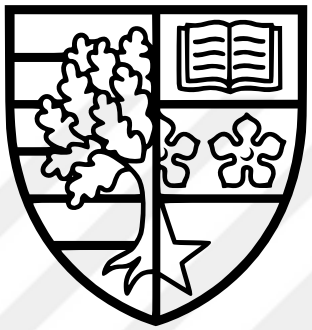
BEHAVIOUR-BASED SECURITY WITH MACHINE LEARNING ON IOT NETWORKS

by

Kahraman Kostas



Submitted for the degree of
Doctor of Philosophy

Department of Computer Science
School of Mathematical and Computer Sciences
Heriot-Watt University
October 2023

Supervised by Dr Mike Just & Dr Michael A. Lones

The copyright in this thesis is owned by the author. Any quotation from the thesis or use of any of the information contained in it must acknowledge this thesis as the source of the quotation or information.

Abstract

The proliferation of Internet of Things (IoT) devices has transformed various aspects of human life, yet has brought forth significant security challenges due to device heterogeneity and limited resources. Addressing this, the thesis focuses on reliable and reproducible IoT security measures, specifically device identification (DI) and attack detection (AD). With over 10 billion devices currently connected and a projected 80 billion by 2026, securing IoT devices is critical. Traditional security approaches face hurdles due to device diversity, while IoT devices are prone to rapid attacks. Behaviour-based methods, particularly utilising machine learning, offer potential solutions for both DI and AD. However, existing studies suffer from limitations in addressing IoT heterogeneity, analysing information leakage features, understanding machine learning insights, and ensuring reproducibility.

This research aims to bridge these gaps by developing robust, transparent, and generalizable solutions for IoT DI and AD. For DI, a novel aggregation algorithm addresses IP and non-IP device challenges, significantly improving accuracy. Comprehensive feature selection results in an optimal feature set, validated across diverse datasets. In AD, a packet-level expanding and rolling windows method detects attacks earlier, outperforming conventional flow methods. The models are evaluated on isolated first-time-seen attack datasets, showcasing their adaptability to novel attacks. Furthermore, machine learning models and features are analysed for deeper attack insights.

The thesis underscores the interdependence of device identification and attack detection within IoT security, emphasising their mutual reinforcement for network safety. By offering reproducible methodologies, transparent analyses, and adaptable models, this work contributes to enhancing the security of IoT devices and networks. Ultimately, this research paves the way for a more secure IoT ecosystem by addressing the unique challenges posed by IoT heterogeneity, resource limitations, and dynamic attack patterns.

Keywords: IoT security, fingerprinting, machine learning, device identification, attack detection

benem ol ḳahramān-ı pür-hüner kim
elimde ĥāmedür şemşīr-i ġamġam

Sabri (Divan: K. 14/24) [249]

Acknowledgements

I would like to extend my heartfelt gratitude to the Turkish Ministry of National Education for their unwavering support, covering my living expenses and tuition fees during my PhD. This scholarship was not only a financial aid but a testament to their belief in my potential. I would also like to express my profound appreciation to my guarantors and dear friends, Zehra, Fadime, and Hatice, who took on a considerable responsibility to ensure my academic journey. Your trust and support were invaluable.

I am deeply indebted to my exceptional supervisors, Dr Mike Just and Dr Michael Lones, who not only guided my PhD studies but also served as remarkable mentors and friends. Their continuous support and expertise significantly contributed to the quality and depth of my research.

Dr Wei Pang played a pivotal role by diligently reviewing my studies during my PhD journey, and offering valuable advice and guidance that enhanced the quality of my work.

During my industrial internship, I had the privilege of working under the guidance of Dr Firas Alsehly and Dr Francisco Zampella. Their mentorship and camaraderie made the experience truly enriching.

I would like to express my heartfelt appreciation to the friends I had the privilege of meeting and collaborating with during the course of my PhD journey. Their camaraderie, support, and shared experiences made this academic endeavour more meaningful. Their encouragement and friendship were invaluable throughout the challenges and successes we encountered together. It is with sincere gratitude that I recognize the role of these friends in making this journey memorable.

As the grandson of four illiterate grandparents and the first university graduate in my family, I am acutely aware of the immense sacrifices and unwavering support of my family that made this journey possible. My family, particularly my late mother Şemsi, provided me with opportunities far beyond what they themselves had. Their sacrifices, love, and encouragement have been the bedrock upon which my educational pursuits were built. I carry with me the legacy of their resilience and determination, and I am forever grateful for their belief in my potential.

Last but certainly not least, my deepest appreciation goes to my wife, Rabia, whose unwavering support and understanding have been the cornerstone of my success. We embarked on this academic journey together, and her presence has been my greatest source of strength and encouragement. With her by my side, I know that we are not only graduating together but embarking on new adventures hand in hand.

Together, all of you have played an indispensable role in my thesis and my life, and I am profoundly thankful for your contributions to my journey.

Inclusion of Published Works Form

Please note you are only required to complete this form if your thesis contains published works. If this is the case, please include this form within your thesis before submission.

Declaration

This thesis contains one or more multi-author published works. I hereby declare that the contributions of each author to these publications is as follows:

Citation details	Kahraman Kostas, Mike Just, and Michael A. Lones. IoTDevID: A Behavior-Based Device Identification Method for the IoT. *IEEE Internet of Things Journal*, 9(23):23741–23749, 1 Dec 2022. https://doi.org/10.1109/JIOT.2022.3191951
Author 1	Conducting the work described, writing, editing.
Author 2	Supervision, writing, editing.
Author 3	Supervision, writing, editing.

Please included additional citations as required.

Contents

1 Introduction **1**

1.1 Motivation . . . 1
1.2 Research Objectives . . . 5
1.3 Thesis Statement . . . 7
1.4 Contributions . . . 7
1.5 Organisation . . . 10

2 Background **13**

2.1 Overview . . . 13
2.2 Network Models . . . 13
2.2.1 Physical Layer . . . 15
2.2.2 Link Layer . . . 16
2.2.3 Network Layer . . . 16
2.2.4 Transport Layer . . . 17
2.2.5 Application Layer . . . 21
2.3 Machine Learning . . . 21
2.3.1 Basic concepts . . . 22
2.3.1.1 Training, Validation, and Test Set . . . 22
2.3.1.2 Overfitting and Underfitting . . . 23
2.3.1.3 Cross-validation . . . 24
2.3.2 ML Algorithms . . . 25
2.3.2.1 Naïve Bayes . . . 25
2.3.2.2 Support Vector Machine . . . 26
2.3.2.3 k-Nearest Neighbours . . . 26
2.3.2.4 Decision Tree . . . 27
2.3.2.5 Random Forest . . . 28
2.3.2.6 Gradient Boosting . . . 30

2.3.2.7 Logistic Regression . . . 30
2.3.2.8 Artificial Neural Networks and Multilayer Perceptrons . 31
2.3.3 Genetic Algorithms . . . 33
2.4 Evaluation . . . 33
2.4.1 Accuracy . . . 34
2.4.2 Recall . . . 35
2.4.3 Precision . . . 35
2.4.4 F1 Score . . . 35
2.4.5 Cohen's Kappa Coefficient . . . 36
2.5 Conclusions . . . 36

I IoT Device Identification 38

3 Literature Review for IoT Device Identification 39
3.1 Overview . . . 39
3.2 Literature Review . . . 39
3.2.1 According to Years . . . 40
3.2.2 According to Classification Method . . . 43
3.2.3 According to Data . . . 43
3.2.4 According to Features . . . 48
3.2.5 According to Machine Learning and Results . . . 52
3.2.6 According to Vulnerabilities . . . 54
3.3 Conclusions . . . 60

4 Device Identification Methodology 63
4.1 Overview . . . 63
4.2 Materials and Methods . . . 65
4.2.1 IoT Data Selection . . . 66
4.2.2 Individual, Aggregated and Mixed Method Packets . . . 67
4.2.3 Feature Extraction . . . 71
4.3 Model Selection . . . 73
4.3.1 Feature Selection . . . 73
4.3.2 Algorithm Selection . . . 78
4.4 Performance Evaluation . . . 80
4.5 Comparison with Previous Work . . . 87

4.6 Limitations . . . 88
4.7 Conclusions . . . 90

5 Externally Validating the Device Identification Methodology 92
5.1 Overview . . . 92
5.2 Case Study . . . 93
5.2.1 CICIoT22 dataset . . . 93
5.2.2 Feature Extraction and Labelling . . . 94
5.3 Results . . . 95
5.3.1 Analysis of Data Quality . . . 95
5.3.2 Dataset Construction . . . 98
5.3.3 IoTDevID Evaluation . . . 99
5.3.4 Evaluation by Including non-IP Devices . . . 103
5.4 Conclusions . . . 105

6 Measuring Model Instance Generalisability 110
6.1 Overview . . . 110
6.2 MonIoTr Dataset . . . 111
6.3 Experiments . . . 114
6.3.1 Comparison of VPN and Normal Sessions in Same Laboratory. . . . 115
6.3.2 Measurement of Model Generalizability . . . 120
6.4 Conclusions . . . 126

II IoT Attack Detection 128

7 Literature Review for IoT Attack Detection 129
7.1 Overview . . . 129
7.2 Related Work . . . 130
7.2.1 Literature Review Criteria . . . 130
7.2.2 Data . . . 135
7.2.3 Features . . . 139
7.2.4 Machine Learning and Evaluation Metrics . . . 140
7.3 Conclusions . . . 141

8 The Risks of Using Only Individual Packet Features 142
8.1 Overview . . . 142

8.2 Why Should IPF not be Used? . . . 143
8.3 IPF Usage in the Literature . . . 144
8.4 How does IPF Succeed? . . . 147
8.4.1 Identifying Features . . . 147
8.4.2 Low Data Complexity . . . 150
8.5 Conclusion . . . 156

9 Generalisable Models for IoT Attack Detection 157
9.1 Overview . . . 157
9.2 Materials and Methods . . . 158
9.2.1 Data Selection . . . 159
9.2.2 ML Algorithm Selection . . . 162
9.2.3 Feature Extraction Methods . . . 162
9.2.3.1 Individual Packet-Based Features . . . 162
9.2.3.2 Flow-Based Features . . . 163
9.2.3.3 Window-Based Features . . . 163
9.2.4 Feature Selection . . . 165
9.2.4.1 Feature elimination . . . 165
9.2.4.2 Feature Selection with GA and External Feedback . . . 166
9.3 Performance Evaluation . . . 167
9.3.1 Comparison with Flow-Based Features . . . 168
9.3.2 Feature Effectiveness . . . 168
9.3.2.1 Flood attacks . . . 171
9.3.2.2 ARPS, BF, and PS Attacks . . . 175
9.3.2.3 Flow-less Attacks . . . 180
9.3.2.4 Summary . . . 180
9.4 Conclusions . . . 181

10 Conclusions, Limitations and Future Work 184
10.1 Conclusions . . . 184
10.2 Summary . . . 191
10.3 Limitations . . . 193
10.3.1 Device Identification Limitations . . . 194
10.3.2 Attack Detection Limitations . . . 195
10.4 Future Work . . . 196

A Appendices **198**
A.1 Feature Comparison: Individual Packet Headers vs. Flow-Based Features 198
A.2 Deep Learning for IoT Device Identification and Attack Detection 200
A.2.1 CNN-based IoT Device Identification 200
A.2.2 LSTM-based IoT Device Identification 201
A.2.3 CNN-based IoT Attack Detection 202
A.2.4 BERT-based Attack Detection 203
A.2.5 Conclusion 203
A.3 IoTDevID 206
A.4 MonIoTr 213
A.4.1 Comparison of active and idle states 218
A.5 All Attacks Graphs 218
A.6 Machine Learning Hyperparameters: Model Configuration Details 220
A.6.1 Genetic Algorithm Hyperparameters 220
A.6.2 Device Identification Hyperparameters 221
A.6.3 Attack Detection Hyperparameters 222
A.7 IoTGeM 229
A.7.1 Problems with the IoTID20 Dataset 229
A.7.2 IoTGeM extended ML Results 231

Bibliography **238**

List of Tables

2.1 Fields of the IP header and their descriptions. 18
2.2 Fields of the TCP header and their descriptions. 20

3.1 Comparison of related work between 2017-2021. 41
3.1 Comparison of related work between 2022-2023. 42
3.2 Open access IoT datasets used in the literature and some of their properties. 44
3.3 Vulnerabilities in device identification studies. 61

4.1 Device MAC addresses and labels in Aalto dataset. 67
4.2 Which data in the UNSWM dataset were taken from which date range. . 72
4.3 Results of feature selection. 79
4.4 Comparison of ML algorithms with average and standard deviation . . . 79
4.5 Results and their SD obtained using Individual, Aggregated and Mixed. . 81
4.6 Device proportions over the entire dataset, F1 scores per device (Aalto). 83
4.7 Device proportions over the entire dataset, F1 scores per device (UNSWM) 84
4.8 Device proportions over the entire dataset, F1 scores per device (LSIF). . 85
4.9 Comparison of feature sets on the Aalto, UNSWM, and LSIF datasets. . 88

5.1 Results for four device conditions using DT. 96
5.2 Results for 4 conditions with individual and aggregated methods. 100
5.3 Class-based results for individual and aggregated methods. 102
5.4 The total number of packets generated by the Zigbee devices. 103
5.5 Class-based results of IP and non-IP devices with AA case. 104
5.6 Total number of packets generated by devices. 108
5.7 Class-based F1 score results for AA case, showing means of 100 repeats. . 109

6.1 Data separated by country, active/idle and VPN state. 113
6.2 Active sessions, USA∩UK, USA, UK. 114
6.3 Comparison of VPN and normal data 116

6.4 Results obtained with data collected from different country laboratories . 121
6.5 Aggregated class-based results of two different laboratories 122
6.6 Intersecting devices in UK and US laboratories 123
6.7 The IoTDevID method and its study datasets and results. 127

7.1 Comparison of related work on IoT attack detection (1). 131
7.1 Comparison of related work on IoT attack detection (2). 132
7.2 The datasets used in the literature and some properties of them. 134

8.1 Studies in the literature review using individual packet features. 145
8.2 Near-perfect classification of 2 datasets afflicted by low data complexity. . 155

9.1 Datasets that can be used for IoT attack detection 160
9.2 F1 scores of attack detection models. 169
9.3 F1 score comparison of window and flow-based approaches. 170

A.1 Feature Comparison: Cross-validation 199
A.2 Feature Comparison: Session vs. Session 199
A.3 Feature Comparison: Different Laboratories 200
A.4 The comparative analysis of DI with CNN and DT 201
A.5 The comparative analysis of DI with LSTM and DT 201
A.6 Comparison of time series graphs' results 203
A.7 Results of BERT and DT binary classification models. 204
A.8 Results of BERT and DT multiclass classification models. 205
A.9 Feature list and descriptions (1/4) 209
A.9 Feature list and descriptions (2/4) 210
A.9 Feature list and descriptions (3/4) 211
A.9 Feature list and descriptions (4/4) 212
A.10 UK and UK-VPN Class based results 213
A.11 US and US-VPN Class based results 214
A.12 Class-based results of the evaluation using data from two different labs . 215
A.13 Hyperparameter configurations for DT (Attack Detection) 222
A.14 Hyperparameter configurations for RF (Attack Detection) 223
A.15 Hyperparameter configurations for XGB (Attack Detection) 224
A.16 Hyperparameter configurations for kNN (Attack Detection) 225
A.17 Hyperparameter configurations for NB (Attack Detection) 226
A.18 Hyperparameter configurations for LR (Attack Detection) 227

A.19 Hyperparameter configurations for MLP (Attack Detection) 228
A.20 Hyperparameter configurations for SVM (Attack Detection) 229
A.21 Comparison of IoT-NID raw data and the IoTDI20 dataset. 230
A.22 Window approach CV results. 232
A.23 Window approach Session results 233
A.24 Window approach Test results. 234
A.25 Flow approach CV results. 235
A.26 Flow approach Session results 236
A.27 Flow approach Test results. 237

List of Figures

1.1 IoT application areas and key participants. 1
1.2 Change in the number of connected IoT devices by years. 2
1.3 Visualisation of the relationship between ID and AD 8
1.4 The visual abstract of the thesis. 11

2.1 OSI and TCP/IP network models. 14
2.2 Encapsulation/decapsulation at each layer during data transfer. 15
2.3 The MAC address of Fitbit Aria WiFi-enabled scale from Aalto dataset. 16
2.4 Structure and parts of the IPv4 header. 17
2.5 Structure and parts of the TCP header. 19
2.6 Structure and parts of the User Datagram Protocol (UDP) header. . . . 21
2.7 Underfit, ideal, and overfit model 23
2.8 Segmentation processes of data for machine learning 24
2.9 Distribution of ML algorithms used in DI, and AD. 25
2.10 Representation of the SVM kernel method. 26
2.11 Representation of the behaviour of the KNN algorithm. 27
2.12 Binary classification of an IoT device using the DT algorithm. 28
2.13 Representation of the change in the decision tree after the data is rotated. 29
2.14 Binary classification of an IoT device using the RF algorithm. 29
2.15 The working principle of GB. 31
2.16 The function used by LR for classification. 31
2.17 Representation of an artificial neuron. 32
2.18 The working principle of MLP. 32
2.19 Representative visualization of the confusion matrix. 34

3.1 Distribution of device identification studies by years. 43
3.2 The change of labels of the CICIoT22 dataset according methods. 44
3.3 Datasets used in the literature and their usage rates. 45

3.4 Feature types, usage frequency, and temporal distribution. 49
3.5 The raw bytes of a network packet from the Fitbit Aria. 51
3.6 Distribution of ML algorithms and the evaluation methods in the literature 53
3.7 The 28x28 images of the device's raw bytes in the Aalto dataset. 54
3.8 Word cloud of domain names of 3 different devices from the same brand . 57
3.9 Example transfer problem in the Aalto dataset. 58
3.10 The impact of the number of classes on chance success 60

4.1 The steps of the aggregation algorithm. 69
4.2 Results of feature importance voting on the Aalto dataset. 74
4.3 Generality of identifying features, measured using isolated and CV. . . . 75
4.4 Word clouds of port numbers used by two devices of the same type. . . . 76
4.5 Representation flow diagram of genetic algorithm. 77
4.6 Relationship between aggregation group size and performance. 80
4.7 Confusion matrix of devices with poor discrimination. 86

5.1 Comparison of sessions with the dataset's same session ID. 97
5.2 Creating datasets with larger representation by combining sessions. . . . 99
5.3 Confusion matrix for case AA, showing means of 100 results. 101
5.4 Confusion matrix for case AA with non-IP devices. 107

6.1 Venn diagram showing the distribution of devices by country. 113
6.2 The data collection visualisation of MonIoTr dataset. 114
6.3 VPN and Normal data comparison MonIoTr UK laboratory 117
6.4 VPN and Normal data comparison MonIoTr US laboratory 118
6.5 VPN and Normal data comparison MonIoTr UK and UK laboratories . . 119

7.1 Datasets, ML, feature types, and evaluation metrics in the literature. . . 133
7.2 Distribution of attacks in the literature by dataset and attack groups. . . 138

8.1 Legitimate and malicious use of 3-way handshake 143
8.2 An 80/20 split for a hypothetical dataset containing 40 samples. 148
8.3 Comparison of CV and isolated data. 149
8.4 Visualization of high-achieving decision tree models. 150
8.5 The distribution of packet size malicious data (HTTP Flood). 150
8.6 Distribution of labels in Kitsune and IoT-NID datasets. 152
8.7 Complexity analysis of UDP Flood, Mirai, and Video injection attacks. . 153

8.8 Comparison of the change in the complexity value and model success. . . 155

9.1 The visualisation of steps in system implementation. 159
9.2 Classification of datasets according to where they will be used. 161
9.3 The process of extracting window-based features. 164
9.4 Utilizing diverse window sizes for Kitsune MitM attack detection. 165
9.5 Voting process during the feature elimination step. 166
9.6 Visualisation of feature selection with Genetic Algorithm. 167
9.7 Comparison of window and flow-based features for the ACK attack. . . . 172
9.8 Comparison of window (WB) and flow (FB) models for HTTP Flood attack. 173
9.9 Comparison of window (WB) and flow (FB) models for SYN attack. . . . 174
9.10 Comparison of window (WB) and flow (FB) models for UDP attack. . . 175
9.11 Comparison of window (WB) and flow (FB) models for ARPS attack. . . 177
9.12 Comparison of window and flow-based models for BF attack. 178
9.13 Comparison of window and flow-based models for the PS attack. 179
9.14 SHAP graphs of models for MHD, SHD, OSD attacks. 181

10.1 Proposed Integration of Device Identification and attack Detection 191

A.1 Creating a dataset from a time series graph. 202
A.2 Confusion matrix of BERT and DT binary classification models. 204
A.3 Confusion matrix of BERT and DT multiclass classification models. . . . 206
A.4 Confusion matrix for UNSW Merged (UNSWM) dataset 207
A.5 Confusion matrix for Locality-Sensitive IoT Fingerprinting (LSIF) dataset 208
A.7 Confusion matrices of MonIoTr dataset (UK vs US case) 216
A.8 Confusion matrices of MonIoTr dataset (US vs UK case) 217
A.9 Comparison of all states in the MonIoTr dataset with each other 218
A.10 Distribution of attacks in the literature by dataset and attack groups. . . 219
A.11 Hyperparameters utilized for the three datasets in Chapter 4. 221
A.12 Hyperparameters utilized for the three datasets in Chapter 5 & 6. 221

Glossary

AA Active vs Active.

Aalto Aalto University - IoT Devices Captures.

AB Adaboost, Adaptive Boosting.

ACK Acknowledgement.

AD Attack detection.

AE AutoEncoder.

AI Active vs Idle.

ANN Artificial Neural Network.

ARP Address Resolution Protocol.

ATM Asynchronous Transfer Mode.

BF Bruteforce.

BLE Bluetooth Low Energy.

BOOTP Bootstrap Protocol.

CART Classification And Regression Trees.

CB Catboost, Category Boosting.

CI Class Identification.

CICIoT22 CIC IoT Dataset 2022.

CNN Convolutional Neural Network.

CSV Comma-separated values.

CV Cross-Validation.

CWR Congestion Window Reduced.

DDoS Distributed Denial of Service.

DI Device Identification.

DNS Domain Name System.

DoS Denial of Service.

DT Decision Tree.

ECE Echo.

F Flooding.

F1 F1 Score.

FDDI Fiber Distributed Data Interface.

FIN Finish.

FN False Negative.

FP False Positive.

FTP File Transfer Protocol.

GA Genetic Algorithm.

GAN Generative Adversarial Network.

GB Gradient Boosting.

HTTP Hypertext Transfer Protocol.

HTTPS Hypertext Transfer Protocol Secure.

IA Idle vs Active.

IAT Inter-Arrival Times.

ICMP Internet Control Message Protocol.

ID3 Iterative Dichotomiser 3.

IEEE Institute of Electrical and Electronics Engineers.

IETF Internet Engineering Task Force Software.

II Idle vs Idle.

IoT Internet of Things.

IP Internet Protocol.

IPF Individual Packet Feature.

IPsec Internet Protocol Security.

kappa Cohen's kappa coefficient.

KNN k-Nearest Neighbours.

LAN Local Area Network.

LGBM LightGBM,Light Gradient-Boosting Machine.

LLC Logical Link Control.

LR Logistic (Logit) Regression.

LSIF Locality-Sensitive IoT Fingerprinting.

LSTM Long short-term memory.

MAC Media Access Control.

MHD Mirai Host Discovery.

ML Machine Learning.

MLP MultiLayer Perceptron.

MonIoTr dataset for "information exposure for consumer IoT devices" (IMC'19).

MQTT MQ Telemetry Transport.

NAT Network Address Translation.

NB Naïve Bayes.

NTP Network Time Protocol.

OSD Operating System Version Discovery.

OSI Open Systems Interconnection.

PCA Principal Component Analysis.

pcap Packet Capture.

PPP Point-to-Point Protocol.

PS Port Scanning.

PSH Push.

RF Random Forest.

RNN Recurrent Neural Network.

RST Reset.

S Spoofing.

SDN Software-Defined Networking.

SHAP SHapley Additive exPlanations.

SHD Scanning Host Discovery.

SLIP Serial Line Internet Protocol.

SMOTE Synthetic Minority Oversampling TEchnique.

SMTP Simple Mail Transfer Protocol.

SSDP Simple Service Discovery Protocol.

SUIT Updates for Internet of Things.

SVM Support Vector Machine.

SYN Synchronize.

TCP Transmission Control Protocol.

TI Type Identification.

TN True Negative.

TP True Positive.

TTL Time to Live.

UDP User Datagram Protocol.

UI Unique Identification.

UK United Kingdom.

UNSW-AD UNSW IoT Benign And Attack Traces, Data for ACM SOSR 2019.

UNSW-DI UNSW IoT Traffic Traces, Data for IEEE TMC 2018.

UNSWM UNSW Merged.

URG Urgent.

URL Uniform Resource Locator.

US United States of America.

VPN Virtual Private Network.

WiFi Wireless Fidelity.

WSN Wireless Sensor Network.

WWW World Wide Web.

XAI Explainable Artificial Intelligence.

XGB XGBoost, eXtreme Gradient Boosting.

Chapter 1

Introduction

1.1 Motivation

Internet of Things (IoT) is a collection of objects with embedded systems that can communicate with each other, wired or wirelessly. The "things" can be any physical items that we have used now or will use in the future [92]. IoT contributes greatly to human life in many critical areas such as [123] smart homes/cities, infrastructure, retail, healthcare, transportation, agriculture, military, petrochemical industry, and manufacturing. IoT application areas and key participants are visualised in Figure 1.1. According to [216], the number of devices currently exceed 15 billion and will reach approximately 30 billion devices by 2030 (see Figure 1.2), and the total market will be worth about 1.1 trillion USD [96]. In other words, more than 150,000 devices join the global IoT network every minute.

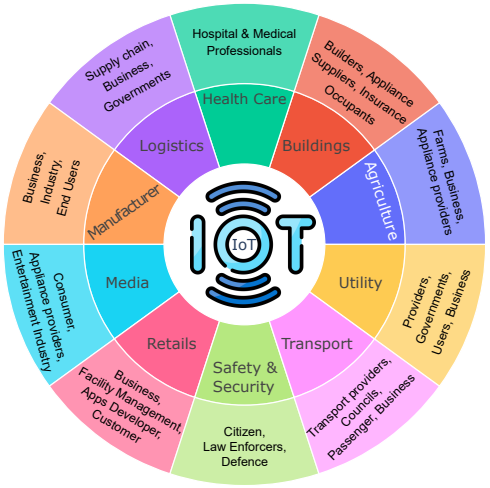


Figure 1.1: IoT application areas and key participants. Adapted from [109].

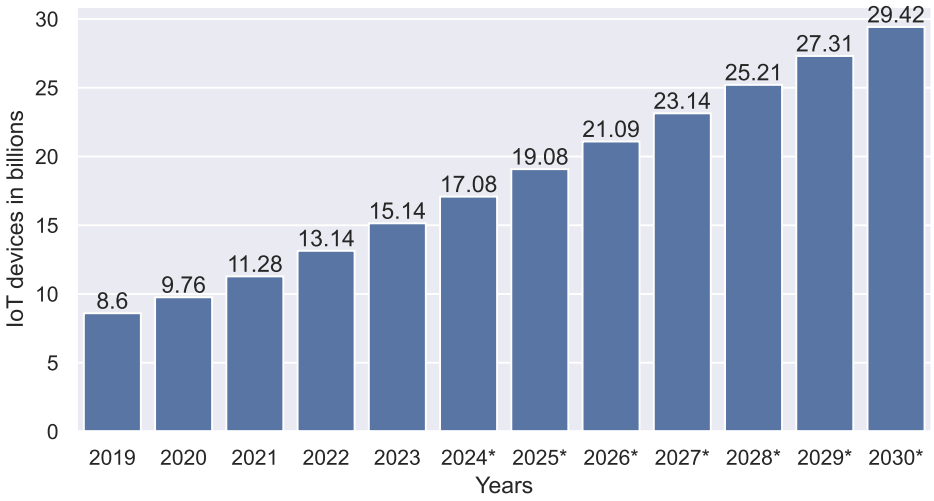


Figure 1.2: Change in the number of connected IoT devices by years, including both real data and forecasted figures (*). Adapted from [216].

In parallel with this rapid development of devices and providers, security remains an important issue. Securing IoT devices can be challenging as a result of the heterogeneity of the devices [36,92] and the difficulty of implementing traditional security solutions due to limited resources, such as processor, battery and bandwidth [244]. A device can have many sensors (temperature, humidity, motion, light, etc.), and the channels it uses to communicate with other devices may need very different requirements (such as Ethernet, Wireless Fidelity (WiFi), Bluetooth, Zigbee, Z-Wave etc. [171]). Thus, it is challenging to treat all IoT devices homogeneously. Although some research focuses on specialist areas such as home appliances [11,176] or smart cities [62,179], many devices have very different characteristics even under this classification. For example, baby monitors and smart kettles are both classified under home devices. However, these devices are very different from each other in terms of both software and hardware.

IoT devices tend to have more vulnerabilities than conventional devices because they have very varied hardware and software, most of them do not have familiar interfaces, and they have a wide variety of manufacturer profiles. According to the statistics [162], a new IoT device included in the network receives its first attack within 5 hours and becomes the target of the specific attack within 24 hours. Most attacks targeting IoT devices exploit vulnerabilities in those devices [151]. Beyond the susceptibility of compromised devices to attacks, another concern is their potential role as tools in botnet attacks, as exemplified by notorious examples such as Mirai, URSNIF and BASHLITE [20]. In these attacks, attacker-captured IoT devices are transformed into armies of bots,

organizing high-volume Distributed Denial of Service (DDoS) attacks [20]. Although it is important to detect these attacks and take the necessary measures at the time of this attack, it is equally important to identify the vulnerability that causes these attacks and to take precautions according to this vulnerability, as it will greatly reduce the risk of attack. This perspective aligns with the Internet Engineering Task Force Software (IETF) Updates for Internet of Things (SUIT) Working Group's standards, particularly RFC 9124 [168], which establishes a robust framework for secure software updates and device identification. This standard mandates unique device identifiers, comprehensive software and hardware configuration details, and manufacturer-signed product software notifications, ensuring authenticity, preventing tampering, and guaranteeing the receipt of secure updates. In this context, using a device identification method, recognising the devices in our system and taking preventive measures for vulnerabilities are also essential for a secure IoT network.

Signature-based and behaviour-based methods are used for ensuring the security of IoT devices. Signature-based methods in IoT security rely on predefined patterns or signatures to identify known threats, while behaviour-based methods analyse the actions and patterns of device behaviour to detect anomalies indicative of potential security issues [49]. Although the signature-based method promises high-accuracy detection in known situations, it does not work in unknown situations. Behaviour-based approaches, on the other hand, are more flexible and can detect unseen situations [103]. Another point is that 75% of web traffic is transmitted in encrypted form and this rate is increasing day by day [180]. It is not practical to apply the signature-based method since the content of the data streams cannot be accessed. This is because the signature-based approaches aim to find the signatures that will examine the payload of the network packets and compare them with the previously found attacks. However, if these payloads are encrypted, the encryption protocol prevents the detection system so the signature-based method cannot examine the payload [124]. On the other hand, the behaviour-based approach that examines not the content of packets but the format and statistics, so it is effective both on encrypted data flow and in detecting new situations. For this reason, many researchers today focus on detecting behaviour patterns and automating this detection with machine learning methods in both device identification and attack detection.

With the rapid rise of machine learning, as in many machine learning studies [106], reproducibility is a serious problem in IoT security applications. While there are many studies in the literature that describe results with very high accuracy, many of these are

not reproducible [152]. The reliability of non-generalisable models, many of which suffer from overfitting, is highly questionable.

In this context, the following is a non-exhaustive list of problems observed in existing studies:

- Failure to accommodate the inherent heterogeneity of IoT. Ignoring devices using non-Internet Protocol (IP) media such as Zigbee, Z-Wave, and Bluetooth, which are very common among IoT devices. For example, we have identified the transfer problem that occurs when non-IoT devices are used together with IP devices (see Chapter 3) and we have implemented a solution to this problem (see Chapter 4).

- Information leakage arising from the features employed in IoT security represents a prevalent challenge. Case studies addressing this issue are thoroughly examined in Chapter 3 focusing on device identification and Chapter 8 centred on attack detection.

- There is a widespread tendency to depend on a single dataset for obtaining results. To illustrate, in the studies we analysed in the literature reviews, out of 71 studies focusing on device identification, 56 exclusively utilized a single dataset, and in the context of attack detection, 55 out of 68 studies followed the same pattern. Notably, the application of cross-validation with diverse datasets is more rare occurrence. In the realm of attack detection, only one study [183] has ventured into cross-validation, and as far as device identification is concerned, to the best of our knowledge, no instance of cross-validation has been reported—except in our study detailed in Chapter 6. Consequently, for the majority of studies, there is a lack of insights into the generalizability of models when applied to dissimilar datasets with potentially distinct data.

- The prevalence of imbalanced data, stemming from the heterogeneity of IoT devices, constitutes a common challenge in IoT security. When confronted with imbalanced datasets, depending solely on a general evaluation metric such as accuracy can yield misleading results (see Section 2.4). Our literature review reveals that among the studies on device identification, 22 out of 71 exclusively relied on accuracy (see Table 3.1), and in the realm of attack detection, 13 out of 68 studies (see Table 7.1) adopted a similar approach.

- Feature analysis often receives insufficient attention in various studies. Our literature review (see Chapter 3 & 7) revealed a lack of emphasis on feature selection in

27 out of 68 attack detection studies and 45 out of 71 device identification studies. Commonly utilized techniques for feature selection include feature importance scores [53, 213], dimensionality reduction [203, 224], and genetic algorithms [8, 78]. In our approach, we addressed potential information leakage by analysing and subsequently eliminating relevant features. Employing a two-step feature selection strategy, we initially utilized a voting mechanism, followed by the application of genetic algorithms. Furthermore, we integrated Explainable Artificial Intelligence (XAI) methods [130] to enhance the depth of attack nature analysis in attack detection.

- While notable contributions in device identification [57, 64] and attack detection [51, 83] have enhanced transparency and reproducibility by sharing their scripts, a significant portion of research in both domains lacks such practices. Moreover, approximately 23% of the datasets employed for device identification are not openly accessible (see Figure 3.3), presenting a substantial risk to the reliability of results. In our study, we exclusively utilized openly accessible data and made our code openly available, adhering to principles of transparency and reproducibility.

Conducting generalizable and reproducible IoT security studies is crucial for advancing detection capabilities while steering clear of the common pitfalls outlined in above. In this context, this thesis aims to provide reproducible and generalisable IoT device identification and attack detection with realistic results.

1.2 Research Objectives

According to our analysis in the Motivation section 1.1, a machine learning approach using behaviour-based features is ideal for a flexible and generalisable IoT security study. From our perspective, a secure IoT network should be supported by two elements. The first is device identification and the second is attack detection. In this context, device identification ensures that IoT devices are up-to-date and vulnerability-free, while attack detection is responsible for eliminating the attacks that occur. In this context, our goals are: To develop a device identification method that identifies IoT devices with high success rate regardless of the medium they use (IP or non-IP), thereby preventing vulnerable IoT devices from being exploited for participation in Distributed Denial of Service (DDoS) attacks against substantial infrastructures. Additionally, to develop an

attack detection method that can detect attacks against IoT devices or by IoT devices early and with high success. The common elements for both methods are reproducibility, transparency and generalisability. In this context, the main objectives of our thesis can be expressed in the following research questions.

- **RQ1** How can IoT device identification be done using machine learning?
  - **RQ1.1** What are the machine learning-based studies in the literature that focus on IoT device identification, and what are the specific features, datasets, machine learning methods, and evaluation metrics utilised in these studies?
  - **RQ1.2** What are the essential characteristics that the features, datasets, machine learning methods, and evaluation metrics should possess to ensure effective and accurate device identification?
  - **RQ1.3** What are the challenges and limitations associated with the current device identification techniques?
  - **RQ1.4** Can the developed device identification method successfully identify devices using different network environments, considering variations in network configurations and communication protocols?
  - **RQ1.5** How does the type of data collected from IoT devices (e.g., idle or in use) impact the performance of the device identification model, and are there any specific data types that pose challenges for the identification process?
  - **RQ1.6** Can the proposed device identification method be adapted to operate effectively in encrypted networks, and how does the presence of encryption influence the accuracy of device identification?
  - **RQ1.7** How can we systematically evaluate and test the generalizability of our device identification method across diverse datasets and scenarios to ensure its reliability in real-world IoT environments?
  - **RQ1.8** Can the device identification method designed in this study successfully detect and identify devices that have not been encountered before (first-time-seen devices), demonstrating its capability to handle novel and unseen IoT devices?
- **RQ2** How can machine learning be used for attack detection in IoT networks?

- **RQ2.1** What is the extent of existing research in the literature that concentrates on the application of machine learning for attack detection within IoT networks?
- **RQ2.2** In these research studies, what specific features, datasets, machine learning methods, and evaluation metrics have been employed to investigate and develop effective attack detection mechanisms?
- **RQ2.3** What are the challenges and limitations associated with the current IoT attack detection techniques?
- **RQ2.4** What are the essential characteristics that the features, datasets, machine learning methods, and evaluation metrics should possess to ensure effective and accurate attack detection?
- **RQ2.5** Given the intertwined nature of device identification and attack detection in IoT networks, can the features derived from device identification methodologies also be effectively utilized for attack detection mechanisms?
- **RQ2.6** Can our attack detection method successfully detect attacks that have not been encountered before (first-time-seen attacks, e.g., an attack recorded in another dataset), demonstrating its capability to handle novel and unseen attacks?

1.3 Thesis Statement

Although attack detection and device identification are considered by many researchers to be separate areas, they are in fact systems that need and support each other. Both are needed for a secure IoT network. The relationship between Device Identification (DI) and Attack detection (AD) is visualised in Figure 1.3. This thesis investigates and analyses up-to-date, transparent, and generalisable solutions for IoT security in the context of device identification and attack detection.

1.4 Contributions

This study underscores the interplay between device identification and attack detection, elucidating the necessity for incorporating attack detection alongside device identification. Device identification serves the purpose of maintaining device currency by assessing vulnerability levels and establishing an authorization framework based on these levels.



Figure 1.3: Visualisation of the relationship between ID and AD. Individual packet features refer to characteristics extracted directly from individual network packets, whereas related or dependent features concentrate on the interconnections and relationships between packets.

Consequently, vulnerable devices are isolated within an IoT network, reducing the potential for attacks. Conversely, our proposed device identification methodology relies on benign individual packet attributes derived from network packets. To facilitate effective device identification, it is imperative to have attack detection mechanisms that ensure the benign nature of network traffic. However, attack detection methods necessitate features reflecting packet relationships rather than individual packet attributes. Hence, while device identification and attack detection are mutually indispensable, their operational methodologies differ significantly. We contend that integrating these approaches within a unified framework is crucial for enhancing IoT security. Our work contributes to the device identification and attack detection areas of IoT security as listed below.

- **C1** For device identification:
  - **C1.1** We conducted a comprehensive literature review, analysing various studies, datasets, and machine learning algorithms used in device identification. We identified the mistakes made in the studies in the literature and investigated how these mistakes can be corrected.
  - **C1.2** In response to the transfer problem resulting from combining IP and non-IP devices in IoT networks, we developed the aggregation algorithm. This innovative method significantly enhances device identification success by effectively handling both IP and non-IP devices across different mediums.
  - **C1.3** To ensure the generalizability of our approach, we carefully analysed the features used in device identification, removing any potential barriers to

generalization. Through a multi-layer feature selection method, we curated an ideal feature set tailored to device identification.

