbf{V} \in \mathbb{R}^{N \times N}$ as follows

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^H, \quad (\text{A.1})$$

where \mathbf{U} and \mathbf{V} are the matrices containing the left singular vectors and right singular vectors, respectively, whereas $\mathbf{\Sigma} = \text{diag}\{\sigma_1, \dots, \sigma_N\}$ is a diagonal matrix containing the singular values σ_n for $n = 1, \dots, N$.

Since \mathbf{V} is a unitary matrix, then $\mathbf{V}\mathbf{V}^H = \mathbf{I}_N$, where $\mathbf{I}_N \in \mathbb{R}^{N \times N}$ is the identity matrix. In this sense, \mathbf{V} contains the eigenvectors of the matrix $\mathbf{X}^H\mathbf{X}$, since

$$\mathbf{X}^H\mathbf{X} = \mathbf{V}\mathbf{\Sigma}^H(\mathbf{U}^H\mathbf{U})\mathbf{\Sigma}\mathbf{V}^H = \mathbf{V}(\mathbf{\Sigma}^H\mathbf{\Sigma})\mathbf{V}^H = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^H, \quad (\text{A.2})$$

where $\mathbf{\Lambda} = \mathbf{\Sigma}^H\mathbf{\Sigma} = \text{diag}\{\lambda_1, \dots, \lambda_N\}$ is a diagonal matrix containing the eigenvalues of $\mathbf{X}^H\mathbf{X}$. Therefore, the singular values matrix $\mathbf{\Sigma}$ can be expressed as a function of the eigenvalues λ_n for $n = 1, \dots, N$ as follows

$$\begin{aligned} \mathbf{\Sigma} &= \text{diag}\{[\sigma_1, \dots, \sigma_N]\} \\ &= \text{diag}\{[\sqrt{\lambda_1}, \dots, \sqrt{\lambda_N}]\}, \end{aligned} \quad (\text{A.3})$$

where $\sigma_1 \geq \dots \geq \sigma_N$, i.e., the singular values are sorted in descending order.

After the basic concepts discussed so far, the SVD based noise attenuation for matrices is presented as follows. The dataset matrix can be modeled as $\mathbf{X} = \mathbf{X}_0 + \mathbf{N}$, where $\mathbf{X}_0 \in \mathbb{R}^{M \times N}$ is the noise-free matrix and $\mathbf{N} \in \mathbb{R}^{M \times N}$ is the noise samples matrix. Assuming that the noise and the signal are independent and zero-mean, the covariance matrix $\mathbf{R}_{\mathbf{xx}} \in \mathbb{R}^{N \times N}$ of \mathbf{X} is defined as [161]

$$\begin{aligned}\mathbf{R}_{\mathbf{xx}} &= \mathbb{E}\{\mathbf{X}^H \mathbf{X}\} \\ &= \mathbb{E}\{\mathbf{X}_0^H \mathbf{X}_0\} + \mathbb{E}\{\mathbf{N}^H \mathbf{N}\} \\ &= \mathbf{R}_{\mathbf{ss}} + \mathbf{R}_{\mathbf{nn}},\end{aligned}\tag{A.4}$$

where $\mathbb{E}\{\cdot\}$ denotes the expectation function. Moreover, $\mathbf{R}_{\mathbf{ss}} = \mathbb{E}\{\mathbf{X}_0^H \mathbf{X}_0\} \in \mathbb{R}^{N \times N}$ is the signal covariance matrix and $\mathbf{R}_{\mathbf{nn}} = \mathbb{E}\{\mathbf{N}^H \mathbf{N}\} = \sigma^2 \mathbf{I}_N \in \mathbb{R}^{N \times N}$ denotes the noise covariance matrix, where σ^2 is the noise power.

If the rank d of \mathbf{X} is known, the eigenvalue decomposition of $\mathbf{R}_{\mathbf{xx}}$ in (A.1) can be expressed as

$$\begin{aligned}\mathbf{R}_{\mathbf{xx}} &= \mathbf{V}_s \mathbf{\Lambda}_s \mathbf{V}_s^H + \mathbf{V}_n \mathbf{\Lambda}_n \mathbf{V}_n^H \\ &= \mathbf{V}_s \mathbf{\Lambda}_s \mathbf{V}_s^H + \sigma^2 \mathbf{V}_n \mathbf{V}_n^H,\end{aligned}\tag{A.5}$$

where $\mathbf{\Lambda}_s = \text{diag}\{\lambda_1, \dots, \lambda_d\} \in \mathbb{R}^{d \times d}$ and $\mathbf{\Lambda}_n = \text{diag}\{\sigma^2, \dots, \sigma^2\} = \sigma^2 \mathbf{I}_{N-d} \in \mathbb{R}^{(N-d) \times (N-d)}$ are diagonal matrices containing the signal eigenvalues and noise eigenvalues, respectively. Furthermore, since the columns of \mathbf{V}_s and \mathbf{V}_n contain the signal eigenvectors and noise eigenvectors of $\mathbf{R}_{\mathbf{xx}}$, they are known as signal subspace matrix and noise subspace matrix, respectively.

Instead of computing the eigenvalue decomposition in (A.5), we can perform the SVD of the dataset matrix \mathbf{X} and, consequently, the signal subspace \mathbf{V}_s can be obtained from the d dominant left singular vectors [161]. In this sense, SVD performs the noise attenuation of the dataset matrix \mathbf{X} by truncating the matrices $\mathbf{U} \in \mathbb{R}^{M \times N}$, $\mathbf{\Sigma} \in \mathbb{R}^{N \times N}$ and $\mathbf{V} \in \mathbb{R}^{N \times N}$ up to the signal subspace. Therefore, the SVD low-rank approximation of \mathbf{X} , denoted as $\tilde{\mathbf{X}} \in \mathbb{R}^{M \times N}$, can be expressed as

$$\tilde{\mathbf{X}} = \mathbf{U}_s \mathbf{\Sigma}_s \mathbf{V}_s^H,\tag{A.6}$$

where $\mathbf{\Sigma}_s = \text{diag}\{\sigma_1, \dots, \sigma_d\} \in \mathbb{R}^{d \times d}$ is a diagonal matrix containing the singular values σ_n for $n = 1, \dots, d$. Further, the columns of $\mathbf{U}_s \in \mathbb{R}^{M \times d}$ and $\mathbf{V}_s \in \mathbb{R}^{N \times d}$ correspond to the singular vectors of $\tilde{\mathbf{X}}$.

The connection between the singular values and the matrix rank is one of the most important properties of the SVD [162]. The rank d corresponds to the number of singular values greater than or equal to a given threshold, which can be computed by state-of-the-art model order selection techniques, such as AIC [77, 105], EDC [78] and MDL [79, 105]. Fig. A.1 illustrates the best rank- d approximation of the dataset matrix $\mathbf{X} \in \mathbb{R}^{M \times N}$ based on the Singular Value Decomposition. First, \mathbf{X} is decomposed into the matrices $\mathbf{U} \in \mathbb{R}^{M \times M}$, $\mathbf{\Sigma} \in \mathbb{R}^{M \times N}$ and $\mathbf{V} \in \mathbb{R}^{N \times N}$, as it can be seen in Fig. A.1a. After truncating each matrix to the signal subspace, according to the rank d , the denoised data matrix $\tilde{\mathbf{X}}$ is computed by (A.6), as shown in Fig. A.1b. For the sake of illustration, in Fig. A.1a, the singular value profile σ_n is depicted by a bar plot above the matrix $\mathbf{\Sigma}$. Note that the rank d is defined by the number of singular values greater than or equal to the threshold th [162].

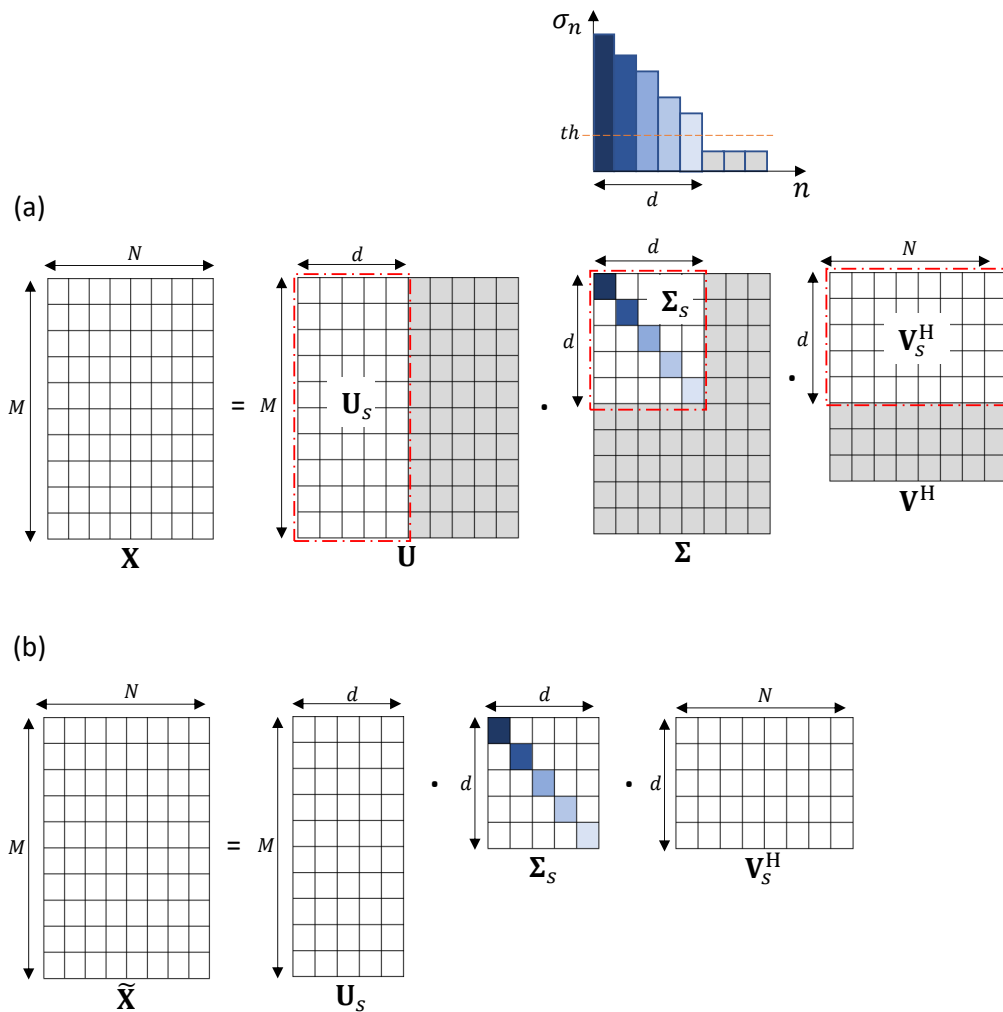


Figure A.1 – (a) SVD of the dataset matrix $\mathbf{X} \in \mathbb{R}^{M \times N}$. The rank d is used to truncate the matrices $\mathbf{U} \in \mathbb{R}^{M \times M}$, $\mathbf{\Lambda} \in \mathbb{R}^{M \times N}$ and $\mathbf{V} \in \mathbb{R}^{N \times N}$ to the signal subspace. (b) Denoised dataset matrix $\tilde{\mathbf{X}} \in \mathbb{R}^{M \times N}$, corresponding to the product $\mathbf{U}_s \mathbf{\Sigma}_s \mathbf{V}_s^H$, where $\mathbf{U}_s \in \mathbb{R}^{M \times d}$, $\mathbf{V}_s \in \mathbb{R}^{N \times d}$ and $\mathbf{\Sigma}_s \in \mathbb{R}^{d \times d}$.

A.2 HIGHER ORDER SINGULAR VALUE DECOMPOSITION (HOSVD)

In this section, a multidimensional extension of the SVD low-rank approximation is defined for higher order tensors. The dataset matrix $\mathbf{X} \in \mathbb{R}^{M \times N}$ can be represented in the tensor form, given by $\mathfrak{X} \in \mathbb{R}^{N_1 \times \dots \times N_R \times M}$, where $N = \prod_{r=1}^R N_r$. If the number of instances M is written as N_{R+1} for simplicity, the Higher Order Singular Value Decomposition of \mathfrak{X} can be expressed as

$$\mathfrak{X} = \mathfrak{G} \times_1 \mathbf{U}_1 \cdots \times_R \mathbf{U}_R \times_{R+1} \mathbf{U}_{R+1}, \quad (\text{A.7})$$

where $\mathfrak{G} \in \mathbb{R}^{N_1 \times \dots \times N_R \times N_{R+1}}$ is the core tensor and $\mathbf{U}_r \in \mathbb{R}^{N_r \times N_r}$ for $r = 1, \dots, R + 1$ are the factor matrices. Each matrix \mathbf{U}_r is computed by applying the SVD on the r -th unfolding matrix of \mathfrak{X} [162], i.e.,

$$[\mathfrak{X}]_{(r)} = \mathbf{U}_r \boldsymbol{\Sigma}_r \mathbf{V}_r^H, \quad (\text{A.8})$$

where $\boldsymbol{\Sigma}_r = \text{diag}\{\sigma_1^{(r)}, \dots, \sigma_{N_r}^{(r)}\} \in \mathbb{R}^{N_r \times N_r}$ is a diagonal matrix containing the singular values of $[\mathfrak{X}]_{(r)}$.

Next, from \mathbf{U}_r for $r = 1, \dots, R + 1$ computed from (A.8), the core tensor \mathfrak{G} can be obtained as follows

$$\mathfrak{G} = \mathfrak{X} \times_1 \mathbf{U}_1^H \cdots \times_R \mathbf{U}_R^H \times_{R+1} \mathbf{U}_{R+1}^H. \quad (\text{A.9})$$

Analogously to the SVD low-rank approximation, the core tensor \mathfrak{G} as well as the factor matrices \mathbf{U}_r for $r = 1, \dots, R + 1$ can be truncated according to the multilinear rank (d_1, \dots, d_{R+1}) . Consequently, the core tensor truncated to the signal subspace is given by $\mathfrak{G}^{[s]} \in \mathbb{R}^{d_1 \times \dots \times d_R \times d_{R+1}}$, whereas the truncated factor matrices are denoted as $\mathbf{U}_r^{[s]} \in \mathbb{R}^{N_r \times d_r}$ for $r = 1, \dots, R + 1$.

In this sense, the denoised data tensor $\tilde{\mathfrak{X}}$ can be expressed as

$$\tilde{\mathfrak{X}} = \mathfrak{G}^{[s]} \times_1 \mathbf{U}_1^{[s]} \cdots \times_R \mathbf{U}_R^{[s]} \times_{R+1} \mathbf{U}_{R+1}^{[s]}, \quad (\text{A.10})$$

where the HOSVD low-rank approximation considers only the first d_r left columns of \mathbf{U}_r corresponding to the d_r largest singular values of $[\mathfrak{X}]_{(r)}$. Fig. A.2 illustrates an example of the HOSVD of a three-dimensional dataset tensor $\mathfrak{X} \in \mathbb{R}^{N_1 \times N_2 \times M}$. The tensor \mathfrak{X} is decomposed into the core tensor $\mathfrak{G} \in \mathbb{R}^{N_1 \times N_2 \times N_3}$ and the factor matrices $\mathbf{U}_r \in \mathbb{R}^{N_r \times N_r}$ for $r = 1, \dots, 3$ in Fig. A.2a. Following, in Fig. A.2b, the core tensor and factor matrices are truncated to the signal subspace according to the multilinear rank (d_1, d_2, d_3) , generating $\mathfrak{G}^{[s]} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ and $\mathbf{U}_r^{[s]} \in \mathbb{R}^{N_r \times d_r}$ for $r = 1, \dots, 3$. Then, the denoised dataset tensor is

given by $\tilde{\mathcal{X}} = \mathcal{G}^{[s]} \times_1 \mathbf{U}_1^{[s]} \times_2 \mathbf{U}_2^{[s]} \times_3 \mathbf{U}_3^{[s]}$. Note that the singular value profiles $\sigma_n^{(1)}$, $\sigma_n^{(2)}$ and $\sigma_n^{(3)}$, with thresholds th_1 , th_2 and th_3 , are depicted above their respective factor matrices.

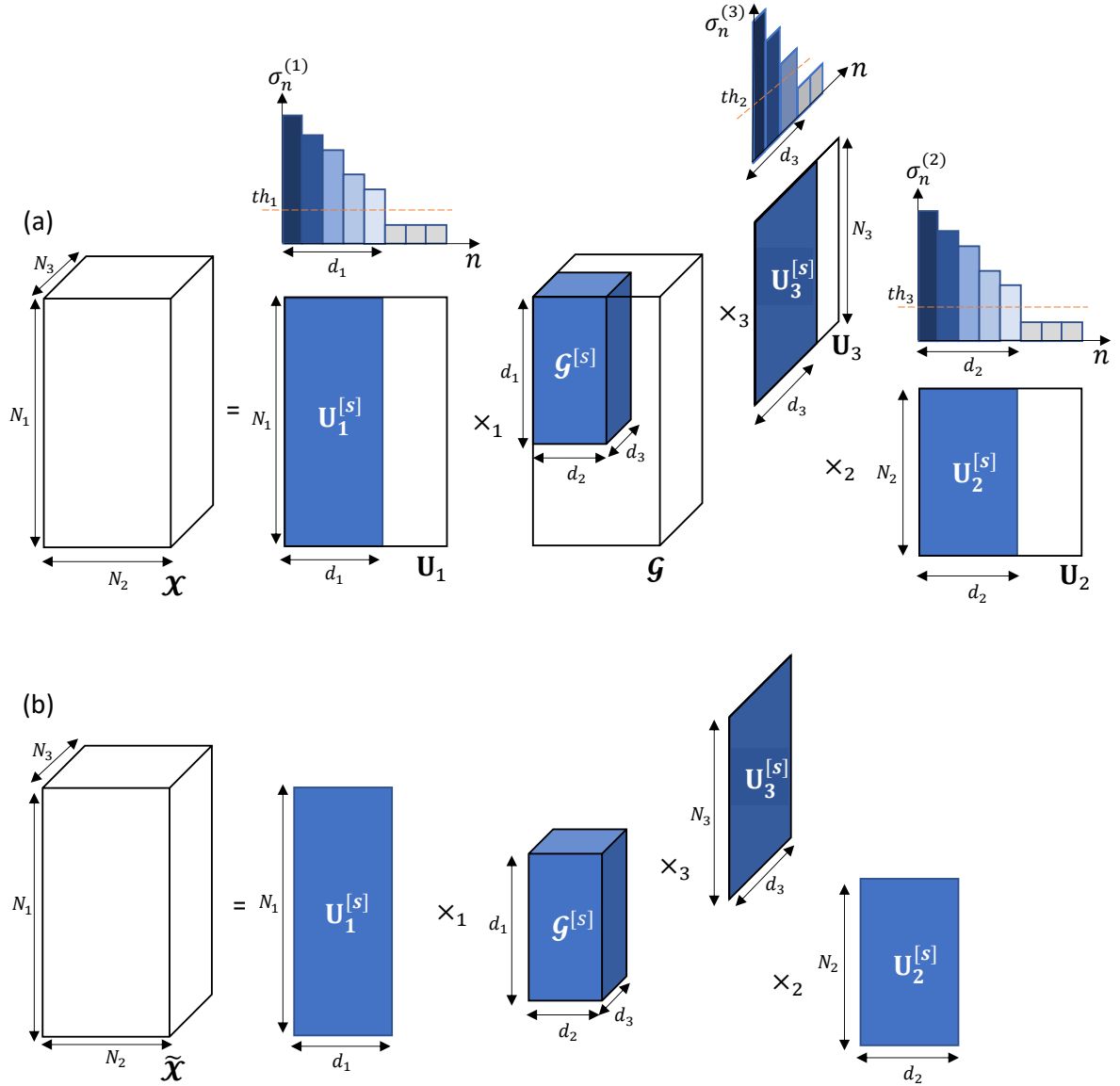


Figure A.2 – (a) HOSVD of the dataset tensor $\mathcal{X} \in \mathbb{R}^{N_1 \times N_2 \times N_3}$. The multilinear rank (d_1, d_2, d_3) is used to truncate the core tensor $\mathcal{G} \in \mathbb{R}^{N_1 \times N_2 \times N_3}$ and the factor matrices $\mathbf{U}_r \in \mathbb{R}^{N_r \times N_r}$ for $r = 1, \dots, 3$. (b) Denoised dataset tensor $\tilde{\mathcal{X}} \in \mathbb{R}^{N_1 \times N_2 \times N_3}$, corresponding to the product $\mathcal{G}^{[s]} \times_1 \mathbf{U}_1^{[s]} \times_2 \mathbf{U}_2^{[s]} \times_3 \mathbf{U}_3^{[s]}$, where $\mathcal{G}^{[s]} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ and $\mathbf{U}_r^{[s]} \in \mathbb{R}^{N_r \times d_r}$ for $r = 1, \dots, 3$.

A.3 HIGHER ORDER ORTHOGONAL ITERATION

Similarly to the HOSVD, the Higher Order Orthogonal Iteration can be considered as a multilinear generalization of the SVD low-rank approximation for matrices. However, HOOI is an iterative technique which finds the best rank- (d_1, \dots, d_{R+1}) tensor $\tilde{\mathbf{X}} \in \mathbb{R}^{N_1 \times \dots \times N_R \times M}$ such that the least squares cost function $\|\mathbf{X} - \tilde{\mathbf{X}}\|^2$ is minimal [108].

The HOOI algorithm estimates the singular matrices $\mathbf{U}_r^{[s]}$ for $r = 1, \dots, R + 1$ by sequentially applying the SVD at each iteration j until some stopping criterion is satisfied [88], i.e.,

$$\begin{aligned}
 [\mathbf{X} \times_2 \mathbf{U}_2^{(j)[s]\text{H}} \times_3 \mathbf{U}_3^{(j)[s]\text{H}} \dots \times_{R+1} \mathbf{U}_{R+1}^{(j)[s]\text{H}}]_{(1)} &= \mathbf{U}_1^{(j+1)[s]} \boldsymbol{\Sigma}_1^{(j+1)[s]} \mathbf{V}_1^{(j+1)[s]\text{H}}, \\
 [\mathbf{X} \times_1 \mathbf{U}_1^{(j+1)[s]\text{H}} \times_3 \mathbf{U}_3^{(j)[s]\text{H}} \dots \times_{R+1} \mathbf{U}_{R+1}^{(j)[s]\text{H}}]_{(2)} &= \mathbf{U}_2^{(j+1)[s]} \boldsymbol{\Sigma}_2^{(j+1)[s]} \mathbf{V}_2^{(j+1)[s]\text{H}}, \\
 &\vdots \\
 [\mathbf{X} \times_1 \mathbf{U}_1^{(j+1)[s]\text{H}} \times_2 \mathbf{U}_2^{(j+1)[s]\text{H}} \dots \times_R \mathbf{U}_R^{(j+1)[s]\text{H}}]_{(R+1)} &= \mathbf{U}_{R+1}^{(j+1)[s]} \boldsymbol{\Sigma}_{R+1}^{(j+1)[s]} \mathbf{V}_{R+1}^{(j+1)[s]\text{H}},
 \end{aligned} \tag{A.11}$$

where, at $j = 0$, the HOOI is initialized with the factor matrices obtained from (A.8) through HOSVD.

From the factor matrices $\mathbf{U}_r^{(J)[s]}$ for $r = 1, \dots, R + 1$ obtained after the J -th iteration of the HOOI algorithm in (A.11), where J is the iteration in which the stop condition was satisfied, the core tensor $\mathfrak{G}^{[s]} \in \mathbb{R}^{d_1 \times \dots \times d_R \times d_{R+1}}$ can be computed as follows

$$\mathfrak{G}^{[s]} = \mathbf{X} \times_1 \mathbf{U}_1^{(J)[s]\text{H}} \dots \times_R \mathbf{U}_R^{(J)[s]\text{H}} \times_{R+1} \mathbf{U}_{R+1}^{(J)[s]\text{H}}. \tag{A.12}$$

Finally, from the core tensor $\mathfrak{G}^{[s]}$ found in (A.12), as well as the factor matrices $\mathbf{U}_r^{(J)[s]}$ for $r = 1, \dots, R + 1$ obtained from (A.11), the denoised dataset tensor $\tilde{\mathbf{X}} \in \mathbb{R}^{N_1 \times \dots \times N_R \times M}$ computed via the HOOI algorithm is given by

$$\tilde{\mathbf{X}} = \mathfrak{G}^{[s]} \times_1 \mathbf{U}_1^{(J)[s]} \dots \times_R \mathbf{U}_R^{(J)[s]} \times_{R+1} \mathbf{U}_{R+1}^{(J)[s]}. \tag{A.13}$$

B TRIANGULATION

In this appendix, we illustrate how triangulation techniques can be used to derive the coordinates of an intruding Unmanned Aerial Vehicle in a Cartesian coordinate system by using the position coordinates of two antenna arrays.

Let us consider the antenna array based UAV localization system composed by U 2D URAs depicted in Figure 2.1a of Chapter 2. By using the origin $(0, 0, 0)$ of the Cartesian coordinate system as reference, two URAs are sufficient to determine the UAV coordinates. Fig. B.1a illustrates the localization of the q -th UAV, via triangulation, by using the u -th and v -th URAs, whereas the same technique, projected onto the xy -plane, is shown in Fig. B.1b. As mentioned in Chapter 2, the coordinates $(\hat{x}_q^{u,v}, \hat{y}_q^{u,v}, \hat{z}_q^{u,v})$ of the q -th UAV, estimated by both URAs, are given by

$$\begin{cases} \hat{x}_q^{u,v} = x_u - \hat{l}_u^q \sin \hat{\theta}_{u,\text{los}}^q \sin \hat{\varphi}_{u,\text{los}}^q, \\ \hat{y}_q^{u,v} = y_u + \hat{l}_u^q \sin \hat{\theta}_{u,\text{los}}^q \cos \hat{\varphi}_{u,\text{los}}^q, \\ \hat{z}_q^{u,v} = \hat{l}_u^q \cos \hat{\theta}_{u,\text{los}}^q, \end{cases} \quad (\text{B.1})$$

if the u -th URA is used as a reference, or

$$\begin{cases} \hat{x}_q^{u,v} = x_v - \hat{l}_v^q \sin \hat{\theta}_{v,\text{los}}^q \sin \hat{\varphi}_{v,\text{los}}^q, \\ \hat{y}_q^{u,v} = y_v + \hat{l}_v^q \sin \hat{\theta}_{v,\text{los}}^q \cos \hat{\varphi}_{v,\text{los}}^q, \\ \hat{z}_q^{u,v} = \hat{l}_v^q \cos \hat{\theta}_{v,\text{los}}^q, \end{cases} \quad (\text{B.2})$$

if, alternatively, the v -th URA is used as a reference, regardless of the relative position between the URAs and the UAV. In both systems of equations, $\hat{\theta}_{u,\text{los}}^q$ and $\hat{\theta}_{v,\text{los}}^q$ are the elevation angles at the u -th and v -th URAs, respectively. Furthermore, $\hat{\varphi}_{u,\text{los}}^q$ and $\hat{\varphi}_{v,\text{los}}^q$ correspond to the angles between the y -direction and the LOS from the UAV projected onto the xy -plane in the counterclockwise rotation at the u -th and v -th URAs, as described in Section 2.3 of Chapter 2. Additionally, \hat{l}_u^q and \hat{l}_v^q are the distances between $(\hat{x}_q^{u,v}, \hat{y}_q^{u,v}, \hat{z}_q^{u,v})$ and the coordinates of each URA, (x_u, y_u, z_u) and (x_v, y_v, z_v) , respectively.

Since all of the URAs are positioned along the xy -plane, the triangulation process used to estimate the coordinates of the q -th UAV can be projected onto the mentioned plane, as shown in Fig. B.2 to B.5, in a process similar to the one performed in Fig. B.1. In each figure, the xy -plane is divided into six regions, each of which corresponding to a different relative position between the UAV and the URAs. The projection of the q -th UAV onto the xy -plane is represented by cross markers in orange color, whereas URAs are represented by circle markers in black color. Moreover, the lines-of-sight from the UAV to the u -th and

v -th URAs, projected onto the xy -plane, are represented by lines in red color and denoted as $\hat{l}_u^q \sin \hat{\theta}_{u,\text{los}}^q$ and $\hat{l}_v^q \sin \hat{\theta}_{v,\text{los}}^q$, respectively.

In Fig. B.2 and B.3, the u -th URA is considered at a top position compared to the v -th URA ($x_u < x_v$), whereas the opposite situation is shown in Fig. B.4 and B.5 ($x_u > x_v$). Furthermore, the azimuth angles are written as a function of $\hat{\varphi}_{u,\text{los}}^q$ and $\hat{\varphi}_{v,\text{los}}^q$. Tables B.1 and B.2 present the coordinates of the q -th UAV, computed by using the u -th and v -th URAs, for each region of the Cartesian coordinate system shown in Fig. B.2 to Fig. B.5. In each table, after some mathematical operations, the results shown in rows 1 to 12 boil down to the same coordinates presented in the last row. In this sense, we conclude that the coordinates of the q -th UAV, obtained by triangulation techniques applied on the u -th and v -th URAs, generate the same results, regardless of the relative positions between the UAV and URAs.