- **C1.4** To validate the effectiveness of our feature set, we applied it to diverse datasets, ensuring its robustness across different scenarios.
- **C1.5** To obtain more reliable results, we strictly isolated training and test data and utilized a wide range of evaluation criteria. This approach enhances the credibility of our findings and fosters a more comprehensive assessment of the device identification process.
- **C1.6** We performed validation of our device identification methodology on datasets containing diverse data types, including both encrypted and unencrypted (using dataset for "information exposure for consumer IoT devices" (IMC'19) (MonIoTr)), in-use and idle scenarios (using MonIoTr & CIC IoT Dataset 2022 (CICIoT22) datasets), to assess the influence of different data types on the device identification process.
- **C1.7** Additionally, we conducted testing on independent datasets to evaluate the generalizability of our feature set (using MonIoTr & CICIoT22 datasets) and models (using MonIoTr dataset). Our results demonstrated that our feature set and models are capable of effectively identifying devices in different datasets, even when encountering first-time-seen data.

- **C2** For attack detection:
  - **C2.1** Through a comprehensive literature review on attack detection, we compiled and analysed studies, datasets, features, and machine learning algorithms used. We identified errors in the literature and explored potential corrections.
  - **C2.2** We investigated the usage of individual features in attack detection and highlighted the issues of information leakage and overfitting that may arise from their implementation.
  - **C2.3** To achieve earlier detection, we developed an expanding and rolling windows method at the packet level and compared it with the commonly used flow method, demonstrating its superiority in different attack scenarios.
  - **C2.4** We showcased the generalizability of our models by evaluating their performance on first-time-seen attack datasets isolated from the training process.

– **C2.5** For deeper insights into network attack nature, we analysed machine learning models and features, aiming to obtain more explainable models.

1.5 Organisation

This thesis consists of 10 chapters, each exploring various aspects of device identification and attack detection. A visual abstract is provided for enhanced comprehension (see Figure 1.4).

Chapter 2 presents core concepts shared between device identification and attack detection, encompassing: network models, machine learning algorithms, genetic algorithms, and evaluation methods.

Chapter 3 is the start of Part I (Device Identification). It conducts a comprehensive review of the device identification literature, categorising and analysing studies based on types, datasets, features, machine learning methods, and results (RQ1.1, RQ1.2, RQ1.3).

Chapter 4 introduces our device identification methodology, providing detailed descriptions of the three datasets used. A multi-layer feature selection process is applied to the first dataset (Aalto University - IoT Devices Captures (Aalto)), involving analysis, pruning, and elimination of redundancies through a genetic algorithm. Selected features and models are tested on the other two datasets (UNSW Merged (UNSWM), Locality-Sensitive IoT Fingerprinting (LSIF)), forming a foundational aspect of device identification (RQ1.2, RQ1.3, RQ1.4, RQ1.7).

In Chapter 5, the CICIoT22 dataset, containing active and idle device data, is employed to assess the impact of data quality on identification performance. Additionally using CICIoT22 dataset, the methodology is validated on a large, independent dataset, exploring its performance with non-IP devices (RQ1.4, RQ1.5, RQ1.7).

Chapter 6 tests the device identification methodology on encrypted networks, demonstrating its effectiveness using the MonIoTr dataset(RQ1.6). Furthermore, the implementation is assessed using data recorded in different laboratory settings to ascertain model generalizability. In particular, this allows us to measure a model's ability to recognise the same device in diverse environments (RQ1.4, RQ1.6, RQ1.7, RQ1.8).

Chapter 7 is the start of Part II (Attack Detection). It conducts a comprehensive review of the IoT attack detection literature, categorising and analysing studies based on types, datasets, features, machine learning methods, and results (RQ2.1, RQ2.2, RQ2.3, RQ2.4).

Chapter 8 extensively explores the drawbacks of employing individual packets for

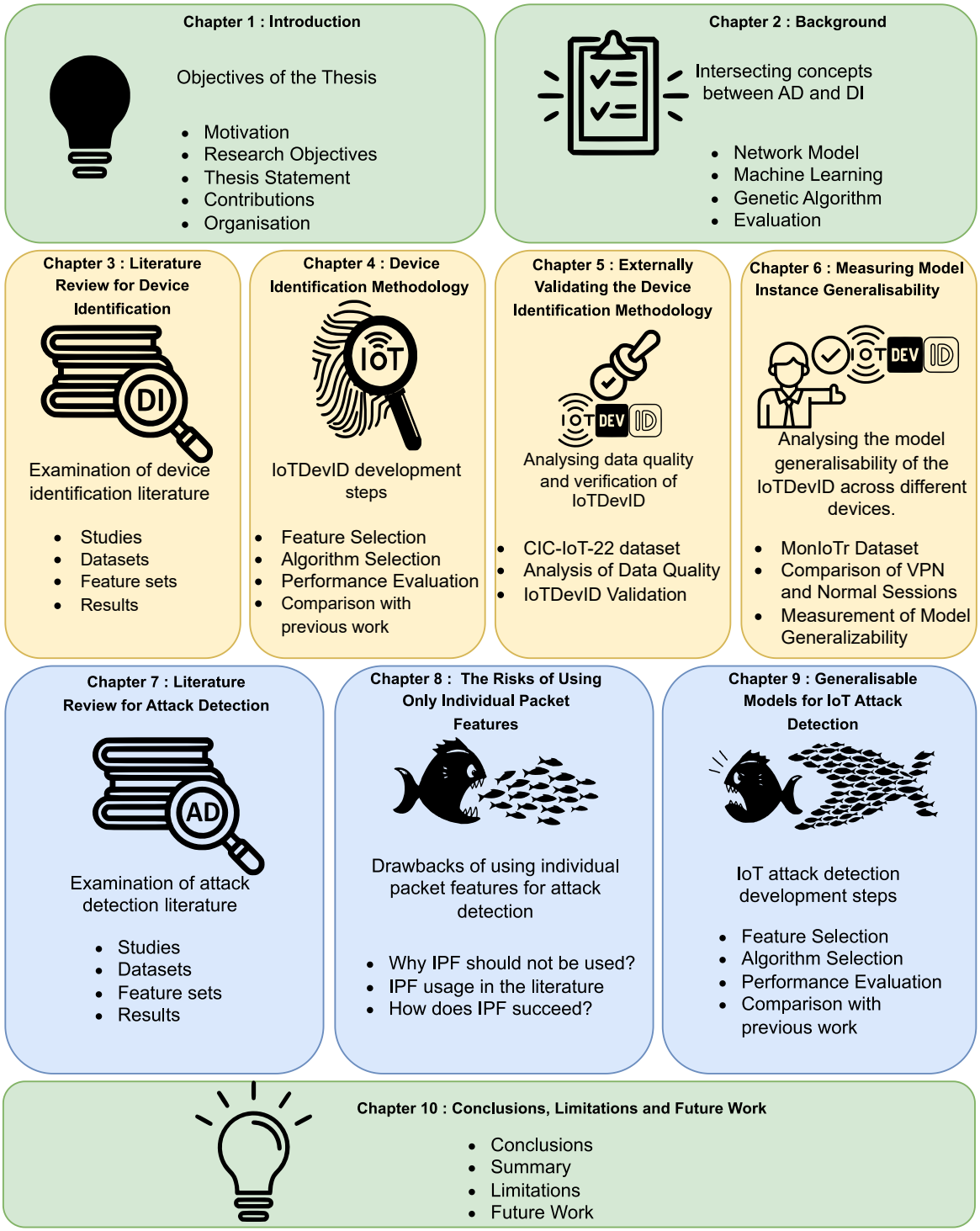


Figure 1.4: The visual abstract of the thesis. Intersection: , Device identification: , Attack detection: .

attack detection in prior research. Through the review of relevant studies, it critically evaluates the merits and limitations of this approach (RQ2.1, RQ2.2, RQ2.3, RQ2.5).

Chapter 9 presents a strategy for IoT network attack detection, emphasising generalizability and enhanced performance. The approach involves refining feature extraction with a rolling window technique and implementing a multi-step feature selection process to counteract overfitting. Models are developed and evaluated on isolated training and testing datasets to avoid information leakage, enhancing generality. The evaluation encompasses various machine learning models, metrics, and datasets. Trust in the models is fortified through XAI techniques, identifying key features for accurate attack detection (RQ2.2, RQ2.3, RQ2.4, RQ2.6).

The last chapter, Chapter 10, concludes the thesis by briefly summarising all chapters of the thesis in parallel with the research questions. In the remainder of this chapter, limitations and future work are discussed.

Chapter 2

Background

2.1 Overview

This background section explores some of the fundamental concepts common to both IoT device identification and IoT attack detection. The key concepts that cut across both studies are as follows. In Section 2.2 (Network Models), network models are reviewed, including network layers and the corresponding data available at each layer. Such data will be used for identifying devices and their network behaviour. In Section 2.3 (Machine Learning), the Machine Learning (ML) algorithms used in both studies are listed, highlighting their working principles and some of their features and properties, including their data requirements and situations in which they are more or less effective. In Section 2.4 (Evaluation), evaluation methods that allow us to determine the level of success from the results of the assessment studies are introduced and the advantages and disadvantages of these methods are emphasised.

2.2 Network Models

Familiarity with network concepts in both device identification and attack detection is essential for a better understanding of the work being done. The main reason for this is that the machine learning features used for both processes are derived from network communication data. Network communication is mediated by protocols that specify the rules by which entities communicate with each other. To make the network structure easier to understand, network models are divided into layers. This allows the functions of each layer and its relationships to other layers to be described in a more modular structure. The layered system allows developers and researchers to focus on a specific

area by separating each layer according to the task it performs, allowing people with different areas of expertise to make maximum contributions to the protocols [63]. In these layered models, as you move up from the lowest layer, which represents the physical connection between entities, you reach the upper layers, which are less concerned with the dynamics of network traffic and more focused on the details of the end user's applications [35]. Each of these layers provides a hierarchical classification of the protocols it contains. Open Systems Interconnection (OSI) (seven layers), Transmission Control Protocol (TCP)/Internet Protocol (IP) (Internet model - four layers) and hybrid model are the most widely used network models [221]. Figure 2.1 shows the structure and protocol relationships of these three models. During data transmission, the data generated by the user application at the top layer is sent down to the lower layers. At each layer it passes through, it is encapsulated by adding the header of a protocol at that layer. You can think of it as boxes that are put into a different box at each step. When it reaches the receiver, the data is transmitted to the upper layer by performing the appropriate decapsulation process starting from the lowest layer to top layer. This process is visualised in Figure 2.2.

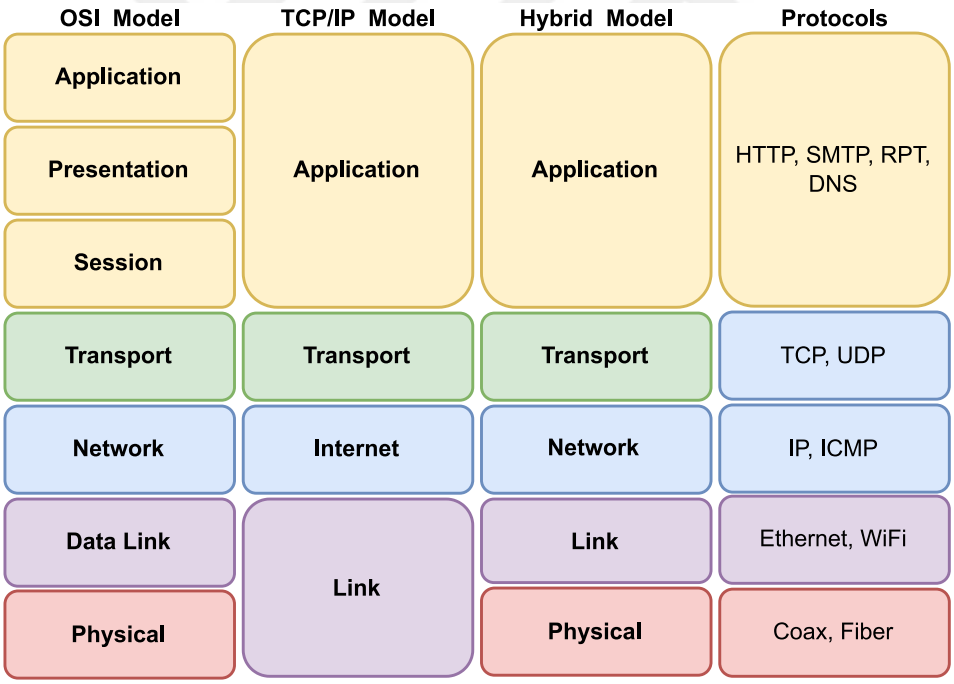


Figure 2.1: OSI and TCP/IP network models. Some protocols that these models are related to according to their layers.

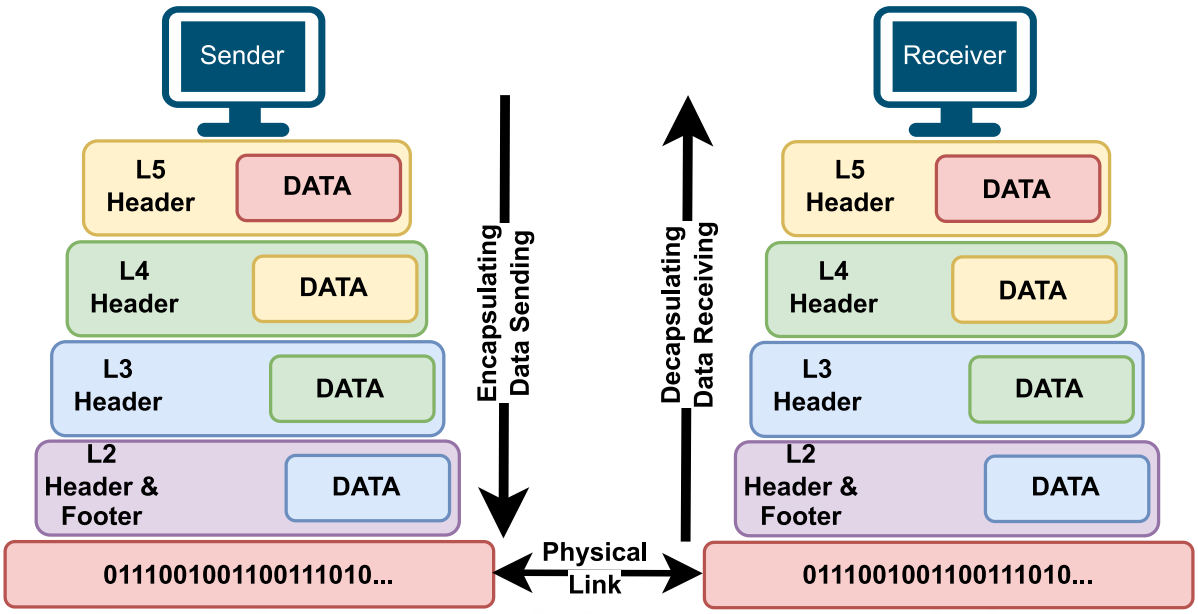


Figure 2.2: Data is encapsulated by the appropriate protocol at each layer as it is sent and decapsulated at each layer at the source.

We will continue using the hybrid model approach since it is sufficiently broad but not too detailed. The layers of this model and some of the protocols in the layers are described below. Most importantly, we highlight the relevant data contained at each layer.

2.2.1 Physical Layer

The first and lowest layer of the network reference model is the physical layer, which is responsible for transmitting the raw data to the communication channel [221]. This layer is responsible for ensuring that the data in bits is sent to the other side smoothly [221]. Unlike the other layers, this layer also physically moves the data. Data that is processed by various protocols at upper layers is physically transmitted (raw bits) to the destination at this layer. Classic hubs and repeaters that use only raw data also operate at this layer. However, this layer does not only consist of physical objects (such as the network interface or cables) but also describes the network operations [125]. It performs a number of tasks such as adapting to the network hardware, converting bits into signals and signals into bits, maintaining the desired quality and speed of the connection, and error correction [125]. There is no specific data at this layer that would be used for describing network behaviour.

2.2.2 Link Layer

The second layer of the reference model, the link layer, is responsible for ensuring the interoperability of different network technologies. In this context, it handles wired and wireless Local Area Network (LAN) management. This layer performs basic tasks such as data framing, addressing, error detection and error correction, and defining physical layer standards. This layer has two sub-layers, Logical Link Control (LLC), which establishes and controls the logical connections between devices, and Media Access Control (MAC), which controls access to the network environment. The protocols that work in relation to this layer can be listed as follows: Ethernet, Wireless Fidelity (WiFi), Token Ring, Asynchronous Transfer Mode (ATM), Serial Line Internet Protocol (SLIP), Fiber Distributed Data Interface (FDDI), and Point-to-Point Protocol (PPP). Hardware associated with this layer includes network interface cards, bridges, switches, and brouters [125]. An important concept related to this layer is the MAC address.

The MAC address is a six-byte (12 hexadecimal digits) unique identification number that uniquely identifies the network interface of a device. The first 3 bytes of this number are assigned by the Institute of Electrical and Electronics Engineers (IEEE) to identify the manufacturer. The last three bytes are assigned by the manufacturer to identify the device network interface [35] (see Figure 2.3). In device type identification studies, this unique number is commonly used as a device label.

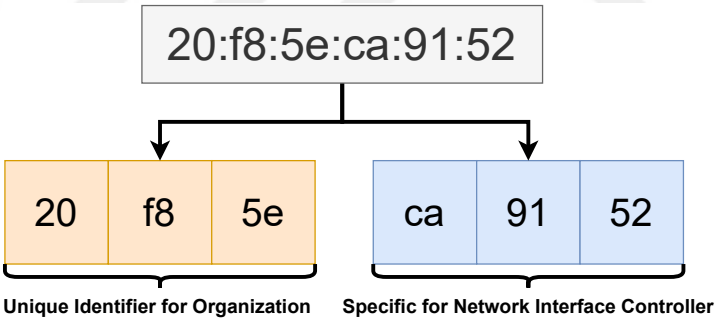


Figure 2.3: The MAC address of Fitbit Aria WiFi-enabled scale from Aalto dataset.

2.2.3 Network Layer

The third layer of the reference model is the Network layer. This layer is responsible for the transmission of data between computers on different networks and for the management of internetworks. At this layer, datagrams are encapsulated, and logical addressing and routing are performed. Fragmentation and reassembly of the transmitted data, de-

tection and correction of errors are also under the responsibility of this layer [125]. In logical addressing, which is the most basic task of this layer, data is assigned a unique address such as an IP address. An IP address is a number that uniquely identifies a device, such as an IoT device, on the local network or the Internet [35]. There are 2 versions IPv4/32-bit and IPv6/128-bit, but IPv4 is currently the dominant one used on the Internet [35]. With this address, data travels between networks to its destination. Figure 2.4 illustrates the IPv4t header and Table 2.1 describes the fields of this header.

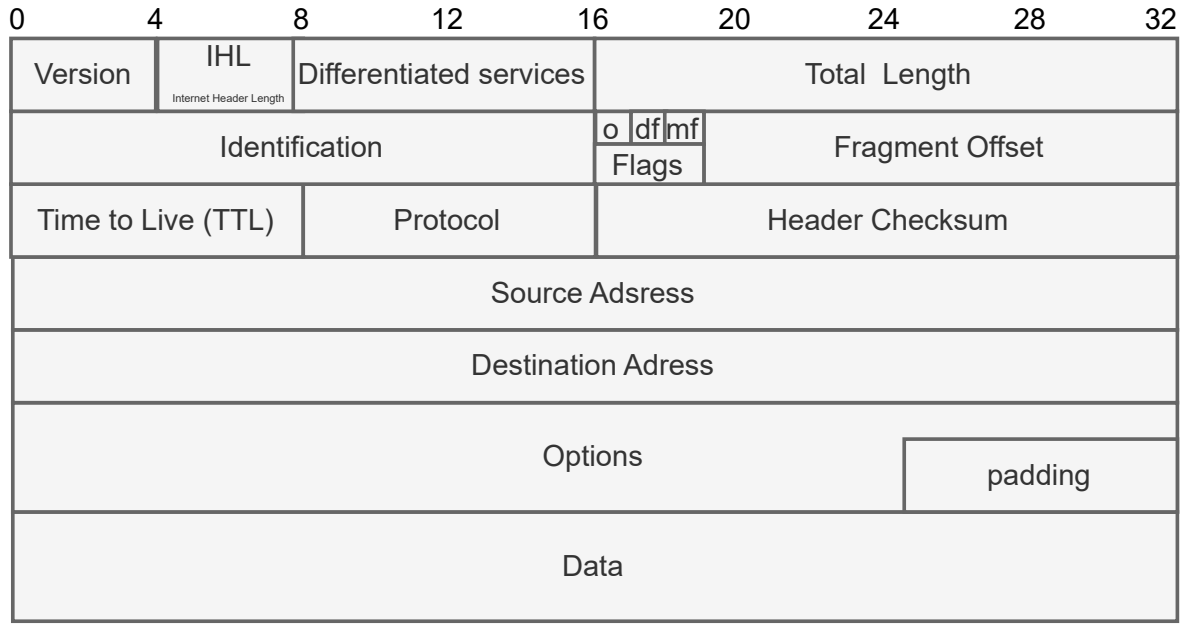


Figure 2.4: Structure and parts of the IPv4 header. Adapted from [63, 221].

2.2.4 Transport Layer

The fourth layer of the reference model is the Transport layer. It acts as a bridge between the software orientated application layer above it and the data transmission orientated layers below it [125]. This layer is responsible for splitting the data sent from the application into pieces for transmission and assembling them at the destination. In connection-oriented protocols, it is also responsible for activities such as initiation, maintenance and termination in managing the connection. In addition, the flow control mechanism that determines the data rate also works at this layer [125]. The two important end-to-end protocols at this layer are described in detail.

TCP is a reliable connection-oriented protocol that ensures error-free transmission of data from one machine to another machine. The network layer fragments the data to

Table 2.1: Fields of the IP header and their descriptions [63, 221].

Field Name	Description
Version	Version number. For IPv4 it is always 4.
Internet Header length	Header size.
Differentiated services	Contains information such as packet priority, reliability, speed, etc.
Total length	The total size of the IP packet.
Identification	The number given to reassemble fragmented data.
Flags	Indicate the fragmentation status of the packet.
Fragmented offset	contains sequence information for fragmented packets.
Time to live	Indicates the packet's hop lifetime. It decreases by 1 each time it passes through a router.
Protocol	Indicates a higher layer protocol (TCP, User Datagram Protocol (UDP) or Internet Control Message Protocol (ICMP)).
Checksum	Used to verify if there are errors in the packet.
Source IP address	Contains the source address.
Destination IP address	Contains the destination address.
Options	Contains information such as security, source, routing, etc. to be used when necessary.
Padding	Completes the package size in multiples of 32 bits.
Data	Transmitted data.

transmit it to the other side and combines it at the other side. By ensuring that this process is error-free, it corrects lost or incorrect data. As long as there is no confirmation of delivery from the destination, it continues to send the same data. It also provides flow control by determining the speed of the transmitter according to the capacity of the receiver [221]. To ensure connectivity between the sender and receiver, the TCP protocol establishes a connection between them called a 3-way handshake. Data transmission starts after this connection is established. During the 3-way handshake process, the client that wants to establish a connection sends a Synchronize (SYN) packet to the server. The server receiving this packet sends a SYN and Acknowledgement (ACK) packet to the client. Finally,the client responds with the ACK packet, and the TCP connection is established. As long as this connection lasts, an acknowledgement packet is sent each time a packet is received. The TCP session is typically terminated by sending a Finish (FIN) packet. Figure 2.5 shows the TCP header. The tasks of the areas illustrated in this header are listed in Table 2.2.

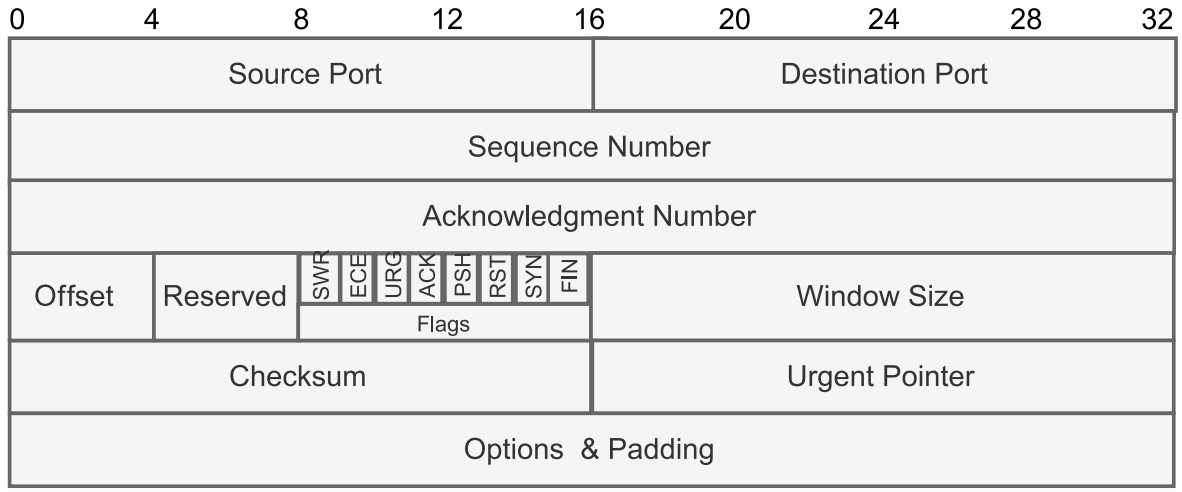


Figure 2.5: Structure and parts of the TCP header. Adapted from [63].

UDP is another important protocol in the transport layer. As an unreliable and connectionless protocol, UDP does not perform flow control or queuing, making it ideal for situations where speed is more important than accurate transmission, such as one-off server client transactions or simultaneous video audio transmission. Applications using UDP either do not need queuing and flow control or do it themselves on the upper layer [221]. The header of UDP, which is much simpler than the TCP header, is visualised in Figure 2.6.

Table 2.2: Fields of the TCP header and their descriptions [63].

Field Name	Description
Source & Destination port	These fields, consisting of integer values from 0 to 65536, are used in combination with the IP address to uniquely identify each connection.
Sequence Number	A number between 0 and $2^{32} - 1$ is given to represent each segment in the segmented data.
Acknowledgement Number	It is sent in response to the received SYN packet, it contains the next SYN number it expects to receive.
Offset	The length of the header.
Reserved	It is a 4-bit field reserved for future use.
Checksum	Used to verify if there are errors in the packet.
Window Size	Provides flow control, allowing the receiver to receive as much data as it can handle
Flags	Indicates the control bits associated with the packet. Each of the 8 bits indicates whether a flag is set or not. These flags are: Congestion Window Reduced (CWR), Echo (ECE), Urgent (URG), FIN, Push (PSH), Reset (RST), and FIN.
Urgent Pointer	This field is used to indicate that the transmission of data is urgent. It works in parallel with the URG flag.
Option & Padding	Specifies the maximum segment size that can be received by the sender in the reverse direction.

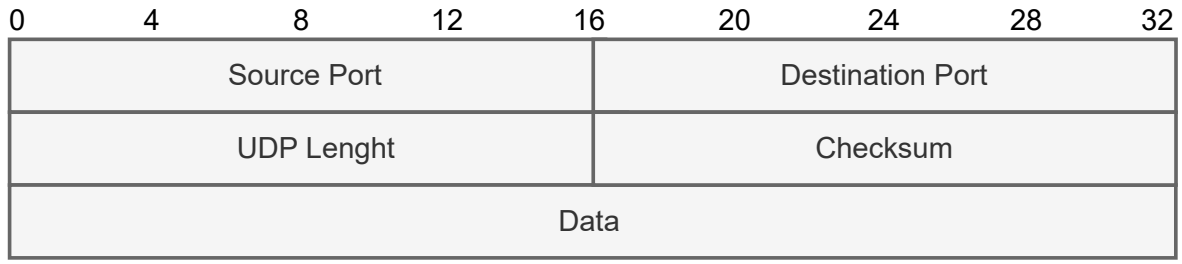


Figure 2.6: Structure and parts of the UDP header. Adapted from [63].

2.2.5 Application Layer

At the top of the reference model is the application layer (the fifth layer). This layer includes the presentation and session layers of the OSI model. There are many high-level protocols in this layer, such as Hypertext Transfer Protocol (HTTP)/Hypertext Transfer Protocol Secure (HTTPS) used to request a page from the World Wide Web (WWW), Domain Name System (DNS) for mapping domain names and network addresses, File Transfer Protocol (FTP), and Simple Mail Transfer Protocol (SMTP). This list is expanding as new protocols are added to these protocols over time [221]. There are many different protocols at this layer, and their specific headers can be used for device identification or attack detection. For example, it is common to use features from the DNS protocol for device identification [126, 129, 226] and features from the MQ Telemetry Transport (MQTT) protocol for attack detection [68, 88]. Due to the extensive range of such protocols and their divergence from our study's scope, detailed elaboration is withheld.

2.3 Machine Learning

Machine Learning is the adaptation of a computer program or model to improve its performance in a new task, taking advantage of previously acquired data or experience [74]. In this way, a learning algorithm can produce a solution to a new problem that it is not specifically programmed to solve with minimal human intervention. We can divide machine learning algorithms into 3 different groups depending on whether their data is labelled or not.

Supervised learning: In this type of learning, the training data used by the learning algorithm are labelled. The algorithm performs the learning process using the data labels

and classifies the test data provided to it [74]. The success of this method is typically high, but it is costly because it requires a pre-labelled dataset. This method includes Naïve Bayes (NB), Support Vector Machine (SVM), k-Nearest Neighbours (KNN), Decision Tree (DT), Random Forest (RF), MultiLayer Perceptron (MLP), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) etc.

Unsupervised learning: In this method using unlabelled data, the machine learning algorithm divides the given data into different groups according to their characteristics. Because it does not require pre-labelled datasets, it typically costs less than supervised learning [74]. In addition to the cost of labelling, there are situations where data labelling may not be feasible, and in such cases, unsupervised learning is employed to uncover inherent patterns in the data. However, the performance of unsupervised methods is typically lower because they lack the training signal benefits provided by labelled data. Examples of this method are k-Means, AutoEncoder (AE) and Principal Component Analysis (PCA).

Semi-supervised learning [92]: This method is the hybrid of supervised and unsupervised learning. The aim is to take the advantages (high performance for supervised learning, low cost for unsupervised learning) of both methods and avoid disadvantages (high cost for supervised learning, low performance for unsupervised learning). To ensure this, very little of the data is labelled. Thus, it aims to obtain an optimum profit/cost ratio.

2.3.1 Basic concepts

In this section, we introduce some basic concepts of machine learning. These concepts are aimed at facilitating a better understanding of algorithms, pipelines, and results.

2.3.1.1 Training, Validation, and Test Set

When using ML algorithms in any task, the data is divided into two or three parts (Training, validation and testing) [72]. **Training**, the largest part, is used as input to the ML algorithm to train the model, that is, to develop the rules of the model. **Validation** uses a portion of the data in the ML algorithm for operations such as feature and parameter selection of the model trained with the training data input. Validation data is specialized for the optimization of the model. **Test** data is the data used to

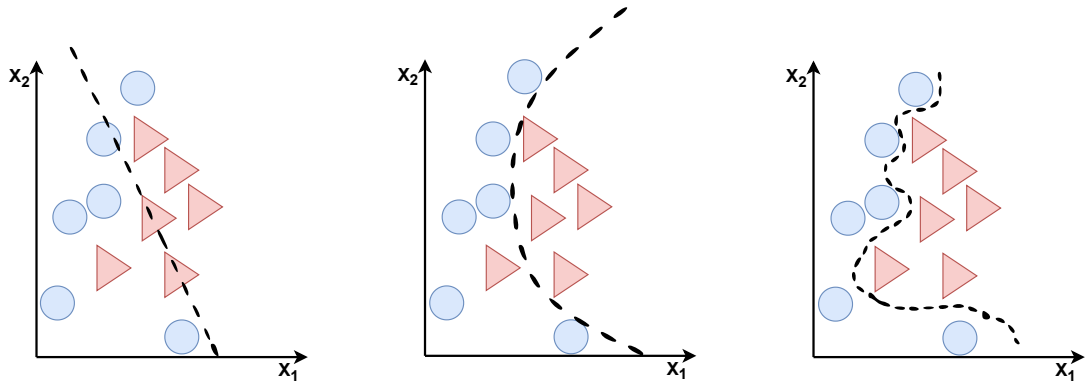
evaluate the performance of the model trained using the training data. This data needs to be isolated from the data used in training and optimization. By using isolated data, the success of the model can be measured more realistically on data that has not been encountered before.

2.3.1.2 Overfitting and Underfitting

Overfitting is a common problem in machine learning. It is characterised by the inability of the model to transfer its over-performance on the training data to another dataset. When this happens, the model has no real predictive power, it merely memorises the training data. The issue of overfitting arises when unwanted and irrelevant information, known as noise, disrupts the learning process. The reason is that the model focuses more on noise than on the actual features. However, overfitting can also occur due to information leakage. The overfitting problem usually occurs when the model complexity is allowed to increase excessively [72, 185].

A similar problem is underfitting. In this case, unlike overfitting, the model is too simple. Since the model is not complex enough to capture the pattern found in the training data well, it performs also very poorly on unseen data [185].

An ideal model should be free from both overfitting and underfitting. This is possible by balancing between bias and variance. This is because models suffering from overfitting are too complex and have high variance, whereas models suffering from underfitting are too simple and have high bias [185]. Ideal, overfitted and underfitted models are visualised in Figure 2.7.



(a) Underfitting/High bias (b) Good compromise (c) Overfitting/High variance

Figure 2.7: Underfit (a), ideal (b), and overfit (c) model. Adapted from [185].

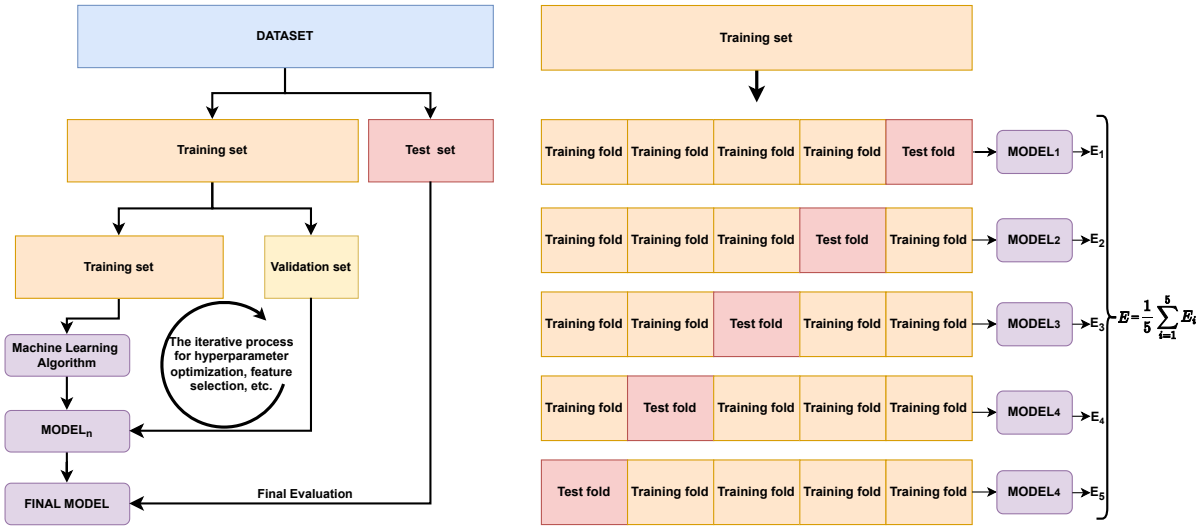
2.3.1.3 Cross-validation

It is crucial to evaluate the machine learning model's performance on unseen data to avoid overfitting or underfitting issues. To address this, the training data should not be used for validation or testing [185]. Two popular methods for splitting data into training and testing are as follows:

Holdout method [185]: The data is initially divided into training and test sets. Then, the training data is further split into training and validation sets. The model is fitted using the training data and optimized using the validation set. The test data remains untouched throughout the process and is only used to evaluate the final optimized model. This process is visualised in Figure 2.8a.

k-fold cross-validation [185]: The training data is randomly split into k equal subsets, often referred to as 'folds'. In each iteration of k-fold cross-validation, one of these k subsets is designated as the test set, while the remaining k-1 subsets are used for training the model. This process is repeated k times, with each subset taking a turn as the test set. Finally, the average performance across all k iterations is used to estimate the model's overall performance. This process is illustrated in Figure 2.8b.

Both methods, cross-validation and holdout methods, serve the crucial purpose of reliably estimating the model's generalization performance. Cross-validation is particularly valuable when dealing with limited data, ensuring robust performance evaluation while avoiding the pitfalls of relying solely on the training data for testing and validation.



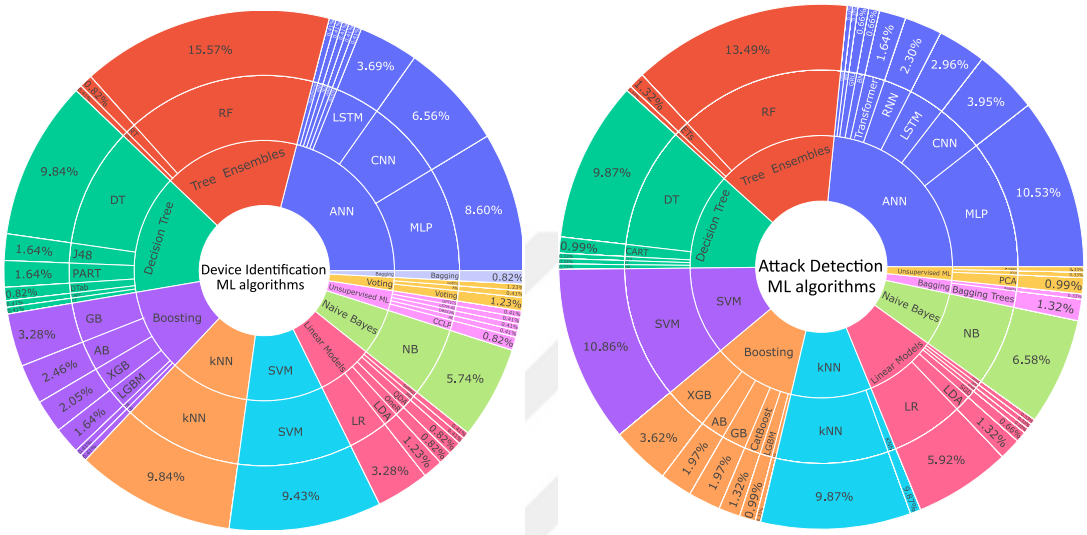
(a) Data split with holdout method

(b) Data split with 5-fold cross-validation

Figure 2.8: Segmentation processes of data as training validation and testing for machine learning. Adapted from [185].

2.3.2 ML Algorithms

Although there are many other learning algorithms in theory and use, we will focus on the supervised algorithms such as NB, SVM, KNN, DT, RF, MLP, Logistic (Logit) Regression (LR), Gradient Boosting (GB) that are used extensively in IoT security [9,92]. See Figure 2.9 for the frequency of use of these algorithms in IoT security. Figures 2.9a and 2.9b give an overview of algorithm usage within the DI and ML literature (and will be considered in more detail in Chapters 3 and 7 when this literature is reviewed).



(a) DI ML map (b) AD ML map

Figure 2.9: Distribution of ML algorithms used in (a) device identification, and (b) attack detection (b) based on our literature review (see Table 3.1 and Table 7.1).

2.3.2.1 Naïve Bayes

NB is a classification algorithm used in supervised learning. Basically, it is based on the conditional probability principle. It creates probability networks that predict labels with the features in the dataset. However, when doing this, it considers each feature it uses to be independent from each other [9]. For example, there is a clear relationship between the TCP and SYN, ACK, FIN flags: All packets containing these flags are TCP packets. If NB is used in an algorithm that classifies network flow by using the protocol type (UDP and TCP) and flags, it ignores this relationship between the TCP and flags. It evaluates each of these features (every flag and protocol type) as an independent feature from each other [9].

NB is a very simple, easy-to-apply method with low calculation costs. Also, it does not need optimization since it has no free parameters. Although it is ideal for small datasets that are not too complicated, it generally does not achieve high performance in large and multi-dimensional datasets [212] [9].