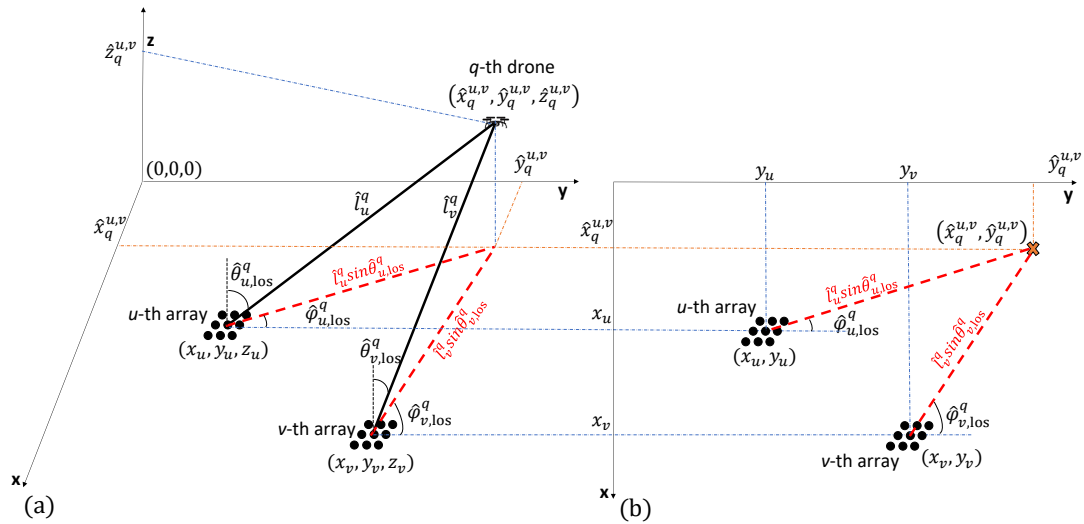
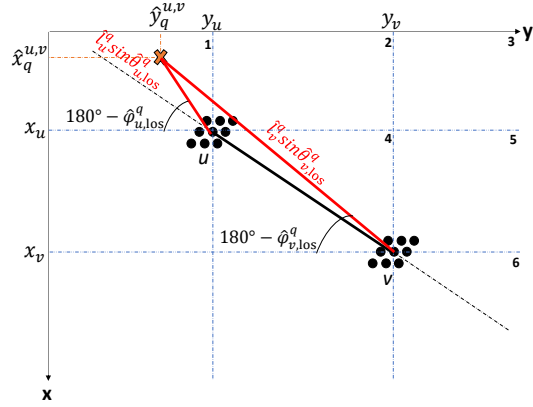
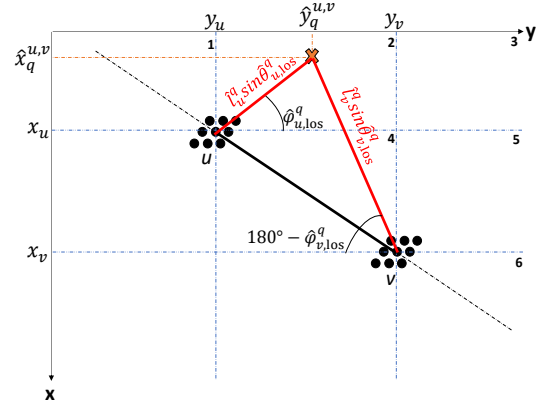


Figure B.1 – (a) Triangulation technique used to localize the q -th UAV by using the u -th and v -th URAs. (b) Triangulation technique, projected onto the xy -plane.

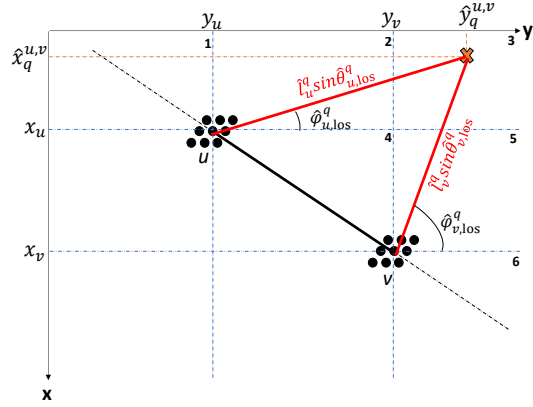
(a) Position 1



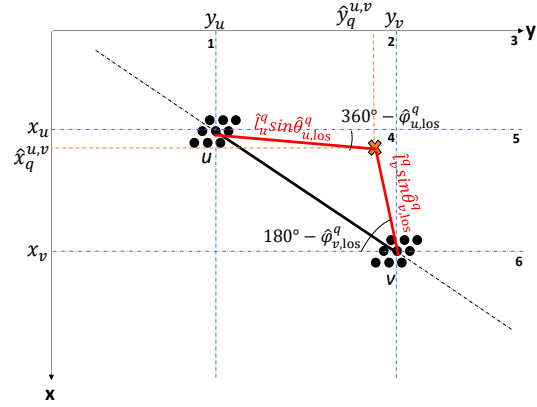
(b) Position 2



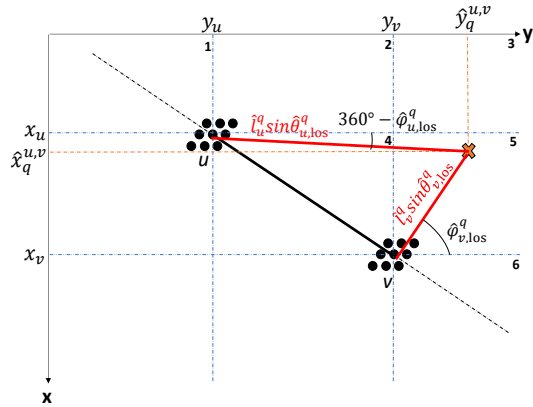
(c) Position 3



(d) Position 4



(e) Position 5



(f) Position 6

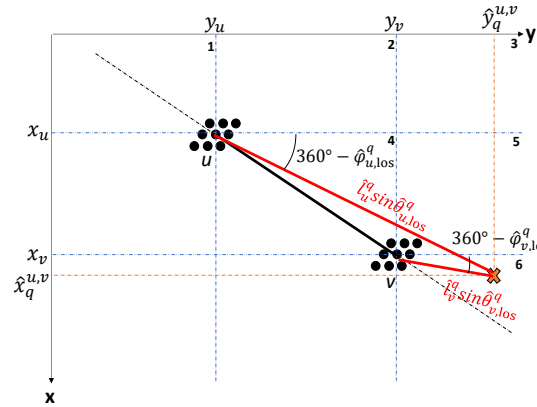
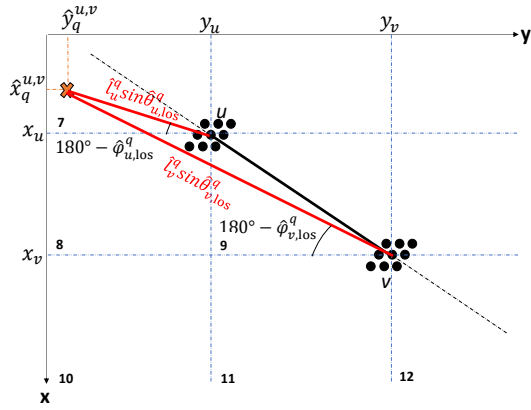
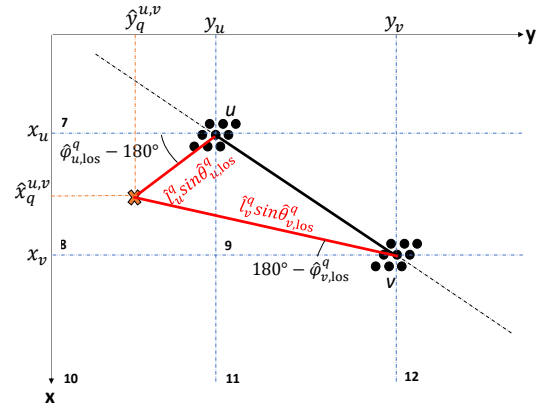


Figure B.2 – Localization of the q -th UAV via triangulation, projected onto the xy -plane, considering positions 1 to 6 ($x_u < x_v$).

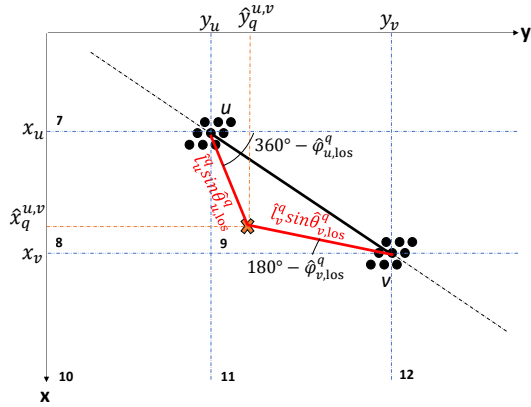
(a) Position 7



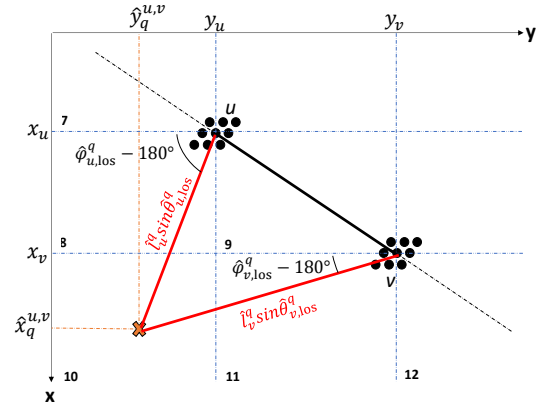
(b) Position 8



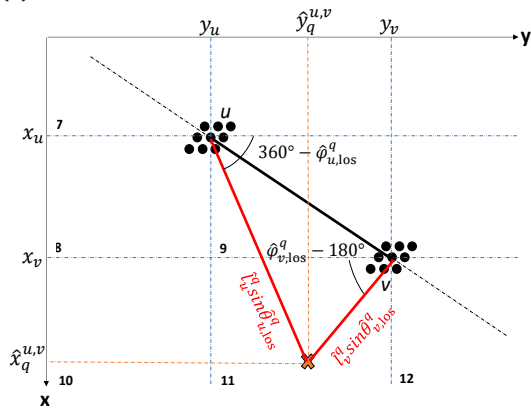
(c) Position 9



(d) Position 10



(e) Position 11



(f) Position 12

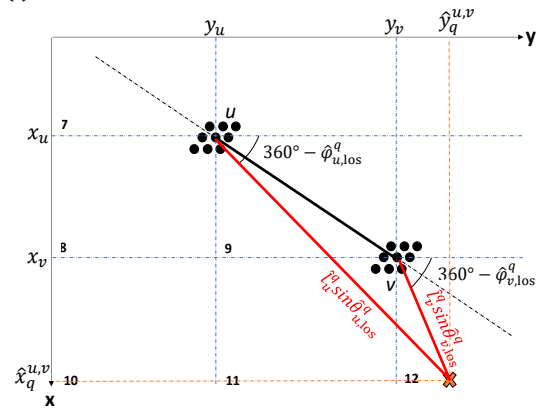
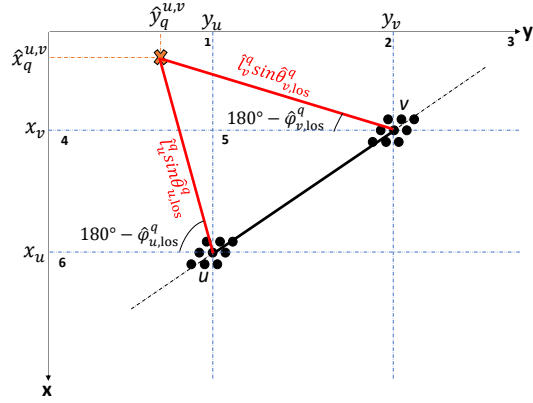
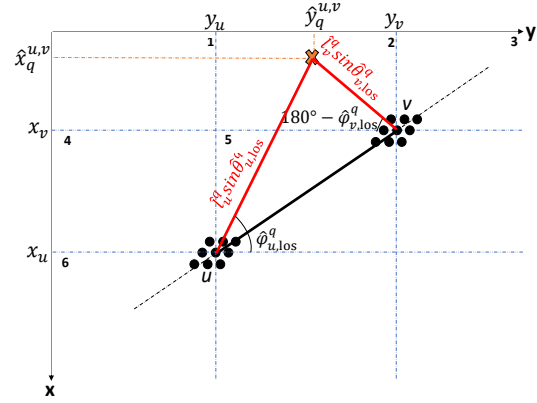


Figure B.3 – Localization of the q -th UAV via triangulation, projected onto the xy -plane, considering positions 7 to 12 ($x_u < x_v$).

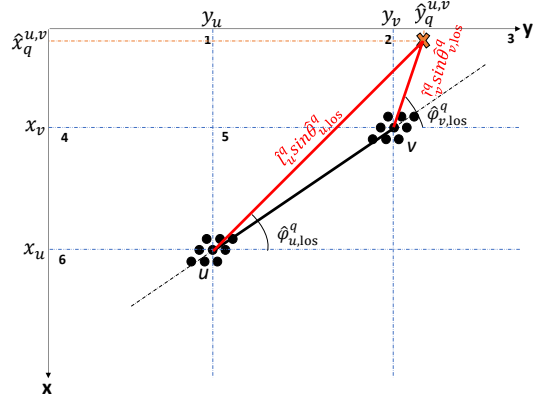
(a) Position 1



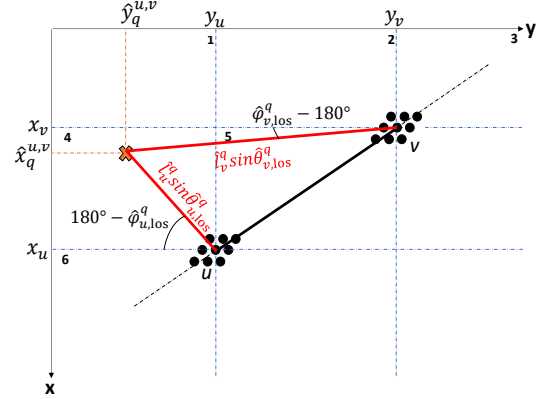
(b) Position 2



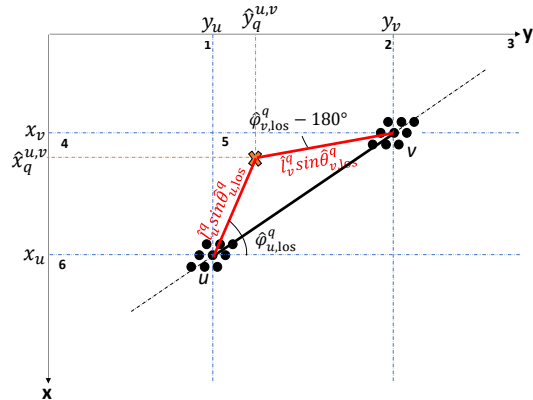
(c) Position 3



(d) Position 4



(e) Position 5



(f) Position 6

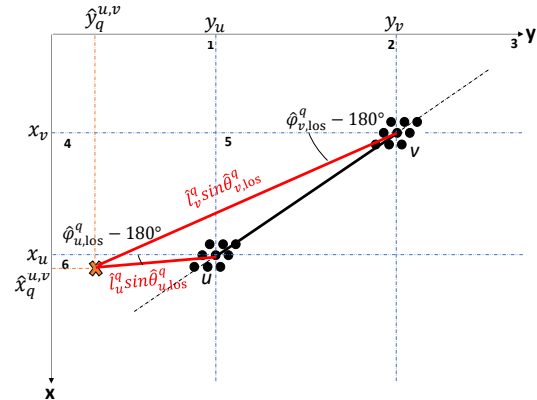
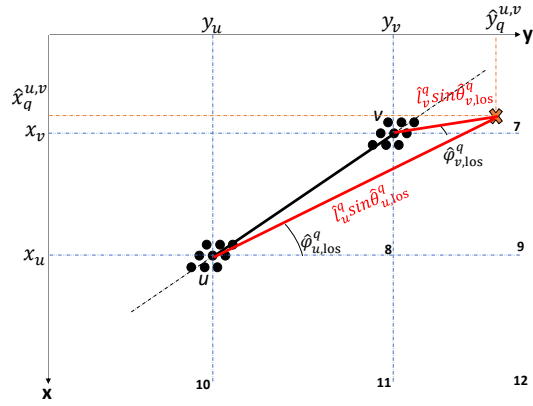
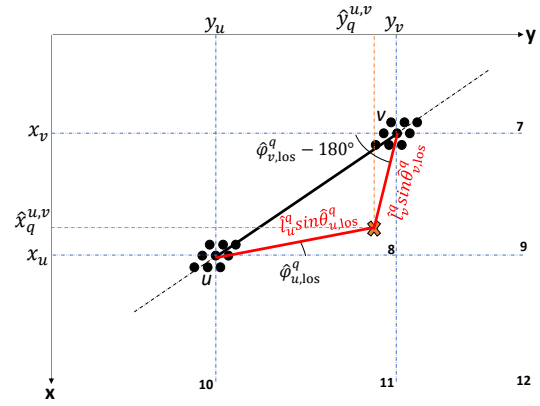


Figure B.4 – Localization of the q -th UAV via triangulation, projected onto the xy -plane, considering positions 1 to 6 ($x_u > x_v$).

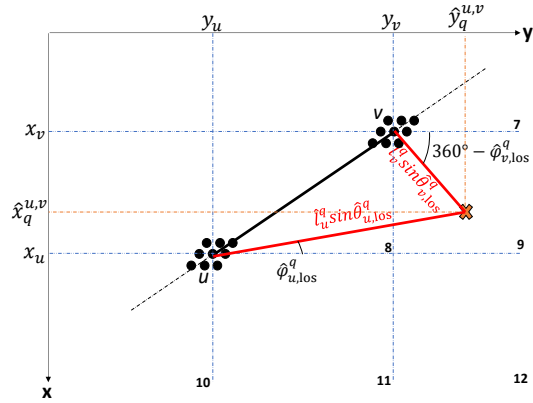
(a) Position 7



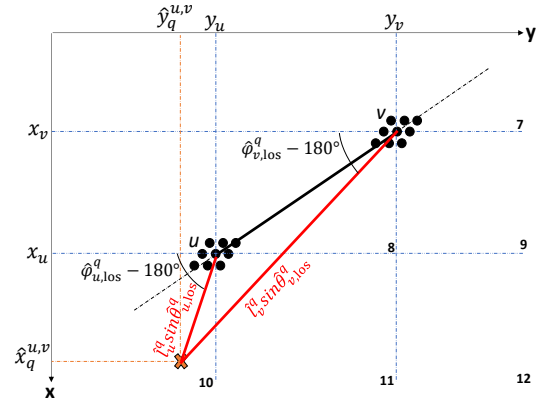
(b) Position 8



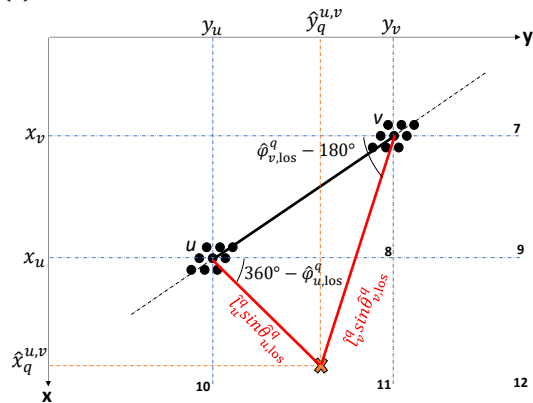
(c) Position 9



(d) Position 10



(e) Position 11



(f) Position 12

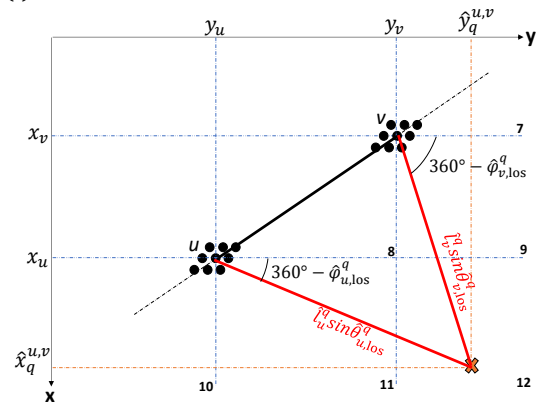


Figure B.5 – Localization of the q -th UAV via triangulation, projected onto the xy -plane, considering positions 7 to 12 ($x_u > x_v$).

C RF SENSING BASED DRONE IDENTIFICATION AND DDoS ATTACK DATASETS

In this appendix, details regarding the RF-based drone identification dataset used in Chapter 2 are introduced. Furthermore, the DDoS attack benchmark datasets used in Chapters 3 to 5 are discussed as well.

C.1 RF SENSING BASED DRONE IDENTIFICATION DATASET

This section details the main characteristics of the DroneRF database, which is the RF sensing based drone identification dataset used in this thesis. The number of instances collected from the DroneRF dataset and used for model validation are shown in Table 2.5 of Chapter 2.

DroneRF is a novel radio frequency based dataset which contains 227 recorded segments collected from three different types of commercial drones, namely, Parrot Bebop 1 [163], Parrot AR 2.0 Elite [164] and DJI Phantom 3 [165]. Each segment is composed by two equally sized parts containing 1 million samples each, with 454 record files in total. Such samples correspond to the amplitude of the received RF signals in the time domain. The data has been collected by RF receivers which intercepted communications between the UAV and its flight controller. The DroneRF database is publicly available for students and researchers in CSV format [85], and can be used to train and test machine learning based models for detecting and identifying the three above-mentioned UAV models. Furthermore, the dataset contains signatures of drones operating in different modes, organized into three identification levels, as follows:

- Level 1: the intruding drone is detected as “on” or “off” by the IDS. In the former case, the drone RF activities are recorded in the dataset, whereas only RF background activities are recorded in the latter case.
- Level 2: once the intruding drone is detected as “on”, then the IDS identifies its model as one of the following options: “Parrot Bebop 1”, “Parrot AR 2.0 Elite” or “DJI Phantom 3”.
- Level 3: once the intruding drone model is identified, then the IDS identifies its flight

mode as one of the following options: “on and connected to the controller”, “hovering automatically”, “flying without video recording”, or “flying with video recording”.

C.2 DDoS ATTACK DATASETS

This section introduces the NSL-KDD, CIC-IDS2017, CIC-IDS2018 and CIC-DDoS2019 benchmark datasets, provided by the Canadian Institute for Cybersecurity (CIC) of the University of New Brunswick (UNB). The DDoS attack types and the corresponding number of instances collected from each dataset and used for model validation are shown in the respective Simulation Results sections of Chapters 3 to 5.

NSL-KDD is a benchmark dataset used to design network IDSs proposed by Tavallaee et al. [166]. It is a distilled version of the well-known KDD Cup 99 dataset, which presented several redundant records and, consequently, biased evaluation results were shown by NIDS trained with such data. The training and testing samples of the NSL-KDD dataset are contained in two different sets called KDDTrain+ and KDDTest+, respectively. Moreover, NSL-KDD has 41 features and different types of network attacks divided into four major categories: Probe, Denial of Service (DoS), User to Root (U2R) and Remote to Local (R2L) [167]. In a probing attack, an attacker gathers information about a network in order to find its vulnerabilities. On the other hand, in DoS attacks, legitimate users are prevented from using a service after a massive attack is launched on the target server. Moreover, in U2R attacks, an attacker with access to a normal user account exploits some vulnerability in order to gain super-user privilege. Finally, in R2L attacks, an attacker tries to gain access to the victim’s machine without having an account on it [116]. The DoS attacks present in the NSL-KDD dataset and used in this thesis are: Neptune, Teardrop, Smurf, Pod, Back, Land, UDPStorm, Apache2, ProcessTable and MailBomb. Further, five of the 41 features of the NSL-KDD dataset were ignored, namely, count, protocol_type, service, flag and num_outbound_cmds. The first and second one were neglected because they presented only zero values, while the other three features were deleted because they were all nominal. Table C.1 illustrates the NSL-KDD features used in Chapter 3. Further, all legitimate and DDoS attack instances are labeled as 0 and 1, respectively.

CIC-IDS2017 is a completely labeled benchmark dataset provided by the Canadian Institute of Cybersecurity for network intrusion detection models, with 87 network traffic features. The dataset contains legitimate traffic and the most up-to-date common network attacks, such as DDoS, Denial of Service (DoS), Brute Force, Cross-Site Scripting (XSS), SQL Injection, Infiltration, Port Scan and Botnet [168]. All legitimate and malicious traffics are stored in CSV and PCAP files, organized according to the date and time of the data capturing, and are publicly available in [131]. Since this thesis focus on Distributed Denial of

Table C.1 – NSL-KDD dataset features used in Chapter 3.

Nr	Feature Name	Nr	Feature Name
1	duration	19	srv_count
2	src_bytes	20	serror_rate
3	dst_bytes	21	srv_serror_rate
4	land	22	rerror_rate
5	wrong_fragment	23	srv_rerror_rate
6	urgent	24	same_srv_rate
7	hot	25	diff_srv_rate
8	num_failed_logins	26	srv_diff_host_rate
9	logged_in	27	dst_host_count
10	num_compromised	28	dst_host_srv_count
11	root_shell	29	dst_host_same_srv_rate
12	su_attempted	30	dst_host_diff_srv_rate
13	num_root	31	dst_host_same_src_port_rate
14	num_file_creations	32	dst_host_srv_diff_host_rate
15	num_shells	33	dst_host_serror_rate
16	num_access_files	34	dst_host_srv_serror_rate
17	is_host_login	35	dst_host_rerror_rate
18	is_guest_login	36	dst_host_srv_rerror_rate

Service attack detection, only legitimate and DDoS instances are extracted.

Furthermore, CIC-IDS2018 is an updated version of the CIC-IDS2017 which presents multiple network attack and legitimate traffic instances recorded in PCAP and CSV files [168]. The dataset contains ten days of benign and malicious activities performed in a controlled simulation environment, from Feb 14th, 2018, to March 2nd, 2018, and is publicly available in [132]. Malicious samples include common network attacks, such as FTP-Brute Force, DoS-Golden Eye, SQL Injection, DDoS-HOIC and Infiltration. In addition, since we focus on DDoS attack detection, legitimate and DDoS attack samples were collected from the traces captured in February 20th, 2018, and February 21st, 2018, respectively.

Finally, CIC-DDoS2019 is one of the most up-to-date DDoS attack datasets available on the web [89]. It is completely labeled and contains more than 80 network traffic features with millions of instances and different DDoS attack types. The authors divided the DDoS attacks present in the CIC-DDoS2019 dataset into two categories: reflection-based and exploitation-based [120]. In the first category, an attacker, using spoofed Internet Protocol (IP) addresses, sends several request packets to a server which replies directly to the forged IPs, overwhelming the victim’s bandwidth or resources. Such attacks become more efficient when traffic amplification is used, i.e., when the response size is much larger than the request size. The reflection-based DDoS attacks used in this thesis include Domain Name System (DNS), Lightweight Directory Access Protocol (LDAP), Microsoft Structured Query Language (MSSQL), Network Basic Input/Output System (NetBIOS), Network Time Protocol (NTP), Simple Network Management Protocol (SNMP), Simple Service Discovery Pro-

ocol (SSDP), Trivial File Transfer Protocol (TFTP) and DNS based attacks. On the other hand, exploitation-based DDoS attacks usually consume server resources by exploiting protocol vulnerabilities. In this thesis, SYN flood and UDP flood are used as exploitation-based attacks. Such attacks can also be performed by a reflection structure composed by several compromised machines and spoofed IP addresses, similarly to the reflection-based category. The capturing days of the training and testing sets are January 12th, 2019 and March 11th, 2019, respectively.

From the original features present in the CIC-IDS2017, CIC-IDS2018 and CIC-DDoS2019 datasets, only 64 are considered in this thesis. The feature selection process is summarized as follows. Nine features were eliminated because they presented only zero values and, consequently, would not provide any contribution to the machine learning classification algorithms. Such features are: FwdURGFlags, BwdPSHFlags, BwdURGFlags, FwdAvgBytes_Bulk, FwdAvgPackets_Bulk, FwdAvgBulkRate, BwdAvgBytes_Bulk, BwdAvgPackets_Bulk and BwdAvgBulkRate. Moreover, one feature was wrongly written twice in the dataset (FwdHeaderLength and FwdHeaderLength_1) and, consequently, one of such identical copies was ignored. Furthermore, as we intend to develop a DDoS attack detection system regardless of IP addresses, protocols and date/time information, four more features were neglected: Source IP, Destination IP, Protocol and Timestamp. Finally, the features Unnamed_0, FlowID, SimilarHTTP, Inbound, FlowIATStdDev, FwdIATStdDev, BwdIATStdDev, FwdPUSHFlag and StdDevTimeIdleFlow were also eliminated, since they did not provide any useful information for our proposed technique. Table C.2 describes all the CIC-DDoS2019, CIC-IDS2018 and CIC-IDS2017 dataset features used in Chapters 3 to 5. More details about each feature can be found in [169]. In addition, the types of network traffic present in the CIC-DDoS2019, CIC-IDS2018 and CIC-IDS2017 datasets, along their respective capturing days, are summarized in Table C.3.

Table C.2 – CIC-IDS2017, CIC-IDS2018 and CIC-DDoS2019 dataset features used in Chapters 3 to 5.

Nr	Feature Name	Nr	Feature Name	Nr	Feature Name
1	Source Port	23	Fwd IAT Max	44	CWE Flag Count
2	Destination Port	24	Fwd IAT Min	45	ECE Flag Count
3	Flow Duration	25	Bwd IAT Total	46	Download/Upload Ratio
4	Total Fwd Packet	26	Bwd IAT Avg	47	Avg Packet Size
5	Total Bwd Packet	27	Bwd IAT Max	48	Avg Fwd Segment Size
6	Total Length Fwd Packet	28	Bwd IAT Min	49	Avg Bwd Segment Size
7	Total Length Bwd Packet	29	Fwd Header Length	50	Subflow Fwd Packets
8	Fwd Packet Length Max	30	Bwd Header Length	51	Subflow Fwd Bytes
9	Fwd Packet Length Min	31	Fwd Packet/s	52	Subflow Bwd Packets
10	Fwd Packet Length Avg	32	Bwd Packet/s	53	Subflow Bwd Bytes
11	Fwd Packet Length Std Dev	33	Packet Length Min	54	Fwd Win Bytes
12	Bwd Packet Length Max	34	Packet Length Max	55	Bwd Win Bytes
13	Bwd Packet Length Min	35	Packet Length Avg	56	Fwd Active Data Packet
14	Bwd Packet Length Avg	36	Packet Length Std Dev	57	Fwd Min Segment Size
15	Bwd Packet Length Std Dev	37	Packet Length Var	58	Avg Time Active Flow
16	Flow Bytes/s	38	FIN Flag Count	59	Std Dev Time Active Flow
17	Flow Packets/s	39	SYN Flag Count	60	Max Time Active Flow
18	Flow IAT Avg	40	RST Flag Count	61	Min Time Active Flow
19	Flow IAT Max	41	PUSH Flag Count	62	Avg Time Idle Flow
20	Flow IAT Min	42	ACK Flag Count	63	Std Dev Time Idle Flow
21	Fwd IAT Total	43	URG Flag Count	64	Min Time Idle Flow
22	Fwd IAT Avg				

Table C.3 – Capturing days and network traffic present in the CIC-IDS2017, CIC-IDS2018 and CIC-DDoS2019 datasets.