2.3.2.2 Support Vector Machine

SVM is a versatile supervised learning algorithm that aims to optimally distinguish two or more-dimensional data from each other [74]. The basic logic of this algorithm is to create a decision hyperplane with maximum separation, where the two classes are closest to each other. A prominent feature of SVM is that it can also separate nonlinear data. SVM utilizes the kernel method to create this separator surface in nonlinear data. Using this method, SVM transfers this data to a higher dimensional space to classify data that cannot be separated linearly. Thus, in this new multi-dimensional space, a surface that will provide linear separation can be created [185]. In Figure 2.10, data that cannot be separated as 2-dimensional linear has been transferred to a new 3-dimensional area and made linearly separable.

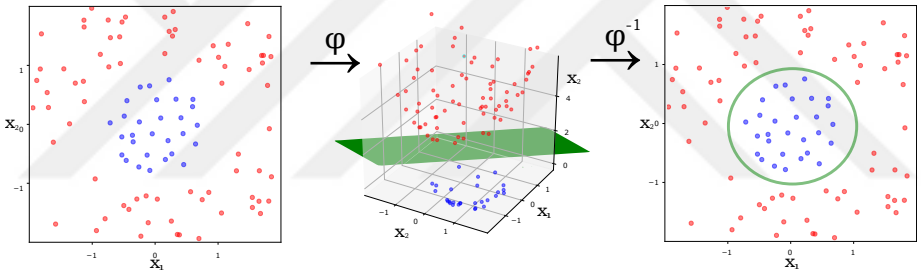


Figure 2.10: Representation of the SVM kernel method. Adapted from [185].

This algorithm stands out for its ability to generalize, and it often promises higher accuracy than alternatives and less overfitting. Another advantage is that its performance is independent of the dataset size and it can handle multi-dimensional data. On the other hand, it is a complex and rather slow algorithm, and the performance of the algorithm is subject to sufficient hyperparameter optimisations [212] [9].

2.3.2.3 k-Nearest Neighbours

KNN is a supervised machine learning algorithm. In KNN algorithm, the distance between an unknown/unlabelled sample and labelled/known samples is measured. As a result of this measurement, the unlabelled data is labelled with the label of the nearest

class. The k value determines how many neighbours to look at. The group, which constitutes the majority among the number of neighbours that are looked after, decides what the label of the unlabelled sample will be [212] [9].

For example, there are examples representing good (green) and bad (red) network traffic in Figure 2.11, and the yellow circle represents unlabelled data. In this example unlabelled data will be labelled as bad for k value 1, bad for k = 3 and good for k = 10. As can be understood from the example, the appropriate k value varies according to the distribution of the data. This algorithm is suitable for multi-class datasets and it has a simple structure. However, optimization is required to find the appropriate value of k, which is a costly calculation. It is also sensitive to noise and the algorithm's performance is affected by the size of the data, as it has to calculate the distance for each data [212] [9].

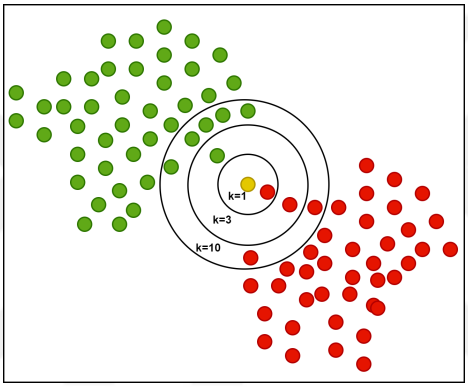


Figure 2.11: Representation of the behaviour of the KNN algorithm when the k number takes different values (1,3,10).

2.3.2.4 Decision Tree

DT is a learning algorithm that provides classification using a tree-structured decision process. Starting from the root node, the transition to the next node or leaf through the branches is achieved. Each sub-node consists of a condition containing a property and associated values. The leaf refers to the classification result. DT is divided into subtypes such as Iterative Dichotomiser 3 (ID3), C4.5, C5.O and Classification And Regression Trees (CART) according to the criteria used such as Gain Rate and Knowledge Gain, Gini Coefficient [212]. Figure 2.12 shows the decision tree depicting the classification of an Internet of Things (IoT) device. This binary classification model determines whether a given sample belongs to the *Fitbit Aria WiFi-enabled scale* device or not.

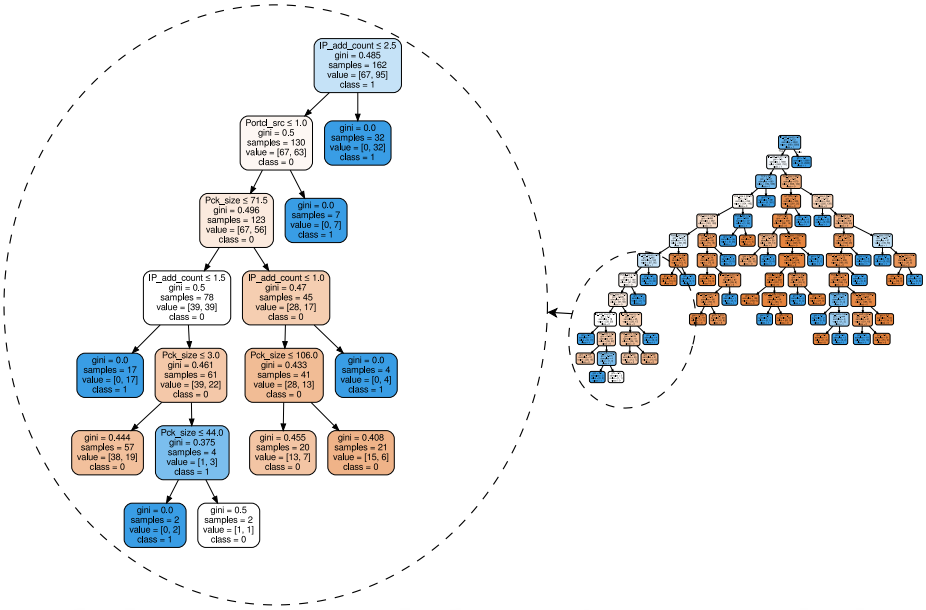


Figure 2.12: Binary classification of an IoT device using the DT algorithm and network statistics.

DTs have low complexity and high explainability. The calculation cost to performance ratio is low, so it is seen as an efficient algorithm. It is also an algorithm that is successful in generalizing and is resistant to noise. However, as the size of the data increases, it becomes difficult to process and if the pruning (removing unnecessary branches) is not done properly, it may tend to overfit. Another disadvantage is that decision trees are sensitive to rotations in the training data, as they only form boundaries parallel to the axis [74]. To give an example, the figures below show the first version (Figure 2.13-a) of the sample data and the second version (Figure 2.13-b), rotated 45 degrees. In the first, the decision tree is able to create a general and simple rule for separating the data, while in the second picture, a less generalized and complex rule must be created because of the perpendicular boundary lines [74].

2.3.2.5 Random Forest

RF is a supervised ML algorithm. The basic idea in this method is to create a model that produces more generalizable performance and less overfitting by bringing together a lot of decision trees, each individually suffering from weakness and high variance [185]. Therefore, this algorithm consists of many randomly generated decision trees. The final decision is made according to the results that collect the most votes from the decision trees [212] [9].

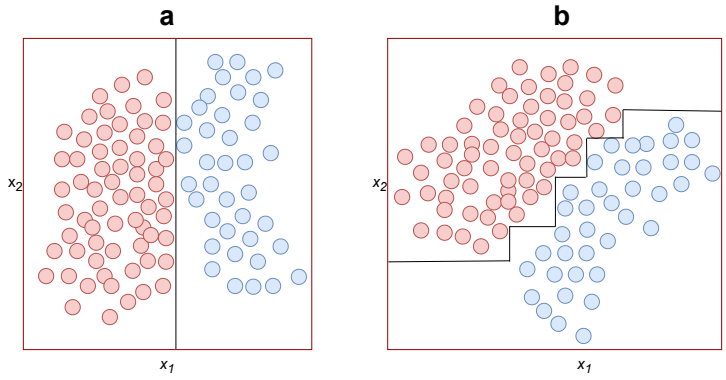


Figure 2.13: Representation of the change in the decision tree after the data is rotated. Adapted from [74].

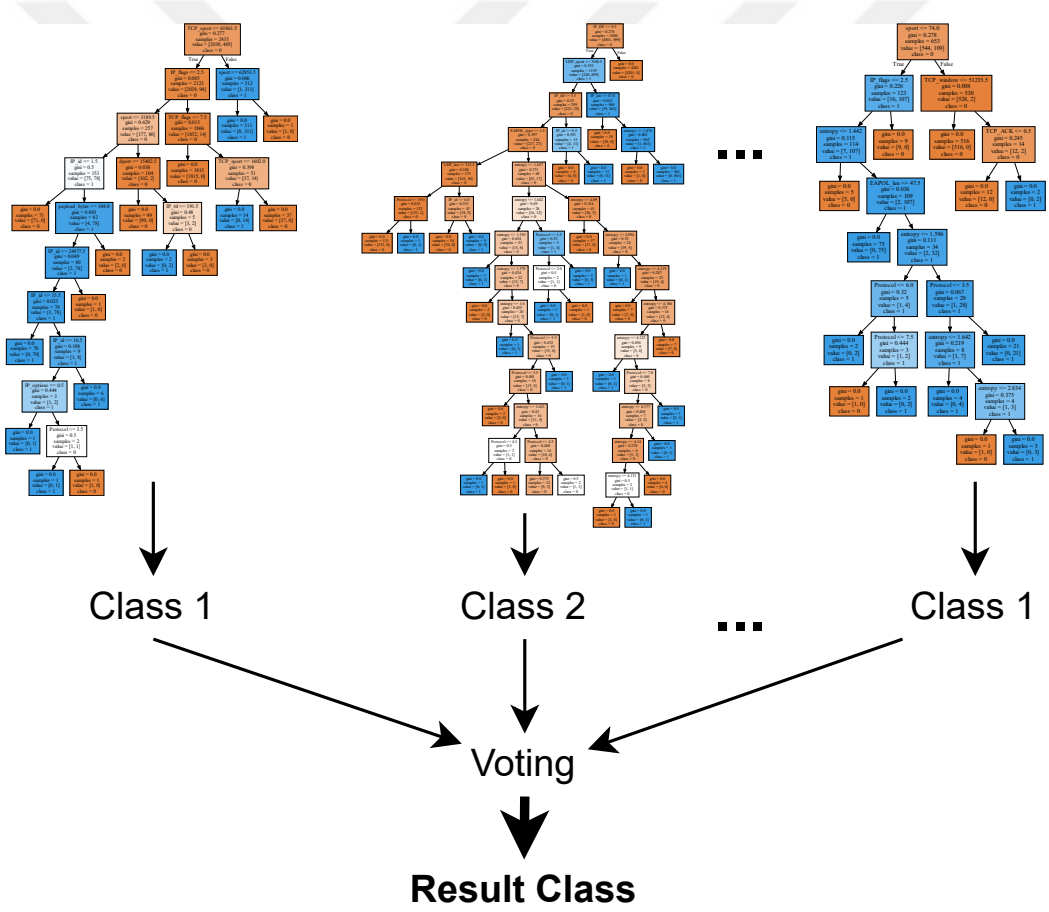


Figure 2.14: Binary classification of an IoT device using the RF algorithm and network statistics.

Random forests inherit many good features of decision trees, such as success in generalizing and noise resistance. In addition, it is strong against the overfitting problem that DTs suffer from. On the other hand, since it is necessary to create multiple decision trees during this process, they can be inefficient/slow in real-time applications when dealing with large datasets [212] [9]. In addition, RF leads to a complex and difficult-to-understand model. Figure 2.14 shows the random forest depicting the classification of an IoT device.

2.3.2.6 Gradient Boosting

Boosting (hypothesis boosting) is another ensemble method that aims to build a stronger model by combining several weak models. GB is a widely used boosting algorithm [74]. It is based on the principle that at each step a new model is added to successive weak models (e.g., a decision tree) and the new model corrects the errors in the previous model. In this way, a model with a large domain is constructed that compensates for the weaknesses of its component models [185]. The working principle of this method is illustrated in Figure 2.15.

In Figure 2.15, ① the initial learner creates a coarse discrimination that does not perfectly divide the sample. Some samples at the bottom left of the graph are misclassified. In the following example ②, the learner focuses on the errors made by the previous model and produces a new model that correctly classifies those misclassified samples (the bold squares). However, this model misclassifies three examples at the top of the graph. In ③, by focusing on the errors of the previous model, the learner develops a refined model that correctly classifies those three examples. Finally, in ④, the combination of the three refined models yields a model that achieves perfect classification on all examples. In addition to GB, other popular boosting algorithms include Adaboost, Adaptive Boosting (AB), Catboost, Category Boosting (CB), LightGBM,Light Gradient-Boosting Machine (LGBM), and XGBoost, eXtreme Gradient Boosting (XGB). XGB, an optimized Python implementation of gradient boosting, has gained significant popularity in recent years [74].

2.3.2.7 Logistic Regression

LR is a machine learning method based on the principle of calculating the probability that a sample belongs to a certain class (e.g., whether a network packet is benign or not). It provides a classification solution by assigning a value of 1 if the probability for this class is greater than 50% and 0 otherwise [74]. The logistic function (logistic

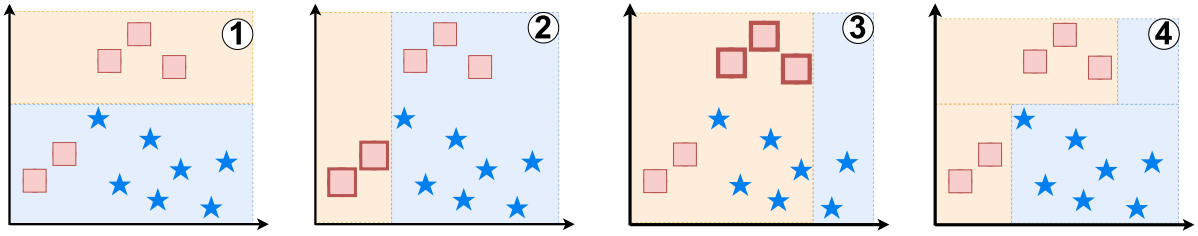
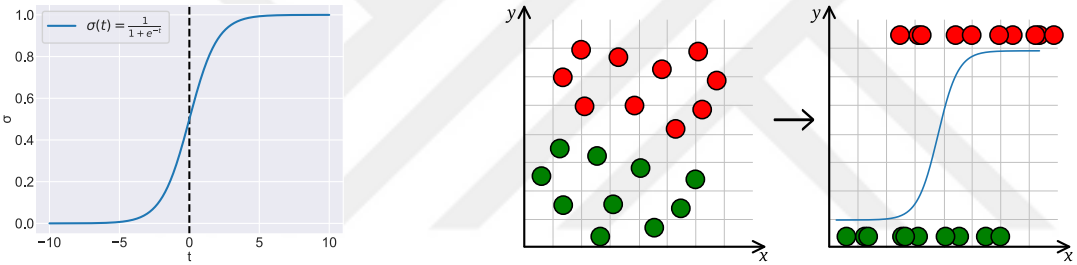


Figure 2.15: The working principle of GB. Adapted from [185].

curve) used for this process is shown in Figure 2.16a. The use of LR in classification is visualised in Figure 2.16b.

It is a robust algorithm suitable for both linear and binary classification tasks, and it can also be applied in scenarios where linear separation of classes is not feasible. Additionally, it can be extended to tackle multi-class classification problems. Due to its simplicity of implementation, high interpretability of its results and high performance for linearly separable classes, it is a very popular algorithm and is widely used in industry [185].



(a) The logistic function

(b) The usage of LR for classification

Figure 2.16: The function used by LR for classification and how this function works.

2.3.2.8 Artificial Neural Networks and Multilayer Perceptrons

Before we talk about the following learning algorithms, we will briefly explain the concepts of artificial neural networks and deep learning. Inspired by biological nerve cells, Artificial Neural Network (ANN)s are formed by adding layers of artificial nerve cells/nodes together, one after the other [101]. The working principle of the artificial neuron, which you can see in Figure 2.17, can be summarized as follows [185]: It creates a total value by multiplying the values coming to it from all inputs (x_1, x_2, ..., x_m) connected to the artificial neuron by its weight values (w_1, w_2, ..., w_m). Then, by adding a bias value (b) on the total value and it is converted to a value in a certain range (for example

in the range 0-1) through the activation and unit step function. The result of these operations is given as output ($\hat{y}$).

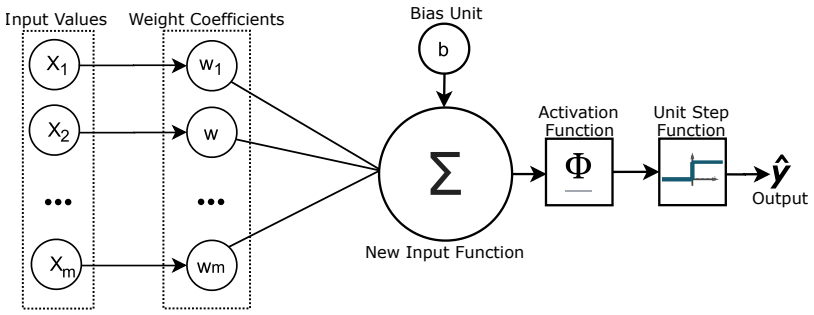


Figure 2.17: Representation of an artificial neuron. Adapted from [185].

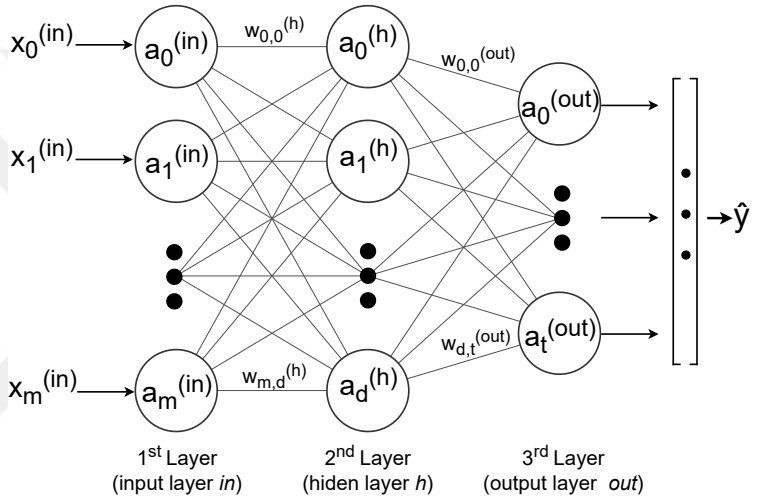


Figure 2.18: The working principle of MLP. Adapted from [185].

One of the most widely used types of ANN is MLP. MLP consists of three steps. The first layer of MLP receives the data, the middle layers process the data, and the last layer classifies the processed data and outputs it. These layers are called input, hidden and output layers (MLP visualised in Figure 2.8b). The number of hidden layers can be increased, and architectures with many layers in the hidden layer are called "deep". Learning in MLP takes place in two stages. In the first one, the value of each cell is transferred to all cells in the next layer under the influence of a weight value and adding a bias value. At the end of this process, the results obtained are compared with the desired results. In the second stage, the weight values that affect each cell are adjusted backwards to minimise errors [101]. Although their training steps are time-consuming, complex and difficult to understand, ANNs and especially deep learning approaches are

often superior for non-linear and complex problems [212] and this currently makes them popular in the world of machine learning.

2.3.3 Genetic Algorithms

Genetic Algorithms are not considered classical machine learning algorithms, but they can be used as part of the ML modelling process, including feature selection. Genetic Algorithm (GA) is an optimisation algorithm inspired by natural selection based on the survival of the fittest principle [55]. The basic elements of the GA can be summarized as follows [55]:

The algorithm starts with a group of individuals known as the population. Each individual's genetic makeup is represented by a chromosome containing genes, which can be composed of bits, numbers, or characters. The selection of individuals for reproduction is based on their fitness value, where those with higher fitness have a greater chance of being chosen. The generation of a new population is primarily facilitated by crossover and mutation operations. Crossover enhances the exploration of the search space during the early stages of population evolution, while mutation is responsible for reintroducing lost information through local or global adjustments in the search space. This iterative process continues until specific stopping criteria are satisfied or an optimal solution is achieved.

Feature selection is crucial in machine learning as when they work together, some features can enhance performance while others can hinder it. However, exhaustively exploring all possible feature subsets using the wrapper method, which involves iteratively evaluating different subsets and selecting the best-performing one, through trial and error is time-consuming and impractical due to the exponential number of combinations. For instance, a dataset with 100 elements would yield 2^{100} possibilities. To address this challenge, when creating our models for device identification (Part I) and attack detection (Part II), we employed a GA to efficiently generate feature subsets within a reasonable time frame.

2.4 Evaluation

Evaluation measures are essential for assessing the model's performance by comparing predicted values with actual values. For example, in the context of binary attack detection, there are four possible scenarios in a sample containing Benign and Attack data:

True Positive (TP): The sample is correctly predicted as an attack.

True Negative (TN): The sample is correctly predicted as non-attack.

False Positive (FP): The sample is incorrectly predicted as an attack when it is actually a non-attack.

False Negative (FN): The sample is incorrectly predicted as non-attack when it is actually an attack.

These situations are represented in the confusion matrix, which displays the distribution of these scenarios. While the confusion matrix can also be shown with reversed rows and columns, we utilize the scikit-learn[1] version for consistency.

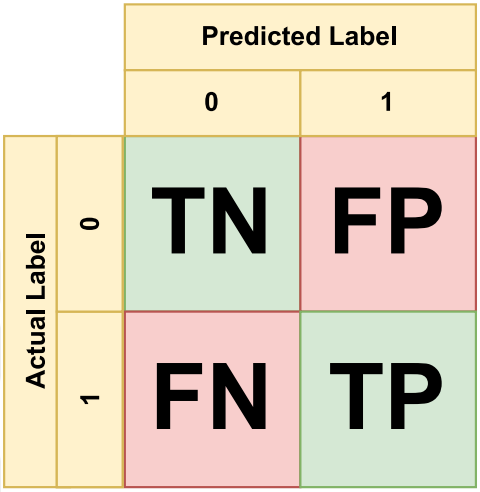


Figure 2.19: Representative visualization of the confusion matrix.

Among the evaluation metrics, the most commonly used are accuracy, recall, precision, and F1 score.

2.4.1 Accuracy

Accuracy is found by dividing all correct answers (TP+TN) by all answers (TP+TN+FP+FN). It is the most widely used evaluation method. Although it is quite common, it is not a valid evaluation method in datasets that suffer from unbalanced distribution (both device identification data and attack detection data generally have a very unbalanced distribution). We can explain with an example: In a dataset of 100 samples with a total of 95 benign and 5 attack cases, an erroneous model that labelled every sample

[1] https://scikit-learn.org/stable/

as benign would have an accuracy score of 0.95, which is highly misleading.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.1}$$

Another problem with Accuracy is the inability to obtain class-based scores in multi-class situations. It is used for only overall evaluation. Despite these disadvantages, accuracy has become the standard method of evaluation due to its wide use in classification problems in machine learning. We also have used the accuracy score, along with others, in our studies because it provides an opportunity for comparison to previous work.

2.4.2 Recall

Recall, also known as detection rate, sensitivity, or true positive rate, measures the proportion of correctly predicted positive samples to all positive samples (e.g., actual attack). In the case of predicting attacks, it represents the ratio of true positives (TP) to the total number of attacks (TP + FN). It is commonly used in multi-class situations and can provide class-based results. Balanced accuracy, the average of recalls in multi-class classification, serves as an alternative to accuracy in addressing unbalanced datasets.

$$Recall = \frac{TP}{TP + FN} \tag{2.2}$$

$$Balanced\ Accuracy = \frac{1}{k} \sum_{n=1}^{k} Recall\ (where\ k = number\ of\ classes) \tag{2.3}$$

2.4.3 Precision

Precision, also known as positive predictive value, is the proportion of correctly predicted positive samples in the model's positive predictions. In the case of predicting attacks, it is the ratio of correctly predicted attacks (TP) to all values predicted as attacks (TP + FP).

$$Precision = \frac{TP}{TP + FP} \tag{2.4}$$

2.4.4 F1 Score

F1 Score (F1) is the harmonic average of precision and recall. It shows more realistic results by indicating how much the balance between recall and precision is maintained.

If we go back to the unbalanced data example we discussed in Section 2.4.1 for Accuracy, the F1 score of the results in the same situation is 0.49.

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN} \tag{2.5}$$

We will use F1 as the main evaluation criterion in our studies due to its advantages such as being suitable for unbalanced data, giving class-based results and being common in the literature related to our research. In the multiclass cases, we used a macro F1 score, which is an unweighted average of the class-based F1 scores, to deal with imbalances in the number of samples in different classes.

2.4.5 Cohen's Kappa Coefficient

Additionally, the Cohen's kappa coefficient (kappa) score is used to assess the agreement between the model's predictions and the actual values. kappa is rarely used in the IoT security literature but is particularly useful in situations where measuring the percentage agreement between the true value and the model prediction can be misleading. This is because the percentage of agreement can be affected by the class distribution. Unlike other evaluation measures, the kappa score ranges from -1 to +1. With a kappa score of 1, which is the highest score, the agreement that can be obtained by chance is close to 0. A value below 0 indicates less than the success that can be achieved by chance. E.g., the kappa value of the above example for Accuracy (Section 2.4.1) and F1 (Section 2.4.4) is 0. Although rarely used in the literature, we found it useful to include the kappa score in the evaluation sections as it indicates random success based on chance.

$$\kappa \equiv \frac{p_o - p_e}{1 - p_e} = 1 - \frac{1 - p_o}{1 - p_e} \ (where\ p_o = observed, p_e = chance\ probabilities) \tag{2.6}$$

2.5 Conclusions

This Chapter delved into fundamental concepts shared between IoT device identification and IoT attack detection. These overarching concepts include network models explored in Section 2.2, encompassing network layers and associated data used for device identification and network behaviour analysis. Section 2.3, highlighted the machine learning algorithms utilised in both studies, outlining their operational principles, features, and properties, including their varying efficacy based on data requirements and scenarios.

Evaluation methods were introduced in Section 2.4, providing insight into assessing the outcomes of assessment studies. The rest of the thesis is divided into two main parts. Part I focuses on IoT device identification, and Part II focuses on IoT attack detection.

Part I

IoT Device Identification

Chapter 3

Literature Review for IoT Device Identification

> Knowing yourself is the beginning of all wisdom.
>
> Aristotle

3.1 Overview

This section contains a literature review on Internet of Things (IoT) device identification, which is the first of the two themes of our study. With this review, we aim to provide a comprehensive overview of the research landscape on IoT device identification. In this way, we aim to familiarise ourselves with the work in the literature, the emerging issues and their possible solutions. In addition, we provide information on the tools, methods, algorithms and results obtained in device identification. This analysis aims to provide a comprehensive understanding of the performance, accuracy and limitations of device identification techniques.

3.2 Literature Review

The initial phase of this literature review focused on acquiring a comprehensive yet moderately detailed understanding of the research landscape within the chosen field. While not adhering to the rigorous structure of a systematic review, this exploratory review aims to provide a solid foundation by gathering pertinent information. The objective was to familiarise oneself with the key concepts, methodologies, and trends

prevalent in the field. While the depth of analysis may vary, the collected insights serve as a valuable backdrop for subsequent discussions and analyses within this study.

Within the scope of this literature review, 200 publications that appeared as a result of searching the phrase "IoT Device Identification" in Google Scholar in May 2023 were examined. Among these publications, those that were not related to device identification or those that make radio frequency-based identifications, which are outside our field of interest, were eliminated. IoT identification based on radio frequency uses Radio Frequency Identification (RFID) technology to uniquely identify and track devices. The reason for keeping radio frequency work separate is that radio frequency is not used for device type and category identification using network behaviour, which is our focus. The remaining studies were analysed based on the study year, the utilised dataset, the quantity of identified devices, the feature type, and the employed machine learning techniques. Table 3.1 present the data extracted from this literature review for the periods 2017-2023. This framework allows us to systematically examine the literature according to the following categories: publication year, classification method, dataset, device number, machine learning algorithms, features, evaluation methods and results.

3.2.1 According to Years

Our literature review covered studies without any specific time constraints. Analysis of the publication years revealed that the earliest studies on IoT device identification were conducted in 2017, indicating its relatively recent emergence as a research area. However, there has been a consistent increase in the number of studies in this field over the years, as shown in Figure 3.1. This increase over time can be seen as a natural consequence of the developments in IoT. However, we believe that another important factor in this increase is the IoT data produced for device identification purposes over time and made available for general use.

Table 3.1: Comparison of related work between 2017-2021. Only the best scores are included in the evaluation section. The vulnerabilities are listed in Table 3.3 and explained in Section 3.2.6. (Class Identification (CI),Unique Identification (UI),Type Identification (TI)).

Year	Study	Method	Datasets																		Device Number	Machine learning algorithms																Feature Type										Evaluation Method						Vulnerability
			Aalto	CIC-IoT	DEFT	D-LinkIoT	IoT-23	IoT-deNAT	IoT-traffic-dataset	LSIF	Lwhbench-RASPI	N-BAIOT	Private	ProfiliIoT	UNSW-AD	UNSW-DI	Yourthings		CNN	Bagging	Boosting	Decision Tree	Ensemble Trees	kNN	Linear Models	MLP	Naive Bayes	RNN	SVM	Transformer	Unsupervised ML	Voting	Flow	Header	Packet Size	Packet Size Statistics	IAT-Time	DNS	payload	raw data	Protocol	Header Statistics	Confusion Matrix	Accuracy	Recall	Precision	F1 Score	AUC	
2017	[155]	CI											•					17					•										•								•	99.49					1|7
	[108]	TI|CI											•					9										•						•	•								83.68				1|8
	[158]	TI	•															27					•										•								•	81.50					3|6
	[154]	TI												•				10			•		•											•							•	99.28					7
2018	[201]	CI														•		20					•									•			•	•						99.00	98.70	98.70	98.70		7
	[26]	CI														•		15	•		•	•	•	•	•	•	•	•							•		•			•		99.70					2|7
	[30]	TI											•					14			•	•		•										•				•				99.00					1|2|7
	[204]	TI											•					4				•	•	•		•	•		•											•	99.90	99.90	99.90	99.90		1|7|8
	[214]	TI														•		28					•				•						•					•			•	99.98					3|7
	[224]	TI			•													15					•	•			•						•									98.00	97.00	97.00	97.00		7
2019	[53]	TI			•													15		•		•	•				•		•												•	99.00			98.10		7
	[149]	CI											•					33						•									•								•	98.26	96.59	97.75	97.12		1|7
	[8]	TI	•															23				•			•									•								•	95.00					2
	[15]	TI	•										•					22|6				•											•					•				98.00	98.00				1|2|6
	[178]	CI											•					72										•			•		•									83.00			91.00		1|7
																•		21										•			•		•									66.00			79.00		
													•			•		72|21										•			•		•									61.00			76.00		
	[182]	TI														•		14					•	•					•		•				•								96.00	97.00	97.00	97.00		2|7
	[215]	TI	•															10			•		•	•	•				•											•	•	•	93.20					2|6
	[80]	UI	•															31				•	•	•		•			•												•	90.30					6
	[129]	TI											•			•		72|22										•									•				99.00					1|4
2020	[156]	TI					•											8			•				•				•				•										99.99			99.20	7|8
	[218]	TI													•			14					•	•		•			•						•							99.50	99.50	99.50	99.50		1|2|7
													•					14					•	•		•			•						•							99.40	99.40	99.40	99.40		
	[246]	TI											•					8						•								•										95.00	NA	NA	NA		1|7|8
	[65]	TI														•		24	•											•		•			•				•	•	99.00					7
	[16]	TI	•															27				•	•		•		•	•					•					•				97.00	97.00	97.00	97.00		6
	[126]	TI														•		23				•	•	•	•									•		•		•				100					7
	[28]	TI												•				10					•								•										•	•	92.90					7
	[41]	TI|CI														•		19					•											•				•					97.78					3
			•															27					•											•				•					83.35					
	[172]	UI	•															31										•						•								•	92.00	92.00	93.10	92.50		6
																•		28										•						•								•	99.00	99.00	95.40	97.20		
	[111]	TI											•					43	•			•	•		•			•													•				62.30		1|7
	[57]	TI						•										10					•					•						•	•		•			•		•	99.20					2|7
	[24]	CI											•					39					•	•	•	•		•									•					•		93.60	93.30			1|7
2021	[59]	TI														•		18						•												•						•	99.81					2|7
	[47]	CI											•					41				•											•										99.81	99.90	99.90	99.90		1|7
	[90]	TI														•		20	•						•			•											•	•		99.19			99.09		5
	[110]	TI											•					3				•			•			•								•					•	•	99.10					1|7|8
	[120]	TI														•		9																						•	•		•	99.80	99.80	99.80	99.80	99.80	2|5|8
	[219]	TI														•		10				•		•	•		•														•	•	•	99.20	99.25	100	99.61		2|3|8
	[86]	TI												•				10	•										•												•	•	•	88.20	88.20	83.20	84.90		7
	[173]	TI														•		6					•	•		•							•					•				99.41					2|7|8
	[242]	TI														•		18	•															•					•	•		•	99.91	99.11	99.35	99.23	99.99	2|3
																	•	17	•															•					•	•		•	99.68	99.47	98.63	99.01	99.99	
	[38]	TI	•															10					•																			•		88.50				2|6
	[67]	TI	•										•					12|8				•	•	•					•			•									•	99.50				100	2|6
	[98]	TI											•					25				•	•	•	•		•		•				•				•						93.37					1
	[230]	TI					•											3					•										•										100	100	100	100	7|8
	[226]	TI											•					27								•												•				92.64			89.00		2|4
	[116]	TI	•															27				•	•	•		•		•						•				•							99.20				3
																•		31				•	•	•		•		•						•				•							99.50				
																	•	47				•	•	•		•		•						•				•							99.80				

Table 3.1: Comparison of related work between 2022-2023. Only the best scores are included in the evaluation section. The vulnerabilities are listed in Table 3.3 and explained in Section 3.2.6 (Continued).

Year	Study	Method	Datasets																	Device Number	Machine learning algorithms																		Feature Type										Evaluation Method						Vulnerability
			Aalto	CIC-IoT	DEFT	D-LinkIoT	IoT-23	IoT-deNAT	IoT-traffic-dataset	LSIF	Lwhbench-RASPI	N-BAIOT	Private	ProfilIoT	UNSW-AD	UNSW-DI	Yourthings		CNN	Bagging	Boosting	Decision Tree	Ensemble Trees	kNN	Linear Models	MLP	Naive Bayes	RNN	SVM	Transformer	Unsupervised ML	Voting	Flow	Header	Packet Size	Packet Size Statistics	IAT-Time	DNS	payload	raw data	Protocol	Header Statistics	Confusion Matrix	Accuracy	Recall	Precision	F1 Score	AUC	
2022	[89]	TI														•		22	•																				•	•						90.68		3
	[200]	TI									•							45	•		•	•	•	•	•	•	•	•										•				•	96.02	96.02	96.26	96.02		3
	[142]	TI														•		15	•			•	•	•				•	•			•										•	99.97	93.47	96.73	94.61		2
															•			15																•								•	99.99	98.68	98.12	98.18		
						•												3																•								•	99.89	98.99	99.39	99.14		
				•														37																•								•	86.06	65.81	65.73	64.59		
	[64]	CI													•			24	•												•		•			•			•			99.68					7
																	•	45	•												•		•			•			•			97.20					
	[132]	TI													•			14	•						•						•						•	•				93.03	90.77	91.01	92.96		5|2
	[127]	TI	•															27				•			•			•				•		•					•		•	•	84.50			83.30		3|6
	[222]	TI													•			19				•	•	•		•															•	99.00					2|7
	[58]	TI													•			22					•	•									•									99.50	99.50	99.50	99.50		7
	[197]	TI												•				7	•			•	•					•					•								•	94.47	81.51	81.59	81.42		1|7|8
	[122]	TI													•			9							•														•	•	•	99.80	99.80	99.80	99.80	99.80	2|5|8
	[143]	TI												•				11			•		•		•				•				•	•							•	99.00	99.80				1|7
	[133]	TI|CI													•			18	•						•									•							•	99.00	99.00	99.00	99.00		2
	[236]	TI												•				14				•		•	•				•				•									78.27					1|7
	[43]	UI	•															31				•												•									83.90					3
															•			21				•												•									97.20					
						•												8				•												•									99.00					
	[21]	TI												•				5					•										•								•	96.00					1|3|7|8
															•			16					•										•								•	81.00					
	[225]	TI								•								22							•														•			98.00			83.64		3
	[228]	TI	•											•				10|27							•								•									91.00			87.00		3|6|7
																•		22							•								•									96.00			94.00		
	[117]	TI	•															27			•	•	•		•	•			•				•							•		•					86.10		
																•	•	32			•	•	•		•	•			•				•							•		•					93.70		
2023	[121]	TI														•		8	•												•							•	•				100					2|5|8
	[243]	TI	•															27	•				•										•							•	•		•	99.71	99.85	99.76	99.80		2|3
																•		19	•				•										•							•	•		•	99.89	99.88	99.79	99.83		
	[42]	UI														•		22		•		•	•									•	•										99.69		94.00			3
			•															31		•		•	•									•	•										91.43		91.10			
														•				7		•		•	•									•	•										93.22		93.20			
	[46]	TI													•			10			•	•	•	•	•		•		•												•	84.37	85.96	86.14	85.97		7
	[167]	CI												•		•		42					•	•				•	•				•											94.00	95.00	95.00		1|7
	[141]	TI											•					9	•			•	•	•	•	•		•		•		•	•									•	100			100		7|8
	[217]	TI														•		31							•								•							•			82.00	82.00	88.00	84.00		3
			•															26							•								•							•			79.00	79.00	82.00	79.00		
															•			31							•								•							•			90.00	90.00	92.00	90.00		

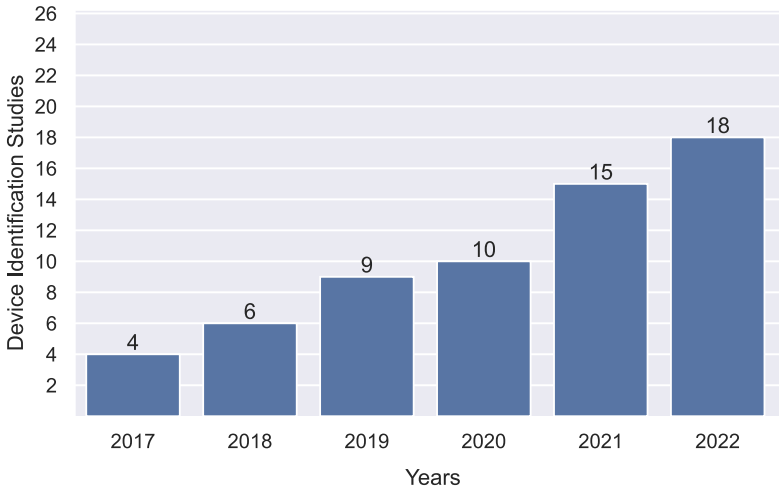


Figure 3.1: Distribution of device identification studies by years. The data were collected in May 2023, so 2023 has not been included.

3.2.2 According to Classification Method

There are three different approaches to the classification process [240]: UI: Each device is considered a "unique" device and a separate class label is created for each [42,43,80,172]. TI: Identification is performed according to the device type. If there are multiple devices of the same brand and model, they are seen as a single class [158]. CI: Different devices that are not the same but have similar features, are gathered under a single class such as camera, speaker, or smart lamp [48, 65, 167].

By way of illustration, we have depicted in Figure 3.2 how devices within a dataset can be categorized using these three distinct classification methods. Figure 3.2 shows the labelling of some devices in the CIC IoT Dataset 2022 (CICIoT22) dataset with three viewpoints. In the example containing 20 devices in total, 20 different labels are formed in the unique method, 13 different labels in the type method, and 3 different labels (Smart Lamp/Bulb, Speakers, and Smart Plug respectively) in the class method.

3.2.3 According to Data

One of the biggest problems in device identification studies is the lack of adequate datasets. The feasibility of applying simulations from various networking studies to device identification research may be limited. Simulated devices often exhibit uniform characteristics, which diverge significantly from the diverse hardware and software profiles observed in real-world devices. A primary challenge arises from the necessity of

Labels			Devices
Class	Type	Unique	
1	1	1	SmartThings Bulb 1
		2	SmartThings Bulb 2
		3	SmartThings Bulb 3
	2	4	Philips Hue White 1
		5	Philips Hue White 2
	3	6	HeimVision Lamp
	4	7	Globe Lamp
2	5	8	Amazon Echo Studio
	6	9	Amazon Echo Dot 1
		10	Amazon Echo Dot 2

Labels			Devices
Class	Type	Unique	
2	7	11	Google Nest Mini
	8	12	Amazon Echo Spot
	9	13	Sonos One Speaker
3	10	14	Yutron Plug 1
		15	Yutron Plug 2
	11	16	Amazon Plug
	12	17	Fibaro Wall Plug 1
		18	Fibaro Wall Plug 2
	13	19	Teckin Plug 1
		20	Teckin Plug 2

Figure 3.2: The change of device labels of the CICIoT22 dataset according to three identification methods.

Table 3.2: Open access IoT datasets used in the literature and some of their properties.

Dataset	Year	Physical Device Number	Device Class Number	Raw data	Benign Data	Attack Data	Label	Multiple sessions	Non-IP devices	Multiple examples of same devices
Aalto [148]	2016	31	27	●	●		●	●	●	●
UNSW-DI [214]	2016	24	24	●	●		●	●		
ProfilIoT [154]	2017	10	10		●		●			
IoT-23 [70]	2018	3	3	●	●	●	●	●		
IoT-traffic-dataset [57]	2018	7	7	●	●		●	●		
N-BAIOT [153]	2018	9	9		●	●	●			
UNSW-AD [81]	2018	28	28	●	●	●	●	●		
Yourthings [11]	2018	51	51	●	●	●	○	●		
LSIF [37]	2019	22	22	●	●		●	●		
MonIoTr [189]	2019	81	56	●	●		●	●		●
D-LinkIoT [43]	2020	14	5	●	●		○	●	●	●
IoT-deNAT [156]	2020	8	8		●		●			
CIC-IoT-22 [48]	2021	60	43	●	●	●	●	●	●	●
Lwhbench-RASPI [220]	2021	45	45		●		●			●

employing authentic device data for the creation of device identification datasets. No-

tably, our observation from the existing literature indicates a prevailing trend among researchers to utilise datasets containing real devices. For example, Table 3.2 shows the datasets used for device identification. All datasets in this table were created using real IoT devices. A dataset with a large set of devices and data, generated from real activities is the one of the keys to building generalisable ML models for device identification. To create a proper device identification dataset, many types of IoT devices are needed, and the supply of these devices is a serious financial burden. In addition, normal data collection requires a long time, labour, and a specialised work environment. Therefore, the use of publicly available data is very common in device identification studies (see Figure 3.3). In this way, both costs are reduced and comparable studies are obtained.

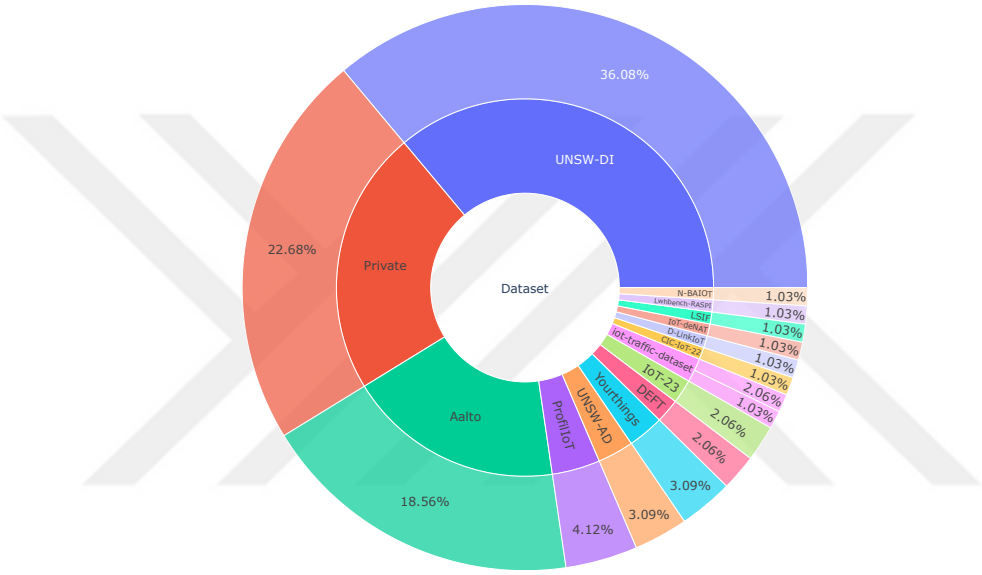


Figure 3.3: Datasets used in the literature and their usage rates.

We analysed approximately 100 experiments in 70 studies for device identification. Among these studies, 25% utilised private datasets, while the majority employed publicly available datasets. These datasets are free of charge and can be accessed directly on the Internet or after filling out an information form. Private datasets often lack detailed information, whereas open access datasets are essential for conducting transparent and reproducible research. They are also essential to facilitate comparisons with other studies. Table 3.2 shows publicly available datasets in the literature and some of their properties. With reference to this table, we will investigate the datasets used in IoT device identification studies.

Although newer datasets are emerging each year, Aalto University - IoT Devices Captures (Aalto) and UNSW IoT Traffic Traces, Data for IEEE TMC 2018 (UNSW-

DI) remain prominent benchmarks in device identification studies. These datasets were collected in 2016 and continue to be extensively utilised. More than half of all device identification studies have been performed using these two datasets.

The inclusion of a large number of devices is a crucial aspect of device identification studies. It provides a realistic evaluation of model performance and allows observation of the model's effectiveness against various device types, making the research more applicable in real-life scenarios. The IoT-23, N-BAIOT, IoT-deNAT, and IoT-traffic-dataset datasets contain a very small number of devices with less than 10 devices. In particular, when evaluating a multi-class classifier, the random chance of success is quite high in the IoT-23 dataset, which contains only 3 devices. In this context, the number of devices should be taken into account when evaluating the results of device identification studies.

The number of devices can be considered from two perspectives. In device identification, devices are classified based on brand and model. If there are multiple instances of the same device model, they are combined into a single label. If no such instances exist, the number of physical devices is equal to the number of device classes in the dataset. For example, in the Aalto dataset, *EdimaxCam1* and *EdimaxCam2*, which are physically distinct devices with separate Media Access Control (MAC) addresses, are treated as a single device labelled as *EdimaxCam*, regardless of their origin.

Another important criterion in the evaluation of datasets is the availability of raw network data (Packet Capture (pcap) files). pcap files are essential for feature extraction. However, feature extraction was performed on some datasets and the extracted features were saved as Comma-separated values (CSV). When converting pcap files to CSV format, researchers often select specific features of interest for extraction. However, this selection process may result in the omission of relevant features that were not initially identified and potentially important information may be lost. In the absence of pcap files, the ability to extract new features or modify old ones can be very limited, as you can only use previously extracted features in the CSV file. In this context, it is especially important that the datasets have pcap files in order to conduct comparable experiments. So, only CSV datasets such as ProfilIoT, N-BAIOT, Lwhbench-RASPI, IoT-deNAT are disadvantaged in this way.

For device identification, the activity of the devices in their normal state is required. In this context, only benign data is used for device identification. However, many datasets developed for attack detection also contain benign data. In this context, it is possible to filter only normal data from attack detection datasets and use this filtered

data for device identification. In many cases such as [116, 141, 142, 230] researchers have chosen to do so using attack detection datasets like IoT-23, N-BAIOT, and UNSW IoT Benign And Attack Traces, Data for ACM SOSR 2019 (UNSW-AD).

IoT devices can also be varied according to their network connection environments. In this context, classical network mediums such as Wifi and Ethernet are generally used, while some devices may use different technologies, non-Internet Protocol (IP) and low-energy protocols, such as Bluetooth, ZigBee, or ZWave. Typically Device Identification (DI) datasets consist of IP devices, but some datasets such as Aalto, CICIoT22 and D-LinkIoT, also have examples of non-IP devices. These datasets represent a network environment where IP and non-IP devices are used together. In order to test the behaviour of a device identification system on non-IP devices, or to test whether it will work on these devices, it is a significant advantage for a dataset to have non-IP devices.

Another important point about datasets is labelling. In the absence of raw data, the CSV files are labelled with device names. For the labelling of the raw data, the MAC/IP name pairs of the devices shared by the dataset creators can be used for labelling. In this context, there is usually no labelling problem in device identification studies. However, MAC and IP addresses are not identifiable in the case of what we call the transfer problem, which is caused by non-IP devices transferring data over other devices (this concept is explored in-depth in Section 3.2.6). In such cases, the originator is expected to isolate these files (this is the solution in the Aalto and CICIoT22 datasets) or share additional information such as frame number or timestamp. There are two of these datasets that have a labelling problem. In the Yourtings dataset, IoT devices are scanned for vulnerabilities. The interaction of this process with devices is very similar to malicious activities. However, there is no information in this dataset that shows the time of these scans. In the D-linkIoT dataset, it is not possible to distinguish between D-LinkDoorSensor data, which is a non-IP device, and D-LinkHomeHub data. In this context, it is not recommended to use at least the sections related to these devices for device identification.

A further feature that many datasets have is the ability to have multiple sessions (by session we mean data that is separated from each other by date, task, or function). Having these sessions makes it possible to properly partition the datasets into training and test data, and to limit information leakage during this partitioning. In the course of our device identification research, we have utilised a selection of these datasets, namely the Aalto, UNSW-DI, UNSW-AD, Locality-Sensitive IoT Fingerprinting (LSIF), CICIoT22, and dataset for "information exposure for consumer IoT devices" (IMC'19) (MonIoTr)

datasets (see Table 3.2). The rationale for our selection of these datasets stems from their advantageous attributes outlined below:

- They have raw data, so data extraction can be carried out without any problems.
- The number of devices they contain is quite high (more than 20).
- They are free from labelling problems.
- They have multiple sessions.

These datasets are described and analysed in more detail in the following chapters (Chapter 4: Aalto, UNSW-DI, UNSW-AD, LSIF; Chapter 5: CICIoT22; Chapter 6: MonIoTr).

An additional aspect relevant to device identification studies that warrant attention here is the number of datasets used and the diversity of devices included. Most device identification studies focus on a single dataset, but some studies utilise multiple datasets to assess the generalizability of features and methods across different data sources. This approach allows for understanding the performance of the method on diverse datasets with varying characteristics.

Table 3.1 also provides insights into the number of devices used in the studies. This information is valuable for understanding the size of private datasets and the selection of devices from publicly accessible datasets. While datasets may contain a specific number of devices, not all devices are used in every study.

3.2.4 According to Features

Features are fundamental to device identification and play a crucial role in the overall system performance. The quality of features is paramount, as the saying goes, "garbage in, garbage out". In this context, we can say that a device identification system can only be as good as its features. Although it is possible to say similar things about data, poor feature selection can lead to a poorly performing model. In datasets whose raw data is not shared, features are limited to what has already been extracted, and this disadvantage gives these datasets a serious limitation. On the other hand, datasets that provide raw data allow for the extraction of different types of features using various approaches. The set of features used to identify a device, i.e., each of the samples that make up the dataset, is known as a fingerprint. Figure 3.4 presents the types of features used in the literature, the frequency of their usage, and their distribution over time.

Among these feature sources, the most commonly utilized is the **Flow** [21,47,58,141, 149,156,167,197,224,228,230,236,246]. Flow refers to the transfer of data between two

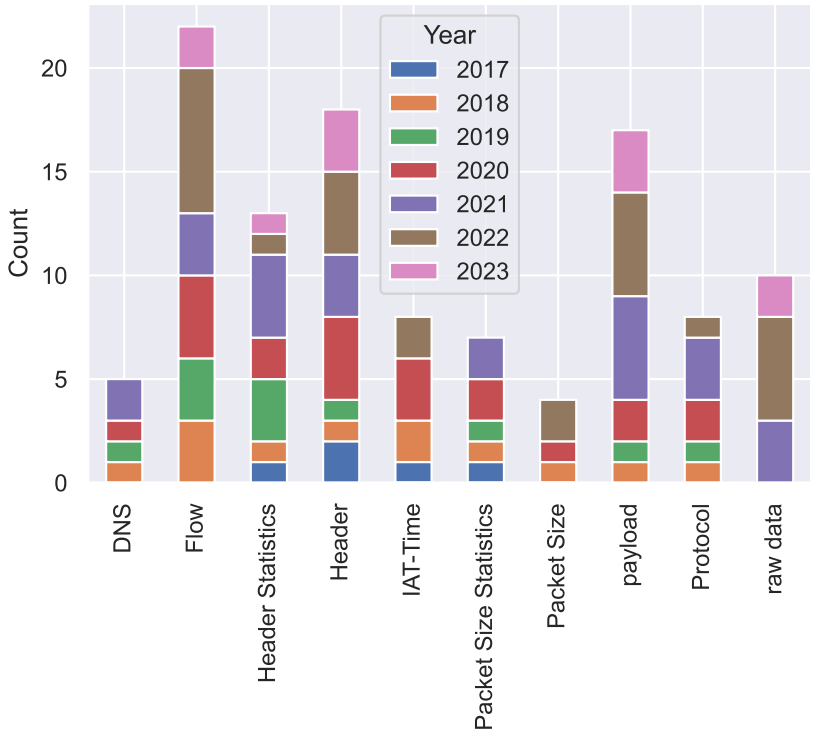


Figure 3.4: Feature types, usage frequency, and temporal distribution.

entities on the network that share common characteristics such as the same protocol, source and destination IP and port numbers. Tools like CICFlowMeter [128] or NetFlow [175] are employed to extract flow statistics, focusing on factors like time (Flow IAT Mean[1], Flow IAT Std, Flow IAT Max, etc.), size (Fwd Packet Length Min, Fwd Packet Length Max, Fwd Packet Length Mean etc.), or flags (SYN Flag Count, RST Flag Count, PSH Flag Count etc.), from raw network data.

Another method, **Header** [8, 30, 41–43, 57, 98, 116, 117, 127, 143, 158, 172, 217, 242], is the use of features derived from protocol headers. The most important difference from flow is that it is not a holistic approach like flow, but uses individual packets. Some studies leverage only features from individual packets [8, 42, 116, 117] to create feature sets, while others [30, 158] combine features from a certain number of different packets to create larger fingerprints. Even if these packets are put together, the relationships between them are not analysed. Therefore, the purpose of this combination is not to see the relationship between the packets but to increase the identifiability of the fingerprint by obtaining larger fingerprints. For example, in IoTSentinel [158] and IoTSense [30]

[1]The complete list of features and their description can be found here: https://github.com/ahlashkari/CICFlowMeter/blob/master/ReadMe.txt

studies, 12 and 20 individual packets are combined to obtain larger fingerprints, respectively. We demonstrate in Chapter 4 that smaller fingerprints whose relationships are considered can lead to more accurate classification results.

Header statistics [28, 38, 53, 67, 80, 111, 154, 204, 222] is another method that leverages packet headers as a source of information. Instead of analysing individual headers, it focuses on the relationships between multiple headers, such as mean, min, max, and standard deviation for a specific number of Time to Live (TTL) values. The determination of the number of packets to group together can be done through various approaches. Design specifications may dictate the number [80], flow standards can be adopted [142], or sliding window techniques can be used to obtain in-window statistics [200].

Another method is Domain Name System (DNS) based [126, 129, 226]. DNS is a protocol used to map between an entity's machine-oriented network address and its human-oriented Uniform Resource Locator (URL). For example, it translates the IP-based machine-oriented address *172.217.23.196* to the human-oriented URL *www.google.com*. IoT devices connect to service provider sites, especially when using cloud services. In this context, this feature method is based on the assumption that information from the DNS protocol can provide knowledge about the devices, depending on the sites they are connected to.

Protocol is another method that is extracted from individual packets [26, 86, 215, 219]. This characteristic, which gives information about the presence or absence of a protocol, is usually translated into binary tables and is included in the fingerprints , e.g., 1 if TCP is used in the packet being analysed, 0 if it is not. This method is used to support other methods rather than being used alone.

Packet sizes/size statistics are generally used in or together with header/flow-based features as well, but there are also studies [59, 142, 182, 218] in the literature that focus only on packet size for device identification.

Statistical features derived from the **time** differences between packets using the timestamps contained in network packets can also be used for device identification. These features can be used to support and enrich other feature types [26, 200, 201], or they can be used independently [24, 65] for device identification.

In some studies [89, 90, 120–122, 132, 225, 242, 243], especially those involving deep learning, bytes of **raw data** are converted to a dataset without performing feature extraction (see Figure 3.5 for raw bytes and the components they represent). In this process, bytes of raw data are given as input to the machine learning model. To standardize the input size, truncation or zero-padding operations can be performed.

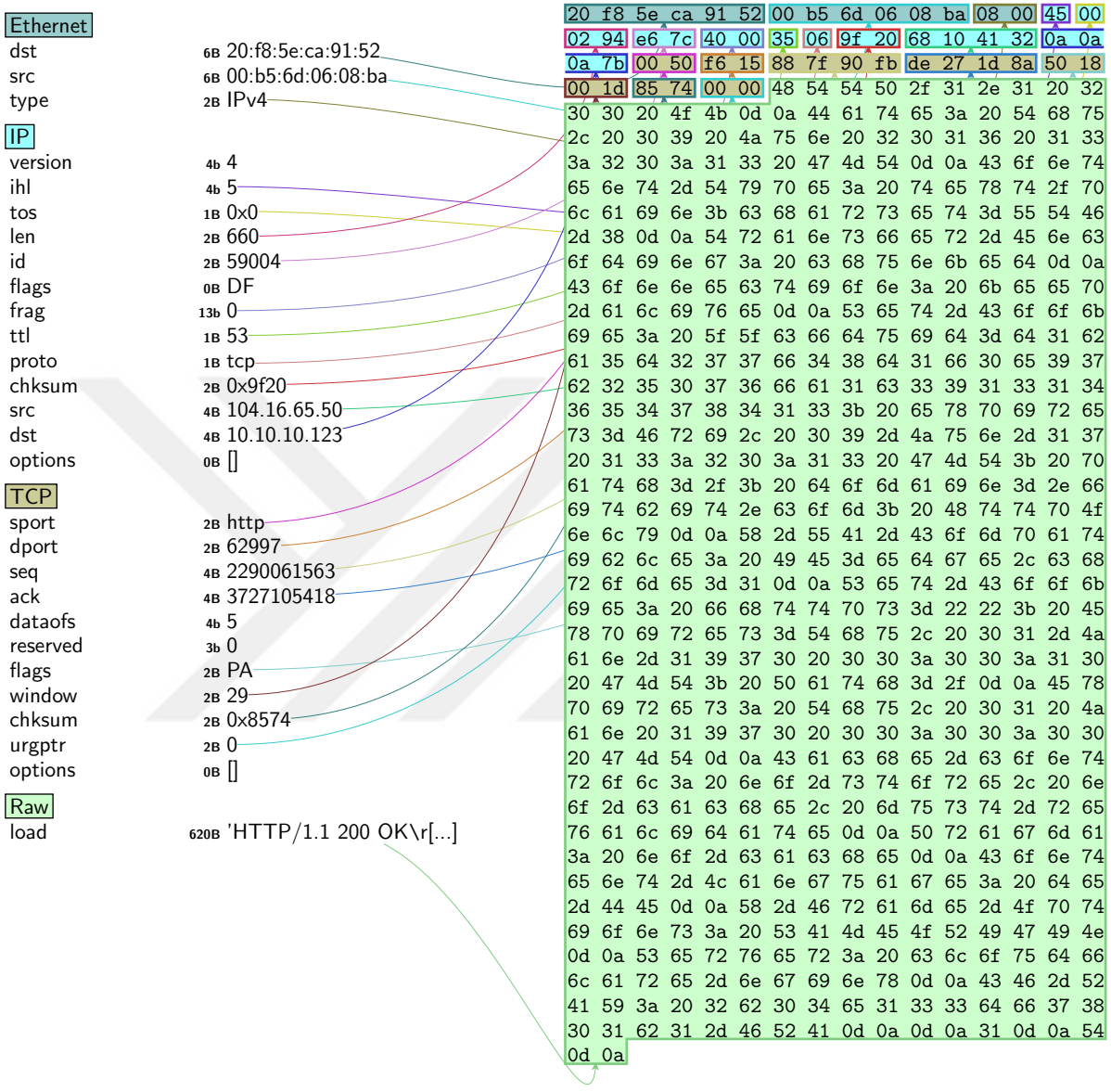
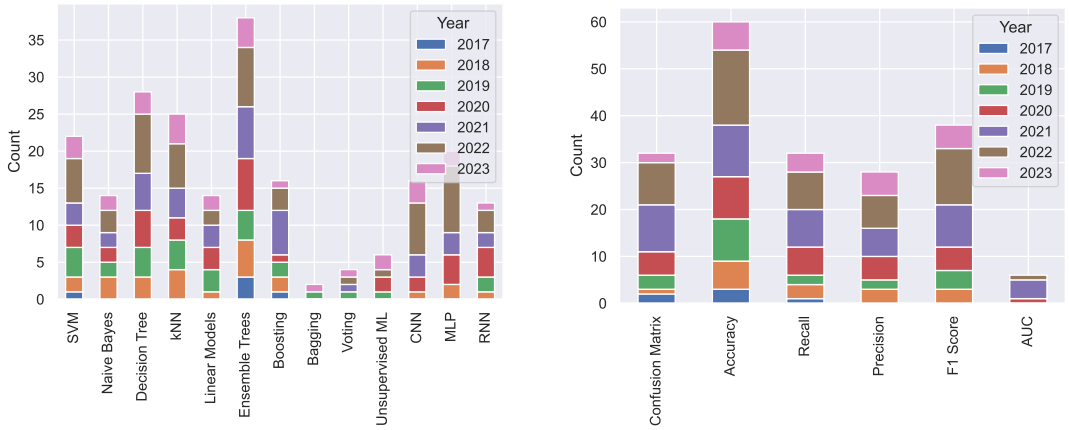


Figure 3.5: The raw bytes of a network packet from the Fitbit Aria WiFi-enabled scale device and the components they represent.

In the use of **payload**, the focus is on the payload of the network packet. In some cases [30, 98, 116, 127], some payload-based features, such as payload size and payload entropy, may be used to support and enrich other types of features. On the other hand, there are some cases [90, 120–122, 132] where only the raw data is used. In this case, the bytes that make up the payload used as input in a very similar way to the use of raw data.

3.2.5 According to Machine Learning and Results

Figure 3.6 visualises the distribution of machine learning methods and evaluation metrics as shown in Table 3.1. From the analysis of Figure 3.6a, it is evident that the most commonly used method is Ensemble trees, particularly Random Forest (RF). Classical Machine Learning (ML) algorithms such as Decision Tree (DT), k-Nearest Neighbours (KNN), and Support Vector Machine (SVM) follow it. Despite the increasing popularity of deep learning techniques, classical machine learning methods remain prevalent in device identification. This can be attributed to the fact that the features utilised in device identification, namely flows and headers, are tabular data. Notably, classical methods like decision trees often outperform deep learning methods, especially when evaluating tabular data lacking temporal relationships [139], as observed in device identification studies. In the literature, in many studies [127, 173, 197, 219] based on tabular data where deep learning and classical ML are used together, classical methods have given better results. The results of our preliminary studies also support this view (see Appendix A.2).



(a) Machine Learning algorithms.

(b) Evaluation methods.

Figure 3.6: Distribution of ML algorithms and the evaluation methods used in the literature by years.

On the other hand, deep learning is very common and successful in time series studies using basic features such as packet size or time [26, 200]. Deep learning also achieves remarkable success when extracting features from raw bytes. However, as discussed later in Section 3.2.6, this may well be due to overfitting. In particular, converting raw data or payloads to images first (this process is visualised in Figure 3.7) and then using these images in highly successful algorithms in visual processing such as CNN is another common device identification method [89, 90, 120–122, 132, 242, 243].

Figure 3.6b reveals that accuracy is the predominant metric used in nearly all studies, with a notable increase in the utilisation of additional metrics in recent years. Furthermore, upon examination of Table 3.1, it becomes evident that certain studies [8, 15, 21, 26, 28, 30, 41, 43, 57, 59, 64, 80, 110, 121, 126, 154, 155, 158, 173, 178, 215, 236] solely rely on accuracy as a metric. Given the inherent imbalance in IoT data distribution and the unsuitability of accuracy for evaluating such imbalanced data, these studies possess a significant likelihood of inducing misleading outcomes.

It is important to consider that not all results are directly comparable. For instance, datasets with a small number of devices may exhibit higher scores due to chance factors. Therefore, it would be unfair to compare studies that identify only three devices with those that identify approximately 30 devices. Another point is the comparability of datasets independent of the number of devices. For example, if we compare the two most used datasets Aalto and UNSW, Aalto is a very challenging dataset compared to UNSW due to the large number of similar devices and the limited data. In this respect, Aalto's results are always expected to be less successful. Comparisons should be made between

studies using similar datasets, and caution is warranted when interpreting results from specialised datasets, given their unknown nature and characteristics.

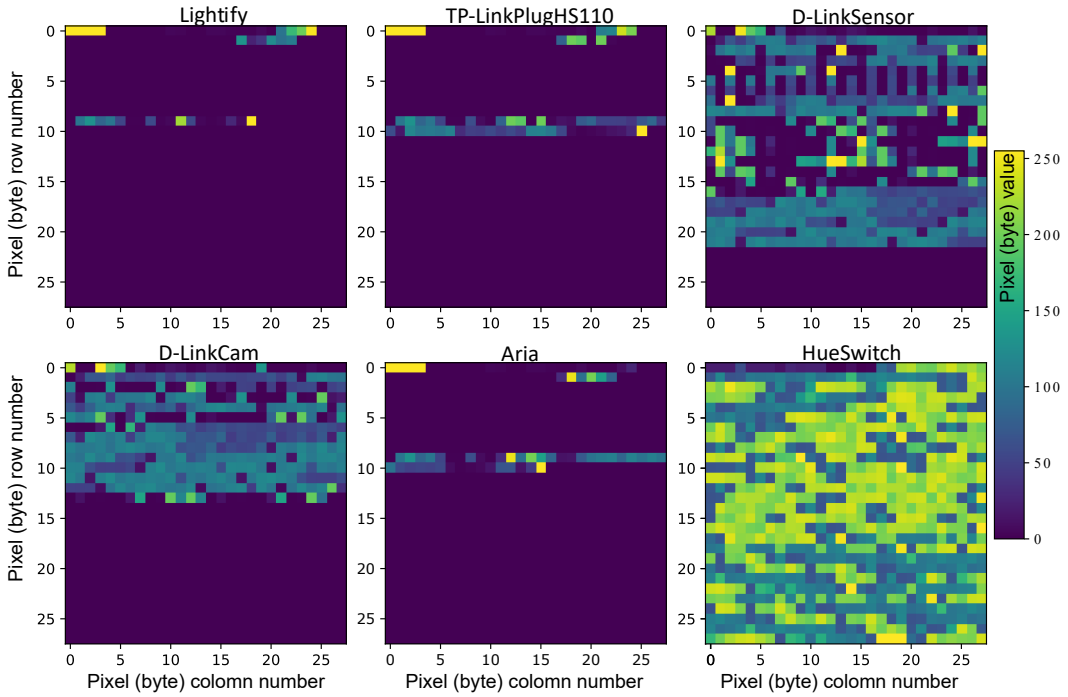


Figure 3.7: The 28x28 images of the raw bytes contained in the packets of six different devices on the Aalto dataset. Each image represents a network packet 784 bytes (28x28). The colour of each pixel represents the value of the byte, with darker colours representing lower values and lighter colours representing higher values.

3.2.6 According to Vulnerabilities

Recognising the intricacies and limitations inherent in existing literature is paramount to facilitating a comprehensive, encompassing, and unimpeded academic investigation. Within this segment, our focus shall be directed toward elucidating certain deficiencies or omissions observed in the studies present within the literature corpus. We will subsequently delve into the potential ramifications of these identified limitations. Within this framework, we shall elucidate the eight enumerated weaknesses, as presented in Table 7.1 for every study, accompanied by illustrative instances for enhanced clarity and comprehension.

1- Usage of Private Datasets: In the literature we have reviewed, about 25% of the datasets used in experiments are private datasets. Since these datasets are not

shared, the information we can obtain about the properties and contents of the datasets is severely limited. However, it is very important to test the developed method on more than one dataset, regardless of whether they are public or private datasets. Otherwise, the developed method may develop overfitting to this dataset and the results obtained may not be realistic. In addition, when the studies are examined, it is seen that more than one dataset is used in 15 studies [41–43, 64, 116, 117, 121, 142, 172, 178, 217, 218, 225, 228, 242, 243] and only one dataset is used in the others. Although such an approach significantly affects the reliability of the results, in recent years there has been an increase in the use of multiple datasets. We believe that the increasing number of publicly available datasets is an important factor in this.

2- Partial Utilisation of Datasets: Another prevalent issue in the literature is the selective utilisation of datasets. A notable observation arises when comparing the number of devices employed in experiments, as outlined in Table 3.1, with the total devices available within the datasets. In several instances, substantial disparities become evident. As an example, consider the UNSW-DI dataset, a prominent choice among researchers, encompassing 24 devices. However, a considerable number of studies [21, 26, 41, 59, 120, 122, 132, 133, 142, 173, 182, 201, 219] utilizing this dataset do so with fewer than 20 devices. Similarly, certain investigations [8, 15, 38, 67, 215] utilising the widely adopted Aalto dataset, encompassing 27 devices, selectively employ only a subset of the available devices. The deliberate omission of devices from a dataset, particularly without providing transparent justification, or conducting analyses exclusively on a subset of devices raises concerns regarding the robustness and credibility of the findings. Well-motivated reasons for removing devices from a study may include:

Quality Control: Certain devices within a dataset could exhibit data quality concerns, hardware disparities, or anomalies. Eliminating such devices safeguards the overall study quality and dependability.

Specific Research Focus: Researchers might be delving into a precise facet of the data pertinent to a subset of devices (e.g., studies centred on cameras or sensors). Employing solely relevant devices in such instances simplifies analysis and heightens result clarity.

Data Constraints: In the feature extraction phase, researchers could target specific features (e.g., TCP or DNS protocols only, payload-based studies). This approach renders some devices ineligible for inclusion if they lack the requisite volume of such data or possess none at all.

However, transparency in reporting the reasons for excluding devices is essential to

maintain the credibility of the study and enable other researchers to understand the scope and limitations of the research.

3- Incorporation of Features Prone to information leakage: Another problem we see is the information leakage that occurs depending on the features used. In many cases this is not systematic, but we can give individual examples. The *IP address count* feature used in the IoT Sentinel [158] study, in addition to packet headers and protocol specifications, is unlikely to generalise beyond the Aalto dataset, as it refers to the number of devices each device communicates with, depending on the environment. Another example, in Chowdhury et al.'s study [42], during feature extraction, features that could uniquely identify sessions (e.g., port numbers, TCP sequence, and TCP acknowledgement) were used. Since data sessions were not considered when splitting training and testing data, information leakage from the training data would very likely cause an overestimation of a model's performance on the test dataset. Another more systematic example of information leakage is when raw data is used. While studies [89,225,242,243] analysing raw data at the individual packet level have shown promise, overfitting may be a concern due to the presence of numerous identifiers in raw data. Some studies [89, 225] have not taken precautions against identifiers such as IP/MAC. Although in others [242, 243] measures are taken to censor identifiers like IP/MAC, it is evident that identifiers extend beyond these (such as addresses in other protocols, checksums, port numbers, ID numbers etc.). For example, the Internet checksum feature stores the source and destination IP addresses in hashed form and is therefore highly identifying. On the other hand, since it is not a statistics-based feature the use of raw data obtained after rigorous discarding of identifying features is very promising.

4- Excessive Focus on the DNS Protocol: Regarding the use of DNS for device identification [126, 129, 226], it is evident that the websites to which devices connect are not directly related to device behaviour. While insights can be derived from device connections to specific sites, this process lacks generalizability. For instance, Figure 3.8 illustrates the DNS outputs of three distinct devices. Despite performing different tasks, their DNS outputs exhibit similarity due to shared manufacturer and cloud infrastructure utilisation. Although we acknowledge that relying solely on a DNS-based method for device identification lacks generalizability and may not accurately reflect device behaviour, we treat the inclusion of this feature similarly to packet payload-based features. We advocate its use in conjunction with other types of features to enhance the overall

feature set.



Figure 3.8: Word cloud of domain names of 3 different devices manufactured by the same brand. Figure adapted from [214].

5- Excessive Focus on Payload: An alternative approach involving raw bytes focuses on payload analysis, offering the potential for discarding identifying features. Some studies [90, 120–122, 132] in the literature focus only on the payload and extract features from the raw payload bytes. Nevertheless, the proliferation of encryption within contemporary internet data prompts scepticism concerning the effectiveness of payload-centric methodologies. Given the surge in encrypted network traffic, features derived solely from payload content exhibit a significant drawback. Nonetheless, we advocate for the incorporation of payload-based features (such as payload size or payload entropy) in conjunction with other feature types to enhance the comprehensiveness of feature sets.

6- Suffering from the Transfer Problem: Within the Aalto dataset there are non-IP devices using Z-Wave and Zigbee protocols. These devices do not communicate directly with the gateway where the data is collected but through packets transmitted via another device. As a result of this process, some MAC/IP addresses in the Aalto dataset represent more than one device in the local network. We call this situation the "transfer problem". It is explained with an example in Figure 3.9. MAC/IP addresses are the main element in the extraction of features in many studies [15, 16, 38, 67, 80, 127, 158, 172, 215, 228]. If a method dependent on MAC/IP addresses is applied, it is not possible to distinguish between non-IP devices and the devices used to transmit them.

The fact that one IP can represent multiple devices also applies to devices (not necessarily IoT) behind a Network Address Translation (NAT). NAT works at the IP layer of the network protocol stack and is primarily used to map private IP addresses in a local network to a single public IP address. However, the transfer problem is not related to the multiple representations caused by NAT, but only to the multiple representations in the local network.

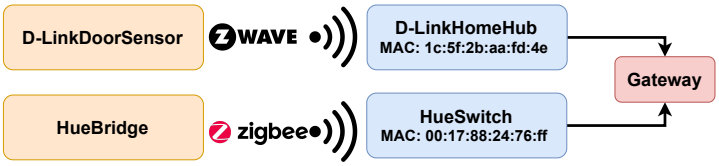


Figure 3.9: Example transfer problem in the Aalto dataset. Network data is collected at the gateway. Between D-LinkDoorSensor and D-LinkHomeHub there is only Z-Wave as a communication medium. Between D-LinkHomeHub and the gateway, there is only Ethernet. Data from the D-LinkDoorSensor is decapsulated at the D-LinkHomeHub and re-encapsulated for transmission to the gateway. As a result, both the D-LinkDoorSensor and D-LinkHomeHub share the same MAC address as the data is encapsulated on the same device (the same happens between HueBridge, HueSwitch and gateway). Therefore, studies that use MAC addresses to concatenate packets cannot overcome this problem.

7- The Usage of Statistics-based Features: Statistics-based features, including flow or header statistics, are among the most commonly employed types of features for device identification (see Figure 3.4). However, we have identified certain issues with these approaches. Statistical methods, which emphasise device interaction within a network rather than individual device behaviour, may not be suitable for device type identification. This is because when the device or model operates in a different network, the identification method may become ineffective due to the potential change of features depending on the statistics. Our preliminary experiments give results that support this idea (see Appendix A.1). In these experiments, we compared flow-based features with features derived from individual packet headers, and found that features derived from individual packets were more resilient to changes in the network, while flow-based features based on network statistics were negatively affected by network changes.

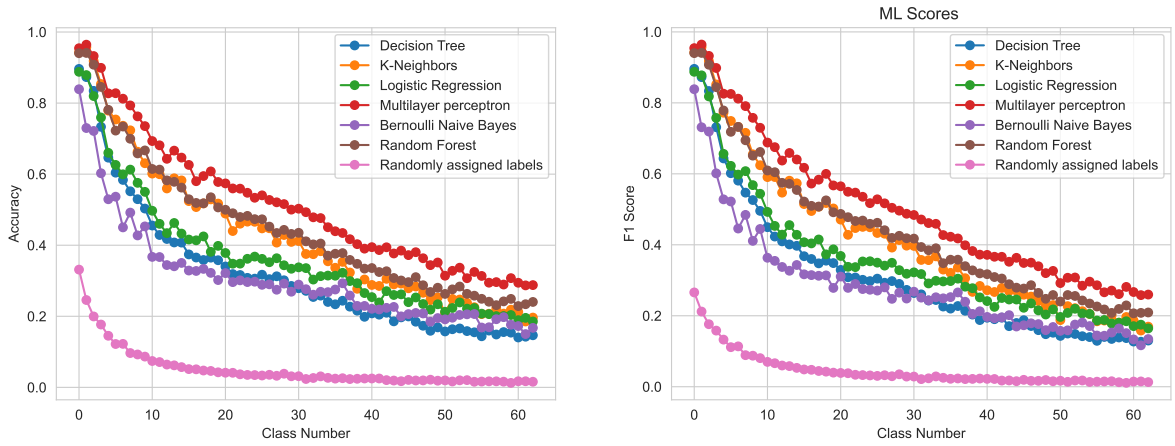
Furthermore, statistical methods rely on data exchange between source and destination files, often correlating values based on IP/MAC addresses. However, when the transfer problem occurs, non-IP devices lack identifying information such as IP/MAC, rendering statistical methods inadequate for detecting them. An additional drawback of the statistical approach lies in its substantial reliance on attributes like IP/MAC addresses during the statistical feature extraction process. For instance, a study utilising the Aalto dataset will encounter challenges addressing the transfer problem when employing statistical features. While transfer problems have not been noted in datasets apart from Aalto and CICIoT22, such scenarios are plausible within real-world applications.

8- Utilisation of Datasets with a Minimal Number of Devices: Another critical factor that warrants consideration when evaluating results in the literature is the number of devices utilised. Upon analysing the examples, it becomes evident that a number of studies involve a relatively limited pool of devices such as [21, 108, 110, 120–122, 141, 156, 173, 197, 204, 219, 230, 246]. The utilisation of a scant number of devices (e.g., 10 or fewer) within an IoT device identification dataset can give rise to several drawbacks.

When dealing with a small number of devices, the dataset might not effectively encompass the full spectrum of diversity and intricacy exhibited by IoT devices in real-world scenarios. Consequently, this limitation can engender a skewed or incomplete comprehension of device behaviours and attributes. A dataset of diminutive size could fail to capture the subtleties and variations characteristic of distinct device types, thereby causing a deficiency in the models' capacity to generalise. While models trained on such modest datasets might demonstrate commendable performance on training data, they might falter in their ability to extrapolate to novel, unseen data.

It is important to note that IoT environments typically encompass a wide array of devices, including those that might be considered relatively rare. A constrained dataset might lack a sufficient number of examples showcasing these less common devices, thereby impeding the accurate identification and differentiation of such devices. Furthermore, the issue of increased chance of success comes into play. Particularly in multi-class scenarios, the likelihood of achieving success purely by chance rises inversely with the number of classes. In instances where the number of devices remains small, the probability of chance success becomes elevated.

We conducted an experiment to assess the impact of the number of classes on chance success. 63 synthetic datasets were generated with class counts ranging from 2 to 64, and their classification success was observed. Figure 3.10 depicts the results, illustrating an inverse relationship between the number of classes and success. It's crucial to note that factors beyond the number of classes, such as class separability, distribution, and the nature of the classification problem, also influence outcomes.



(a) Change in Accuracy according to the number of classes.