Dataset	Days	Captured Network Traffic
CIC-DDoS2019	Jan. 12th, 2019	PortMap, NetBIOS, LDAP, MSSQL, UDP, UDP-Lag, Syn
	Mar. 11th, 2019	NTP, DNS, LDAP, MSSQL, NetBIOS, SNMP, SSDP, UDP, UDP-Lag, WebDDoS, Syn, TFTP
CIC-IDS2018	Feb 20th, 2018	Legitimate
	Feb 21st, 2018	HTTP and UDP-based DDoS
CIC-IDS2017	July 3rd, 2017	Legitimate
	July 7th, 2017	HTTP and UDP-based DDoS
NSL-KDD	–	Legitimate
	–	Neptune, Teardrop, Smurf, Pod, Back, Land, UDPStorm, Apache2, ProcessTable and MailBomb

D PERFORMANCE EVALUATION METRICS

In this appendix, we present the main metrics adopted for performance and noise-robustness evaluation of the NIDSs assessed throughout this thesis. All of the metrics can be extracted from the well-known confusion matrix, represented in Table D.1. Such matrix allows us to visualize all of the possible cases of classification. Based on the values of True Positives (TPs), False Positives (FPs), True Negatives (TNs) and False Negatives (FNs) provided by the confusion matrix, several metrics commonly used for performance evaluation can be computed.

Table D.1 – Confusion matrix.

		Predicted Class	
		Positive	Negative
Actual Class	Positive	TP	FN
	Negative	FP	TN

Accuracy (Acc), Precision (Prec), Recall (Rec), False Positive Rate (FPR), False Negative Rate (FNR), F-Score and Matthews Correlation Coefficient (MCC) are the performance evaluation metrics used in this work, which are defined as follows:

- Accuracy (Acc): the ratio between the correctly predicted instances and the total number of instances,

$$\text{Acc} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}. \quad (\text{D.1})$$

- Precision (Prec): the ratio between the correctly predicted positive instances and the total number of predicted positive instances,

$$\text{Prec} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \quad (\text{D.2})$$

- Recall (Rec) or Detection Rate (DR): the ratio between the correctly predicted positive instances and the total number of actual positive instances,

$$\text{Rec} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \quad (\text{D.3})$$

- False Positive Rate (FPR) or False Alarm Rate (FAR): the ratio between the number of negative instances wrongly classified as positives and the total number of actual

negative instances,

$$\text{FPR} = \frac{\text{FP}}{\text{TN} + \text{FP}}. \quad (\text{D.4})$$

- False Negative Rate (FNR): the ratio between the number of positive instances wrongly classified as negatives and the total number of actual positive instances,

$$\text{FNR} = \frac{\text{FN}}{\text{FN} + \text{TP}}. \quad (\text{D.5})$$

- F-Score or F-Measure: corresponds to the harmonic mean of precision and recall,

$$\text{F-Score} = \frac{2 \cdot \text{Prec} \cdot \text{Rec}}{\text{Prec} + \text{Rec}}. \quad (\text{D.6})$$

- Matthews Correlation Coefficient (MCC): measures the quality of binary classifications. It ranges from -1 to $+1$ such that higher values represent better performance. The MCC is defined as

$$\text{MCC} = \frac{(\text{TP} \cdot \text{TN}) - (\text{FP} \cdot \text{FN})}{\sqrt{(\text{TP} + \text{FP}) \cdot (\text{TP} + \text{FN}) \cdot (\text{TN} + \text{FP}) \cdot (\text{TN} + \text{FN})}}. \quad (\text{D.7})$$

In addition, the Relative Loss of Accuracy (RLA) and Relative Loss of Detection Rate (RLDR) are adopted as noise-robustness evaluation metrics. Both can be expressed as follows:

- Relative Loss of Accuracy: measures the percentage of variation of the accuracy of the classifiers at the noise level $x\%$, $\text{Acc}_{x\%}$, with respect to the original case with no additional noise, $\text{Acc}_{0\%}$,

$$\text{RLA} = \frac{\text{Acc}_{0\%} - \text{Acc}_{x\%}}{\text{Acc}_{0\%}}. \quad (\text{D.8})$$

- Relative Loss of Detection Rate: measures the percentage of variation of the detection rate of the classifiers at the noise level $x\%$, $\text{DR}_{x\%}$, with respect to the original case with no additional noise, $\text{DR}_{0\%}$,

$$\text{RLDR} = \frac{\text{DR}_{0\%} - \text{DR}_{x\%}}{\text{DR}_{0\%}}. \quad (\text{D.9})$$

E SIMULATION PARAMETERS ADOPTED FOR ML AND DL CLASSIFICATION ALGORITHMS

In this appendix, we present the main simulation parameters adopted for state-of-the-art machine learning and deep learning classification algorithms throughout this thesis, which are summarized in Table E.1.

Table E.1 – Main simulation parameters adopted for the state-of-the-art ML and DL classification algorithms used in this thesis.

ML/DL Classifier	Parameters
Convolutional Neural Network-1D 1 layer	Batch size: 128, Nr epochs: 20, 1 X Conv1D, Nr conv filters: 16, Conv filter size: 3, Stride: 1, Padding: "same", Batch Normalization: "yes", Activation function: ReLU, Pooling function: Avg Pooling, Pooling filter size: 2, Dropout: 0.2, Classification function: Softmax, Loss function: categorical crossentropy, Optimizer: Adam
Convolutional Neural Network-2D 1 layer	Batch size: 128, Nr epochs: 20, 1 X Conv2D, Nr conv filters: 16, Conv filter size: 3×3 , Stride: 1, Padding: "same", Batch Normalization: "yes", Activation function: ReLU, Pooling function: Avg Pooling, Pooling filter size: 2×2 , Dropout: 0.2, Classification function: Softmax, Loss function: categorical crossentropy, Optimizer: Adam
Convolutional Neural Network-1D 2 layer	Batch size: 128, Nr epochs: 20, 2 X Conv1D, Nr conv filters: 16 and 32, Conv filter size: 3, Stride: 1, Padding: "same", Batch Normalization: "yes", Activation function: ReLU, Pooling function: Avg Pooling, Pooling filter size: 2, Dropout: 0.2, Classification function: Softmax, Loss function: categorical crossentropy, Optimizer: Adam
Convolutional Neural Network-2D 2 layer	Batch size: 128, Nr epochs: 20, 2 X Conv2D, Nr conv filters: 16 and 32, Conv filter size: 3×3 , Stride: 1, Padding: "same", Batch Normalization: "yes", Activation function: ReLU, Pooling function: Avg Pooling, Pooling filter size: 2×2 , Dropout: 0.2, Classification function: Softmax, Loss function: categorical crossentropy, Optimizer: Adam
Gated Recurrent Unit 2 layer	Batch size: 128, Nr epochs: 20, 2 X GRU, Nr Units: 20 and 10, Activation function: tanh, Recurrent activation: sigmoid, Kernel initializer: glorot uniform, Recurrent initializer: orthogonal, Dropout: 0.2, Classification function: Softmax, Loss function: categorical crossentropy, Optimizer: Adam
Long Short-Term Memory 2 layer	Batch size: 128, Nr epochs: 20, 2 X LSTM, Nr Units: 20 and 10, Activation function: tanh, Recurrent activation: sigmoid, Kernel initializer: glorot uniform, Recurrent initializer: orthogonal, Unit forget bias: "true", Dropout: 0.2, Classification function: Softmax, Loss function: categorical crossentropy, Optimizer: Adam
Simple Recurrent Neural Network 2 layer	Batch size: 128, Nr epochs: 20, 2 X SimpleRNN, Nr Units: 20 and 10, Activation function: tanh, Kernel initializer: glorot uniform, Recurrent initializer: orthogonal, Dropout: 0.2, Classification function: Softmax, Loss function: categorical crossentropy, Optimizer: Adam
Multilayer Perceptron 5 layer	Batch size: 128, Nr hidden layers: 5, Nr epochs: 20, Neuron config: 100/80/60/40/20, Activation function: ReLU, Classification function: Softmax, Optimizer: Adam
AdaBoost	Base estimator: decision tree, Nr estimators: 100, Learning rate: 1, Algorithm: SAMME.R, Random state: "none"

Continued on next page...

Table E.1 – continued from previous page.

ML/DL Classifier	Parameters
Decision Trees	Criterion: "gini", Splitter: "best", Max depth: "none", Min samples split: 2, Min samples leaf: 1, Max features: "None", Max leaf nodes: "None", Min impurity decrease: 0, Class weight: "none"
Extra Trees	Nr estimators: 100, Min samples split: 2, Min samples leaf: 20, Max features: "auto", Max depth: 90, Criterion: "gini"
Gradient Boosting	Nr estimators: 100, Min samples split: 2, Min samples leaf: 20, Max features: "auto", Max depth: 90, Criterion: "friedman_mse", Loss: "deviance", Learning rate: 0.1
k-Nearest Neighbors	Nr neighbors: 5, Weights: "uniform", Algorithm: "auto", Leaf size: 30, Metric: "Minkowski", Power parameter: 2
Linear Discriminant Analysis	Solver: "svd", Shrinkage: "none", Tol: 1e-4, Priors: "none", Nr components: min(Nr classes - 1, Nr features)
Logistic Regression	Solver: "lbfgs", Max iter: 1000, Penalty: "l2", Tol: 1e-4, C: 1.0, Intercept scaling: 1, Class weight: "none", Multi class: "auto", l1 ratio: "none", Random state: "none"
Naïve Bayes	Prior class probability: "default", Var smoothing: 1e-9
Random Forest	Nr estimators: 100, Min samples split: 2, Min samples leaf: 20, Max features: "auto", Max depth: 90, Criterion: "gini"
Support Vector Machine	C: 1.0, Kernel: "rbf", Probability: True, Gamma: "auto", Degree: 3, Probability: "false", Tol: 1e-3, Class weight: "none", Max iter: -1, Decision function shape: "ovr"

BIBLIOGRAPHY

PUBLICATIONS AS FIRST OR CO-AUTHOR

- [1] J. P. A. Maranhão, J. P. C. L. da Costa, E. Javidi, C. A. B. de Andrade, and R. T. de Sousa Jr., “Tensor based framework for Distributed Denial of Service attack detection,” *Journal of Network and Computer Applications*, vol. 174, 102894, Jan. 2021, doi: 10.1016/j.jnca.2020.102894.
- [2] J. P. A. Maranhão, J. P. C. L. da Costa, E. P. de Freitas, E. Javidi, and R. T. de Sousa Jr., “Noise-robust multilayer perceptron architecture for Distributed Denial of Service attack detection,” *IEEE Communications Letters*, vol. 25, no. 2, pp. 402-406, Feb. 2021, doi: 10.1109/LCOMM.2020.3032170.
- [3] J. P. A. Maranhão, J. P. C. L. da Costa, E. P. de Freitas, E. Javidi, and R. T. de Sousa Jr., “Error-robust Distributed Denial of Service attack detection based on an average common feature extraction technique,” *Sensors*, vol. 20, no. 20, 5845, pp. 1-21, Oct. 2020, doi: 10.3390/s20205845.
- [4] J. P. A. Maranhão, J. P. C. L. da Costa, E. Javidi, J. M. Junior, and R. T. de Sousa Jr., “Multidimensional antenna array based framework for drone localization in multipath environments,” *2019 13th International Conference on Signal Processing and Communications Systems (ICSPCS)*, pp. 1–8, Dec. 2019, doi: 10.1109/ICSPCS47537.2019.9008610.
- [5] J. P. A. Maranhão, J. P. C. L. da Costa, R. T. de Sousa Jr., A. J. de Barros Braga, and G. Del Galdo, “Antenna array based framework with multipath mitigation for signal emitter detection and localization,” *2017 11th International Conference on Signal Processing and Communications Systems (ICSPCS)*, pp. 1–5, Dec. 2017, doi: 10.1109/ICSPCS.2017.8270462.
- [6] J. P. A. Maranhão, and J. P. C. L. da Costa, “Antenna array based framework for detection and localization of correlated signals,” *2018 Workshop on Communication Networks and Power Systems (WCNPS)*, pp. 1–5, Nov. 2018, doi: 10.1109/WCNPS.2018.8604387.
- [7] J. P. A. Maranhão, J. P. C. L. da Costa, E. P. de Freitas, M. A. M. Marinho, and G. Del Galdo, “Multi-hop cooperative XIXO transmission scheme for delay tolerant wireless sensor networks,” *20th International ITG Workshop on Smart Antennas (WSA 2016)*, pp. 1–5, Mar. 2016. Accessed: Oct. 20, 2019. [Online]. Available: <https://ieeexplore.ieee.org/document/7499143>.

- [8] J. Milanezi Junior, J. P. C. L. da Costa, J. P. A. Maranhão, R. T. de Sousa Jr., and G. Del Galdo, “A Chinese Remainder Theorem based perfect Secret Sharing Scheme with enhanced secret range values using tensor based operations,” 2019 13th International Conference on Signal Processing and Communications Systems (ICSPCS), pp. 1–6, Dec. 2019, doi: 10.1109/ICSPCS47537.2019.9008712.
- [9] K. H. M. Gularte, L. F. Q. Martins, J. A. R. Vargas, J. P. A. Maranhão, W. A. M. Galaban, and J. F. A. Romero, “A scheme for encryption/decryption based on hyperchaotic systems and Lyapunov theory,” 2019 13th International Conference on Signal Processing and Communications Systems (ICSPCS), pp. 1–9, Dec. 2019, doi: 10.1109/ICSPCS47537.2019.9008413.
- [10] K. H. M. Gularte, L. M. Alves, J. A. R. Vargas, J. P. A. Maranhão, G. C. Carvalho, S. C. A. Alfaro, and J. F. A. Romero, “A chaotic synchronization scheme for information security,” 2019 13th International Conference on Signal Processing and Communications Systems (ICSPCS), pp. 1–10, Dec. 2019, doi: 10.1109/ICSPCS47537.2019.9008760.
- [11] D. G. Rega, R. K. Miranda, E. Javidi, J. P. A. Maranhão, J. P. C. L. da Costa, and G. P. M. Pinheiro, “ESPRIT-based step count for wearable devices,” 2019 13th International Conference on Signal Processing and Communications Systems (ICSPCS), pp. 1–5, Dec. 2019, doi: 10.1109/ICSPCS47537.2019.9008702.
- [12] P. H. de Pinho, M. F. K. B. Couras, G. Favier, J. P. C. L. da Costa, A. L. F. de Almeida, and J. P. A. Maranhão, “Semi-supervised receivers for MIMO systems with multiple Khatri-Rao coding,” 2019 13th International Conference on Signal Processing and Communications Systems (ICSPCS), pp. 1–7, Dec. 2019, doi: 10.1109/ICSPCS47537.2019.9008759.
- [13] E. Javidi, J. P. C. L. da Costa, R. K. Miranda, J. P. A. Maranhão, and J. A. R. Vargas, “Modified information theoretic criteria for low complexity estimation of the amount of components in MEG measurements,” 2019 13th International Conference on Signal Processing and Communications Systems (ICSPCS), pp. 1–6, Dec. 2019, doi: 10.1109/ICSPCS47537.2019.9008721.
- [14] —, “Unsupervised framework for the identification of visual evoked potential in MEG measurements,” 2019 13th International Conference on Signal Processing and Communications Systems (ICSPCS), pp. 1–6, Dec. 2019, doi: 10.1109/ICSPCS47537.2019.9008577.
- [15] B. J. G. Praciano, J. P. C. L. da Costa., J. P. A. Maranhão, F. L. L. de Mendonça, R. T. de Sousa Jr., and J. B. Prettz, “Spatio-temporal trend analysis of

- the Brazilian Elections based on Twitter data,” 2018 IEEE International Conference on Data Mining Workshops (ICDMW), pp. 1355–1360, Nov. 2018, doi: 10.1109/ICDMW.2018.00192.
- [16] D. V. de Lima, J. P. C. L. da Costa, J. P. A. Maranhão, and R. T. de Sousa Jr., “Time delay estimation via Procrustes estimation and Khatri-Rao factorization for GNSS multipath mitigation,” 2017 11th International Conference on Signal Processing and Communications Systems (ICSPCS), pp. 1–7, Dec. 2017, doi: 10.1109/ICSPCS.2017.8270464.
- [17] G. C. Ornelas, M. V. V. da Silva, J. M. Junior, J. P. C. L. da Costa, J. P. A. Maranhão, F. E. G. de Deus, and G. Del Galdo, “First step towards a smart grid communication architecture for the Brazilian Federal District,” 4th IFAC Symposium on Telematics Applications TA 2016, IFAC-PapersOnLine, vol. 49, no. 30, pp. 251–256, 2016, doi: 10.1016/j.ifacol.2016.11.120.

REFERENCES BY OTHER AUTHORS

- [18] S. Han, M. Xie, H. Chen, and Y. Ling, "Intrusion detection in Cyber-Physical Systems: Techniques and challenges," *IEEE Systems Journal*, vol. 8, no. 4, pp. 1052–1062, Dec. 2014, doi:10.1109/JSYST.2013.2257594.
- [19] E. A. Lee, "CPS foundations," *Proceedings of the 47th Design Automation Conference (DAC'10)*, pp. 737–742, June 2010, doi:10.1145/1837274.1837462.
- [20] H. Sadreazami, A. Mohammadi, A. Asif, and K. N. Plataniotis, "Distributed-graph based statistical approach for intrusion detection in Cyber-Physical Systems," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 4, no. 1, pp. 137–147, Mar. 2018, doi: 10.1109/TSIPN.2017.2749976.
- [21] H. Wang, H. Zhao, J. Zhang, D. Ma, J. Li, and J. Wei, "Survey on unmanned aerial vehicle networks: A Cyber-Physical System perspective," *IEEE Communications Surveys & Tutorials*, pp. 1027-1070, Second Quarter 2020, doi: 10.1109/COMST.2019.2962207.
- [22] S. Hosseini, and M. Azizi, "The hybrid technique for DDoS detection with supervised learning algorithms," *Computer Networks*, vol. 158, pp. 35–45, July 2019, doi: 10.1016/j.comnet.2019.04.027.
- [23] M. Wang, Y. Lu, and J. Qin, "A dynamic MLP-based DDoS attack detection method using feature selection and feedback," *Computers & Security*, vol. 88, pp. 1-15, Jan. 2020, doi: 10.1016/j.cose.2019.101645.
- [24] D. Sathyamoorthy, "A Review of Security Threats of Unmanned Aerial Vehicles and Mitigation Steps," *Journal of Defense and Security*, vol. 6, no. 1, Oct. 2015, pp. 81–97.
- [25] M. F. Al-Sa'd, A. Al-Ali, A. Mohamed, T. Khattab, and A. Erbad, "RF-based drone detection and identification using deep learning approaches: An initiative towards a large open source drone database," *Future Generation Computer Systems*, vol. 100, Nov. 2019, pp. 86-97, doi: 10.1016/j.future.2019.05.007.
- [26] Z. Alkhalisi, "Dubai deploys a 'drone hunter' to keep its airport open," 2016. Accessed: Feb. 25, 2021. [Online]. Available: <https://money.cnn.com/2016/11/04/technology/dubai-airport-drone-hunter/index.html>.
- [27] M. Gajewski, "Drone strikes commercial aircraft in Quebec: Garneau," 2017. Accessed: Feb. 25, 2021. [Online]. Available: <https://www.ctvnews.ca/canada/drone-strikes-commercial-aircraft-in-quebec-garneau-1.3633035>.

- [28] I. Lee, “Drone Pilot Fined \$20,000 For Landing Drone at McCarran Airport, Las Vegas,” 2019. Accessed: Feb. 25, 2021. [Online]. Available: <https://uavcoach.com/drone-pilot-fines/>.
- [29] M. Pinhoni and R. Gallo, “Presença de drone no entorno de Congonhas fez aeroporto fechar por 20 minutos nesta terça,” 2020. Accessed: Feb. 25, 2021. [Online]. Available: <https://g1.globo.com/sp/sao-paulo/noticia/2019/01/09/presenca-de-drone-no-entorno-de-congonhas-fez-aeroporto-fechar-por-20-minutos-na-terca.ghtml>.
- [30] S. Shackle, “The mystery of the Gatwick drone,” 2020. Accessed: Feb. 25, 2021. [Online]. Available: <https://www.theguardian.com/uk-news/2020/dec/01/the-mystery-of-the-gatwick-drone>.
- [31] J. Lee, “Watch: Drone crashes into Space Needle during New Year’s Eve fireworks setup,” 2017. Accessed: Feb. 25, 2021. [Online]. Available: <https://www.seattletimes.com/photo-video/video/watch-drone-crashes-into-space-needle-during-new-years-eve-fireworks-setup/>.
- [32] J. Spires, “A photography drone crashed into a Sydney high-rise building,” 2021. Accessed: Feb. 25, 2021. [Online]. Available: <https://dronedj.com/2021/01/28/a-photography-drone-crashed-into-a-sydney-high-rise-building/>.
- [33] M. D. Shear and M. S. Schmidt, “White House Drone Crash Described as a U.S. Worker’s Drunken Lark,” 2015. Accessed: Feb. 25, 2021. [Online]. Available: <https://www.nytimes.com/2015/01/28/us/white-house-drone.html>.
- [34] BBC News, “Drone crash causes Hollywood electricity blackout,” 2015. Accessed: Feb. 25, 2021. [Online]. Available: <https://www.bbc.com/news/technology-34656820>.
- [35] S. French, “This drone crashing into a bike race is every cyclist’s nightmare,” 2017. Accessed: Feb. 25, 2021. [Online]. Available: <https://www.marketwatch.com/story/this-drone-crashing-into-a-bike-race-is-every-cyclists-nightmare-2017-05-09>.
- [36] J. Trevithick, “Two MQ-9 Reaper Drones Collide Over Syria After Days Of Sightings From Those On The Ground,” 2020. Accessed: Feb. 25, 2021. [Online]. Available: <https://www.thedrive.com/the-war-zone/35775/two-mq-9-reaper-drones-collide-over-syria-after-days-of-sightings-from-those-on-the-ground>.
- [37] CNBC, “Smugglers used drones to bring \$79.8 million worth of iPhones into China. They just got busted,” 2018. Accessed: Feb. 25, 2021. [Online]. Available: <https://www.cnbc.com/2018/03/30/china-busts-smugglers-using-drones-to-transport-smartphones.html>.

- [38] BBC, “Charges over drone drug smuggling into prisons,” 2018. Accessed: Feb. 25, 2021. [Online]. Available: <https://www.bbc.com/news/uk-england-43413134>.
- [39] Y. Chen, X. Ma, and X. Wu, “DDoS detection algorithm based on preprocessing network traffic predicted method and chaos theory,” *IEEE Communications Letters*, vol. 17, no. 5, pp. 1052–1054, May 2013, doi: 10.1109/LCOMM.2013.031913.130066.
- [40] J. Galeano-Brajones, J. Carmona-Murillo, J. F. Valenzuela-Valdés, and F. Luna-Valero, “Detection and mitigation of DoS and DDoS attacks in IoT-based stateful SDN: an experimental approach,” *Sensors*, vol. 20, no. 3, 816, pp. 1-18, Feb. 2020, doi: 10.3390/s20030816.
- [41] A. Praseed, and P. S. Thilagam, “DDoS attacks at the application layer: challenges and research perspectives for safeguarding web applications,” *IEEE Communications Surveys & Tutorials*, vol. 21, Firstquarter 2019, pp. 661–685, doi:10.1109/COMST.2018.2870658.
- [42] R. Vishwakarma, and A. K. Jain, “A survey of DDoS attacking techniques and defence mechanisms in the IoT network,” *Telecommunication Systems*, vol. 73, pp. 3–25, Jan. 2020, doi: 10.1007/s11235-019-00599-z.
- [43] F. S. Dantas Silva, E. Silva, E. P. Neto, M. Lemos, A. J. V. Neto, and F. Esposito, “A taxonomy of DDoS attack mitigation approaches featured by SDN technologies in IoT scenarios,” *Sensors*, vol. 20, no. 11, 3078, pp. 1-28, May 2020, doi: 10.3390/s20113078.
- [44] Cisco, “Cisco Annual Internet Report (2018–2023) White Paper,” 2020. Accessed: Feb. 26, 2021. [Online]. Available: <https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html>.
- [45] Kaspersky, “DDoS attacks in Q4 2020,” 2021. Accessed: Feb. 26, 2021. [Online]. Available: <https://securelist.com/ddos-attacks-in-q4-2020/100650/>.
- [46] KrebsOnSecurity, “Krebsonsecurity hit with record DDoS,” 2016. Accessed: Oct. 30, 2020. [Online]. Available: <https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos/>.
- [47] KrebsOnSecurity, “The democratization of censorship,” 2016. [Online]. Accessed: Oct. 30, 2020. Available: <https://krebsonsecurity.com/2016/09/the-democratization-of-censorship/>.
- [48] Dyn Status, “Dyn status updates,” 2016. Accessed: Oct. 30, 2020. [Online]. Available: <https://www.dynstatus.com/incidents/5r9mppc1kb77>.

- [49] M. Prince, “Technical details behind a 400 Gbps NTP amplification DDoS attack,” 2014. Accessed: Oct. 30, 2020. [Online]. Available: <https://blog.cloudflare.com/technical-details-behind-a-400gbps-ntp-amplification-ddos-attack/>.
- [50] S. Kottler, “February 28th DDoS Incident Report,” 2018. Accessed: Oct. 30, 2020. [Online]. Available: <https://github.blog/2018-03-01-ddos-incident-report/>.
- [51] AWS, “AWS Shield - Threat Landscape Report – Q1 2020,” 2020. Accessed: Oct. 30, 2020. [Online]. Available: <https://aws-shield-tlr.s3.amazonaws.com/2020-Q1-AWS-Shield-TLR.pdf>.
- [52] Google Cloud, “Exponential growth in DDoS attack volumes,” 2020. Accessed: Oct. 30, 2020. [Online]. Available: <https://cloud.google.com/blog/products/identity-security/identifying-and-protecting-against-the-largest-ddos-attacks>.
- [53] L. Constantin, “DDoS attacks against US banks peaked at 60 Gbps,” 2012. Accessed: Oct. 30, 2020. [Online]. Available: <https://www.cio.com/article/2389721/ddos-attacks-against-us-banks-peaked-at-60-gbps.html>.
- [54] P. Olson, “The largest cyber attack in history has been hitting Hong Kong sites,” 2014. Accessed: Oct. 30, 2020. [Online]. Available: <https://www.forbes.com/sites/parmyolson/2014/11/20/the-largest-cyber-attack-in-history-has-been-hitting-hong-kong-sites/?sh=2e8d4d1d38f6>.
- [55] C. Morales, “NETSCOUT Arbor confirms 1.7 Tbps DDoS attack; The Terabit attack era is upon us,” 2018. Accessed: Oct. 30, 2020. [Online]. Available: <https://www.netscout.com/blog/asert/netscout-arbor-confirms-17-tbps-ddos-attack-terabit-attack-era>.
- [56] I. Management Association, “Securing the Internet of Things: Concepts, Methodologies, Tools, and Applications: Concepts, Methodologies, Tools, and Applications”, IGI Global, 2019.
- [57] Z. Kaleem, and M. H. Rehmani, “Amateur drone monitoring: State-of-the-art architectures, key enabling technologies, and future research directions,” *IEEE Wireless Communications*, vol. 25, no. 2, pp. 150–159, Apr. 2018, doi: 10.1109/MWC.2018.1700152.
- [58] J. Kang, K. Park, and H. Kim, “Analysis of localization for drone-fleet,” 2015 International Conference on Information and Communication Technology Convergence (ICTC), pp. 533–538, Oct. 2015, doi: 10.1109/ICTC.2015.7354604.
- [59] X. Chang, C. Yang, J. Wu, X. Shi, and Z. Shi, “A surveillance system for drone localization and tracking using acoustic arrays,” 2018 IEEE 10th Sensor Array

- Multichannel Signal Processing Workshop (SAM), July 2018, pp. 573–577, doi: 10.1109/SAM.2018.8448409.
- [60] R. Roy, and T. Kailath, “ESPRIT-estimation of signal parameters via rotational invariance techniques,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 7, pp. 984–995, July 1989, doi: 10.1109/29.32276.
- [61] T. J. Shan, M. Wax, and T. Kailath, “On spatial smoothing for direction-of-arrival estimation of coherent signals,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 33, no. 4, pp. 806–811, Aug. 1985, doi: 10.1109/TASSP.1985.1164649.
- [62] M. Haardt, F. Roemer, and G. Del Galdo, “Higher-Order SVD-based subspace estimation to improve the parameter estimation accuracy in multidimensional harmonic retrieval problems,” *IEEE Transactions on Signal Processing*, vol. 56, no. 7, pp. 3198–3213, July 2008, doi: 10.1109/TSP.2008.917929.
- [63] M. A. M. Marinho, E. P. de Freitas, J. P. C. L. da Costa, and R. T. de Sousa Jr., “Sensor localization via diversely polarized antennas,” *2014 IEEE International Conference on Distributed Computing in Sensor Systems*, pp. 333–337, May 2014, doi: 10.1109/DCOSS.2014.54.
- [64] M. A. M. Marinho, P. R. L. Gondim, and J. P. C. L. da Costa, “Continuous authentication via localization using triangulation of directions of arrival of line of sight components,” *9th International Conference on Computer Science and Cyber Law (ICoFCS 2015)*, pp. 31–35, June 2015, doi: 10.5769/C2015004.
- [65] M. A. M. Marinho, A. Vinel, F. Tufvesson, F. Antreich, and J. P. C. L. da Costa, “Performance assessment for distributed broadband radio localization,” *2018 52nd Asilomar Conference on Signals, Systems, and Computers*, pp. 20–23, Oct. 2018, doi: 10.1109/ACSSC.2018.8645490.
- [66] M. A. M. Marinho, A. Vinel, F. Antreich, J. P. C. L. da Costa, and E. P. de Freitas, “Antenna array based localization scheme for vehicular networks,” *2017 IEEE International Conference on Computer and Information Technology (CIT)*, Aug. 2017, pp. 142–146, doi: 10.1109/CIT.2017.64.
- [67] P. R. B. Gomes, A. L. F. de Almeida, J. P. C. L. da Costa, and R. T. de Sousa Jr., “A nested-PARAFAC based approach for target localization in bistatic MIMO radar systems,” *Digital Signal Processing*, vol. 89, pp. 40–48, June 2019, doi: 10.1016/j.dsp.2019.02.005.
- [68] S. Y. Nam, and G. P. Joshi, “Unmanned aerial vehicle localization using distributed sensors,” *International Journal of Distributed Sensor Networks*, vol. 13, no. 9, pp. 1–8, Sep. 2017, doi: 10.1177/1550147717732920.