(b) Change in F1 score according to the number of classes.

Figure 3.10: The impact of the number of classes on chance success. The datasets consist of n classes with 10 features, generated using the make_classification[2] function of the scikit-learn[3] library. Results represent the average of 5 times 5-fold cross-validation. : https://github.com/kahramankostas/Impacts-of-class-numbers-in-ML.

These eight vulnerabilities undermine the ability to perform device identification with high accuracy. In this thesis, one of our focus areas is to provide an effective and generalizable high accurate IoT device identification that is free from these vulnerabilities. However, in addition to these vulnerabilities, although outside the scope of our research, it is worth mentioning that high fidelity IoT device identification has the potential to be used for user profiling and privacy invasion.

High-fidelity IoT device identification has significant implications for user profiling. The ability to accurately identify and track IoT devices can lead to the prediction of users' future actions, posing a significant threat to user privacy [248]. In this context, it is important to anonymize user information in device identification efforts, and to censor identifying fields when raw data is used, as well as those that contain private data of the user.

3.3 Conclusions

This section offers a comprehensive literature review encompassing datasets, features, and machine learning techniques employed in IoT device identification research (R1.1, R1.2, C1.1). It also assesses potential shortcomings within this literature, elucidating their causes and repercussions (RQ1.3, C1.1). As a result, critical gaps and notable

Table 3.3: Vulnerabilities in device identification studies and studies that suffer from these vulnerabilities.

	Vulnerability	The studies affected by this vulnerability
1	Private dataset usage.	[15, 21, 24, 30, 47, 98, 108, 110, 111, 129, 143, 149, 155, 167, 178, 197, 204, 218, 236, 246]
2	Partial dataset utilisation.	[8, 15, 26, 30, 38, 57, 59, 67, 120–122, 132, 133, 142, 173, 182, 215, 218, 219, 222, 226, 242, 243]
3	Use of features that may lead to information leakage.	[21, 41–43, 89, 116, 127, 158, 200, 214, 217, 219, 225, 228, 242, 243]
4	Excessive focus on the DNS Protocol.	[129, 226]
5	Excessive focus on the payload.	[90, 120–122, 132]
6	Suffering from the transfer problem.	[15, 16, 38, 67, 80, 127, 158, 172, 215, 228]
7	The Usage of Statistics-based Features and Transfer Problem Probability.	[21, 24, 26, 28, 30, 46, 47, 53, 57–59, 64, 65, 86, 110, 111, 126, 141, 143, 149, 154–156, 167, 173, 178, 182, 197, 201, 204, 214, 218, 222, 224, 228, 230, 236, 246]
8	Utilisation of Datasets with a Minimal Number of Devices	[21, 108, 110, 120–122, 141, 156, 173, 197, 204, 219, 230, 246],

issues in the literature are brought to the forefront.

This chapter establishes the groundwork for the subsequent chapters concerning device identification. Chapter 4 takes these identified weaknesses into consideration and introduces a device identification method devoid of these limitations. In Chapter 5, the effectiveness of data collected in different situations on this device identification system is investigated and the method is validated by an independent dataset. Chapter 6 marks the final stage of device identification, examining the efficacy of our method on both encrypted and unencrypted data. The system is tested on previously unseen devices, culminating in a comprehensive assessment.

Chapter 4

Device Identification Methodology

> What is difficult in training will become easy in a ~~battle~~ testing.
>
> Alexander Suvorov

In this chapter we introduce IoTDevID, a generalisable method for device identification that can achieve high performance at the individual packet level, identifying devices using different network environments, including non-Internet Protocol (IP) devices. The following publication has been utilised in the preparation of this chapter.

- K. Kostas, M. Just, and M. A. Lones. IoTDevID: A behavior-based device identification method for the IoT. IEEE Internet of Things Journal, 9(23):23741–23749, 2022. :github.com/kahramankostas/IoTDevIDv2

4.1 Overview

Securing Internet of Things (IoT) devices using conventional security solutions can be quite challenging due to the inherent limitations of these devices, including constraints on processing power, battery life, and bandwidth [244].

Additionally, device heterogeneity introduces further complexities [92]. For instance, a single device may incorporate various sensors (such as temperature, humidity, motion, and light), each with its unique communication requirements. While some research has delved into specialised areas like home appliances [11] or smart cities [179], it's important to note that even within these categories, devices can exhibit diverse characteristics. Moreover, IoT security tends to be overlooked by users, primarily because of the absence of a familiar user interface [11]. Given the widespread proliferation and diversity of

today's IoT devices, it is unreasonable to expect users to recognise and comprehend the security implications of all these devices. Hence, the automatic identification of devices within the network becomes crucial in addressing potential security issues.

An effective Device Identification (DI) system can identify devices on a network, allowing for necessary actions such as updates, access restrictions, or isolation of devices with security vulnerabilities. By doing so, it becomes possible to proactively prevent potential security threats stemming from IoT devices on the network.

In this chapter, we introduce a new Internet of Things (IoT) DI method that models the behaviour of the network packets communicated by the devices. IoTDevID analyses the behaviour of IoT devices in order to identify their brand and model. The resulting normal behaviour profiles can then be used to secure devices, e.g., by finding devices with known vulnerabilities. More details about the threat model can be found in Section 4.6. Our approach has similarities to the work reviewed in Section 3.2.4, in that we use Machine Learning (ML) methods to train device fingerprints from information stored in packet headers. However, we approach this in a more systematic way, using feature selection to identify fingerprints that generalise in a meaningful way from the training data, and with an emphasis on developing a model that can be deployed within real network environments. With IoTDevID, devices are identified by their behaviour, a critical first step to the subsequent detection of anomalous behaviour. It identifies device behaviours at the individual packet level using generalizable features. In this way, it can detect different kinds of devices, while offering high detection success rate with its aggregation method. This work offers the following novel contributions from previous work:

1. Our method does not strictly depend on identifying features such as the Media Access Control (MAC) and IP address in the feature extraction step. This gives it wider applicability than previous approaches that depended on this information, such as [30, 80, 158]. Notably, it allows identification of devices that use non-IP and low-energy protocols such as Bluetooth, ZigBee, or ZWave.

2. Our DI model uses a feature set that summarises the network behaviour of IoT devices. We carried out a systematic study of packet-level features, using various feature selection techniques to build a feature set that improves on previous approaches. In contrast to previous research [8, 80, 214], our approach involved a rigorous process for identifying and eliminating features such as session IDs and port numbers that are not generalisable across datasets. To achieve this, we devel-

oped a test that specifically targets features that are prone to information leakage. In this evaluation, we assessed the performance of each feature on both an isolated and non-isolated dataset. Features that showed performance differences were either removed or modified (see Section 4.3.1).

3. Our approach was developed using a rigorous experimental process. We took steps to prevent information leaking from the test set into training, and used a second, independent, dataset to demonstrate the generality of our approach. Consequently, we believe that our results are more realistic and generalizable than much of the previously published work in this area.

As a consequence, our DI method is more suited to real world IoT network environments than previous approaches, not only because it supports a wider range of environments, but also because we have more confidence that the approach generalises to environments that were unseen during development. To promote transparency and reproducibility, we have made our dataset, scripts, and supplementary material public[1].

The basic concepts of machine learning, network and evaluation used in this chapter are subject to a comprehensive explanation in Background Chapter 2. Similarly, the underlying principles in the literature on device identification, covering aspects such as methodologies, datasets and vulnerabilities, are described in detail in Literature Review for IoT DI Chapter 3. This chapter is organised as follows. Section 4.2 describes the methods we use to model network packets, and the datasets used for evaluation. Section 4.3 gives an overview of our model selection approach, Section 4.4 evaluates the selected model, and Section 4.5 compares it against other DI approaches. Limitations are discussed in Section 4.6, and conclusions are presented in Section 4.7.

4.2 Materials and Methods

This section is divided into three subsections that outline the experimental framework and procedures employed for the investigation. These subsections are:

IoT Data Selection (Section 4.2.1): Three IoT device datasets are introduced, serving as the basis for performance evaluation. In addition to the technical information of these datasets, the distinguishing features, pros and cons of each dataset are explained.

Individual, Aggregated, and Mixed Method Packets (Section 4.2.2): This subsection introduces innovative strategies for device fingerprinting. Individual packet-based

[1]Materials available at:github.com/kahramankostas/IoTDevIDv2

discrimination is emphasised for efficiency and accuracy. Additionally, an aggregated method is presented, along with a mixed approach to address exceptional cases and enhance success rates.

Feature Extraction (Section 4.2.3): In this subsection, how the training and test data are separated, the path and methods followed in the feature extraction process are expressed.

4.2.1 IoT Data Selection

We used three public datasets to measure and compare the performance of our DI approach. All of these contain real device data. Since our approach is designed for benign networks, the datasets do not contain attack data. The first, the Aalto University - IoT Devices Captures (Aalto) dataset [158] has 31 devices and contains only the device installation data, but this installation process was repeated 20 times for each device to increase the amount of data [158]. Whilst there are a total of 31 devices in this dataset, four of these devices are in pairs (see Table 4.1). For example, there are two WeMoSwitches with different MAC/IP addresses. Our objective is to detect device types based on behaviour. Therefore, these pairs of devices are treated as a single device rather than two separate or unique devices. Therefore, although the dataset contains 31 devices, it contains 27 classes. Another characteristic of the dataset is that two devices using low energy protocols connect to the gateway where the data is collected via other devices. Therefore, these two devices (D-LinkDoorSensor and HueSwitch) do not have identifying features such as their own MAC and IP address. They use the IP/MAC address of the devices they communicate through (D-LinkHomeHub and HueBridge respectively). This is an example of the transfer problem described in Section 3.2.6. The device MAC addresses and labels related to this problem are shared in Table 4.1. Notably, when using an individual packet-based approach (which is independent of MAC/IP addresses), our method is able to identify devices that are suffering from the transfer problem.

The second, UNSW IoT Traffic Traces, Data for IEEE TMC 2018 (UNSW-DI) [214] contains the day-to-day network logs of 28 IoT devices, each recorded over a period of 26 weeks. However, only a 60-day portion of this data is publicly available. This limited data does not include the four devices (Ring Door Bell, Hello Barbie, August Doorbell Camera, Belkin Camera) from the work of Sivanathan et al. [214]. Therefore, we extracted data for these devices from the UNSW IoT Benign And Attack Traces, Data for ACM SOSR 2019 (UNSW-AD) dataset [81], another study by the same institution. We have thus compensated for devices not included in the UNSW-DI dataset. This

merged dataset is referred to as UNSW Merged (UNSWM) in our work and the UNSWM dataset we use contains a total of 32 IoT devices and 7 non-IoT devices. Following the example of another study which used this dataset [214], we gathered non-iot devices under one class label, resulting in 33 total labels.

The third, Locality-Sensitive IoT Fingerprinting (LSIF) dataset contains a 20-day record of a testbed consisting of 22 devices. Each Packet Capture (pcap) file is named with the date of the day it was collected. During this process, data was collected from IoT devices both through automated behaviours (This automation involves devices such as lamps and sockets being activated every thirty minutes and going into inactive mode after five minutes, and devices such as cameras, sensors and doorbells being set to detect these changes.) and through the manual usage in laboratory conditions. It, therefore, contains benign user-interactive data. Since the size of LSIF and UNSWM datasets are quite large, we used the smaller Aalto dataset to formulate our modelling approach and the other two datasets to measure its broader generality.

Table 4.1: Device MAC addresses and labels in Aalto dataset. MAC addresses in bold correspond to more than one label, and labels in bold correspond to more than one MAC address.

MAC Address	Label	MAC Address	Label
94:10:3e:35:01:c1	**WeMoSwitch**	3c:49:37:03:17:db	**EdnetCam**
94:10:3e:34:0c:b5	**WeMoSwitch**	3c:49:37:03:17:f0	**EdnetCam**
1c:5f:2b:aa:fd:4e	D-LinkDoorSensor	**00:17:88:24:76:ff**	HueBridge
1c:5f:2b:aa:fd:4e	D-LinkHomeHub	**00:17:88:24:76:ff**	HueSwitch
94:10:3e:41:c2:05	**WeMoInsightSwitch**	74:da:38:80:7a:08	**EdimaxCam**
94:10:3e:42:80:69	**WeMoInsightSwitch**	74:da:38:80:79:fc	**EdimaxCam**

4.2.2 Individual, Aggregated and Mixed Method Packets

Creating fingerprints by combining multiple packets is a common strategy [30, 80, 158, 214], where identifying features such as a MAC or IP address are used to combine multiple packets. These features uniquely identify the device, allowing us to know the source of the data, but do not provide any information about the device type or behaviour. Assuming that packets from the same address come from the same device, larger fingerprints are often constructed by combining these packets. However, this assumption is not always correct. In cases where there is a *transfer problem*, an IP/MAC address can represent more than one device. Consequently, the packet merging logic will not work, as devices with the same MAC address but different behaviour will be considered as a

single device. Further, there is no standard for the size of this merging process (number of packets, duration, etc.). Without merging, the use of individual packets is beneficial in terms of speed and efficient use of limited resources [94]. For these reasons, unlike many previous studies [30, 80, 158, 214], we chose to use *individual packets*, not merged ones as the basis for discrimination.

However, when viewed individually, some individual packets may be ambiguous and match the behavioural profile of more than one device. Therefore, the success rate from discrimination based on single packets is limited. We address this by introducing an alternative to individual packets, called *aggregated method* that groups packets based on identifier attribute (MAC address) and the results of the ML algorithm at the individual packet level. In this way, we can increase the success rate by ignoring the mislabelled packets. The difference of this approach from previous work which merged packets is that the behaviour analysis in our aggregation is performed *before* the merging process. In other words, we combine not the packets, but the labels assigned to these packets by the ML algorithm. Since, this method does not work for MAC addresses suffering from the transfer problem, we have added an exception to our evaluation phase and followed a *mixed method* to deal with this issue.

The aggregation algorithm can be explained simply as follows. The algorithm consists of two steps (see Figure 4.1), using as input the MAC address and the predicted label. In the first step, it groups the MAC addresses according to the labels assigned to them and then finds the predominant MAC address for each label. If a MAC address is selected as dominant for more than one label, this MAC address is added to the exception list (likely a transfer problem with one MAC address being used for more than one device). In the second step, the predicted labels are gathered together in groups according to their MAC addresses. The most repetitive label among these groups is applied to the whole group to obtain aggregated labels. This procedure is not applied for MAC addresses that have entered the exception list; only the individual results are used for them.

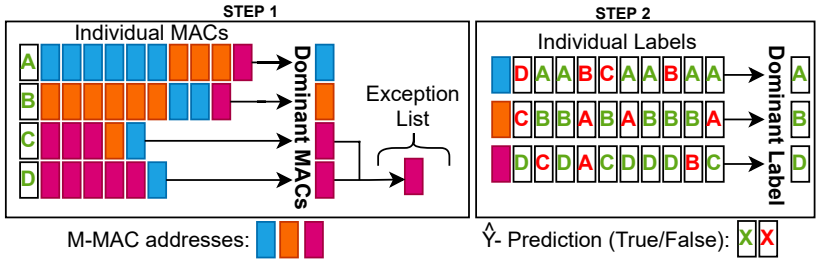


Figure 4.1: The steps of the aggregation algorithm. In the first step, MAC addresses are sorted by labels to find MAC addresses that show more than one device behaviour and added to the exception list. In the second step, labels outside the exception list are grouped according to MAC addresses. The most repetitive label is assigned to the whole group and the aggregated results are created.

This aggregation process is explained with reference to the pseudocode (see Algorithm 1 for pseudocode and corresponding line numbers mentioned below) as follows. Our aggregation algorithm takes the device MAC addresses $M = \{m_1, m_2, \ldots, m_n\}$, the result of the ML algorithm (initial labels) $\hat{Y} = \{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n\}$, and group size ($g$) as input. The labels originating from the same MAC address are collected in the same array (lines 3-11). Each array is divided into small pieces of group number size (lines 12-15). Aggregated results are created by assigning the mode (the most frequent element) to the entire group as a new label $\hat{Y}' = \{\hat{y}'_1, \hat{y}'_2, \ldots, \hat{y}'_n\}$ (lines 16-22). On the other hand, for creating the exception list of aggregation process, the MAC address corresponding to each label is added to the same array (lines 30-33). If a MAC address is the most frequent element in more than one array, it is added to the exception list (lines 34-38). When applying the mixed method, individual labels ($\hat{y}$) are applied for the MAC addresses in the exception list, and aggregated labels ($\hat{y}'$) are applied for the remainder (lines 40-45).

Actively employing the MAC address during the aggregation phase prompts inquiry into the behaviour of IoTDevID within protocols featuring MAC randomization, such as Bluetooth Low Energy (BLE). In BLE-enabled devices, the utilization of MAC randomization is designed to enhance user privacy and impede device tracking by generating a distinct random address upon each wakeup, as opposed to utilizing the original MAC address [102].

While Z-Wave and Zigbee protocols are present in the Aalto dataset, which comprises non-IP instances, no devices utilizing the BLE protocol have been identified (to the best of our knowledge, no device identification datasets (see Table 3.2) include devices employing this protocol). Consequently, we will engage in reasoned analysis rather than

Algorithm 1 Aggregation Algorithm ($\hat{Y}, M$)

$MAC_{ij} \leftarrow [\emptyset][\emptyset]$ ▷ create an empty two-dimensional array MACs & index
$seen \leftarrow [\emptyset]$, $g \leftarrow 12$ ▷ specifying the group size
for each $m \in M$ **do** ▷ discovering unique MAC addresses
  if $m \notin seen$ **then**
    $seen \leftarrow seen \cup m$
    $i \leftarrow$ **length**$(seen) - 1, j \leftarrow 0$
    $MAC_{ij} \leftarrow m$ ▷ create new array and assign m as the first element
for $j \leftarrow 0$, **length**(M) **do** ▷ store the MAC & indices in the 2D array
  for $i \leftarrow 0$, **length**$(seen)$ **do**
    if $MAC_{i0} == M_j$ **then**
      $MAC_i \leftarrow MAC_i \cup j$ ▷ add the index of m to MAC_i
for $i \leftarrow 0$, **length**$(seen)$ **do**
  $C \leftarrow []$ ▷ hold the indices of ms divided into chunks
  for $j \leftarrow 1$, **length**$(MAC_i) - g$, **Step** $= g$ **do**
    $C \leftarrow C \cup MAC_{i[j:j+g]}$ ▷ divide MAC_i into chunks of g
  for each $c \in C$ **do**
    $g_list \leftarrow []$
    for each $j \in c$ **do**
      $g_list \leftarrow g_list \cup \hat{y}_j$ ▷ assign $\hat{y}$ sharing same index with m
    $mode \leftarrow$ **mode**(g_list)
    for j **in** c **do**
      $\hat{y}'_j \leftarrow mode$ ▷ Aggregated labels
$LABEL_{ij} \leftarrow [\emptyset][\emptyset]$ ▷ create an empty 2-dimensional array labels & MACs
$seen \leftarrow [\emptyset]$, $exceptionlist \leftarrow [\emptyset]$, $dominantMACS \leftarrow [\emptyset]$
for each $\hat{y} \in \hat{Y}$ **do** ▷ discovering unique Labels
  if $\hat{y} \notin seen$ **then**
    $seen \leftarrow seen \cup \hat{y}$
    $i \leftarrow$ **length**$(seen) - 1, j \leftarrow 0$
    $LABEL_{ij} \leftarrow \hat{y}$ ▷ create array, assign $\hat{y}$ as the first element
for $j \leftarrow 0$, **length**$(\hat{Y})$ **do** ▷ store Labels & MACs in the 2D array
  for $i \leftarrow 0$, **length**$(seen)$ **do**
    if $LABEL_{i0} == \hat{y}_i$ **then**
      $LABEL_i \leftarrow LABEL_i \cup m_j$ ▷ add the index of m_j to MAC_i
for $i \leftarrow 0$, **length**$(seen)$ **do**
  if mode$(LABEL_i)$ **not in** $dominantMACS$ **then**
    $dominantMACS \leftarrow dominantMACS \cup$ **mode**$(LABEL_i)$
  else
    $exceptionlist \leftarrow exceptionlist \cup$ **mode**$(LABEL_i)$
$FinalReulsts \leftarrow [\emptyset]$
for $j \leftarrow 0$, **length**$(\hat{Y})$ **do** ▷ crate mixed labels
  if m_j **in** $exceptionlist$ **then**
    $FinalResulsts \leftarrow FinalResulsts \cup \hat{y}_j$
  else
    $FinalResulsts \leftarrow FinalResulsts \cup \hat{y'}_j$
return $FinalResulsts$

relying on experimental demonstration.

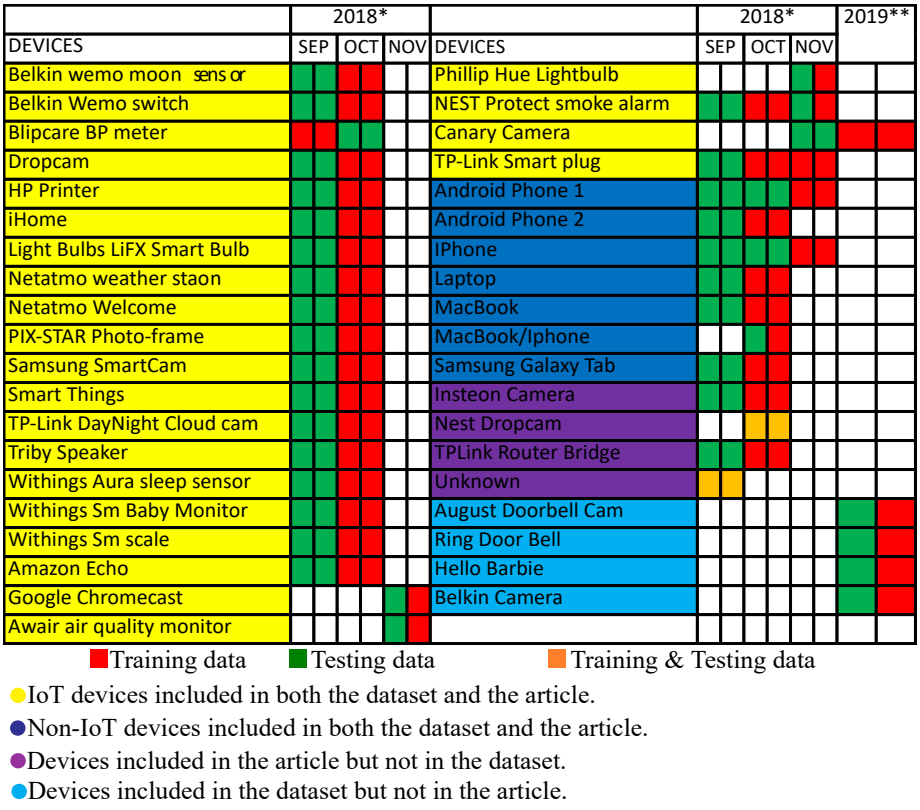
The implication of MAC randomization is the emergence of multiple devices exhibiting identical behavioural characteristics. This scenario parallels the existence of devices with multiple instances, as illustrated in Table 4.1 (such as WeMoSwitch, WeMoInsightSwitch, EdnetCam, and EdimaxCam). Because in these devices, a device is associated with more than one MAC address. In this context, we anticipate MAC randomization to seamlessly align with the aggregation algorithm.

Another issue with the aggregation is that while it works well for DI with benign data, care should be taken in networks with malicious data. Depending on the tolerance of the aggregation algorithm, malicious packets that impersonate an IP/MAC address and are close to the behaviour of the benign packets with the same IP/MAC address may be grouped together (see Limitations 4.6).

4.2.3 Feature Extraction

We performed feature extraction from the packet headers of pcap files. Before this, we separated the pcap files into test and training sets to completely isolate these sets. The Aalto dataset has 20 sessions (pcap files) per device. We divided them in two parts: 16 parts training and four parts test data (80%:20%). The UNSWM dataset consists of a different pcap file for each day's records. To isolate, we constructed the training and test data from data collected on different days (pcap files) (see Table 4.2 for distribution). The LSIF dataset consists of pcap files containing 20 days of usage of the IoT network between 2019.06.29 - 2019.07.19. We used the first 10 days of these records as training and the second 10 days as test data. Section 4.3.1 explains our feature selection process in detail.

Table 4.2: Which data in the UNSWM (*UNSW-DI, **UNSW-AD) dataset were taken from which date range.



In addition to the 111 features extracted from the network packet headers, we included payload entropy [30], protocol (from Transmission Control Protocol (TCP)-IP layers), source and destination port class [158], which are all features found to be useful in previous studies. While payload entropy gives clues about the characteristic of the payload, the protocol and port class features provide summary information about source and destination ports. Source-destination port class features gather port numbers under 14 classes: No port, 0-Reserved, 53-Domain Name System (DNS), 67-Bootstrap Protocol (BOOTP) server, 68-BOOTP client, 80-Hypertext Transfer Protocol (HTTP), 123-Network Time Protocol (NTP), 443-Hypertext Transfer Protocol Secure (HTTPS), 1900-Simple Service Discovery Protocol (SSDP), 5353-mDNS, 49153-ANTLR, 0:1023-well-known ports, 1023:49151-registered ports, and 49151:65535-dynamic (private) ports [158]. The port classification above is ordered so that, for example, a packet with port 53 is not also classified as a well-known port. Section 4.3.1 explains further how port number classes are created.

However, the assumption that standard port numbers will invariably be utilized or

discernible may not hold true in various scenarios, particularly when services employ non-standard ports or when the transport protocol is encrypted. For instance, within the realm of IPv6, nearly any port can be virtually utilized, and the transport protocol may become encrypted through the use of Internet Protocol Security (IPsec). Regrettably, the dataset employed for device identification lacks instances that illustrate this specific context. In our feature extraction code, default values are applied in cases where the data is encrypted or incompatible. For example, in scenarios involving IPv6 virtual ports or the utilization of an encrypted transport protocol, the port class is designated as the initial value: "No port".

MAC/IP addresses are source-destination based identifying features. Identifying features, in this context, refer to attributes that have the potential to cause information leakage. Although they uniquely identify source and destination devices, they do not provide information about device behaviours. For example, two devices with the same behaviour can have different MAC and IP addresses. Because they are not predictable and generalizable, we do not use them in our feature set (though they are used in our aggregation method).

4.3 Model Selection

This section comprises two sub-sections. In the first subsection, Feature Selection 4.3.1, a multi-layer selection process is implemented, involving an initialisation phase based on a feature importance score followed by the application of a genetic algorithm. In the next subsection, Model Selection 4.3.2, the best model is decided by evaluating the performance across six different machine learning algorithms using the selected feature set.

4.3.1 Feature Selection

In the initial phase of feature selection, we discarded insignificant features using a voting mechanism based on feature importance scores. In the task aimed at computing importance scores, we established binary classification for each device (e.g., determining whether a device is a *Fitbit Aria* or not). Feature scores were then computed for each device within this context. The ultimate feature scores were derived by averaging across the 27 devices for each specific feature. We used a feature-importance-based voting

method via the xverse[2] package to eliminate unnecessary features. This method calculates the importance scores of all features for each device using six different techniques, and then uses voting to decide whether to include them (see Figure 4.2 for features and their vote rates). The six scoring techniques used by this method are information value using the weight of evidence, variable importance using Random Forest (RF), recursive feature elimination, variable importance using extra trees classifier, chi-square best variables, and L1-based feature selection. We removed from our feature pool the 26 features that failed to receive votes on any device from any of these six techniques.

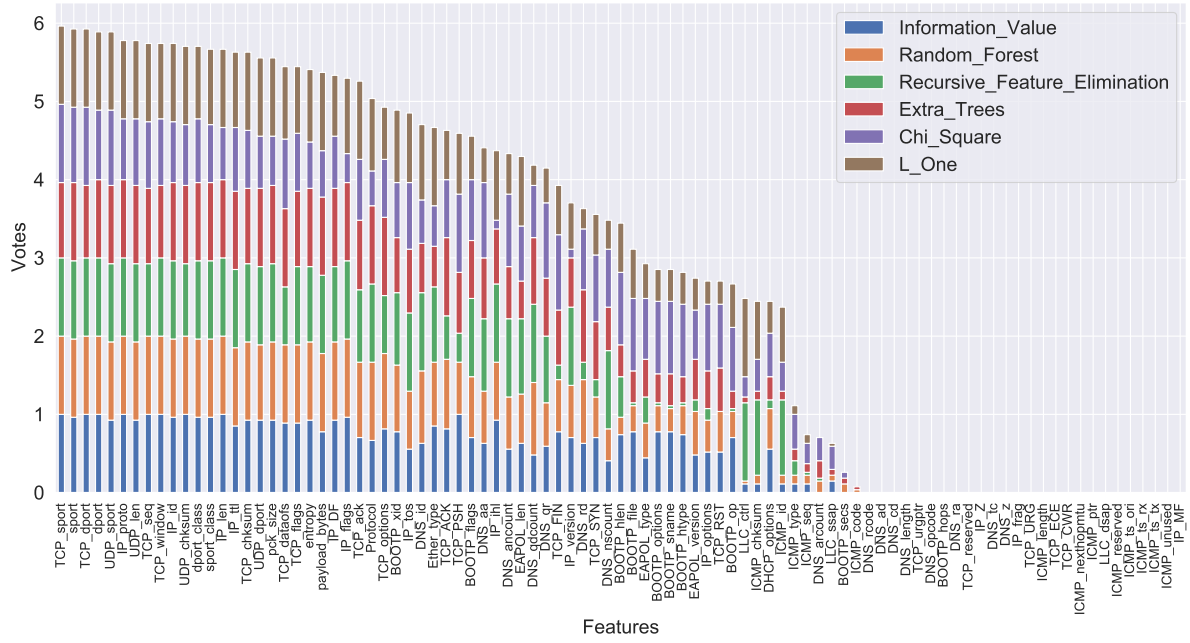


Figure 4.2: Results of feature importance voting on the Aalto dataset. Features rated with six different feature importance methods are listed from the most-to-least voted. The highest ranked features are related to size, payload characteristics and frequently used protocols such as IP, User Datagram Protocol (UDP), TCP, while the lowest ranked features are related to rarely used features of protocols.

A number of the remaining features contain information that is potentially specific to a particular session, and therefore may not be useful for identifying devices more generally. For some features, this is clear, e.g., the initial values of the IP ID, TCP sequence and acknowledgement numbers are randomly assigned, and the next values are consecutive numbers following this initial value; hence, they do not contain information that is useful for DI beyond their current session. For other features, such as TCP port numbers, their generality is less clear. So, to measure whether or not each of these

[2] https://pypi.org/project/xverse/

features is beneficial, we excluded all of them from the feature set and trained an ML model (Decision Tree (DT)) to determine a baseline level of performance. In turn, we then added each of the features to the baseline set and observed whether this improved or impaired the performance.

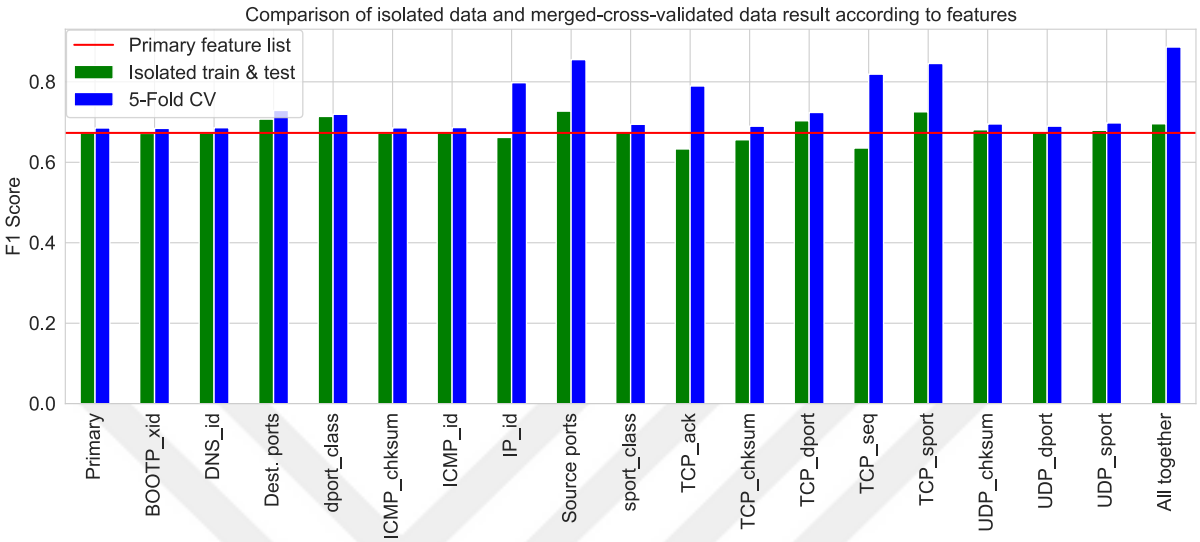


Figure 4.3: Generality of identifying features, measured using isolated test sets and cross-validation. At each step, a identifying feature is incorporated into the list of primary features, encompassing all the non-descriptive features found in the blue, green, and white areas of Table 4.3. It is seen that, due to information leakage from training set to test set, cross-validation considerably over-estimates the value of these features.

The results are shown in Figure 4.3. First of all, it is notable that the results are strongly dependent on whether the models are evaluated using cross-validation or using isolated training and test sets. This is presumably because, when using cross-validation, it is more likely that the training and test data will be sampled from the same session — meaning that session-specific patterns learned in the training set will also generalise to the test set. This may explain why session-specific features were chosen during feature selection, and highlights the danger of using off-the-shelf feature selection algorithms on data of this kind. Using isolated training and test sets, by comparison, it is clear that session-specific features such as the IP ID cause overfitting and impair the generality of models.

On the other hand, Figure 4.3 shows that port-based features can be useful for building generalisable models. For example, consider the port numbers used by one of the devices in the Aalto dataset which contains multiple device instances. Figure 4.4 shows a word cloud built from the data from the two Edimax camera instances. Although

a few port numbers (representing the protocols BOOTP, HTTP, SSDP and mDNS) generalise between the devices, the remaining ports are presumably session-based and do not generalise. To address this, we do not use the raw feature values for port-based features, but rather map them to discrete values representing port classes and protocols (see Section 4.2.3) — in effect, pruning the instance-specific information whilst preserving the behavioural information they provide.

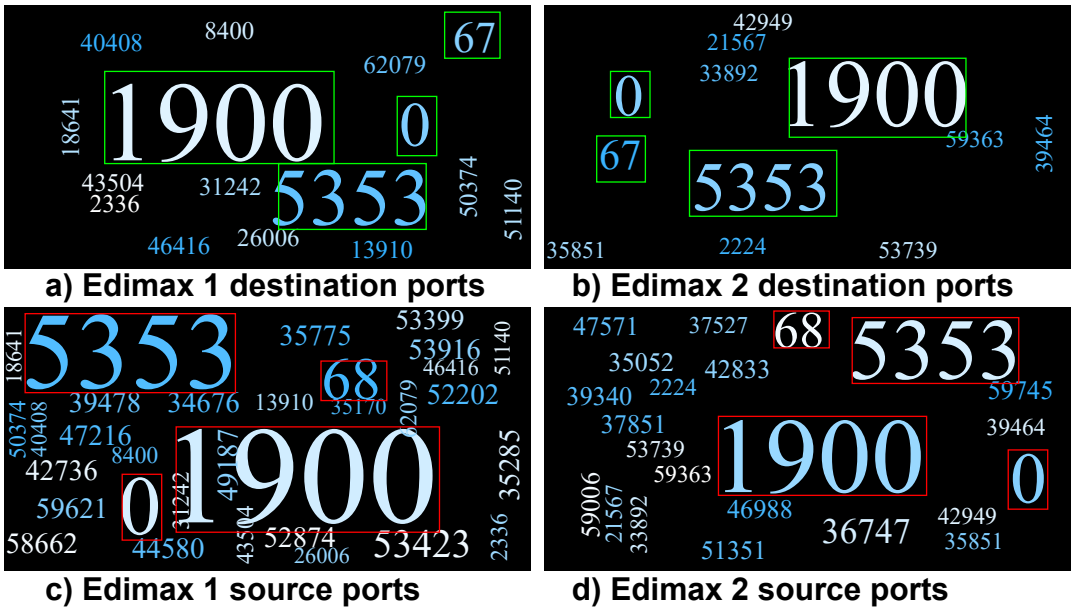


Figure 4.4: Word clouds of port numbers used by two devices of the same type, with digit size showing frequency of use. Red and green boxes, showing port numbers specific to protocols such as SSDP, mDNS, and BOOTP, generalise between device instances. The other port numbers do not generalise.

Finally, after eliminating identifying and redundant features, we used a Genetic Algorithm (GA) to decide on the most appropriate feature set from the remaining 52-member feature pool. Various feature selection methods include [100] filter methods, which evaluate features based on statistical properties such as correlation and mutual information; wrapper methods that evaluate subsets using a specific model and are computationally expensive but often provide better performance; and embedded methods (e.g. LASSO for linear regression) where feature selection is part of the model building process. However, Genetic algorithm (GA) stands out in feature selection due to its capacity to handle a large number of features, handle non-linear relationships, and efficiently search for optimal subsets without the need for extensive examination. The GA uses a wrapper method and thus tests the usefulness of feature sets in the context of a particular

classifier (DT). In this way, we have created a feature subset that detects with higher performance, while decreasing the model complexity by reducing the number of features. Table 4.3 shows (highlighted in green) the final list of features selected by the GA. Refer to Figure 4.5 for the specific structure of the GA used in this study. The algorithm can be summarized as follows [40, 55]:

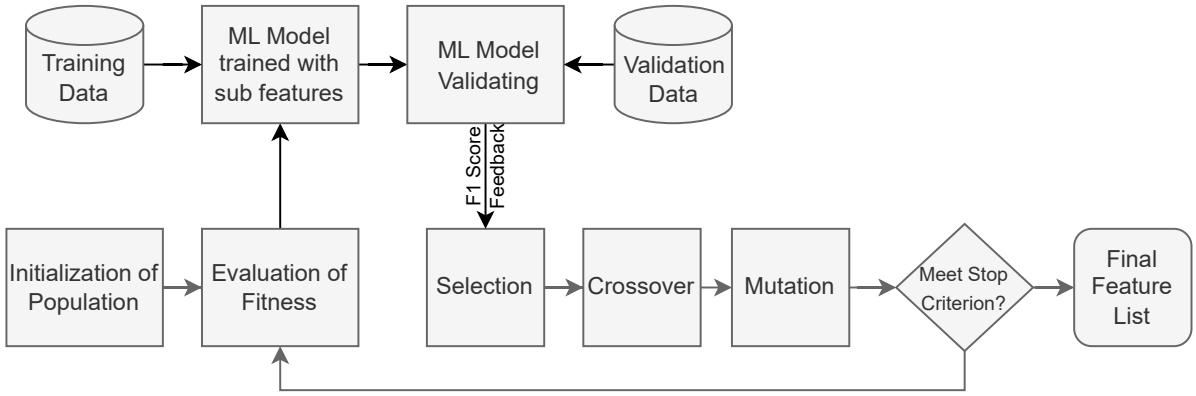


Figure 4.5: Representation flow diagram of genetic algorithm.

Given a set of features representing network behaviour, denoted as A, B, C, D, subsets of features, called chromosomes, are formed by combinations of these individual features (genes). For example, a chromosome can be represented as 1, 1, 0, 1, indicating that traits A, B and D are included in the subset and trait C is excluded. The steps in Figure 4.5 are described as follows.

Initialisation of Population: The initial chromosomes of the genetic algorithm (the population of candidate solutions) are created from the initial feature set with 52 elements. The chromosomes represent a set of features (sub-features) selected for the classification process. In the first run of the algorithm, the chromosomes are created randomly.

Evaluation of Fitness: In this step, the chromosome fitness, i.e., how successful the feature groups are, is determined. Success here means how successful the ML model trained with the training data is on the validation data in terms of F1 score. The higher the F1 score, the better the chromosome fitness.