- [69] X. Shi, C. Yang, W. Xie, C. Liang, Z. Shi, and J. Chen, "Anti-drone system with multiple surveillance technologies: Architecture, implementation, and challenges," *IEEE Communications Magazine*, vol. 56, no. 4, pp. 68–74, Apr. 2018, doi: 10.1109/M-COM.2018.1700430.
- [70] R. K. Miranda, D. A. Ando, J. P. C. L. da Costa, and M. T. de Oliveira, "Enhanced direction of arrival estimation via received signal strength of directional antennas," 2018 IEEE Symposium on Signal Processing and Information Technology (ISSPIT), pp. 162–167, Dec. 2018, doi: 10.1109/ISSPIT.2018.8642668.
- [71] D. A. Ando, R. K. Miranda, J. P. C. L. da Costa, and M. T. de Oliveira, "A novel direction of arrival estimation algorithm via received signal strength of directional antennas," 2018 Workshop on Communication Networks and Power Systems (WCNPS), pp. 1–5, Nov. 2018, doi: 10.1109/WCNPS.2018.8604345.
- [72] M. T. de Oliveira, R. K. Miranda, J. P. C. L. da Costa, A. L. F. de Almeida, and R. T. de Sousa Jr., "Low cost antenna array based drone tracking device for outdoor environments," *Wireless Communications and Mobile Computing*, vol. 2019, no. ID 5437908, pp. 1–14, June 2019, doi: 10.1155/2019/5437908.
- [73] M. Saria Allahham, M. F. Al-Sa'd, A. Al-Ali, A. Mohamed, T. Khattab, and A. Erbad, "DroneRF dataset: A dataset of drones for RF-based detection, classification and identification," *Data in Brief*, vol. 26, 104313, pp. 1-9, Oct. 2019, doi: 10.1016/j.dib.2019.104313.
- [74] M. S. Allahham, T. Khattab, and A. Mohamed, "Deep learning for RF-based drone detection and identification: A multi-channel 1-D Convolutional Neural Networks approach," 2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT), pp. 112-117, Feb. 2020, doi: 10.1109/ICIoT48696.2020.9089657.
- [75] P. R. B. Gomes, J. P. C. L. da Costa, A. L. F. de Almeida, and R. T. de Sousa Jr., "Tensor-based multiple denoising via successive spatial smoothing, low-rank approximation and reconstruction for R-D sensor array processing," *Digital Signal Processing*, vol. 89, pp. 1–7, June 2019, doi: 10.1016/j.dsp.2019.01.005.
- [76] F. R. Farrokhi, A. Lozano, G. J. Foschini and R. A. Valenzuela, "Spectral efficiency of FDMA/TDMA wireless systems with transmit and receive antenna arrays," in *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, pp. 591-599, Oct. 2002, doi: 10.1109/TWC.2002.804078.
- [77] H. Akaike, "A new look at the statistical model identification," *IEEE Transactions on Automatic Control*, vol. 19, no. 6, pp. 716–723, Dec. 1974, doi: 10.1109/TAC.1974.1100705.

- [78] L. C. Zhao, P. R. Krishnaiah, and Z. D. Bai, "On detection of the number of signals in presence of white noise," *Journal of Multivariate Analysis*, vol. 20, no. 1, pp. 1–25, Oct. 1986, doi: 10.1016/0047-259X(86)90017-5.
- [79] A. Barron, J. Rissanen, and B. Yu, "The minimum description length principle in coding and modeling," *IEEE Transactions on Information Theory*, vol. 44, no. 6, pp. 2743–2760, Oct. 1998, doi: 10.1109/18.720554.
- [80] E. Radoi, and A. Quinquis, "A new method for estimating the number of harmonic components in noise with application in high resolution radar," *EURASIP Journal on Advances in Signal Processing*, vol. 2004, no. 8, pp. 1177-1188, July 2004, doi: 10.1155/S1110865704401097.
- [81] G. Yong, X. X. Ci, and Z. Z. Zhong, "Erasing false-location of two stations direction finding cross location in multi-path and multiple sources environments," *2001 International Conference on Radar Proceedings*, pp. 864–868, Oct. 2001, doi: 10.1109/ICR.2001.984848.
- [82] H. Zhu, X. You, and S. Liu, "Multiple ant colony optimization based on Pearson Correlation Coefficient," *IEEE Access*, vol. 7, pp. 61628–61638, May 2019, doi: 10.1109/ACCESS.2019.2915673.
- [83] G. H. Golub, and C. F. V. Loan, *Matrix Computations* (3rd Ed.). Baltimore, MD, USA: Johns Hopkins University Press, 1996.
- [84] S. Liu, F. Qin, J. Zhao, W. Xiong, and Z. Yuan, "Fast direction-finding algorithm by partial spatial smoothing in sparse MIMO radar," *Progress In Electromagnetics Research M (PIER M)*, vol. 93, pp. 127-136, 2020, doi:10.2528/PIERM20031701.
- [85] M. F. Al-Sa'd, M. Saria Allahham, A. Mohamed, A. Al-Ali, T. Khattab, and A. Erbad, "DroneRF dataset: A dataset of drones for RF-based detection, classification, and identification," *Mendeley Data*, V1, Mar. 2019, doi: 10.17632/f4c2b4n755.1.
- [86] A. Alsirhani, S. Sampalli, and P. Bodorik, "DDoS detection system: Using a set of classification algorithms controlled by fuzzy logic system in apache spark," *IEEE Transactions on Network and Service Management*, vol. 16, no. 3, pp. 936–949, Sept. 2019, doi: 10.1109/TNSM.2019.2929425.
- [87] B. Felter, "7 of the most famous recent DDoS attacks," 2020. Accessed: Nov. 20, 2020. [Online]. Available: <https://www.vxchnge.com/blog/recent-ddos-attacks-on-companies>.
- [88] M. R. Zanatta, F. L. L. de Mendonça, F. Antreich, D. V. de Lima, R. K. Miranda, G. Del Galdo, and J. P. C. L. da Costa, "Tensor-based time-delay estimation for second

- and third generation global positioning system,” *Digital Signal Processing*, vol. 92, pp. 1–19, Sept. 2019, doi: 10.1016/j.dsp.2019.04.003.
- [89] Canadian Institute for Cybersecurity, “DDoS Evaluation Dataset (CIC-DDoS2019),” 2019. Accessed: July 08, 2020. [Online]. Available: <https://www.unb.ca/cic/datasets/ddos-2019.html>.
- [90] Canadian Institute for Cybersecurity, “NSL-KDD dataset,” 2009. Accessed: Aug. 10, 2020. [Online]. Available: <https://www.unb.ca/cic/datasets/nsl.html>.
- [91] J. A. Saez, M. Galar, J. Luengo, and F. Herrera, “Tackling the problem of classification with noisy data using multiple classifier systems: Analysis of the performance and robustness,” *Information Sciences*, vol. 247, pp. 1–20, Oct. 2013, doi: 10.1016/j.ins.2013.06.002.
- [92] F. Li and Y. Tang, “False data injection attack for Cyber-Physical Systems with resource constraint,” *IEEE Transactions on Cybernetics*, vol. 50, no. 2, pp. 729–738, Feb. 2020, doi: 10.1109/TCYB.2018.2871951.
- [93] B. M. David, J. P. C. L. da Costa, A. C. A. Nascimento, M. D. Holtz, D. Amaral, and R. T. de Sousa Jr., “Blind automatic malicious activity detection in honeypot data,” *Proceedings of the 6th International Conference on Computer Science and Cyber Law (ICoFCS 2011)*, pp. 142–152, Oct. 2011, doi: 10.5769/C2011016.
- [94] B. M. David, J. P. C. L. da Costa, E. P. de Freitas, A. M. R. Serrano, and R. T. de Sousa Jr., “A parallel approach to PCA based malicious activity detection in distributed honeypot data,” *The International Journal of Forensic Computer Science (ICoFCS 2011)*, vol. 6, no. 1, pp. 8–27, 2011, doi: 10.5769/IJ201101001.
- [95] J. P. C. L. da Costa, E. P. de Freitas, B. M. David, A. M. R. Serrano, D. Amaral, and R. T. de Sousa Jr., “Improved blind automatic malicious activity detection in honeypot data,” *Proceedings of the 7th International Conference on Computer Science and Cyber Law (ICoFCS 2012)*, pp. 46–55, Sept. 2012, doi: 10.5769/C2012008.
- [96] J. P. C. L. da Costa, E. P. de Freitas, A. M. R. Serrano, and R. T. de Sousa Jr., “Improved parallel approach to PCA based malicious activity detection in distributed honeypot data,” *The International Journal of Forensic Computer Science*, vol. 2, pp. 8–20, 2012, doi: 10.5769/IJ201202001.
- [97] D. F. Tenorio, J. P. C. L. da Costa, and R. T. de Sousa Jr., “Greatest eigenvalue time vector approach for blind detection of malicious traffic,” *Proceedings of the 8th International Conference on Computer Science and Cyber Law (ICoFCS 2013)*, pp. 46–51, Aug. 2013, doi: 10.5769/C2013007.

- [98] T. P. B. Vieira, D. F. Tenório, J. P. C. L. da Costa, E. P. de Freitas, G. Del Galdo, and R. T. de Sousa Jr., “Model order selection and eigen similarity based framework for detection and identification of network attacks,” *Journal of Network and Computer Applications*, vol. 90, pp. 26–41, July 2017, doi: 10.1016/j.jnca.2017.04.012.
- [99] E. S. C. Vilaça, T. P. B. Vieira, R. T. de Sousa Jr., and J. P. C. L. da Costa, “Botnet traffic detection using RPCA and Mahalanobis Distance,” *2019 Workshop on Communication Networks and Power Systems (WCNPS)*, pp. 1–6, Oct. 2019, doi: 10.1109/WCNPS.2019.8896228.
- [100] O. Osanaiye, H. Cai, K. R. Choo, A. Dehghantanha, Z. Xu, and M. Dlodlo, “Ensemble-based multi-filter feature selection method for DDoS detection in cloud computing,” *EURASIP Journal on Wireless Communications and Networking*, vol. 130, pp. 1-10, May 2016, doi: 10.1186/s13638-016-0623-3.
- [101] G. S. Kushwah, and S. T. Ali, “Detecting DDoS attacks in cloud computing using ANN and black hole optimization,” *2017 2nd International Conference on Telecommunication and Networks (TEL-NET)*, pp. 1–5, Aug. 2017, doi: 10.1109/TEL-NET.2017.8343555.
- [102] M. Kuhn, and K. Johnson, “Applied predictive modeling,” New York, NY: Springer, 2013, doi: 10.1007/978-1-4614-6849-3.
- [103] L. P. F. Garcia, A. C. P. L. F. de Carvalho, and A. C. Lorena, “Noisy data set identification,” *Hybrid Artificial Intelligent Systems*, pp. 629–638, 2013, doi: 10.1007/978-3-642-40846-5_63.
- [104] A. Thakre, M. Haardt, F. Roemer, and K. Giridhar, “Tensor-based spatial smoothing (TB-SS) using multiple snapshots,” *IEEE Transactions on Signal Processing*, vol. 58, no. 5, pp. 2715–2728, May 2010, doi: 10.1109/TSP.2010.2043141.
- [105] M. Wax, and T. Kailath, “Detection of signals by information theoretic criteria,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 33, no. 2, pp. 387–392, Apr. 1985, doi: 10.1109/TASSP.1985.1164557.
- [106] M. O. Ulfarsson, and V. Solo, “Dimension estimation in noisy PCA with SURE and random matrix theory,” *IEEE Transactions on Signal Processing*, vol. 56, no. 12, pp. 5804–5816, Dec. 2008, doi: 10.1109/TSP.2008.2005865.
- [107] J. P. C. L. da Costa, F. Roemer, M. Haardt, and R. T. de Sousa Jr., “Multi-dimensional model order selection,” *EURASIP Journal on Advances in Signal Processing*, vol. 2011, no. 26, pp. 1–13, Jan. 2011, doi: 10.1186/1687-6180-2011-26.

- [108] L. D. Lathauwer, B. D. Moor, and J. Vandewalle, “On the best rank-1 and rank-(R_1, R_2, \dots, R_n) approximation of higher-order tensors,” *SIAM Journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1324–1342, May 2000, doi: 10.1137/S0895479898346995.
- [109] P. N. Tan, M. Steinbach, and V. Kumar, “Introduction to Data Mining (2nd Edition),” Pearson Education, 2018, ISBN: 0133128903.
- [110] E. S. Gualberto, R. T. de Sousa Jr., T. P. B. Vieira, J. P. C. L. da Costa, and C. G. Duque, “From feature engineering and topics models to enhanced prediction rates in phishing detection,” *IEEE Access*, vol. 8, pp. 76368–76385, Apr. 2020, doi: 10.1109/ACCESS.2020.2989126.
- [111] E. S. Gualberto, R. T. de Sousa Jr., T. P. B. Vieira, J. P. C. L. da Costa, and C. G. Duque, “The answer is in the text: Multi-stage methods for phishing detection based on feature engineering,” *IEEE Access*, vol. 8, pp. 223529–223547, Dec. 2020, doi: 10.1109/ACCESS.2020.3043396.
- [112] R. Ballester-Ripoll, S. K. Suter, and R. Pajarola, “Analysis of tensor approximation for compression-domain volume visualization,” *Computers & Graphics*, vol. 47, pp. 34–47, Apr. 2015, doi: 10.1016/j.cag.2014.10.002.
- [113] Q. Guo, C. Zhang, Y. Zhang, and H. Liu, “An efficient SVD-based method for image denoising,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 5, pp. 868–880, May 2016, doi: 10.1109/TCSVT.2015.2416631.
- [114] S. K. Jha, and R. D. S. Yadava, “Denoising by singular value decomposition and its application to electronic nose data processing,” *IEEE Sensors Journal*, vol. 11, no. 1, pp. 35–44, Jan. 2011, doi: 10.1109/JSEN.2010.2049351.
- [115] A. Rajwade, A. Rangarajan, and A. Banerjee, “Image denoising using the Higher Order Singular Value Decomposition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 4, pp. 849–862, Apr. 2013, doi: 10.1109/T-PAMI.2012.140.
- [116] P. Gogoi, M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, “Packet and flow based network intrusion dataset,” *Contemporary Computing, IC3 2012, Communications in Computer and Information Science*, vol. 306, Springer, Berlin, Heidelberg, pp. 322–334, 2012, doi: 10.1007/978-3-642-32129-0_34.
- [117] A. R. Yusof, N. I. Udzir, A. Selamat, H. Hamdan, and M. T. Abdullah, “Adaptive feature selection for denial of services (DoS) attack,” *2017 IEEE Conference on Application, Information and Network Security (AINS)*, pp. 81–84, Nov. 2017, doi: 10.1109/AINS.2017.8270429.