Selection: At this stage, the best chromosomes that will be the parents of the next generation are selected. Here, the best chromosomes are selected according to the fitness scores in the previous step.

Crossover: At this stage, randomly selected parental chromosomes are joined together to create a new offspring chromosome. Here, a "two-point crossover" is used, in which certain gene fragments from two adjacent parents are exchanged to create offspring chromosomes.

Mutation: In this step, random small changes are made to the chromosomes in order to preserve the diversity of the population and thus prevent premature convergence.

In the genetic algorithm process, the selection, crossover and mutation steps are repeated as many times as the number of generations, aiming to produce new children from the best-performing parents in each generation.

When the features selected by the GA are analysed (see Table 4.3), it can be seen that some features that were previously considered unimportant in the voting process (e.g. ICMP_code) were included. The presence of such features in the final preferred set may seem confusing. However, it is very important to keep in mind that the genetic algorithm does not guarantee a perfect feature set. As a result, the feature set selected by the GA may also include redundant or irrelevant features inherited from the parent feature sets.

Another noteworthy aspect of this feature set is its abundance of IP field features (such as IP_frag, IP_MF, and IP_Z). This prompts the concern that the feature set might lack representativeness for a singular identified device, implying that it may fail to identify the same device when encountered on a different network. However, it's essential to highlight that our validation in Chapter 6, specifically designed for devices on different networks, is independent of IP field. This approach addresses concerns regarding the generalizability and representativeness of the feature set for individual devices.

4.3.2 Algorithm Selection

Based on earlier approaches and surveys of related work [9, 92] which sought to train an ML algorithm to correctly predict a device's type from extracted features, we considered the following seven ML algorithms: RF, k-Nearest Neighbours (KNN), Gradient Boosting (GB), DT, Naïve Bayes (NB), MultiLayer Perceptron (MLP) and Support Vector Machine (SVM). The choice of these algorithms is grounded in their widespread utilization within the literature, as detailed in the literature review chapter (see Chapter 3). We used random search with predefined split cross-validator (as implemented

BOOTP_chaddr	dports	ICMP_nexthopmtu	ICMP_ptr	payload_bytes	DNS_aa
BOOTP_ciaddr	ICMP_chksum	ICMP_ts_ori	ICMP_reservd	EAPOL_version	DNS_ancount
BOOTP_giaddr	ICMP_id	ICMP_ts_rx	DNS_tc	TCP_ACK	DNS_arcount
BOOTP_siaddr	IP_id	ICMP_ts_tx	DNS_z	TCP_dataofs	DNS_nscount
BOOTP_yiaddr	sports	ICMP_unused	ICMP_length	TCP_FIN	EAPOL_len
DNS_an	sport23	IP_frag	dport_class	TCP_window	ICMP_seq
DNS_ar	TCP_ack	IP_MF	EAPOL_type	BOOTP_file	ICMP_type
DNS_ns	TCP_chksum	IP_Z	pck_size	BOOTP_flags	IP_proto
DNS_qd	TCP_dport	LLC_dsap	UDP_len	BOOTP_hlen	IP_version
Ether_dst	TCP_seq	TCP_CWR	Ether_type	BOOTP_options	LLC_ssap
Ether_src	TCP_sport	TCP_ECE	ICMP_code	BOOTP_sname	Protocol
ICMP_addr_mask	UDP_chksum	TCP_reserved	IP_DF	DHCP_options	sport_class
ICMP_gw	UDP_dport	TCP_URG	IP_flags	DNS_qdcount	TCP_flags
IP_chksum	UDP_sport	TCP_urgptr	IP_ihl	DNS_qr	TCP_options
IP_dst	BOOTP_xid	BOOTP_hops	IP_len	DNS_rd	TCP_PSH
IP_src	DNS_id	DNS_ad	IP_options	Payload entropy	TCP_RST
MAC_dst	dport23	DNS_cd	IP_tos	BOOTP_htype	TCP_SYN
MAC_src		DNS_length	IP_ttl	BOOTP_op	DNS_ra
ts (timestamp)		DNS_opcode	LLC_ctrl	BOOTP_secs	DNS_rcode

Source-destination based identifying features
Session based identifying features
Features removed as a result of the voting process
Features selected by the genetic algorithm

Table 4.3: Results of feature selection. Red and orange features were removed due to a lack of generality. Blue features were removed as a result of feature importance voting. Of the remaining features, those in green were identified by the GA as being most useful for discriminating between devices. They include size, payload and fields related to common protocols.

by scikit-learn) to find suitable hyperparameters for each algorithm. The utilization of a predefined split cross-validator in this context is driven by using the validation dataset created beforehand. Using these algorithms, we trained multi-class classifiers to discriminate the 27 classes from the Aalto dataset, training each model 100 times to measure its stability. See Table 4.4 for the results.

Table 4.4: Comparison of ML algorithms with average and standard deviation (SD) of 100 repeats on the Aalto dataset. t is time, in seconds. Only individual results, no grouping.

ML	Accuracy	Precision	Recall	F1 score	Train-t	Test-t
DT	0.705±0.001	0.774±0.003	0.706±0.001	0.727±0.001	0.128	0.004
GB	0.699±0.001	0.789±0.003	0.693±0.001	0.725±0.001	918.3	8.312
KNN	0.705±0.000	0.752±0.000	0.705±0.000	0.718±0.000	0.005	20.20
NB	0.617±0.000	0.584±0.000	0.629±0.000	0.559±0.000	0.433	0.032
RF	0.708±0.001	0.768±0.004	0.708±0.001	0.727±0.002	3.742	0.333
SVM	0.680±0.000	0.697±0.000	0.634±0.000	0.649±0.000	101.3	64.80
MLP	0.573±0.014	0.580±0.035	0.513±0.019	0.515±0.026	8173	2.075

It can be seen that RF, DT and GB are the best performing algorithms, in terms of DI. The fastest in terms of inference time are DT and NB, though the accuracy of NB is very low. Although KNN and GB perform relatively well in terms of accuracy, they are not practical to use due to their slowness. The SVM and MLP algorithms are also not a reasonable option in terms of speed and accuracy. KNN, GB, MLP and SVM are particularly disadvantageous due to their very large inference times. In a real-time device detection system that will operate in a millisecond-level processing environment such as network traffic, inference time in seconds are not acceptable. Based on these observations, we use DT in the remainder of this work, since it offers the best balance between speed and accuracy.

4.4 Performance Evaluation

Based on our model selection we present our results here for the three variations of our method: individual, aggregated, and mixed. First, we looked at the relationship between group size and performance within the context of the aggregation algorithm. From Figure 4.6, there is a positive relationship, showing that larger group sizes are generally more effective. However, many of the IoT devices communicate infrequently, so large group sizes would be impractical in some cases. For this reason, we selected a group size of 13, which is approximately the point where performance starts to plateau.

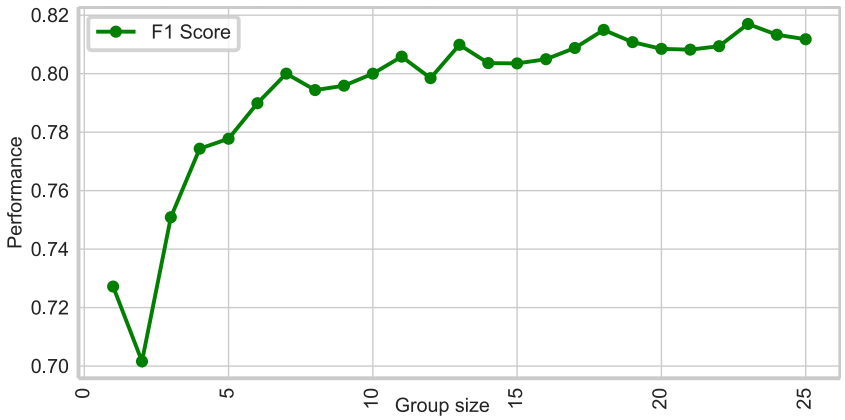


Figure 4.6: Relationship between aggregation group size and performance in Aalto dataset. The benefit of increasing group size appears to plateau around $g = 13$.

Table 4.5 shows overall results for the three datasets. Using the individual packet approach, an F1 score of 73% was achieved in the Aalto dataset, 83% in the UNSWM, and 91% in the LSIF dataset. However, a significant increase in the ability to correctly

identify devices of both datasets was observed by using aggregation. When the running times of the aggregation algorithm are analysed, it shows that this period is significantly longer in the LSIF dataset compared to the other two datasets. The primary factor contributing to this disparity is the significant difference in the total number of samples between the LSIF dataset and the other two datasets. In this context, we can say that the number of samples and the running time increase in parallel (The number of samples can be found in Table 4.6, 4.7, and 4.8). The overall F1 score increased to 81% on the Aalto dataset, to approximately 94% on the UNSWM, and 93% on the LSIF dataset. Since feature selection was done using only the Aalto dataset, it is very encouraging to see such a high level of discrimination on the UNSWM and LSIF datasets. This is a good indication that the selected feature set will generalise well to other IoT environments.

Table 4.5: Results and their SD obtained using Individual, Aggregated and Mixed approaches on Aalto, UNSWM and LSIF datasets. In all cases, training and test data from different sessions of the specified dataset were used. t is time in seconds. Alg-t indicates the running time of the aggregation algorithm.

Method	Dataset	Accuracy	F1 score	Test-t	Alg-t
Individual	Aalto	0.705±0.001	0.727±0.001	0.004	0.000
	UNSWM	0.852±0.011	0.833±0.013	0.0344	0.000
	LSIF	0.989±0.000	0.907±0.006	0.499	0.000
Aggregated	Aalto	0.745±0.011	0.809±0.005	0.007	0.164
	UNSWM	0.943±0.012	0.937±0.016	0.049	0.976
	LSIF	0.994±0.000	0.931±0.001	0.488	27.30
Mixed	Aalto	0.833±0.002	0.861±0.004	0.008	0.216
	UNSWM	0.941±0.014	0.934±0.019	0.047	1.023
	LSIF	0.994±0.000	0.931±0.001	0.483	27.79

Table 4.6 shows the average discrimination performance of DT models at the device level for the Aalto Dataset. As Table 4.6 also shows, the dataset is highly imbalanced, meaning that we need to use a metric that is relatively insensitive to class size imbalance in order to give a meaningful picture of the model's performance at the device level – hence our use of the F1 score. Many works have used accuracy for this, which is an inappropriate metric for imbalanced datasets.

A potential solution for addressing the class imbalance issue is the utilization of data augmentation techniques such as resampling, Synthetic Minority Oversampling TEchnique (SMOTE), or Generative Adversarial Network (GAN). In the earlier version of IoTDevID [116], we tackled this imbalance through resampling and SMOTE. However,

in the subsequent/updated version of IoTDevID [118], we have excluded this step from our methodology, as it was determined to have no evident impact on overall success in our specific context. It's worth noting that the efficacy of these augmentation methods can vary based on the dataset and the specific characteristics of the problem at hand. While we opted to omit these techniques in our current approach, future research or alternative methods may find success with these augmentation strategies in different contexts or datasets.

When the device-based results of the Aalto dataset are examined from Table 4.6, we see that the aggregation algorithm contributes positively to almost all devices, while negatively affecting only four devices. This is because the pairs that make up these four devices suffer from the transfer problem (they share the same MAC address, see Table 4.1). To deal with this, we used our mixed method by adding an exception to our aggregation algorithm. Table 4.5 shows that with the application of the mixed approach, the overall F1 score, which was 81% in the Aalto dataset, increased to 86%. Since there is no transfer problem in the UNSWM and LSIF datasets, this approach does not impact its results significantly — see Table 4.7 and Table 4.8.

It is notable that the performance on the Aalto dataset is significantly lower than the UNSWM and LSIF datasets. This appears to be due to certain device subgroups. A confusion matrix containing only low-performance devices is given in Figure 4.7. In this case, the devices in a subgroup have some similarities: they are either similar purpose devices manufactured by the same companies (e.g., yellow, blue and orange groups in Figure 4.7) or different models of the same device (e.g., red and green groups in Figure 4.7). It does not seem possible to perfectly separate these devices according to their behaviour, at least when observed at the network level. However, it is likely that these devices use very similar hardware and software, and so exhibit similar behaviour, as well as similarities in vulnerabilities and their prevention [158]. Consequently, from a DI perspective, it is plausible to consider these devices under a single label. When doing this, the accuracy for the Aalto dataset increases from 73% to 88% for the individual method, and from 86% to 97% for the mixed method.

In the UNSWM and LSIF datasets, while success rates were comparatively higher than those observed in the Aalto dataset, challenges arose in accurately identifying certain devices. Notably, in the LSIF dataset, the *tp-link LightBulb* and *tp-link SmartPlug* exhibited significant identification issues, frequently being mislabelled as each other. Distinguishing between devices of the same brand that perform similar functions posed a considerable challenge. This intricacy was thoroughly examined within the Aalto

Table 4.6: Device proportions over the entire dataset, F1 scores per device according to different approaches (Aalto Dataset, average of 100 repetitions).

	Packet statistics		Packet discrimination		
Device name	Packets	Percent	Individual	Aggregated	Mixed
Aria	441	0.420	0.932	1.000	1.000
D-LinkCam	6244	5.940	0.891	1.000	0.988
D-LinkDayCam	1063	1.010	0.864	1.000	1.000
D-LinkDoorSensor	1892	1.800	0.762	0.057	0.788
D-LinkHomeHub	8595	8.180	0.681	0.797	0.777
D-LinkSensor	6549	6.230	0.382	0.644	0.626
D-LinkSiren	6186	5.890	0.367	0.640	0.631
D-LinkSwitch	6519	6.200	0.665	0.969	0.964
D-LinkWaterSensor	6435	6.120	0.392	0.624	0.622
EdimaxCam	831	0.790	0.872	1.000	1.000
EdimaxPlug1101W	1160	1.100	0.601	0.827	0.824
EdimaxPlug2101W	1010	0.960	0.453	0.708	0.699
EdnetCam	408	0.390	0.833	1.000	1.000
EdnetGateway	683	0.650	0.908	1.000	1.000
HomeMaticPlug	611	0.580	1.000	1.000	1.000
HueBridge	13936	13.260	0.810	0.217	0.815
HueSwitch	18448	17.560	0.891	0.720	0.891
IKettle2	145	0.140	0.727	1.000	1.000
Lightify	4149	3.950	0.977	1.000	1.000
MAXGateway	567	0.540	0.964	1.000	1.000
SmarterCoffee	149	0.140	0.727	0.983	0.981
TP-LinkPlugHS100	667	0.630	0.693	0.748	0.746
TP-LinkPlugHS110	636	0.610	0.429	0.336	0.328
WeMoInsightSwitch	5962	5.670	0.667	0.874	0.866
WeMoLink	6625	6.300	0.638	0.929	0.924
WeMoSwitch	4477	4.260	0.511	0.769	0.768
Withings	688	0.650	1.000	1.000	1.000

Table 4.7: Device proportions over the entire dataset, F1 scores per device according to different approaches (UNSWM Dataset, average 100 repetitions).

	Packet statistics		Packet discrimination		
Device name	Packets	Percent	Individual	Aggregated	Mixed
Amazon Echo	10000	2.831	0.887	0.984	0.986
AugustDoorbellCam	10000	2.831	0.700	0.960	0.953
AwairAirQualityMon	10000	2.831	0.949	0.993	0.993
Belkin Camera	10000	2.831	0.890	1.000	1.000
BelkinWeMoSwitch	10000	2.831	0.803	0.987	0.988
BelkinWeMoSensor	10000	2.831	0.774	0.987	0.987
BlipcareBPMeter	119	0.034	0.848	1.000	1.000
Canary Camera	10000	2.831	0.774	0.922	0.919
Dropcam	10000	2.831	0.998	1.000	1.000
GoogleChromecast	10000	2.831	0.452	0.349	0.377
HPPrinter	10000	2.831	0.459	0.712	0.710
HelloBarbie	164	0.046	0.682	1.000	1.000
InsteonCam	10000	2.831	0.954	0.997	0.997
LightLiFXSmartBulb	10000	2.831	0.980	1.000	1.000
NEST-PSmokeAlarm	7063	1.999	0.934	1.000	1.000
Nest Dropcam	10000	2.831	0.966	1.000	1.000
NetatmoWelcome	10000	2.831	0.929	0.994	0.995
NetatmoWeatherSt	10000	2.831	0.897	1.000	1.000
Non-IoT	67295	19.050	0.874	0.928	0.930
PIX-STAR-Pframe	10000	2.831	0.594	0.932	0.872
PhillipHLightbulb	10000	2.831	0.972	1.000	1.000
RingDoorBell	10000	2.831	0.852	0.998	0.996
SamsungSCam	10000	2.831	0.685	0.897	0.894
SmartThings	10000	2.831	0.995	1.000	1.000
TP-LinkCloudCam	10000	2.831	0.979	1.000	1.000
TP-Link Smart plug	10000	2.831	0.875	1.000	1.000
TP-LinkRouter	10000	2.831	0.889	0.966	0.959
TribySpeaker	10000	2.831	0.915	0.990	1.000
WithingsSlpSensor	10000	2.831	0.384	0.418	0.388
WithingsBabyMon	10000	2.831	0.984	1.000	1.000
WithingsSScale	8279	2.344	0.973	1.000	1.000
iHome	10000	2.831	0.958	1.000	1.000
unknownIoT	340	0.096	0.682	0.894	0.894

Table 4.8: Device proportions over the entire dataset, F1 scores per device according to different approaches (LSIF Dataset, average of 100 repetitions).

	Packet statistics		Packet discrimination		
Device name	Packets	Percent	Individual	Aggregated	Mixed
Chime_Doorbell	4935	0.311	0.944	1.000	1.000
D-Link_Cam936L	24591	1.547	0.998	1.000	1.000
Gosuna_LightBulb	115734	7.282	0.999	1.000	1.000
Gosuna_Socket	115671	7.278	0.999	1.000	1.000
Goumia_Coffemaker	11629	0.732	0.887	0.996	0.996
LaCrosse_AlarmClock	7246	0.456	0.982	1.000	1.000
Lumiman_Bulb600	114867	7.228	0.999	1.000	1.000
Lumiman_Bulb900	115451	7.264	0.999	1.000	1.000
Lumiman_SmartPlug	115721	7.281	0.995	1.000	1.000
Minger_LightStrip	14730	0.927	0.897	0.997	0.997
Ocean_Radio	77816	4.896	1.000	1.000	1.000
Renpho_SmartPlug	114858	7.227	0.999	1.000	1.000
Ring_Doorbell	1559	0.098	0.836	1.000	1.000
Smart_Lamp	115523	7.269	0.999	1.000	1.000
Smart_LightStrip	114783	7.222	0.999	1.000	1.000
Tenvis_Cam	7426	0.467	0.713	0.825	0.825
Unknown-Device	200919	12.642	0.999	1.000	1.000
Wans_Cam	179858	11.317	0.970	0.978	0.978
Wemo_SmartPlug	1255	0.079	0.950	1.000	1.000
itTiot_Cam	17764	1.118	0.710	0.795	0.795
oossxx_SmartPlug	115299	7.255	0.999	1.000	1.000
tp-link_LightBulb	863	0.054	0.625	0.696	0.697
tp-link_SmartPlug	770	0.048	0.369	0.120	0.119

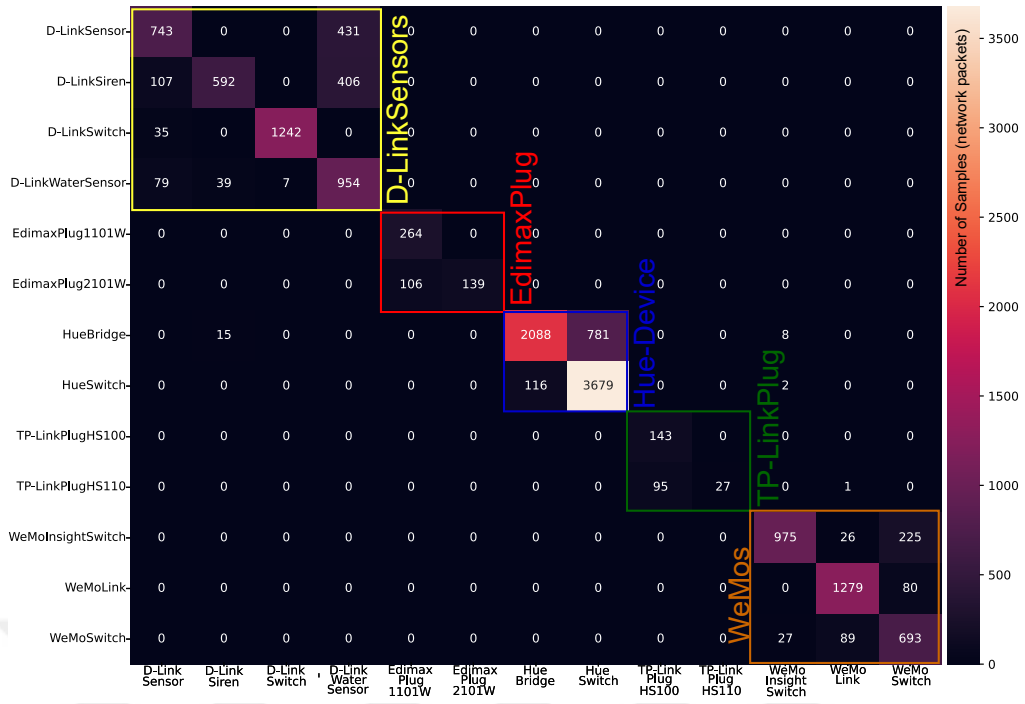


Figure 4.7: Confusion matrix of devices with poor discrimination. The coloured groups show this is mostly due to the difficulty of discriminating between similar devices from the same manufacturer.

database, renowned for its diverse array of such devices.

Examining the UNSWM dataset, we note a substantial reduction in success for two devices. In the case of the Chrome device, a majority of its samples were erroneously categorized as Non-IoT. Non-IoT serves as a comprehensive label encompassing multiple devices. A closer inspection revealed that packages attributed to this device were misclassified as *Android Phone*. *Chromecast*, a compact Google-produced device, endeavours to enhance non-smart televisions by enabling seamless communication with smartphones, tablets, or computers via WiFi, facilitating the streaming of videos, music, and other content.

Given its absence of a dedicated interface, *Chromecast* is managed through an application installed on a secondary device, such as a mobile phone. We posit that the challenge in accurately identifying the *Chromecast* device stems from an undue reliance on another device during its operation.

Another problematic device is the *Withings Aura Smart Sleep Sensor*, experiencing suboptimal performance due to its data being labeled as *HP Printer*. A detailed analysis of the data indicated that approximately half of the anomalous data from the

HP Printer device was IPv6-based. Regrettably, our feature set lacks functionality for feature extraction in IPv6. Consequently, we assert that this issue with the HP Printer device adversely impacts the performance of *the Withings Aura Smart Sleep Sensor*.

4.5 Comparison with Previous Work

Using Table 3.1, we can compare the overall results of our study with previously published results. While, at first glance, our study does not appear to yield higher numeric results than existing work, there are a number of reasons why this is not a fair comparison. First, many approaches use overly specific features that are unlikely to generalise to unseen datasets. As our analysis in Section 4.3.1 showed, this may be a result of information leaking from the test set into the training set, something that is easy to do unintentionally. Figure 4.3 further illustrates that the incorporation of identifying features, although yielding high performance in specific cases, lacks generalizability, as these features may not sustain the same level of effectiveness in an isolated test dataset. Second, there is the widespread use of accuracy figures with imbalanced datasets; since IoT device datasets inherently suffer from unbalanced distributions, using accuracy as the sole evaluation criterion can be highly misleading. Third, there is inconsistency amongst the datasets used, with some studies reporting results for only partial datasets. We summarize these points in Section 3.2.6 and Table 3.3. In our work, by comparison, we were careful to filter out dataset-specific features, to prevent information leakage, to use appropriate metrics, and to report full results. We also explicitly used two more datasets to measure the generality of the approach. Consequently, we believe that our results are unbiased and more likely to indicate success on unseen data than those reported in previous studies.

A key contribution of our work is the determination of a feature set that provides good discrimination across datasets, and is therefore likely to provide a good basis for DI more generally. To give more insight into this, and to offer a more meaningful comparison against previous approaches, we compared this feature set against the feature sets used in IoTSense [30] and IoT Sentinel [158].We applied all three of our approaches (individual, aggregated, and mixed) using all three feature sets and the three (Aalto UNSWM and LSIF) datasets. Table 4.9 shows that, in two cases, Aalto and UNSWM, our feature set resulted in significantly better device discrimination than the previously published feature sets. In the LSIF dataset, IoTDevID is very close to IoTSense, but slightly ahead. Overall, the results show that the IoTDevID feature set outperforms the other

methods on all the datasets we used. Although the IoTSense method is very close on the LSIF dataset, this success cannot be generalised to other datasets.

Table 4.9: Comparison of feature sets on the Aalto, UNSWM, and LSIF datasets.

Data		Method	Accuracy	F1 score	Test-t	Alg-t
Individual	Aalto	IoTDevID	0.705±0.001	0.727±0.071	0.004	N/A
		IoTSense	0.639±0.000	0.561±0.001	0.006	N/A
		IoTSentinel	0.700±0.001	0.602±0.002	0.008	N/A
	UNSWM	IoTDevID	0.852±0.011	0.833±0.013	0.044	N/A
		IoTSense	0.801±0.009	0.766±0.011	0.027	N/A
		IoTSentinel	0.634±0.011	0.572±0.014	0.028	N/A
	LSIF	IoTDevID	0.989±0.000	0.907±0.006	0.499	N/A
		IoTSense	0.986±0.000	0.885±0.005	0.437	N/A
		IoTSentinel	0.395±0.000	0.417±0.000	0.365	N/A
Aggregated	Aalto	IoTDevID	0.745±0.011	0.809±0.005	0.007	0.164
		IoTSense	0.671±0.003	0.657±0.004	0.007	0.142
		IoTSentinel	0.671±0.003	0.639±0.005	0.007	0.136
	UNSWM	IoTDevID	0.943±0.012	0.937±0.016	0.049	0.976
		IoTSense	0.908±0.011	0.898±0.015	0.028	1.169
		IoTSentinel	0.779±0.013	0.728±0.016	0.028	1.159
	LSIF	IoTDevID	0.994±0.000	0.931±0.001	0.488	27.30
		IoTSense	0.993±0.011	0.930±0.001	0.439	26.86
		IoTSentinel	0.410±0.000	0.469±0.001	0.364	26.93
Mixed	Aalto	IoTDevID	0.833±0.002	0.861±0.004	0.008	0.216
		IoTSense	0.748±0.004	0.701±0.005	0.007	0.174
		IoTSentinel	0.778±0.010	0.691±0.008	0.006	0.165
	UNSWM	IoTDevID	0.941±0.014	0.934±0.019	0.047	1.023
		IoTSense	0.908±0.012	0.894±0.016	0.028	1.216
		IoTSentinel	0.774±0.017	0.722±0.021	0.029	1.192
	LSIF	IoTDevID	0.994±0.000	0.931±0.001	0.483	27.79
		IoTSense	0.993±0.000	0.930±0.001	0.438	27.28
		IoTSentinel	0.410±0.000	0.469±0.001	0.357	25.95

4.6 Limitations

In our study, we used all the IoT DI datasets that were publicly available at the time (see Section 4.2.1). However, IoT technologies are a rapidly developing field and new

data is likely to have been released even as we continued our work. We encourage other researchers to use other datasets to further test the generalizability of our results and we provide scripts to make this possible. We used a broad selection of commonly-used ML algorithms (see Section 4.3.2), though this was not exhaustive, and there may be other algorithms that might perform better. However, we have done some preliminary research using deep neural networks (see Appendix A.2), and so far our results suggest that more traditional ML techniques work better on this particular problem. While this study employed pre-existing machine learning models, future research could explore the development of novel methods tailored to the device identification task. Such specialized approaches could potentially enhance the identification accuracy and robustness of device classification schemes.

Although IoTDevID identifies devices well, it does not provide solutions for detecting vulnerable devices and taking necessary precautions about these devices. A Software-Defined Networking (SDN) based network management solution may be useful for this process. An SDN-based method of auditing device vulnerabilities using the CVE database [45] has already been proposed by IoTSentinel [158]. But relying solely on the CVE database poses a potential risk for auditing IoT devices for vulnerabilities and determining authorization levels. We can list them as follows:

- Incomplete CVE Database: The CVE database may not comprehensively cover all IoT device vulnerabilities, especially for newer or niche devices. This can lead to undetected vulnerabilities and potential security breaches.
- Delayed Vulnerability Updates: The CVE database may not be updated immediately with newly discovered vulnerabilities, leaving devices vulnerable during the lag time.
- False Positives: The CVE database may contain inaccurate or outdated information, leading to false positives that unnecessarily restrict or quarantine devices.
- Over-reliance on a Single Source: Relying solely on the CVE database as the sole source of vulnerability information limits the overall security posture.
- Lack of Real-Time Monitoring: The CVE database is a static source of information and may not provide real-time insights into evolving vulnerabilities or device behaviour.

In addition to CVE, RFC 9124 standards can also be used in this context. RFC 9124 defines a set of attributes that can be used to identify IoT devices, including device type,

manufacturer, software version, and vulnerabilities. The SDN controller can use these attributes to determine the appropriate authorization level for each device. For example, devices with known vulnerabilities can be quarantined or restricted, while devices with up-to-date software and no known vulnerabilities can be granted unrestricted access to the network.

Furthermore, IoTDevID is currently intended only for use in benign networks, meaning that there is still a need for an attack detection system to protect against attacks. A further consideration is how a DI model would be deployed, trained and maintained in a particular network. Whilst we focus on the performance of multi-class classifiers, i.e., a single model for discriminating between all devices on the network, this may not be optimal in situations where devices are frequently added and removed. A more practical architecture in this case may be an ensemble of single-class classifiers since this would not require retraining of the entire model.

4.7 Conclusions

In this chapter, we present an ML-based method for identifying IoT devices. We can summarise what has been done in this chapter in line with the contributions in Section 1.4. Identifying the IoT devices with a network is important, both for finding and removing rogue devices and for understanding the security vulnerabilities of a network more generally (RQ.1.3, C1.1). A number of previous studies have attempted to use ML in this process. However, we have identified various factors that may limit the generality of their findings, including the use of session-based identifying features, and the use of inappropriate metrics. In this study, we have attempted to go about this process in a rigorous and transparent manner, with the aim of developing a robust method of DI that reliably generalises beyond the data on which it was trained.

A key contribution of this work is the use of multi-stage feature selection to determine a set of generalizable packet-level features that can be used for building robust models (RQ.1.3, C1.3). Notably, we compared this feature set against those used in previous studies and showed that it supports the training of significantly better ML models. We also demonstrate that the feature set generalises well to a dataset that were unseen during feature selection, giving us confidence that it provides a meaningful basis for the broader discrimination of IoT devices (RQ.1.7, C1.4, C1.5).

We also address the problem of detecting non-IP devices (RQ.1.4, C1.2). A limitation of earlier approaches is that they use IP or MAC addresses to merge packets prior to

applying an ML model, which means they are not applicable in situations where a device does not have an IP or MAC address. Instead, we apply ML at the packet level, and then use an aggregation algorithm that takes into account both the packet-level classification and, if available, the IP or MAC address. If the latter is not available, it is still possible to carry out identification based upon individual packets alone.

From an ML-perspective, we found decision trees to offer the best trade-off between predictive performance and inference time, the latter being an important consideration for deploying a model to monitor network traffic in real time. In forthcoming research, we intend to validate the IoTDevID method across various datasets. Chapter 5 will involve independent validation using the CIC IoT Dataset 2022 (CICIoT22) dataset to assess system performance and account for data quality impact on models. In Chapter 6, we'll employ the dataset for "information exposure for consumer IoT devices" (IMC'19) (MonIoTr) dataset to evaluate IoTDevID's efficacy with encrypted and unencrypted network data and to assess model generalizability. The second part of the thesis (see Part II) will focus on developing an attack detection system that complements device identification to identify network attacks. By coupling our device identification system with effective attack detection, we can identify malicious data and create a secure environment for working with benign data, thereby establishing a comprehensive security solution to mitigate vulnerabilities, prevent attacks, and operate in real-world settings.

Chapter 5

Externally Validating the Device Identification Methodology

In this section, we will analyse the effect of data quality for IoT device identification using data collected from the CIC IoT Dataset 2022 (CICIoT22) dataset under different conditions (idle and in-use). Thus, we investigate the effect of data collected in different situations on model performance. In addition, we will validate our methodology using a new, independent dataset that contains a large number of devices and hosts non-Internet Protocol (IP) devices. In this validation process, the feature selection and algorithm selection processes performed in IoTDevID are not repeated. By applying the features and machine learning algorithm selected there to the new dataset, the operation of the IoTDevID method (feature set and aggregation algorithm) is verified. The following publication was used in the preparation of this chapter.

- Kostas, K., Just, M., & Lones, M. A. (2023). Externally validating the IoTDevID device identification methodology using the CICIoT22 Dataset. arXiv preprint arXiv:2307.08679.:github.com/kahramankostas/iotdevid-cic

5.1 Overview

In this chapter, we validate our IoTDevID solution by applying it to a new dataset, the CICIoT22. This dataset provides an opportunity to test the robustness and generalisability of core components of our solution, namely its feature set and aggregation algorithm. CICIoT22 contains more devices than other prominent datasets used in our original evaluation of IoTDevID: Aalto University - IoT Devices Captures (Aalto) [148, 158] UNSW

Merged (UNSWM) [81, 214], and Locality-Sensitive IoT Fingerprinting (LSIF) [37]. It also has non-IP devices (which UNSWM and LSIF do not) and data collected during use (which Aalto does not). It further contains additional contextual data, such as whether a device is *idle* or *active*, as well as different data usage scenarios. This richer dataset will allow us to further test the generalisability of IoTDevID, and it also allows us to provide some insight into the usefulness of such data for creating more generalisable and robust models. In order to enhance transparency and ensure reproducibility, we have publicly shared our dataset, and scripts[1].

5.2 Case Study

In this section, the CICIoT22 dataset is examined and its features are analysed in depth. The individual packets and aggregation methods used for classification are explained. Finally, how feature extraction and labelling are performed is described.

5.2.1 CICIoT22 dataset

Data was collected during 6 different device states. These states can be summarised as follows [48]. In the **Power** state, each device is isolated from other devices and rebooted and the network packets related to this device are collected. In the **Interactions** state, the device is interacted with by buttons, applications or voice commands and the network packets generated during this process are collected. In **Scenarios**, the network data of these devices are collected in scenarios such as entering the house, leaving the house, unauthorised entry to the house at night and day or user error. In the **Attack** state, data is collected by applying Flood attacks and RTSP Brute Force attacks to the devices. The **idle** state consists of recording every 8-hour period for 30 days in the evening hours when the devices are working but not actively used. The **Active** state contains the data of the devices being used during the day for 30 days. This data is generated by people entering the lab and using the devices.

Some important points about the dataset: In this study, the most important sections for benign device behaviour are *idle* and *active* as these states cover most normal usage and provide a wide range of data about all devices. Although it is stated in the paper [48] that 60 devices were used in this process, according to our experiments and the information provided in the dataset[2], the data for these states covers 40 devices.

[1] Materials available at: github.com/kahramankostas/IoTDevID-CIC
[2] www.unb.ca/cic/datasets/iotdataset-2022.html

These 40 devices are only LAN/Wired or WIFI devices, they do not include Zigbee and Z-Wave devices. Zigbee and Z-Wave devices have data isolated from other devices in the *power* and *interaction* stages. However, these data are both very limited and do not contain normal usage data. Also, the data of the Z-Wave devices is not in pcap format.

5.2.2 Feature Extraction and Labelling

We followed the feature extraction procedure described in Chapter 4 for methods, features, and tools. However, given the dataset's unique characteristics, we made slight adjustments to the labelling. This process can be summarized as follows. Python[3], Scapy[4], and Wireshark[5] were used for feature extraction from packet capture (pcap) files. Only individual packet-based features are used for feature extraction. Many of these features are derived from packet headers, but there are also payload-based features such as payload entropy and payload bytes. Although the feature extraction system created about 100 features[6] in total, only the features[7] selected during the feature selection phase of the IoTDevID study were used in these experiments.

For IP devices, labelling was performed using the list of device names/Media Access Control (MAC) address pairs in the dataset. In each fingerprint (feature set representing a packet) extracted, the source MAC address part was replaced with the given name. The MAC addresses not given in this list (5 MAC addresses that we believe belong to the hub, switch or the computer where the data is collected) were ignored. Due to the non-meaningful nature of MAC addresses for non-IP devices, the dataset creators isolated the network packets of these devices from the rest. In labelling non-IP devices, we have utilised the isolation information provided by this dataset arrangement.

In the CICIoT22 dataset, each of the pcap files we use for feature extraction contains network traffic recorded on a day, and is named with the date it was recorded. For example, data recorded on 24.11.2021 is labelled A211124 if *active* and I211124 if *idle*. In this context, 30 *idle* and 24 [8] *active* sessions were recorded. As a preliminary study, we aimed to test the performance of models trained on data from each session by comparing

[3] https://www.python.org/
[4] https://scapy.net/
[5] https://www.wireshark.org/
[6] Complete feature list: github.com/kahramankostas/IoTDevID-CIC/blob/main/featurelist.md
[7] Selected features are: pck_size, Ether_type, LLC_ctrl, EAPOL_version, EAPOL_type, IP_ihl, IP_tos, IP_len, IP_flags, IP_DF, IP_ttl, IP_options, ICMP_code, TCP_dataofs, TCP_FIN, TCP_ACK, TCP_window, UDP_len, DHCP_options, BOOTP_hlen, BOOTP_flags, BOOTP_sname, BOOTP_file, BOOTP_options, DNS_qr, DNS_rd, DNS_qdcount, dport_class, payload_bytes, entropy
[8] Although the paper [48] states 30 *active* sessions, there are only 24 sessions in the dataset.

them with each other. In order to compare the sessions with each other, they should contain the same devices. Unfortunately, data was not collected from every device in every session, and in some sessions, some devices did not generate any data at all. Table 5.6 shows how much data was generated by each device in each session in terms of network packets. Therefore, we only compare sessions that contain the same devices with each other. For this comparison, we create a session ID. In this ID, each device is represented by a binary digit. If the session has that device, it is indicated with 1, if not, it is indicated with 0. For example, if Session1 contains devices A and C, but not device B, then the ID number is 101(ABC). Session1 can be compared to other sessions with the same ID number without any problem. In this context, we have created a 32-digit ID for each session according to 32 device classes in total. There are 40 physical devices; however, some of these devices are in the same label because they are identical in brand and model.

5.3 Results

We first consider data quality, in terms of its ability to support the training of device identification models, by training and testing ML models using different sessions within the CICIoT22 dataset. Based on the findings of this analysis, appropriate sessions are then merged to produce a dataset that is both representative of the data diversity and which can be reliably used to train device identification models.

5.3.1 Analysis of Data Quality

We begin by training models using data from each session and then testing them on other sessions with the same ID. 1036 session pairs were created for this purpose, with the first session used for training and the second for testing. These pairs were divided into *active* and *idle* categories, resulting in four training vs testing possibilities: Active vs Active (AA) , Active vs Idle (AI) , Idle vs Active (IA) , Idle vs Idle (II).

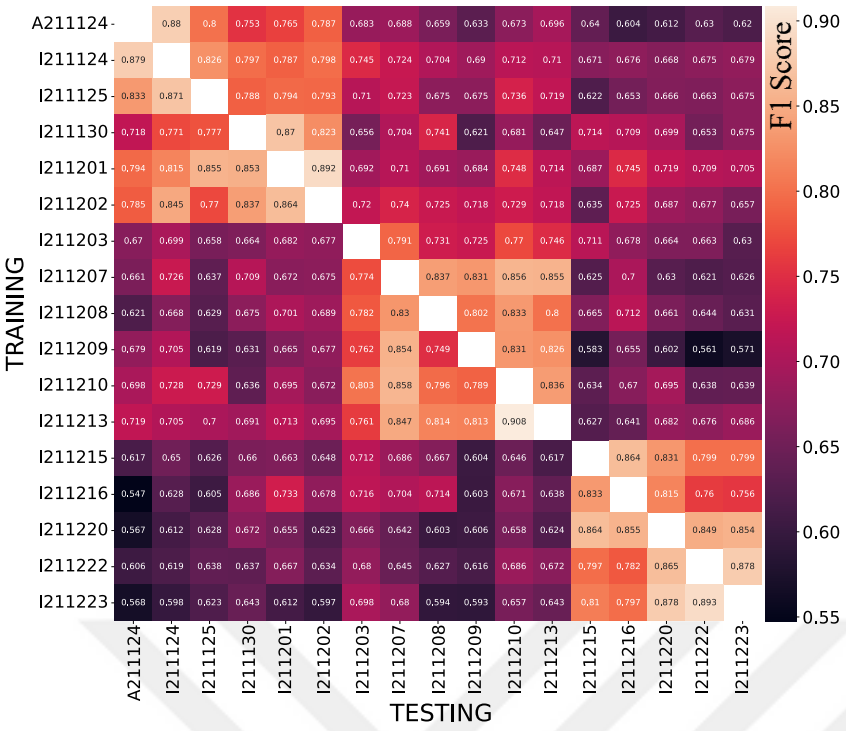
Table 5.1 presents the average results for 1036 session pairs categorised into four conditions. When we focus on individual results, the highest F1 score is achieved in condition II (72%), closely followed by AI (71.4%). The lowest scores are observed in the AA (70.2%) and IA (67.9%) conditions, respectively. When we apply the packet aggregation algorithm, it can be seen that the results reflect those from individual packets, but with an improvement of approximately 5-7 points in each case. Notably, the

utilisation of *idle* data for testing purposes yields higher performance compared to the use of *active* data. This can be attributed to the broader range of data available in *active* scenarios, while *idle* data lacks this diversity. Consequently, a model trained with *active* data exhibits higher success when tested on *idle* data, whereas a model trained with *idle* data shows lower performance when tested on *active* data.

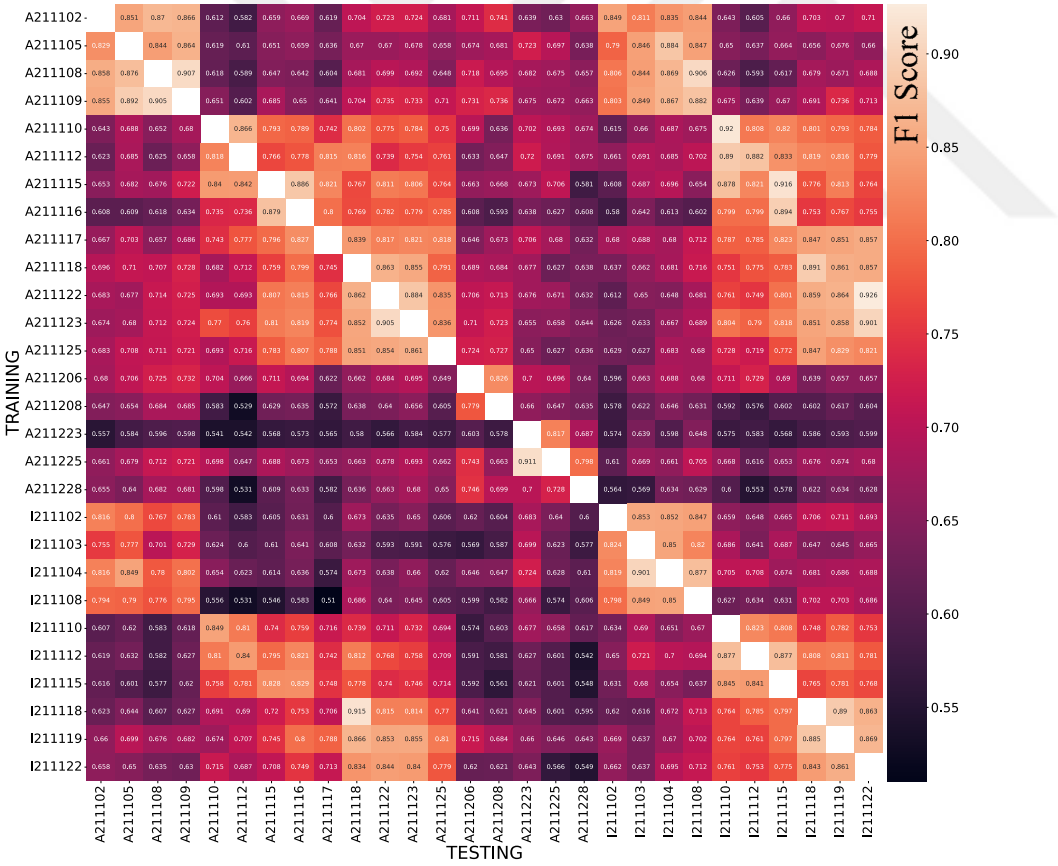
Table 5.1: Results for four device conditions using Decision Tree (DT). Results consisting of accuracy and F1 scores (with standard deviations) are the means of all session comparisons. t specifies the time in seconds. In the data column, the first letter refers to the data used for training (A=active, I=idle), and the second letter refers to the data used for testing..

	Data	Accuracy	F1-Score	Train-t	Test-t	Alg-t
Individual	AA	0.756±0.001	0.702±0.003	1.620	0.185	0
	AI	0.760±0.002	0.714±0.004	1.528	0.191	0
	IA	0.686±0.002	0.679±0.004	1.545	0.171	0
	II	0.768±0.003	0.720±0.005	1.516	0.194	0
Aggregated	AA	0.810±0.001	0.773±0.004	1.570	0.184	7.487
	AI	0.815±0.002	0.780±0.005	1.491	0.190	7.475
	IA	0.731±0.002	0.744±0.005	1.620	0.173	7.285
	II	0.819±0.004	0.778±0.006	1.470	0.175	7.015

To gain a more comprehensive understanding of the data, it is important to analyse individual cases. Figure 5.1 shows a heatmap displaying the F1 scores obtained from session pairs. The vertical axis represents the training data, while the horizontal axis represents the test data. The F1 score ranges from 51% to 93% in pairwise session comparisons. It is important to note that this is a multiple-classification task with approximately 32 classes. In contrast to binary classification, where results above 50% are considered significant, a randomly assigned multiple-classification model would achieve an accuracy of approximately 3.1% (100 divided by 32). Therefore, even an F1 score of 51% represents a substantial improvement over chance/random success.



(a) 17X17 F1 Score list



(b) 28X28 F1 Score list

Figure 5.1: Comparison of sessions with the dataset's same session ID (device distribution). Y-axis shows training and X-axis shows testing data. These results in F1 scores are obtained using Decision Tree (DT).

In Figure 5.1a & b the session IDs that allow the broadest comparison, containing 28 and 17 sessions respectively, are shown. Figure 5.1a predominantly consists of *idle* examples, showing higher success rates when comparing consecutive dates. The heatmap exhibits a somewhat symmetrical structure, albeit imperfect, particularly due to minimal user intervention during *idle* collection. In Figure 5.1b, both *active* and *idle* sessions are mixed. Similarly, success rates are higher for consecutive sessions. However, the involvement of users introduces a more distorted symmetry, especially in *active* sessions. Significant performance drops are observed when using data collected on specific dates coinciding with a national holiday, such as 2021-12-23, 2021-12-25, and 2021-12-28. Additionally, the lowest performances occur when using *idle* as training data and *active* as test data. We believe that *active* sessions offer a broader representation, because *active* use actually includes *idle* use as well, while it is not possible to say the opposite. However, the inherent differences in data collection during *active* sessions change the model's performance. So it is not possible to speak of perfect patterns when human factors are involved. The subsequent subsection explores whether increasing the diversity by combining data from different sessions improves representation and model performance.

5.3.2 Dataset Construction

We aimed to enhance sample diversity and improve model performance by sampling from multiple sessions. The data already included *idle* and *active* sessions, which we further split into training and testing subsets. This resulted in four subsets: *idle*-training, *idle*-testing, *active*-training, and *active*-testing, derived from a total of 54 sessions. Refer to Table 5.6 for the specific assignment of sessions to each subset. The dataset creation process is illustrated in Figure 5.2.

However, due to some problems in the dataset, we have made minor changes to the data. We copied the data of the D-Link Water Sensor, a device not included in the *active* sessions, from the *idle* sessions to the active session data. Another change was related to the LG Smart TV device. The data for this device is only present in three of the 54 sessions. Furthermore, the data for this device is so unbalanced that the data in these three sessions account for about 9% of the total number of packets in all 40 devices. For these reasons, we removed this device from the dataset.

To ensure a balanced dataset that represents session diversity without excessive size, we reduced the number of packets in each of the four datasets to 10% of the total number of packets per dataset by random sampling. This random sampling was employed during this process to maintain consistent packet rates for each device, preserving the natural

distribution of the dataset.

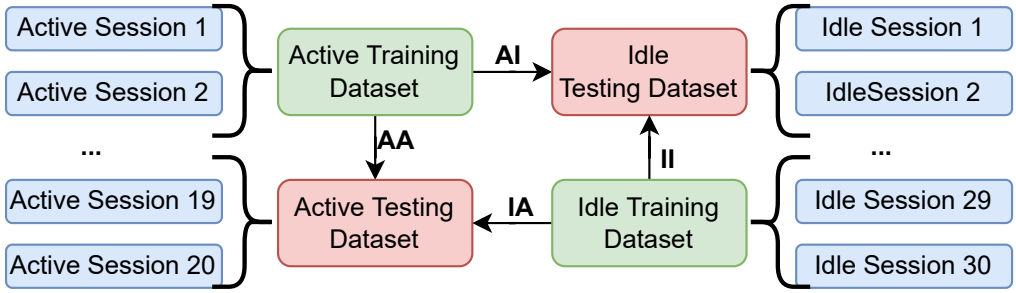


Figure 5.2: Creating datasets with larger representation by combining sessions. Training and testing data of both types (*active* and *idle*) were obtained and they were used in the training and testing of models both for the same category of data (*active* training vs *active* Testing) and for their complement (*active* Training vs *idle* Testing).

5.3.3 IoTDevID Evaluation

Next, we evaluate the performance of the IoTDevID methodology on the datasets described above. Table 5.2 summarises the test performances, both when using individual packets and when using the aggregation algorithm. It can be seen that very good results are obtained in all cases, with all F1 scores being above 81%. When individual packets are used, it is seen that the most successful model is AI with 90.50%, followed by AA with 84.20%, while the results of cases IA and II are very close to each other with a score of around 81%. As in the original IoTDevID study, a significant further improvement is seen when the aggregation algorithm is used. The AA and IA cases improve by about eight points, and the AI and II cases improve by about 10 points. For AI, the models achieve almost perfect discrimination.

Upon comparing these results with Table 5.1, it is evident that all scores have exhibited significant improvements. Analysing individual results, the F1 score has risen from 70.2 to 84.2 for AA, from 71.4 to 90.4 for AI, from 67.9 to 81.8 for IA, and from 72 to 81.4 for II. Notably, the aggregation results demonstrate an even greater increase. The choice of data used exerts a substantial influence on the model's performance, underscoring the importance of constructing a more representative dataset through data combination. This enhanced dataset has substantially bolstered the success of models trained using it.

Returning to Table 5.2, the *active* state has a broader representation as it includes network data both when the devices are used and not used. *Idle* includes only passive states and does not include the states when the devices are used. The AI case, which

Table 5.2: Results for four device conditions with individual and aggregated methods using DT. Results consisting of accuracy and F1 scores (with standard deviations) are the means of 100 repeats. t specifies time in seconds. Alg-t is the duration of the aggregation algorithm.

	Data	Accuracy	F1 Score	Train-t	Test-t	Alg-t
Individual	AA	0.890±0.001	0.842±0.004	1.748	0.204	0
	AI	0.918±0.001	0.905±0.005	1.812	0.287	0
	IA	0.823±0.046	0.818±0.015	1.699	0.223	0
	II	0.821±0.004	0.814±0.007	1.721	0.291	0
Aggregated	AA	0.943±0.001	0.925±0.007	1.962	0.235	9.119
	AI	0.999±0.000	0.999±0.000	1.864	0.299	11.519
	IA	0.850±0.058	0.898±0.017	1.584	0.206	8.46
	II	0.904±0.004	0.912±0.006	1.630	0.313	11.267

employs the *idle* dataset as testing, exhibits an exceptionally high performance that may not reflect practical conditions, as the uniform data distribution of the *idle* dataset creates an "easier" testing environment. In this context, using the *active* state for both training and testing gives more realistic results.

Further analysis at the class level allows for a deeper understanding. In this context, Table 5.3 shows the class-based F1 scores of all devices and Figure 5.3 shows the confusion matrix for the AA case. Focusing on the aggregated results in column AA, it is evident that 22 out of 31 devices achieve near-perfect classification with an F1 score above 99%. Six devices (Globe Lamp, Gosund Plug, HeimVision Lamp, Teckin Plug, Yutron Plug) achieve lower performances, although still above 90%. These devices are lamps or plugs serving similar functions. The dataset is also rich in cameras and speakers, forming another group of devices with similar tasks. However, the classification of these devices does not pose similar challenges. This can be attributed to the fact that devices such as lamps and sockets have simpler structures compared to speakers and cameras, leading to similar data outputs that are more difficult to discriminate. Similar difficulties were encountered in previous experiments with sensors, plugs and switches in the IoTDevID study using the Aalto dataset [117].

Noteworthy are the devices Ring Base Station, Amazon AE Spot and Smart Board, which exhibit significantly lower F1 scores than the other devices. Packets from Ring Base Station devices are often misclassified as speakers (Amazon Alexa family, Sonos Speaker, etc.), likely due to their role as a link between alarm systems in smart homes and management systems like Alexa or other speakers. Analysing the results for the

Amazon AE Spot device, although the recall is high, the precision is exceptionally low (refer to Table 5.7). The Smart Board device poses a challenge as a majority of packets are mislabelled as Amazon AE Spot. Examining the data distribution of the Smart Board, an outlier is observed in the A211126 data, which was added to the *active* test dataset (see Table 5.6). On this specific day, the data collected for this device is twice the combined amount of the other 53 days. Moreover, 78.6% of this unusually large data volume, which is 100 times greater than other sessions, comprises uniformly empty packets that are challenging to classify (TCP packets with the ACK flag set and no payload). Although our study does not analyze the causes of these outliers, it is important to note that the imbalanced data distribution resulting from this outlier greatly complicates the identification of this device during the test phase.

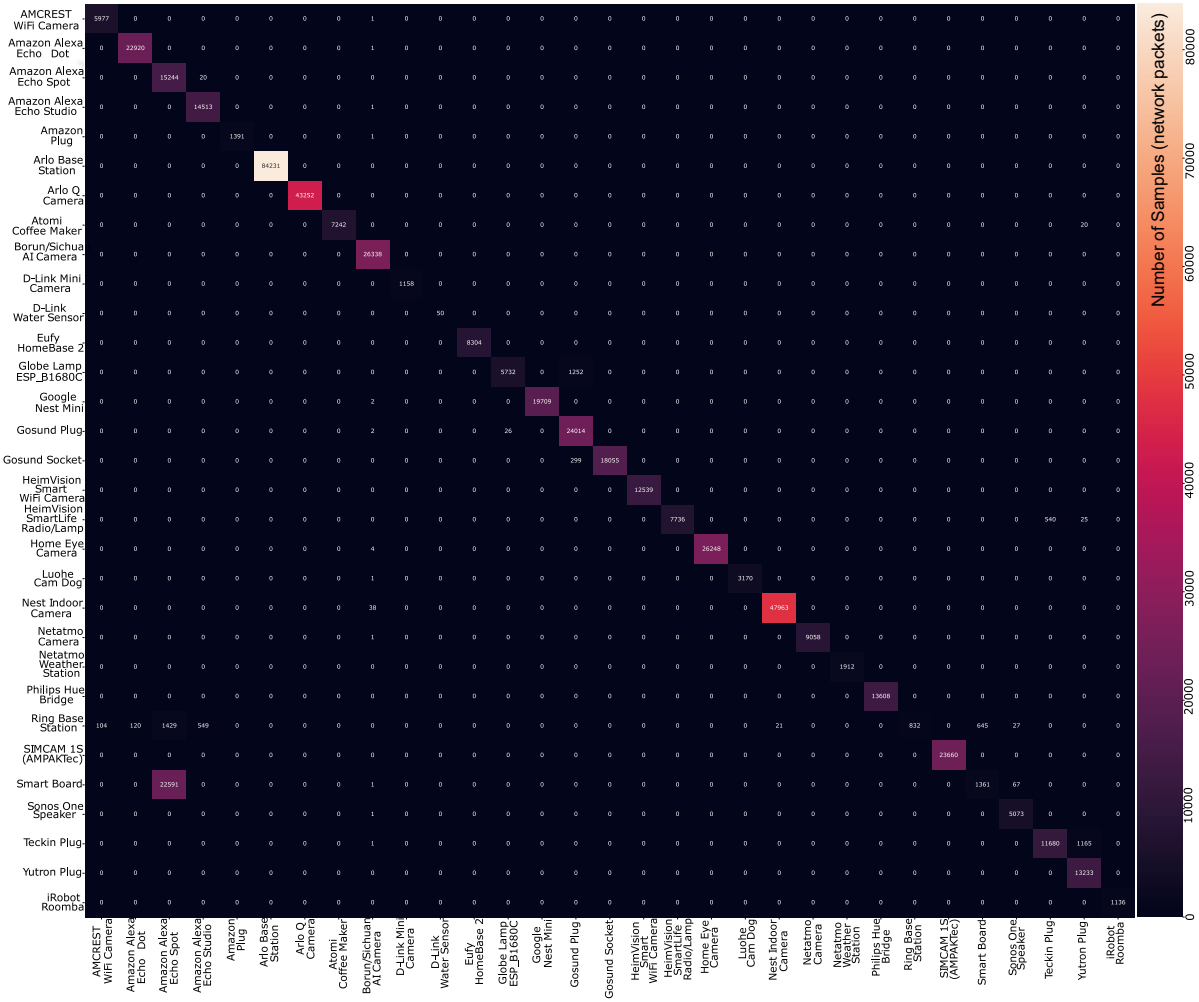


Figure 5.3: Confusion matrix for case AA, showing means of 100 results using DT.

Table 5.3: Class-based results for four device conditions with individual and aggregated methods using DT. Results consisting of F1 scores are the means of 100 repeats.

	Individual				Aggregated			
	AA	AI	IA	II	AA	AI	IA	II
Amcrest WiFi-Cam.	0.968	0.979	0.951	0.959	0.992	1.000	1.000	1.000
Amazon AE Dot	0.933	0.938	0.950	0.947	0.997	1.000	0.998	1.000
Amazon AE Spot	0.555	0.838	0.844	0.837	0.559	1.000	0.999	0.999
Amazon AE Studio	0.821	0.874	0.837	0.736	0.981	1.000	0.999	0.979
Amazon Plug	0.995	0.999	0.997	0.999	1.000	1.000	1.000	1.000
Arlo Base Station	0.984	0.819	0.624	0.864	1.000	0.998	0.454	1.000
Arlo Q Camera	0.987	0.970	0.969	0.952	1.000	1.000	1.000	1.000
Atomi Coff-Maker	0.847	0.892	0.890	0.501	0.999	1.000	1.000	0.638
Borun Camera	0.982	0.981	0.972	0.978	0.999	0.999	0.999	0.999
D-Link Mini Cam.	0.982	0.989	0.342	0.906	1.000	1.000	0.756	1.000
D-Link Water Sen.	0.930	0.936	0.936	0.934	1.000	0.989	1.000	0.989
Eufy HomeBase 2	0.815	0.812	0.782	0.806	1.000	0.998	1.000	0.998
Globe Lamp	0.654	0.823	0.916	0.431	0.900	0.997	1.000	0.246
Google Nest Mini	0.975	0.971	0.952	0.879	1.000	1.000	0.996	0.982
Gosund Plug	0.839	0.899	0.917	0.706	0.968	0.995	0.999	0.724
Gosund Socket	0.868	0.893	0.895	0.532	0.992	0.995	0.999	0.406
HeimVision S Cam.	0.986	0.998	0.934	0.984	1.000	1.000	1.000	1.000
HeimVision Lamp	0.715	0.838	0.857	0.518	0.965	0.999	1.000	0.759
Home Eye Camera	0.930	0.911	0.927	0.910	1.000	1.000	1.000	1.000
Luohe Cam Dog	0.762	0.760	0.759	0.757	1.000	0.995	1.000	0.994
Nest Indoor Cam.	0.998	0.997	0.999	0.910	0.999	0.999	1.000	0.997
Netatmo Camera	0.969	0.986	0.398	0.934	1.000	1.000	0.381	1.000
Netatmo Weather	0.826	0.827	0.868	0.845	1.000	1.000	1.000	1.000
Philips Hue Bridge	0.994	0.990	0.978	0.986	1.000	1.000	1.000	1.000
Ring Base Station	0.305	0.913	0.293	0.697	0.336	1.000	0.250	0.995
SIMCAM 1S	0.996	0.998	0.980	0.998	1.000	1.000	1.000	1.000
Smart Board	0.363	0.721	0.292	0.675	0.105	0.996	0.074	0.996
Sonos One Speaker	0.729	0.844	0.727	0.891	0.991	0.999	0.951	1.000
Teckin Plug	0.675	0.826	0.868	0.578	0.932	1.000	1.000	0.712
Yutron Plug	0.764	0.855	0.867	0.632	0.956	1.000	1.000	0.850
iRobot Roomba	0.955	0.962	0.843	0.946	1.000	1.000	0.993	1.000
Mean	0.842	0.905	0.818	0.814	0.925	0.999	0.898	0.912

5.3.4 Evaluation by Including non-IP Devices

The CICIoT22 dataset also includes non-IP device (Zigbee and Z-Wave) data, specifically in the Power and Interactions states. Only Zigbee devices have records in raw network data format (pcap); Z-Wave device data is not available in this format, limiting the opportunity for feature extraction. Additionally, the non-IP devices lack normal usage data, and the amount of data collected in the Power and Interactions states is relatively small in comparison to IP devices. Table 5.4 shows the number of packets collected in both cases. Despite these limitations, we found the non-IP device data in the experimental suite to be interesting for analysis.

Although the collection method of Zigbee device data does not perfectly align with the transfer problem case, it exhibits similarities due to the shared fixed MAC address (00:00:00:00:00:00) assigned to all devices. This provides an opportunity to explore the exception part of the aggregation algorithm. We incorporated the Zigbee data into the AA state by adding the Zigbee devices from the Interactions state (which had more data) to the training set and using the data from the Power state for testing. The class-based results can be found in Table 5.5.

Table 5.4: The total number of packets generated by the Zigbee devices in Interaction and Power statuses.

Device	Number of Packet		F1 Score
	Power	Interactions	
AeoTec Button	180	13	0.000
AeoTec Motion Sensor	173	25	0.093
AeoTec Multipurpose Sensor	297	173	0.512
AeoTec Water Leak Sensor	340	24	0.034
Philips Hue White	2093	746	0.888
Sengled Smart Plug	367	203	0.149
SmartThings Button	166	12	0.092
SmartThings Hub	198	0	NA
SmartThings Smart Bulb	2870	1412	0.728
Sonoff Smart Plug	148	142	0.386

Table 5.5 reveals that non-IP devices have minimal impact on the results of IP devices, showing almost identical performance to the previous evaluation. The only exception is the Sonos One Speaker, which exhibits a noticeable drop in F1 score, unrelated to the non-IP devices. This can be attributed to the above-mentioned anomaly in the Smart Board device.

Table 5.5: Class-based results of IP and non-IP devices with AA case using individual and aggregated methods.

		Individual			Aggregated		
	Devices	Prec.	Recall	F1 Sc.	Prec.	Recall	F1 Sc.
IP Devices	Amcrest WiFi-Cam.	0.999	0.967	0.983	0.999	1.000	1.000
	Amazon AE Dot	0.913	0.972	0.941	0.974	1.000	0.987
	Amazon AE Spot	0.420	0.852	0.562	0.482	1.000	0.650
	Amazon AE Studio	0.794	0.867	0.829	0.940	1.000	0.969
	Amazon Plug	0.997	0.999	0.998	1.000	0.999	1.000
	Arlo Base Station	0.998	0.970	0.984	1.000	1.000	1.000
	Arlo Q Camera	0.986	0.994	0.990	1.000	1.000	1.000
	Atomi Coff-Maker	0.923	0.777	0.844	1.000	0.998	0.999
	Borun Camera	0.986	0.974	0.980	0.995	1.000	0.997
	D-Link Mini Cam.	0.988	0.980	0.984	0.993	1.000	0.996
	D-Link Water Sen.	1.000	0.912	0.954	1.000	1.000	1.000
	Eufy HomeBase 2	0.727	0.928	0.815	0.995	1.000	0.998
	Globe Lamp	0.707	0.609	0.654	1.000	0.808	0.894
	Google Nest Mini	0.955	0.991	0.973	0.983	1.000	0.991
	Gosund Plug	0.775	0.913	0.839	0.938	1.000	0.968
	Gosund Socket	0.961	0.790	0.867	1.000	0.986	0.993
	HeimVision S Cam.	1.000	0.971	0.985	1.000	1.000	1.000
	HeimVision Lamp	0.861	0.609	0.713	1.000	0.909	0.952
	Home Eye Camera	0.973	0.892	0.931	1.000	0.999	1.000
	Luohe Cam Dog	0.723	0.804	0.762	1.000	0.991	0.996
	Nest Indoor Cam.	0.999	0.999	0.999	1.000	0.999	0.999
	Netatmo Camera	0.950	0.991	0.970	1.000	1.000	1.000
	Netatmo Weather	0.887	0.927	0.906	0.999	1.000	0.999
	Philips Hue Bridge	0.997	0.992	0.994	1.000	1.000	1.000
	Ring Base Station	0.656	0.149	0.242	0.991	0.092	0.167
	SIMCAM 1S	0.996	0.995	0.995	1.000	1.000	1.000
	Smart Board	0.859	0.239	0.374	0.854	0.239	0.373
	Sonos One Speaker	0.583	0.916	0.713	0.701	1.000	0.824
	Teckin Plug	0.698	0.652	0.675	0.942	0.910	0.926
	Yutron Plug	0.669	0.889	0.763	0.916	1.000	0.956
	iRobot Roomba	0.951	0.977	0.964	0.976	1.000	0.988
non-IP — Zigbee Devices	AeoTec Button	0.000	0.000	0.000	0.000	0.000	0.000
	AeoTec Mo Sen.	0.056	0.240	0.091	0.058	0.240	0.093
	AeoTec MP Sen.	0.506	0.514	0.510	0.510	0.514	0.512
	AeoTec WL Sen.	0.020	0.125	0.034	0.020	0.125	0.034
	Philips Hue White	0.839	0.942	0.888	0.840	0.942	0.888
	Sengled Smart Plug	0.307	0.099	0.149	0.308	0.099	0.149
	SmartThings Button	0.056	0.250	0.092	0.057	0.250	0.092
	SmartThings S. Bulb	0.707	0.694	0.700	0.766	0.694	0.728
	Sonoff Smart Plug	1.000	0.239	0.386	1.000	0.239	0.386
	Macro avg	0.761	0.740	0.726	0.831	0.801	0.788
	Weighted avg	0.909	0.890	0.885	0.962	0.949	0.941
	Accuracy		0.890			0.949	

On the other hand, the aggregation algorithm significantly improves the results for IP devices, while it has no effect on non-IP devices. This is due to the shared MAC address among Zigbee devices, which causes the aggregation algorithm to add them to the exception list and bypass aggregation for these devices.

Most of the non-IP devices show relatively low F1 scores. Nevertheless, the primary reason for these low results is likely the insufficient amount of data available for these devices. This performance issue was not encountered in the Aalto dataset, which contains more non-IP devices data, with a similar amount of data to the IP devices. Table 5.4 shows that there is a positive correlation between the number of instances and the F1 score, meaning that as the number of instances increases, the F1 score tends to increase. For example, *SmartThings Smart Bulb* and *Philips Hue White* have the highest F1 scores (0.728 and 0.888 respectively) and also the most training instances (2870 and 2093 respectively). Similarly, the *AeoTec Button*, *AeoTec Motion Sensor* and *SmartThings Button* devices have relatively low F1 scores (0.000, 0.093, 0.092 respectively) and relatively few training examples (180, 173, 166 respectively) and very few test examples (13, 25, 12 respectively). However, there are some exceptions to this trend. For example, the *AeoTec Multipurpose Sensor* has a higher F1 score than the *Sengled Smart Plug* (respectively 0.512, 0.149), but the *Sengled Smart Plug* has more training and test samples. Overall, it is clear that the number of samples is an important factor affecting the F1 score. However, this is not the only factor and other factors such as the quality of the training data and the complexity of the task may also play a role. It should also be noted that training and test data represent different situations rather than being collected under normal conditions. Therefore, whatever the amount of data, we cannot say that the data quality is sufficient in non-IP devices.

5.4 Conclusions

This study validates the feature set and aggregation algorithm of the IoTDevID method using a new dataset. The dataset offers several advantages, including the presence of non-IP devices, multiple instances of the same device, normal usage data, and diverse usage profiles. The results demonstrate that models trained with *active* usage data outperform those trained with *idle* usage data, emphasising the importance of data diversity in achieving better model performance (RQ1.4, RQ1.5, RQ1.6, C2.2, C1.5, C1.6).

The study showed successful external validation. It achieved impressive results with

IP-only devices, achieving an F1 score of 92.50 for 31 device classes when evaluated in the more realistic scenario of devices undergoing *active* use. While non-IP devices faced challenges due to limited data availability, significant success was observed for devices with available data, showcasing the potential of the aggregation algorithm in accurately detecting non-IP devices.

In the original IoTDevID study, the UNSW IoT Traffic Traces, Data for IEEE TMC 2018 (UNSW-DI) dataset (32 IoT devices) achieved an F1 score of 93.70%, similar to the results obtained here. The CICIoT22 and UNSW-DI datasets have similar usage data and device counts, with some structural differences. The Aalto dataset, with a lower number of devices, achieved a lower F1 score of 86%. We attributed this to the dataset's abundance of devices performing similar tasks. Furthermore, the lack of usage data in the Aalto dataset may have contributed to the lower performance, as observed in our experiments on data quality.

This study highlights the importance of validation studies in assessing the robustness and generalizability of machine learning methods. The findings further contribute to the field of IoT device identification and provide insights into the impact of data diversity on model success. However, this study primarily focuses on enhancing the feature set and method generalizability. One notable limitation is the lack of sufficient information about model generalizability, which is a key aspect of device identification. Specifically, it does not ascertain whether a model trained on data from one device can effectively detect another device of the same make and model. This limitation will be addressed in the subsequent Chapter 6 using the dataset for "information exposure for consumer IoT devices" (IMC'19) (MonIoTr) dataset.

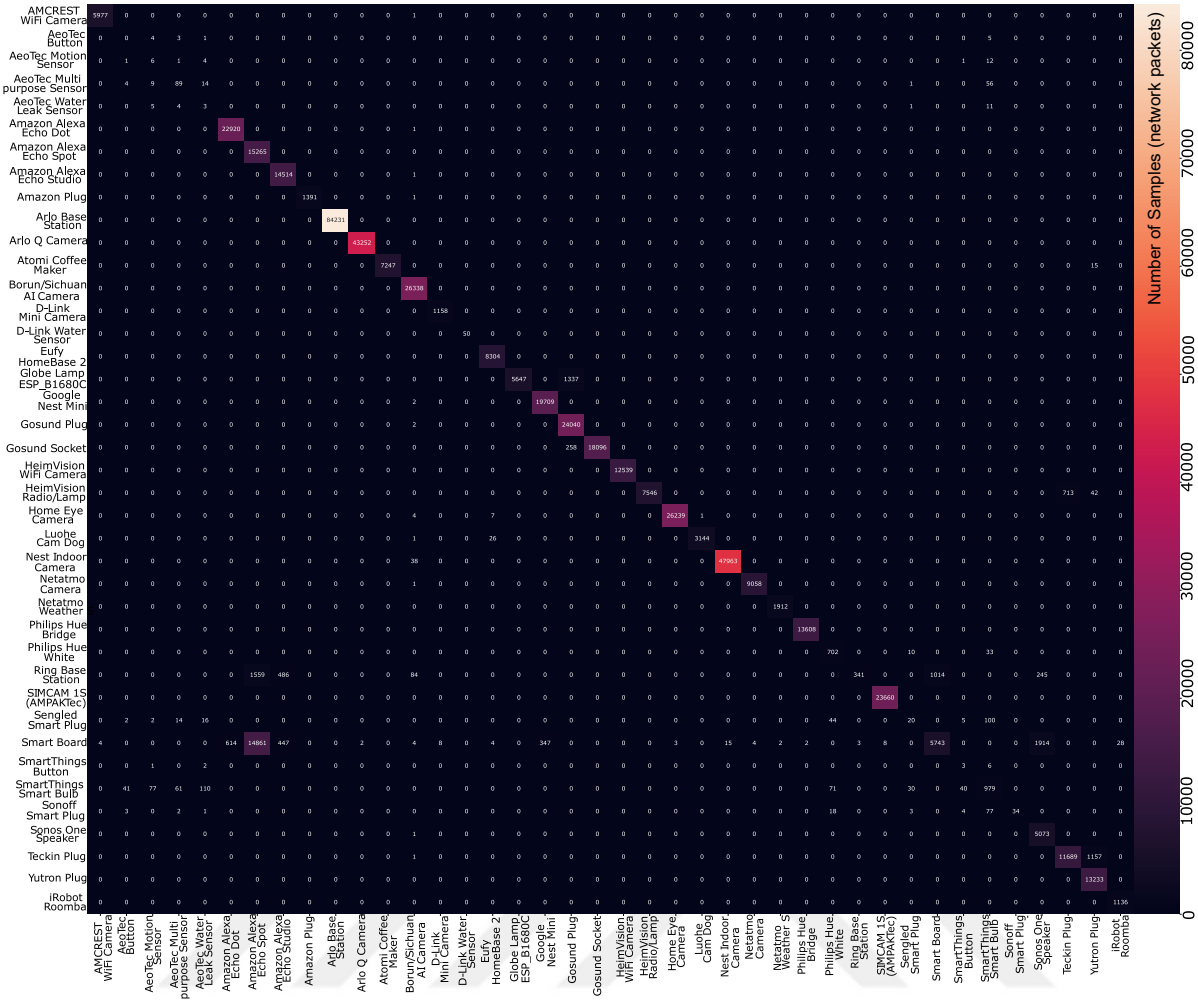


Figure 5.4: Confusion matrix for case AA with non-IP devices.

Table 5.6: Total number of packets generated by devices, number of devices, in each session. Scaled to 1000.