- [118] M. S. Elsayed, N. A. Le-Khac, S. Dev, and A. D. Jurcut, "DDoSNet: A deep-learning model for detecting network attacks," 2020 IEEE 21st International Symposium "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM), pp. 391-396, Sept. 2020, doi: 10.1109/WoWMoM49955.2020.00072.
- [119] M. Shurman, R. Khrais, and A. Yateem, "DoS and DDoS attack detection using deep learning and IDS," *International Arab Journal of Information Technology*, vol. 17, no. 4A, pp. 655-661, Special Issue 2020, doi: 10.34028/iajit/17/4A/10.
- [120] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic Distributed Denial of Service (DDoS) attack dataset and taxonomy," 2019 International Carnahan Conference on Security Technology (ICCST), pp. 1-8, Oct. 2019, doi: 10.1109/CCST.2019.8888419.
- [121] F. Hussain, S. G. Abbas, M. Husnain, U. U. Fayyaz, F. Shahzad, and G. A. Shah, "IoT DoS and DDoS attack detection using ResNet," 2020. Accessed: Nov. 10, 2020. [Online]. Available: <https://arxiv.org/abs/2012.01971>.
- [122] T. Aytac, M. Ali Aydin, and A. Halim Zaim, "Detection DDoS attacks using machine learning methods," *Electrica*, vol. 20, no. 2, pp. 159-167, July 2020, doi: 10.5152/electrica.2020.20049.
- [123] X. Liang, and T. Znati, "An empirical study of intelligent approaches to DDoS detection in large scale networks," 2019 International Conference on Computing, Networking and Communications (ICNC), pp. 821-827, Feb. 2019, doi: 10.1109/ICNC.2019.8685519.
- [124] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, "Understanding the Mirai botnet," *Proceedings of the 26th USENIX Security Symposium (USENIX Secur. 17)*, pp. 1093-1110, Aug. 2017. Accessed: Sept. 14, 2020. [Online]. Available: <https://www.usenix.org/system/files/conference/usenixsecurity17/sec17-antonakakis.pdf>
- [125] R. Doriguzzi-Corin, S. Millar, S. Scott-Hayward, J. Martinez del Rincon, and D. Siracusa, "LUCID: A practical, lightweight deep learning solution for DDoS attack detection," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 876-889, June 2020, doi: 10.1109/TNSM.2020.2971776.
- [126] M. Roopak, G. Yun Tian, and J. Chambers, "Deep learning models for cyber security in IoT networks," 2019 IEEE 9th Annual Computing and Communication

- Workshop and Conference (CCWC), pp. 0452–0457, Jan. 2019, doi: 10.1109/C-CWC.2019.8666588.
- [127] S. Haider, A. Akhunzada, I. Mustafa, T. B. Patel, A. Fernandez, K. R. Choo, and J. Iqbal, “A deep CNN ensemble framework for efficient DDoS attack detection in Software Defined Networks,” *IEEE Access*, vol. 8, pp. 53972–53983, Feb. 2020, doi: 10.1109/ACCESS.2020.2976908.
- [128] J. Chen, Y. Yang, K. Hu, H. Zheng, and Z. Wang, “DAD-MCNN: DDoS attack detection via multi-channel CNN,” *Proceedings of the 2019 11th International Conference on Machine Learning and Computing (ICMLC’19)*, pp. 484–488, Feb. 2019, doi: 10.1145/3318299.3318329.
- [129] A. Saied, R. E. Overill, and T. Radzik, “Detection of known and unknown DDoS attacks using artificial neural networks,” *Neurocomputing*, vol. 172, no. C, pp. 385–393, Jan. 2016, doi: 10.1016/j.neucom.2015.04.101.
- [130] K. J. Singh, K. Thongam, and T. De, “Entropy-based application layer DDoS attack detection using artificial neural networks,” *Entropy*, vol. 18, no. 10, pp. 1–17, Oct. 2016, doi: 10.3390/e18100350.
- [131] Canadian Institute for Cybersecurity, “Intrusion Detection Evaluation Dataset (CIC-IDS2017),” 2017. Accessed: Aug. 09, 2020. [Online]. Available: <https://www.unb.ca/cic/datasets/ids-2017.html>.
- [132] —, “Intrusion Detection Evaluation Dataset (CSE-CIC-IDS2018),” 2018. Accessed: Aug. 02, 2020 [Online]. Available: <https://www.unb.ca/cic/datasets/ids-2018.html>.
- [133] G. Zhou, A. Cichocki, Y. Zhang, and D. P. Mandic, “Group component analysis for multiblock data: Common and individual feature extraction,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 11, pp. 2426–2439, Nov. 2016, doi: 10.1109/TNNLS.2015.2487364.
- [134] I. Kisil, G. G. Calvi, and D. P. Mandic, “Tensor valued common and individual feature extraction: Multi-dimensional perspective,” 2017. Accessed: May 28, 2020. [Online]. Available: <https://arxiv.org/abs/1711.00487>.
- [135] I. Kisil, G. Calvi, A. Cichocki, and D. P. Mandic, “Common and individual feature extraction using tensor decompositions: a remedy for the curse of dimensionality?” *2018 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 6299–6303, Apr. 2018, doi: 10.1109/ICASSP.2018.8461318.

- [136] R. Minster, A. K. Saibaba, and M. E. Kilmer, “Randomized algorithms for low-rank tensor decompositions in the Tucker format,” *SIAM Journal on Mathematics of Data Science*, vol. 2, no. 1, pp. 1-28, 2020, doi: 10.1137/19M1261043.
- [137] J. Kossaifi, Y. Panagakis, A. Anandkumar, and M. Pantic, “TensorLy: Tensor learning in Python,” *Journal of Machine Learning Research*, vol. 20, no. 26, pp. 1-6, Feb. 2019. Accessed: Oct. 20, 2020. [Online]. Available: <http://jmlr.org/papers/v20/18-277.html>.
- [138] Y. Li, Z. Pan, D. Dong, and R. Li, “Adaptive thresholding HOSVD with rearrangement of tensors for image denoising,” *Multimedia Tools and Applications*, vol 79, pp. 19575-19593, Mar. 2020, doi: 10.1007/s11042-020-08624-z.
- [139] M. Aamir, and S. M. Ali Zaidi, “Clustering based semi-supervised machine learning for DDoS attack classification,” *Journal of King Saud University - Computer and Information Sciences*, Feb. 2019, pp. 1-11, doi: 10.1016/j.jksuci.2019.02.003.
- [140] F. S. Lima Filho, F. A. F. Silveira, A. M. Brito Jr., G. Vargas-Solar, and L. F. Silveira, “Smart detection: An online approach for DoS/DDoS attack detection using machine learning,” *Security and Communication Networks*, vol. 2019, pp. 1–15, Oct. 2019, doi: 10.1155/2019/1574749.
- [141] D. Aksu, S. Ustebay, M.A. Aydin, and T. Atmaca, “Intrusion detection with comparative analysis of supervised learning techniques and Fisher score feature selection algorithm,” *Proceedings of the 32nd International Symposium on Computer and Information Science (ISCIS 2018)*, vol. 935, pp. 141-149, Sept. 2018, doi: 10.1007/978-3-030-00840-6_16.
- [142] S. Ustebay, Z. Turgut, and M. A. Aydin, “Intrusion detection system with recursive feature elimination by using random forest and deep learning classifier,” *2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)*, pp. 71-76, Dec. 2018, doi: 10.1109/IBIGDELFT.2018.8625318.
- [143] A. Yulianto, P. Sukarno, and N. A. Suwastika, “Improving AdaBoost-based Intrusion Detection System (IDS) performance on CIC IDS 2017 dataset,” *Journal of Physics: Conference Series*, vol. 1192, pp. 1-9, 2019, doi: 10.1088/1742-6596/1192/1/012018.
- [144] M. Zhu, K. Ye, Y. Wang, and C. Xu, “A deep learning approach for network anomaly detection based on AMF-LSTM,” *Network and Parallel Computing (NPC 2018)*, *Lecture Notes Comput. Sci.*, vol. 11276, pp. 137-141, Dec. 2018, doi: 10.1007/978-3-030-05677-3_13.
- [145] Y. Yao, L. Su, and Z. Lu, “DeepGFL: Deep feature learning via graph for attack detection on flow-based network traffic,” *2018 IEEE Military Communications Conference (MILCOM 2018)*, 2018, pp. 579–584, doi: 10.1109/MILCOM.2018.8599821.

- [146] S. Simpson, S. N. Shirazi, A. Marnierides, S. Jouet, D. Pezaros, and D. Hutchison, "An inter-domain collaboration scheme to remedy DDoS attacks in computer networks," *IEEE Transactions on Network and Service Management*, vol. 15, no. 3, pp. 879–893, Sept. 2018, doi: 10.1109/TNSM.2018.2828938.
- [147] H. Luo, Z. Chen, J. Li, and T. Vasilakos, "On the benefits of keeping path identifiers secret in future internet: A DDoS perspective," *IEEE Transactions on Network and Service Management*, vol. 15, no. 2, pp. 650–664, June 2018, doi: 10.1109/TNSM.2018.2800007.
- [148] Z. Abou El Houda, L. Khoukhi, and A. Senhaji Hafid, "Bringing intelligence to Software Defined Networks: Mitigating DDoS attacks," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2523–2535, Dec. 2020, doi: 10.1109/TNSM.2020.3014870.
- [149] X. Yuan, C. Li, and X. Li, "DeepDefense: Identifying DDoS attack via deep learning," *2017 IEEE International Conference on Smart Computing (SMARTCOMP)*, May 2017, pp. 1–8, doi: 10.1109/SMARTCOMP.2017.7946998.
- [150] C. Ma, X. Mu, and D. Sha, "Multi-layers feature fusion of Convolutional Neural Network for scene classification of remote sensing," *IEEE Access*, vol. 7, pp. 121685–121694, Aug. 2019, doi: 10.1109/ACCESS.2019.2936215.
- [151] D. Yun-Mei, A. Maalla, L. Hui-Ying, H. Shuai, L. Dong, L. Long, and L. Hongsheng, "The abnormal detection of electroencephalogram with three-dimensional deep Convolutional Neural Networks," *IEEE Access*, vol. 8, pp. 64646–64652, Mar. 2020, doi: 10.1109/ACCESS.2020.2984677.
- [152] J. Wang, L. Yu, K. R. Lai, and X. Zhang, "Tree-structured regional CNN-LSTM model for dimensional sentiment analysis," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 581–591, Dec. 2019, doi: 10.1109/TASLP.2019.2959251.
- [153] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, Oct. 2017, doi: 10.1109/ACCESS.2017.2762418.
- [154] Y. Zhang, X. Chen, D. Guo, M. Song, Y. Teng, and X. Wang, "PCCN: Parallel cross convolutional neural network for abnormal network traffic flows detection in multi-class imbalanced network traffic flows," *IEEE Access*, vol. 7, pp. 119904–119916, Aug. 2019, doi: 10.1109/ACCESS.2019.2933165.

- [155] A. Koay, A. Chen, I. Welch, and W. K. G. Seah, “A new multi classifier system using entropy-based features in DDoS attack detection,” in 2018 International Conference on Information Networking (ICOIN), 2018, pp. 162–167.
- [156] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, “A survey of network-based intrusion detection data sets,” 2019, Accessed: Feb. 10, 2020. [Online]. Available: <https://arxiv.org/pdf/1903.02460.pdf>.
- [157] S. Ioffe, and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” Proceedings of the 32nd International Conference on Machine Learning (ICML’15), pp. 448–456, July 2015. Accessed: Nov. 15, 2020. [Online]. Available: <https://dl.acm.org/doi/10.5555/3045118.3045167>.
- [158] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” Journal of Machine Learning Research, vol. 15, no. 56, pp. 1929–1958, 2014. Accessed: Apr. 10, 2020. [Online]. Available: <https://jmlr.org/papers/v15/srivastava14a.html>.
- [159] M. Taghavi, and M. Shoaran, “Hardware Complexity Analysis of Deep Neural Networks and Decision Tree Ensembles for Real-time Neural Data Classification,” 2019 9th International IEEE/EMBS Conference on Neural Engineering (NER), pp. 407–410, 2019, doi: 10.1109/NER.2019.8716983.
- [160] Y. Li, M. Dong, and R. Kothari, “Classifiability-based omnivariate decision trees,” IEEE Transactions on Neural Networks, vol. 16, no. 6, pp. 1547–1560, Nov. 2005, doi: 10.1109/TNN.2005.852864.
- [161] M. Haardt, M. Pesavento, F. Roemer, and M. N. E. Korso, “Chapter 15 - Subspace methods and exploitation of special array structures,” Academic Press Library Signal Processing, vol. 3, Ed. Elsevier, pp. 651–717, 2014, doi: 10.1016/B978-0-12-411597-2.00015-1.
- [162] M. Weis, “Multi-Dimensional Signal Decomposition Techniques for Analysis of EEG Data,” Ph.D. Dissertation, Ilmenau University of Technology, 2015. Accessed: Feb. 25, 2021. [Online]. Available: https://lasp.unb.br/wp-content/uploads/files/research/Thesis_MW.pdf.
- [163] Parrot, “Parrot Bebop Drone - User Guide,” 2020. Accessed: Nov. 21, 2020. [Online]. Available: <https://www.parrot.com>.
- [164] Parrot, “Parrot AR Drone 2.0 - User Guide,” 2020. Accessed: Nov. 21, 2020. [Online]. Available: <https://www.parrot.com>.

- [165] Phantom, “Phantom 3 Standard,” 2020. Accessed: Nov. 21, 2020. [Online]. Available: <https://www.dji.com/br/phantom-3-standard>.
- [166] M. Tavallae, E. Bagheri, W. Lu, and A. A. Ghorbani, “A detailed analysis of the KDD CUP 99 data set,” 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, pp. 1–6, July 2009, doi: 10.1109/CISDA.2009.5356528.
- [167] S. M. H. Bamakan, H. Wang, T. Yingjie, and Y. Shi, “An effective intrusion detection framework based on MCLP/SVM optimized by time-varying chaos particle swarm optimization,” *Neurocomputing*, vol. 199, pp. 90–102, July 2016, doi: 10.1016/j.neucom.2016.03.031.
- [168] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, “Toward generating a new intrusion detection dataset and intrusion traffic characterization,” *Proceedings of the 4th International Conference on Information Systems Security and Privacy - Volume 1: ICISSP*, pp. 108–116, Jan. 2018, doi: 10.5220/0006639801080116.
- [169] Canadian Institute for Cybersecurity, “CICFlowMeter: Network traffic flow analyzer,” 2017. Accessed: Apr. 4, 2019. [Online]. Available: <http://www.netflowmeter.ca/netflowmeter.html>.