ML Input Data Type	Date of Data Collection	State Type (*active*/*idle*)	AMCREST WiFi Camera	Amazon Alexa Echo Dot 1	Amazon Alexa Echo Dot 2	Amazon Alexa Echo Spot	Amazon Alexa Echo Studio	Amazon Plug	Arlo Base Station	Arlo Q Camera	Atomi Coffee Maker	Borun/Sichuan-AI Camera	D-Link Water Sensor	D-Link Mini Camera	Eufy HomeBase 2	Google Nest Mini	Globe Lamp ESP_B1680C	Gosund ESP_032979 Plug	Gosund ESP_039AAF Sckt	Gosund ESP_0C3994 Plug	Gosund ESP_10098F Sckt	Gosund ESP_10ACD8 Plug	Gosund ESP_147FF9 Plug	Gosund ESP_1ACEE1 Sckt	HeimVision Smrt WiFiCam	HeimVision SL R-Lamp	Home Eye Camera	LG Smart TV	Luohe Cam Dog	Nest Indoor Camera	Netatmo Camera	Netatmo Weather Station	Philips Hue Bridge	Ring Base Station	SIMCAM 1S (AMPAKTec)	Smart Board	Sonos One Speaker	Teckin Plug 1	Teckin Plug 2	Yutron Plug 1	Yutron Plug 2	iRobot Roomba	**Total Packet Numbers**
Train	20211102	*active*	6.68	6.91	6.84	14.94	13.20	1.47	12.63	21.08	6.94	24.04	0.00	1.08	8.08	6.75	20.59	6.65	6.62	6.63	0.00	6.66	6.63	6.64	14.30	7.88	22.48	0	3.03	33.40	9.59	1.81	12.50	3.37	21.45	0.40	5.18	6.68	6.34	6.35	6.68	1.12	353.62
Train	20211103	*active*	5.54	7.10	6.98	15.19	13.16	1.48	4.93	18.81	6.93	24.42	0.00	1.10	8.07	6.75	24.46	6.75	6.74	6.75	24.14	6.75	6.74	6.75	14.29	7.88	25.28	600	2.92	24.23	12.21	1.87	12.67	3.26	21.08	0.55	8.71	6.62	6.29	6.53	6.65	1.21	947.57
Train	20211105	*active*	5.52	7.00	6.93	13.72	13.12	1.48	4.95	21.54	6.94	26.20	0.00	1.11	8.07	6.74	21.27	6.74	6.74	6.75	0.00	6.75	6.75	6.76	13.82	7.88	29.66	0	2.78	23.78	26.15	1.70	14.03	3.21	20.91	0.38	4.97	6.61	6.46	6.63	6.06	1.05	361.14
Train	20211109	*active*	6.09	11.44	10.94	14.72	13.17	1.47	45.97	47.26	6.93	24.37	0.00	1.09	7.97	6.75	21.08	6.75	6.74	6.75	0.00	6.75	6.75	6.76	13.76	7.88	25.97	0	3.03	43.68	9.17	1.77	12.47	3.34	21.65	0.44	4.99	6.62	6.08	6.64	6.07	1.05	441.11
Train	20211110	*active*	6.74	44.07	48.50	19.88	14.95	2.39	22.25	37.38	6.93	24.20	0.00	1.09	7.98	6.74	17.82	6.73	6.73	6.74	6.75	6.75	0.00	6.75	13.79	7.86	33.05	0	3.01	26.28	9.00	1.82	13.79	3.27	21.15	6.15	6.87	6.09	6.11	6.46	6.49	1.16	473.70
Train	20211112	*active*	5.65	14.08	40.82	15.70	19.46	1.47	4.89	37.48	6.92	24.09	0.00	1.09	8.06	6.73	25.21	6.73	6.73	6.74	6.75	6.75	0.00	6.73	13.74	7.86	39.65	0	3.02	29.18	10.53	1.91	15.19	3.28	21.48	8.36	6.98	6.25	6.26	6.56	6.39	1.19	439.88
Train	20211115	*active*	5.66	12.14	11.83	14.93	14.21	1.82	1.10	55.18	7.14	17.26	0.00	1.17	8.15	6.88	24.21	6.87	6.89	6.90	6.88	6.90	6.88	6.90	13.61	8.07	23.00	0	3.01	67.01	9.43	1.83	17.50	3.85	21.47	3.04	4.25	6.36	6.28	6.58	6.59	1.21	541.98
Train	20211117	*active*	5.43	11.33	10.96	13.68	13.82	1.47	11.09	98.64	6.93	22.57	0.00	1.11	8.01	6.74	21.04	6.73	6.75	6.77	6.74	6.75	6.74	6.73	13.93	7.91	27.98	0	3.02	84.55	14.13	1.90	15.72	3.32	22.03	8.06	3.84	6.11	6.01	6.26	6.25	1.17	512.23
Train	20211117	*active*	5.36	23.47	22.58	15.33	58.61	1.47	4.88	44.92	6.93	24.42	0.00	1.12	7.95	6.73	14.10	6.73	6.73	6.75	6.74	6.75	6.74	6.73	13.66	7.88	27.78	0	3.02	29.93	9.54	1.91	13.70	3.25	21.23	3.99	4.84	6.15	6.01	6.14	6.35	1.10	451.52
Test	20211118	*active*	5.97	11.78	11.35	13.33	14.21	1.47	4.81	4.97	6.94	24.18	0.00	1.11	8.12	6.73	20.55	6.74	6.74	6.75	6.75	6.74	6.74	6.73	13.66	7.88	21.26	0	3.02	29.55	4.31	1.88	12.86	3.20	21.01	3.63	3.89	6.26	6.21	6.28	6.38	1.34	335.33
Test	20211119	*active*	5.54	11.83	11.15	13.41	14.22	1.47	4.91	3.64	6.92	24.12	0.00	1.11	8.07	6.73	20.25	6.73	6.73	6.75	6.76	6.74	6.73	6.73	13.66	7.87	21.08	0	3.02	29.57	4.10	1.85	12.70	3.21	20.92	1.37	3.82	6.14	6.00	6.13	6.33	1.12	315.75
Test	20211122	*active*	5.62	9.54	9.35	14.91	13.49	1.47	15.10	58.72	6.94	24.12	0.00	1.12	8.15	6.73	20.72	6.71	6.72	6.74	6.72	6.71	6.72	6.74	13.64	7.87	22.20	0	3.02	78.49	9.49	1.82	13.54	3.32	21.26	3.82	3.87	6.13	6.05	6.39	6.38	1.12	451.39
Test	20211123	*active*	6.28	11.53	10.46	15.34	14.58	1.41	7.82	58.50	7.10	24.76	0.00	1.15	8.12	6.89	20.72	6.88	6.87	6.90	6.91	6.89	6.88	6.88	13.90	8.07	26.72	0	3.09	74.48	11.61	1.85	12.97	3.33	28.15	5.24	3.93	6.25	6.18	6.57	6.59	1.06	461.84
Test	20211124	*active*	5.74	11.22	10.28	14.61	13.34	1.47	4.90	17.54	6.93	24.88	0.08	1.10	7.96	6.73	20.23	6.73	6.73	6.74	6.75	6.74	6.73	6.73	0.00	7.88	21.73	0	3.01	23.02	6.47	1.86	12.57	3.29	27.77	4.74	4.11	6.10	6.23	6.46	6.34	1.03	336.83
Test	20211125	*active*	5.76	11.37	10.25	16.00	14.08	1.55	5.15	34.63	6.94	24.23	0.00	1.09	8.07	6.73	20.68	6.73	6.73	6.76	6.76	6.76	6.76	6.73	29.56	7.88	21.17	0	3.03	22.86	5.75	1.90	14.36	3.30	21.16	6.04	5.75	6.10	6.18	6.30	6.31	1.18	382.52
Test	20211126	*active*	7.74	16.24	14.83	20.22	17.21	0.52	326	145.99	9.20	32.01	0.00	1.65	11.30	8.93	26.73	8.93	8.93	8.94	8.92	8.95	8.97	8.94	18.23	10.45	42.15	30	4.05	39.26	19.26	2.47	17.65	5.69	29.41	212	6.21	8.18	8.72	8.65	8.37	1.56	1176.98
Test	20211207	*active*	6.50	11.06	10.68	15.59	15.76	1.56	24.18	33.81	6.94	32.40	0.00	1.09	8.79	6.73	18.50	6.75	6.73	6.74	6.78	6.73	6.74	13.81	7.88	18.93	0	2.97	69.60	11.83	1.77	12.83	3.41	21.78	0.77	4.66	6.08	6.32	6.55	6.55	6.55	1.20	437.73
Test	20211207	*active*	5.66	10.97	10.55	15.57	14.44	1.52	438	41.61	6.93	27.73	0.00	1.09	6.53	6.73	14.97	6.27	2.67	2.69	2.67	1.05	0.98	0.97	15.55	7.89	29.16	1805	3.02	27.10	9.89	1.82	14.18	3.81	21.62	0.43	11.62	6.73	6.25	6.68	6.78	1.05	2903.11
Test	20211223	*active*	6.26	11.73	11.49	13.33	13.33	1.46	4.93	3.79	6.92	24.79	0.00	1.08	7.99	6.72	19.71	6.75	6.72	6.72	6.72	6.72	6.72	6.73	13.57	7.90	17.00	0	3.02	23.24	3.78	1.77	12.81	3.17	20.89	0.74	4.98	6.81	6.67	6.74	6.46	1.20	321.38
Test	20211224	*active*	5.08	16.13	11.76	14.58	14.39	1.46	13.16	23.09	6.93	25.21	0.00	1.08	8.01	6.73	19.97	6.74	6.72	6.72	6.73	6.73	6.72	6.73	13.52	7.89	19.59	0	3.02	16.77	7.10	1.88	12.95	3.20	21.15	2.71	11.82	6.80	6.68	6.75	6.35	1.04	364.74
Test	20211228	*active*	5.45	11.56	11.93	20.39	15.07	1.46	8.51	23.50	6.92	24.60	0.00	1.07	8.45	6.72	13.55	6.75	6.72	6.72	6.72	6.72	6.72	6.72	13.52	7.89	19.59	0	3.02	24.72	5.67	1.81	7.08	3.23	21.19	0.61	44.11	6.79	6.68	6.64	6.17	1.10	386.08
Train	20220103	*active*	5.59	11.72	9.36	14.45	13.35	1.50	4.91	2.75	6.93	25.43	0.00	1.08	8.07	6.72	14.93	6.73	6.72	6.74	6.72	6.73	6.75	6.73	14.93	7.86	16.75	0	0.00	25.41	3.74	1.68	7.07	3.16	20.91	0.48	4.92	6.68	6.37	6.61	6.60	1.39	308.43
Test	20211102	*idle*	7.92	8.94	8.90	19.26	121.47	2.00	6.85	3.82	9.54	32.99	0.00	1.46	11.04	9.26	20.43	9.26	9.25	9.26	9.26	9.25	9.25	9.26	20.23	10.84	28.76	0	4.15	6.63	7.25	2.47	17.05	4.79	28.62	0.35	6.89	9.08	8.70	8.82	9.08	1.43	593.68
Test	20211103	*idle*	7.85	9.09	9.11	18.78	19.42	2.05	6.83	3.82	9.52	33.06	0.00	1.46	11.12	9.25	30.55	9.26	9.27	9.28	9.27	9.25	9.27	9.27	20.05	10.83	28.81	0	3.66	6.48	7.27	2.46	16.93	4.70	28.64	0.20	6.82	9.06	8.71	8.95	9.07	1.42	401.55
Test	20211104	*idle*	8.60	9.32	9.36	18.90	19.41	2.09	6.89	3.94	9.51	34.96	0.00	1.47	11.23	9.26	22.58	9.27	9.26	9.26	9.26	9.28	9.28	9.26	19.78	10.83	28.76	0	3.80	7.69	7.69	2.53	16.88	4.73	28.60	0.24	6.90	8.91	8.84	8.88	8.36	1.51	398.29
Test	20211108	*idle*	7.97	15.31	15.73	17.53	19.50	2.05	6.81	3.81	9.55	35.11	0.00	1.52	10.97	9.27	29.83	9.26	9.25	9.27	9.27	9.27	9.27	9.26	19.07	10.85	28.78	0	4.15	20.06	5.12	2.51	16.89	4.69	28.64	0.05	6.82	9.11	8.36	9.11	8.38	1.52	433.88
Test	20211110	*idle*	8.43	15.27	15.38	19.42	17.88	1.98	6.84	51.55	9.54	34.98	0.00	1.48	11.27	9.25	31.72	9.25	9.25	9.27	9.27	0.00	9.25	18.83	10.82	29.00	0	4.15	8.06	5.17	2.51	17.11	4.68	28.63	3.87	6.86	8.39	8.30	9.12	8.93	1.43	466.40
Test	20211112	*idle*	8.53	15.11	14.88	17.22	17.35	2.00	6.80	47.37	9.54	34.99	0.00	1.47	11.01	9.25	40.40	9.25	9.25	9.27	9.27	9.28	9.28	18.82	10.82	28.81	0	4.15	11.71	5.12	2.55	24.88	4.72	28.64	3.30	6.93	8.41	8.40	9.03	8.70	1.43	487.15
Test	20211115	*idle*	7.44	16.37	15.87	17.82	18.01	1.99	10.25	3.95	9.55	22.25	0.00	1.48	11.08	9.26	28.80	9.26	9.25	9.28	9.25	18.84	10.82	28.50	0	4.12	41.22	5.11	2.58	24.87	4.65	28.63	1.41	5.31	8.48	8.38	8.72	8.83	1.43	450.91
Test	20211116	*idle*	7.50	16.55	15.58	17.92	18.09	1.99	6.81	3.84	9.54	34.99	0.00	1.50	11.12	9.25	32.30	9.25	9.25	9.27	9.27	9.26	9.25	18.83	10.81	28.44	0	4.15	40.95	5.20	2.47	17.21	4.69	28.64	0.00	5.33	8.43	8.29	8.56	8.71	1.50	454.01
Test	20211117	*idle*	7.56	16.46	16.16	17.94	18.52	2.01	6.86	3.84	9.54	33.14	0.00	1.51	11.02	9.25	20.19	9.25	9.25	9.27	9.27	9.27	9.27	18.87	10.83	28.82	0	4.16	40.84	5.26	2.59	17.20	4.75	28.65	0.00	21.86	8.41	8.24	8.68	8.69	1.51	458.17
Train	20211118	*idle*	7.56	16.78	16.06	18.00	18.49	2.00	6.78	3.80	9.53	33.10	0.00	1.47	11.01	9.25	28.82	9.25	9.25	9.28	9.26	9.25	9.26	13.83	10.81	28.82	0	4.15	40.66	5.16	2.63	17.17	4.72	28.64	3.38	5.36	8.42	8.42	8.27	8.64	8.67	1.48	448.29
Train	20211119	*idle*	8.25	16.65	16.41	18.14	18.41	2.00	6.90	5.31	9.54	34.98	0.00	1.46	11.37	9.25	28.66	9.25	9.25	9.29	9.28	9.26	9.28	9.27	9.25	18.79	10.82	29.03	0	4.15	40.78	5.21	2.50	17.28	4.71	28.64	1.29	5.42	8.42	8.26	8.56	8.67	1.42	441.97
Train	20211122	*idle*	7.68	14.84	12.67	20.80	17.41	2.05	6.81	3.80	9.54	33.25	0.00	1.48	10.96	9.25	28.37	9.25	9.25	9.28	9.30	9.29	9.27	9.25	18.81	10.81	28.94	0	4.16	40.82	5.05	2.56	17.25	4.70	28.63	1.34	5.32	8.35	8.39	8.84	8.87	1.46	448.13
Train	20211124	*idle*	7.75	15.58	14.15	18.74	17.16	2.01	6.83	4.34	9.53	33.27	0.07	1.49	11.17	9.24	28.26	9.25	9.25	9.27	9.25	18.79	10.82	29.05	0	4.15	7.00	5.14	2.48	17.18	4.65	37.53	4.55	5.38	8.44	8.38	8.72	8.63	1.43	426.78
Train	20211125	*idle*	7.95	15.88	14.38	18.76	17.59	2.04	6.82	3.81	9.54	33.25	0.04	1.49	10.98	9.25	28.65	9.25	9.25	9.29	9.29	9.27	9.25	21.05	10.82	28.61	0	4.16	6.82	5.07	2.55	17.18	4.68	28.64	0.48	5.37	8.40	8.32	8.44	8.55	1.43	415.90
Train	20211126	*idle*	8.95	16.37	15.58	21.80	19.96	0.00	6.85	4.38	9.55	34.99	0.07	1.50	11.00	9.25	28.43	9.25	9.26	9.28	9.29	9.27	9.25	18.91	10.82	29.32	0	4.18	21.80	5.00	2.50	17.21	4.69	28.62	4.77	5.38	8.37	8.60	8.63	8.62	1.42	442.39
Train	20211129	*idle*	8.49	15.30	14.39	19.23	19.33	1.97	7.04	3.97	9.61	33.24	0.04	1.50	11.01	9.32	19.83	9.34	9.34	9.34	9.38	9.34	9.33	20.23	10.83	23.21	0	4.17	6.52	5.02	2.59	17.31	0.00	28.73	0.39	5.40	8.51	8.62	9.00	9.04	1.48	400.76
Train	20211130	*idle*	7.79	15.77	14.29	19.18	17.51	2.00	10.01	3.80	9.53	33.67	0.07	1.49	11.16	9.25	19.26	0.00	9.25	9.26	9.25	9.25	9.25	18.75	10.84	23.20	0	4.15	6.65	5.08	2.46	17.27	4.70	28.62	0.38	5.34	8.30	8.86	8.25	8.55	1.42	393.10
Train	20211201	*idle*	8.90	15.49	14.60	18.90	17.48	2.01	6.85	3.80	9.53	35.77	0.04	1.49	11.03	9.24	21.10	9.26	9.25	9.25	9.26	9.25	9.25	18.75	10.84	23.24	0	4.14	7.39	5.19	2.56	17.26	4.71	28.63	0.44	5.37	8.28	8.87	8.27	8.71	1.43	405.04
Train	20211202	*idle*	7.77	14.71	14.15	19.08	17.49	1.97	6.90	3.80	9.53	35.70	0.11	2.53	11.16	9.25	20.06	9.28	9.25	9.25	9.25	9.24	9.25	18.84	10.85	23.69	0	4.13	7.83	5.15	2.52	17.25	4.68	28.60	0.50	5.34	8.26	8.80	9.07	8.50	1.43	404.43
Train	20211203	*idle*	8.44	14.90	14.75	18.24	17.32	1.99	6.86	3.82	9.53	35.76	0.07	1.48	11.94	9.25	28.62	9.27	9.25	9.25	9.33	18.80	10.85	23.51	0	4.16	8.64	5.14	2.47	17.70	4.67	28.63	14.91	5.20	8.44	8.60	9.10	8.93	1.00	427.99
Train	20211207	*idle*	8.05	15.23	14.13	18.63	19.22	2.01	6.91	3.81	9.53	35.77	0.04	1.48	11.28	9.24	20.42	9.26	9.25	9.26	9.25	20.20	10.84	23.64	0	4.15	7.06	5.24	2.61	17.33	6.41	28.63	0.38	5.38	8.45	8.51	8.68	9.01	1.50	408.74
Train	20211208	*idle*	7.99	15.18	14.57	18.94	19.50	2.00	6.78	3.82	9.53	35.73	0.07	1.48	11.02	9.25	19.89	1.38	1.32	1.33	1.38	1.33	1.33	20.07	10.85	23.84	0	4.16	12.79	5.05	2.40	17.26	4.70	28.63	0.41	5.29	9.12	8.34	9.10	9.00	1.44	357.63
Train	20211210	*idle*	8.12	15.42	14.77	19.13	19.35	2.28	6.90	3.79	9.52	35.36	0.04	1.48	10.94	9.24	21.11	9.27	9.24	9.25	9.26	9.25	9.25	18.72	10.84	23.33	0	4.15	23.02	5.20	2.37	17.25	4.69	28.59	0.47	6.86	8.31	8.55	8.54	8.79	1.43	423.30
Train	20211213	*idle*	8.47	15.19	15.03	19.43	19.54	2.05	6.74	3.82	9.53	35.38	0.07	1.48	11.03	9.25	20.22	9.24	9.25	9.26	9.25	9.25	9.25	18.82	10.84	23.72	0	4.16	20.58	5.42	2.42	17.26	4.66	28.54	0.42	6.90	8.20	8.62	8.40	8.88	1.42	421.23
Test	20211215	*idle*	8.09	14.14	13.82	18.83	17.68	2.00	7.36	3.81	9.53	35.47	0.07	1.47	11.43	9.24	27.99	9.26	9.24	9.25	9.25	9.25	9.25	18.67	10.84	23.40	0	4.15	21.63	5.16	2.63	17.27	4.66	28.60	0.50	5.38	8.29	8.95	8.27	9.07	1.44	424.56
Test	20211216	*idle*	7.65	12.23	12.27	18.83	17.25	2.00	6.91	3.80	9.52	35.58	0.11	1.46	11.01	9.25	21.12	9.26	9.24	9.25	9.25	9.25	9.25	18.68	10.85	23.40	0	4.15	25.21	5.15	2.46	17.20	4.66	28.61	0.37	5.33	8.28	9.13	8.32	9.11	1.43	416.07
Test	20211220	*idle*	7.78	15.73	14.76	18.35	19.86	2.00	6.80	3.83	9.52	35.51	0.07	1.47	11.01	9.24	27.55	9.26	9.24	9.24	9.25	9.25	9.24	18.66	10.84	23.18	0	4.14	21.32	5.18	2.28	17.32	4.75	28.62	0.52	5.32	9.36	9.15	9.24	8.67	1.58	428.31
Test	20211222	*idle*	7.63	16.77	15.48	18.56	20.42	2.00	6.78	3.79	9.52	35.56	0.07	1.46	11.02	9.24	27.83	9.26	9.24	9.24	9.25	9.25	9.24	18.77	10.84	23.18	0	4.16	22.92	5.20	2.53	17.28	4.69	28.64	2.39	6.84	9.37	9.12	9.28	8.70	1.44	436.18
Test	20211223	*idle*	8.32	16.74	15.62	18.37	20.51	1.98	6.84	3.80	9.52	35.50	0.07	1.46	11.24	9.24	27.58	9.25	9.24	9.24	9.25	9.25	9.24	18.83	10.85	23.19	0	4.16	20.25	5.19	2.54	17.28	4.70	28.62	3.19	6.87	9.36	9.19	9.25	8.08	1.56	435.22

Table 5.7: Class-based F1 score results for AA case, showing means of 100 repeats.

Devices	precision	recall	f1-score	support
Amazon AE Dot	0.995	1.000	0.997	22921
Amazon AE Spot	0.388	0.999	0.559	15265
Amazon AE Studio	0.963	1.000	0.981	14515
Amazon Plug	1.000	0.999	1.000	1392
Amcrest WiFi-Cam.	0.984	1.000	0.992	5978
Arlo Base Station	1.000	1.000	1.000	84231
Arlo Q Camera	1.000	1.000	1.000	43252
Atomi Coff-Maker	1.000	0.997	0.999	7263
Borun Camera	0.998	1.000	0.999	26338
D-Link Mini Cam.	1.000	1.000	1.000	1158
D-Link Water Sen.	1.000	1.000	1.000	50
Eufy HomeBase 2	1.000	1.000	1.000	8304
Globe Lamp	0.995	0.821	0.900	6985
Google Nest Mini	1.000	1.000	1.000	19711
Gosund Plug	0.939	0.999	0.968	24042
Gosund Socket	1.000	0.984	0.992	18355
HeimVision Lamp	1.000	0.932	0.965	8303
HeimVision S Cam.	1.000	1.000	1.000	12539
Home Eye Camera	1.000	1.000	1.000	26253
iRobot Roomba	1.000	1.000	1.000	1136
Luohe Cam Dog	1.000	1.000	1.000	3171
Nest Indoor Cam.	1.000	0.999	0.999	48001
Netatmo Camera	1.000	1.000	1.000	9059
Netatmo Weather	1.000	1.000	1.000	1912
Philips Hue Bridge	1.000	1.000	1.000	13608
Ring Base Station	1.000	0.223	0.336	3732
SIMCAM 1S	1.000	1.000	1.000	23660
Smart Board	0.735	0.057	0.105	24021
Sonos One Speaker	0.983	1.000	0.991	5074
Teckin Plug	0.956	0.909	0.932	12847
Yutron Plug	0.916	1.000	0.956	13233
accuracy		0.943		
macro avg	0.963	0.933	0.925	506309
weighted avg	0.961	0.943	0.932	506309

Chapter 6

Measuring Model Instance Generalisability

> All models are wrong, some are useful.
>
> George Box

6.1 Overview

One of the primary objectives of device identification research is to develop generalisable model instances, where model generalisability in this context means whether a specific model instance trained with data from one device can detect another device of the same make and model. While some researchers have relied on single datasets to develop and test their device identification methods, we sought to validate our approach using multiple datasets in Chapter 4 and Chapter 5. However, this validation was restricted to testing the methodology and feature sets, as obtaining training and test data from different datasets proved challenging. This is because testing the results of a study focused on model generalisability requires samples of the same make and model device in two different datasets (to be used as training and test data). The vast diversity in the IoT ecosystem makes it difficult to find intersecting datasets with identical devices for direct model generalisation between datasets.

The dataset for "information exposure for consumer IoT devices" (IMC'19) (MonIoTr) [189] offers data collected from two different network environments (laboratories) and includes a substantial number of intersecting devices. Thus, we have two different data of the same devices collected under different conditions. This characteristic al-

lows us to isolate the data for intersecting devices from different laboratories, providing concrete evidence of the generalizability of our device identification method.

Secondly, the dataset includes two categories of data for each laboratory: normal data and data collected via VPN. Although the hardware, firmware, and jurisdiction remain uniform in each pair of situations, there is a variation in the exit IP address (and the choice of servers determined by IP address). These two data types, VPN and normal, can be viewed as data collected during separate sessions originating from the same laboratory. By using both data types (VPN and Normal), we address concerns regarding the generalizability and representativeness of the feature set for individual devices arising from the inclusion of features from the IP field discussed in Section 4.3.1 in our feature set.

In addition, our review of device identification methods suggested that those based around network behaviours would function effectively even in the presence of encrypted data, as these focus on packet shapes rather than contents. However, none of the datasets used so far allowed us to verify this claim. In the primary analysis of the MonIoTr dataset [189], an examination of the extent to which the data in this dataset was encrypted was also conducted. According to this analysis, roughly 30% of the transmitted data is encrypted, although this percentage varies depending on the task. Given these characteristics of the MonIoTr dataset, the success of the identification process on this dataset will show that our device identification method can also work on encrypted data.

6.2 MonIoTr Dataset

The MonIoTr dataset consists of 81 physical devices and 56 device classes. Notably, it has the highest device count among existing/reviewed datasets (see device identification datasets from Table 3.2). An important characteristic of this dataset is that data collection took place in two separate laboratories, one in the United Kingdom (UK) and the other in the United States of America (US). Of the 56 device classes, 26 devices were present in both laboratories, with the US laboratory having 47 devices (including 21 unique devices) and the UK laboratory having 34 devices (including 8 unique devices). Figure 6.1 shows the distribution of devices by country.

Data collection in these laboratories spanned 112 hours and encompassed various situations. These situations can be categorised as follows:

- **Idle**: Data was collected for 8 hours at night when the devices were not actively used.

- **Power**: Data collected during the 2-minute period immediately after the device was powered on, without any interaction.
- **Interaction**: Data collected by interacting with the devices in different ways, including physically pressing buttons or using voice commands, interacting with the device through a phone app on the same network, communicating with the device through a phone on a different network to utilise cloud infrastructure, and interacting with the device via *Alexa Echo Spot*.

Furthermore, each device was connected via a Virtual Private Network (VPN) to the other laboratory, allowing data collection across the laboratories for every condition. The same statuses were repeated using VPN tools to establish connections between the laboratories. For instance, idle data was collected by connecting to the UK laboratory from the US via VPN. This process is visualised in Figure 6.2. Here, we emphasize that VPN data is collected post VPN tunnelling completion, extracting the VPN capsule. Normal and VPN data share identical packet structures, differing only in output IP addresses due to the VPN. VPN data lacks additional headers, and both data types utilize the standard (IoTDevID) dataset.

During our analysis, the Power and Interaction data were combined and labelled as "Active". This decision was made due to the extensive interaction of many devices with the local network, applications, tools, and cloud services during startup. Consequently, the device states were simplified to "Idle" and "Active" instead of three separate states (Idle, Power, and Interaction). These states are listed in Table 6.1.

When comparing this dataset to the active scenario in the CIC IoT Dataset 2022 (CICIoT22) dataset, which was examined in Chapter 5, it's important to note that the active situation in the CICIoT22 dataset involved an extended data collection period lasting up to 30 days. Throughout this duration, data was systematically gathered from devices in the active sections at specific intervals. In contrast, the MonIoTr dataset features a significantly shorter collection period, totalling 112 hours. However, the data collection patterns in both scenarios exhibit striking similarities. During this 112-hour window, the devices were actively engaged in various interactions and power-related activities. Therefore, given the significant similarity, the categorisation of this scenario as "Active" is quite consistent.

Using active sessions containing device usage data for device identification is much more effective than using idle states containing idle/passive states (this situation is discussed in Chapter 5). This is because active states also cover the idle state of the

Table 6.1: Data separated by country, active/idle and VPN state.

Idle UK
Idle US
Idle UK VPN
Idle US VPN
Active UK (Power UK + Interaction UK)
Active US (Power US + Interaction US)
Active UK VPN (Power UK VPN + Interaction UK VPN)
Active US VPN (Power US VPN + Interaction US VPN)

device, while the opposite is not possible. Our experiments with the MonIoTr dataset on this subject also support our findings in Chapter 5 (see Appendix A.4.1). For this reason, and in order not to overload the chapter with too many results, we will continue our experiments with active data.

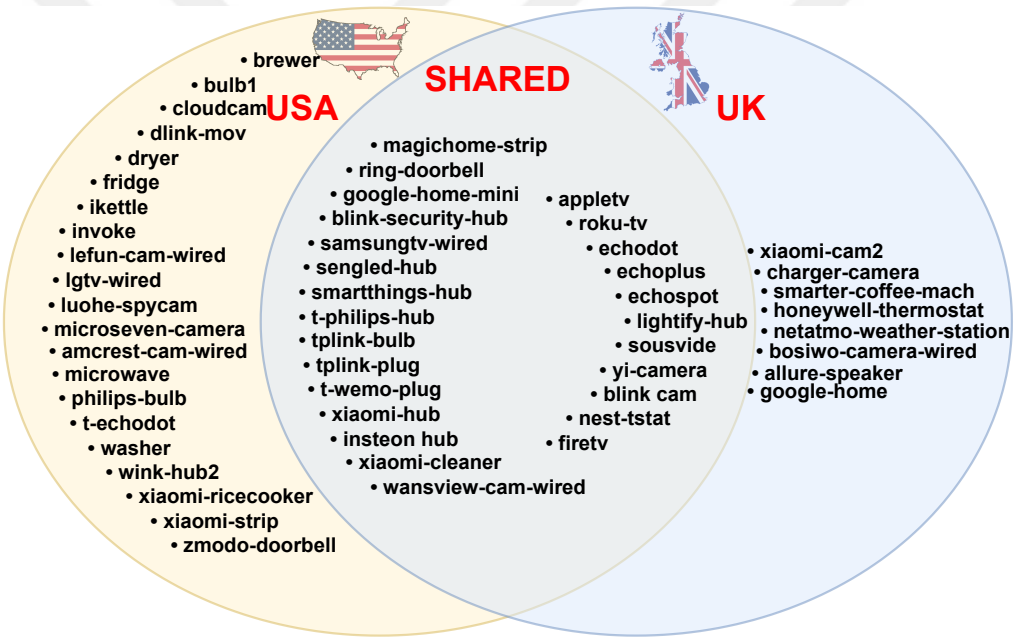


Figure 6.1: Venn diagram showing the distribution of devices by country. Intersection:, USA:, UK:. Adapted from [189]

Table 6.2 presents data generated by active state devices, categorised by country and data transfer status in terms of packet numbers. The table provides insights into the amount of data associated with each device within different countries and statuses. Comparing the information in Figure 6.1 with Table 6.2, it becomes apparent that the Xiaomi-cleaner device should be included in the shared section, but it is instead listed

under the UK-specific devices section. This discrepancy arises due to the absence of packets related to this particular device in the data collected from the US.

Table 6.2: Active sessions,USA∩UK:🟢, USA:🟡, UK:🔵

Device Names	Packet Number			
	Normal	VPN	Normal	VPN
appletv	13319	14685	17977	13775
blink-camera	30560	62151	83855	92434
blink-security-hub	2983	3771	3352	3863
echodot	11955	63302	31471	23483
echoplus	43742	51061	27440	35799
echospot	43504	53605	65150	83536
firetv	19479	20274	10487	25935
google-home-mini	13992	16102	19209	26735
insteon-hub	15067	17724	33	33
lightify-hub	7837	7916	3274	5245
magichome-strip	3667	3336	1138	1521
nest-tstat	13438	13508	25651	26854
ring-doorbell	441388	540796	820444	723017
roku-tv	10606	10095	5876	8900
samsungtv-wired	82579	34790	27148	37328
sengled-hub	4464	4967	3370	1642
smartthings-hub	25970	33576	21271	16856
sousvide	73	69	15842	14271
t-philips-hub	167841	136289	71515	29364
t-wemo-plug	40616	40155	32546	40069
tplink-bulb	19753	49492	14580	6715
tplink-plug	6812	16968	5965	5588
wansview-cam-wired	996276	823173	802384	907544
yi-camera	325660	277292	261276	281210
xiaomi-hub	2897	2588	6394	9298

Device Names	Packet Number	
	Normal	VPN
amcrest-cam-wired	504375	252423
brewer	650	651
bulb1	6897	7122
cloudcam	144224	181797
dlink-mov	394	411
dryer	9064	8348
fridge	93653	79603
ikettle	4522	4328
invoke	7108	7929
lefun-cam-wired	378072	314145
lgtv-wired	56608	24676
luohe-spycam	245945	115184
microseven-cam	146908	45412
microwave	554	557
philips-bulb	3860	3862
t-echodot	19151	7186
washer	8362	9395
wink-hub2	15751	12558
xiaomi-ricecooker	279	242
xiaomi-strip	5932	5543
zmodo-doorbell	157752	164545

Device Names	Packet Number	
	Normal	VPN
allure-speaker	20554	16916
bosiwo-camwired	359580	698552
charger-camera	563	965
google-home	25854	19382
honeywell-thermostat	2779	2503
netatmo-weather-stat	603	340
smarter-coffee-mach	1665	1356
xiaomi-cam2	346227	456611
xiaomi-cleaner	243	295

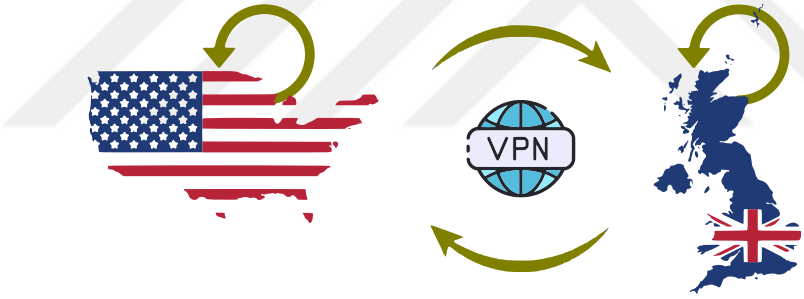


Figure 6.2: Data collection visualisation involves collecting Idle, Power, and Interaction data in each country's laboratory. Additionally, Idle, Power, and Interaction data are collected for each country by utilising a VPN connection from the other laboratory.

6.3 Experiments

In this study, we apply the IoTDevID methodology to the MonIoTr dataset to assess its model generalizability and effectiveness on encrypted data. As a result, individual feature and model selection steps are not part of this work, as we inherit the feature and model selection from IoTDevID.

The pcap files obtained from the MonIoTr dataset source website[1] are converted into Comma-separated values (CSV) files by the feature extraction script. Each network packet's source MAC address is labelled using the device list provided by the dataset creator[2]. While our feature extraction script can generate over 100 features, we will focus on the selected features[3] and the machine learning algorithm (Decision Tree (DT)) from IoTDevID.

6.3.1 Comparison of VPN and Normal Sessions in Same Laboratory.

The MonIoTr dataset includes a version where data collection occurs through a VPN. A VPN establishes a private network over a public network, utilising an encrypted tunnel to ensure secure and private communications. In this dataset, idle data is collected locally in the USA laboratory and then the same data is collected again by connecting from the UK laboratory via VPN. These cases are labeled "US" and "US-VPN" respectively. By using this data collected from the same lab during separate sessions, we once again validate our IoTDevID method, while also addressing the concerns of non-generalizability that the use of IP field features can lead to, thanks to the normal-VPN distinction given by this dataset.

Table 6.3 presents the machine learning experiments conducted using the data collected from both countries and their respective VPN equivalents. Table 6.3 showcases the use of normal data for training and VPN data for testing, as well as the reverse scenario where VPN data is used for training and normal data for testing.

Upon analysing the individual results in Table 6.3, it is evident that a classification process involving 34 classes for the UK case and 46 classes for the US case yields remarkable success across all scenarios. Specifically, in the UK-VPN|UK case, an F1 score of 78.30% is achieved, while in the reverse scenario (UK|UK-VPN), the F1 score is slightly lower at 75.50%, a difference of approximately 3 points. Similarly, in the US-VPN|US case, an F1 score of 74.80% is attained, while in the opposite case, an F1 score of 67.00% is obtained, representing a difference of approximately 7 points. In all four cases, the aggregation algorithm significantly contributes to the performance, ranging

[1] MonIoTr dataset source: moniotrlab.khoury.northeastern.edu/publications/imc19/

[2] Device-MAC list: github.com/NEU-SNS/intl-iot/blob/master/moniotr/etc/devices.txt.example

[3] Selected features are: pck_size, Ether_type, LLC_ctrl, EAPOL_version, EAPOL_type, IP_ihl, IP_tos, IP_len, IP_flags, IP_DF, IP_ttl, IP_options, ICMP_code, TCP_dataofs, TCP_FIN, TCP_ACK, TCP_window, UDP_len, DHCP_options, BOOTP_hlen, BOOTP_flags, BOOTP_sname, BOOTP_file, BOOTP_options, DNS_qr, DNS_rd, DNS_qdcount, dport_class, payload_bytes, entropy

Table 6.3: Comparison of VPN and normal data. The scores given are the mean and standard deviation of 25 repetitions.

	Train | Test	Accuracy	Precision	Recall	F1 Score	Train-t	Test-t	Alg-t
Individual	UK|UK-VPN	0.845±0.002	0.796±0.013	0.760±0.006	0.755±0.013	25.964	1.509	0.000
	UK-VPN|UK	0.882±0.002	0.827±0.010	0.775±0.010	0.783±0.010	23.548	1.633	0.000
	US|US-VPN	0.789±0.003	0.722±0.007	0.680±0.005	0.670±0.010	39.828	2.075	0.000
	US-VPN|US	0.842±0.002	0.776±0.005	0.739±0.007	0.748±0.007	34.957	2.819	0.000
Session	UK|UK-VPN	0.893±0.002	0.904±0.019	0.843±0.007	0.831±0.016	23.662	1.610	55.072
	UK-VPN|UK	0.928±0.001	0.946±0.006	0.861±0.013	0.866±0.014	23.255	1.706	66.287
	US|US-VPN	0.888±0.003	0.877±0.012	0.810±0.005	0.802±0.012	39.179	2.024	61.870
	US-VPN|US	0.942±0.002	0.925±0.005	0.868±0.006	0.883±0.006	29.787	2.190	69.794

from a minimum improvement of 7 points to a maximum improvement of 13 points. The fact that the UK results appear better than the US results can be attributed to the higher number of US devices (the number of US devices is approximately one third higher than the number of UK devices). Additionally, it is worth noting that both scenarios exhibit notably higher success when VPN data is used for training and normal data for testing. To further investigate the underlying reasons for this, we will delve into the class-based results in Figure 6.3 and Figure 6.4. In these figures, the results of both cases in terms of F1 score are grouped for each device in bar graphs. Thus, the increase and decrease between the states can be observed. Numerical results can be found in Appendix Table A.10 and Table A.11.

For the UK data, while the UK-VPN|UK results exhibit significantly higher overall scores, we observe no consistent pattern supporting this increase when comparing individual results in Figure 6.3. When examining the significant differences in individual results (difference of 4 points or more), ten devices (2, 3, 12, 17, 19, 20, 27, 29, 32) achieved higher UK-VPN|UK scores, while eight (7, 8, 11, 13, 14, 24, 34) achieved higher UK|UK-VPN scores, with no significant change observed in the other 18 devices.

Among the variations observed in Figure 6.3, three instances stand out – *blink-camera*, *tplink-plug*, and *xiaomi-cleaner* – due to their substantial differences exceeding 10 points. Of these three devices, xiaomi-cleaner is unique to the UK laboratory, while *blink-camera* and *tplink-plug* intersect with the US laboratory. The *xiaomi-cleaner* exhibits a notable characteristic of exceptionally low packet counts in both sessions (see Table 6.2). Beyond the limitation of a small sample size, this device also presents issues related to sample quality. Examining the raw data for this device reveals that the majority of the data is from power state and usage data is missing. Specifically, there are merely four usage samples for the UK session, and the UK-VPN session lacks any usage

samples altogether. We think that this skewed data distribution has a serious negative impact on data quality.

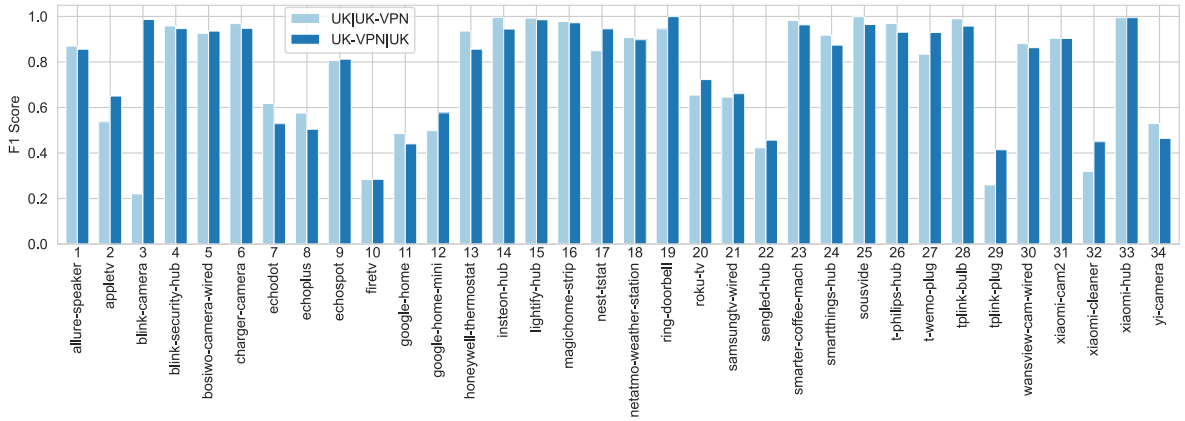


Figure 6.3: VPN and Normal data comparison with data collected from MonIoTr UK laboratory

Similarly, analysing the significant differences for the US in Figure 6.4, 19 devices (1, 3-5, 9-11, 13-15, 18, 19, 25, 30, 37, 38, 42, 44, 46) had higher scores in the US-VPN|US case, while 5 devices (22, 31, 32, 41, 45) had higher scores in the US|US-VPN case. In both scenarios, the VPN data trained case achieved higher scores. In the US laboratory, distinct performance disparities exceeding 10 points are evident between sessions for specific devices, namely *amcrest-cam-wired*, *dryer*, *fridge*, *invoke*, and *microwave*. On all of these devices, the US—US-VPN session is less successful. When we make a comparison based on the number of samples, we see that on many devices in the US dataset the number of samples is higher than US-VPN. For example, in the five devices (*amcrest-cam-wired, microseven-cam, t-echodot, lgtv-wired, luohe-spycam*) the number of samples is 2x or more of the other dataset.

When we examine these redundancies on the raw data, they are usually empty packets that are very difficult to sort (e.g., TCP packets with the ACK flag set and no payload). The sheer number of these examples caused noise in the training data, negatively affecting the classification of some devices. These relationships can be seen more clearly on the confusion matrix. Either the devices themselves showed this degradation, as in the case of the *amcrest-cam-wired* device, or it affected other devices. For example, we can see that all samples of *microwave* devices are almost perfectly classified as this device (high recall), but many *tplink-bulb* devices are classified as microwave (low precision). As a result of these processes, the F1 score of the *microwave* device in this

session is very low.

Dryer, fridge, and *invoke* devices are also negatively affected by this noise and we think that this noise is the reason for the big difference between the two sessions. In addition to devices contributing to the noise, we observe heightened confusion particularly with *echodot, echoplus, echospot*, and *google-home-mini* devices, where low-scoring devices are frequently misclassified as one of these four devices, and vice versa. We think this is mainly because these devices interact with other devices in the dataset, e.g., devices are often used through these four devices.

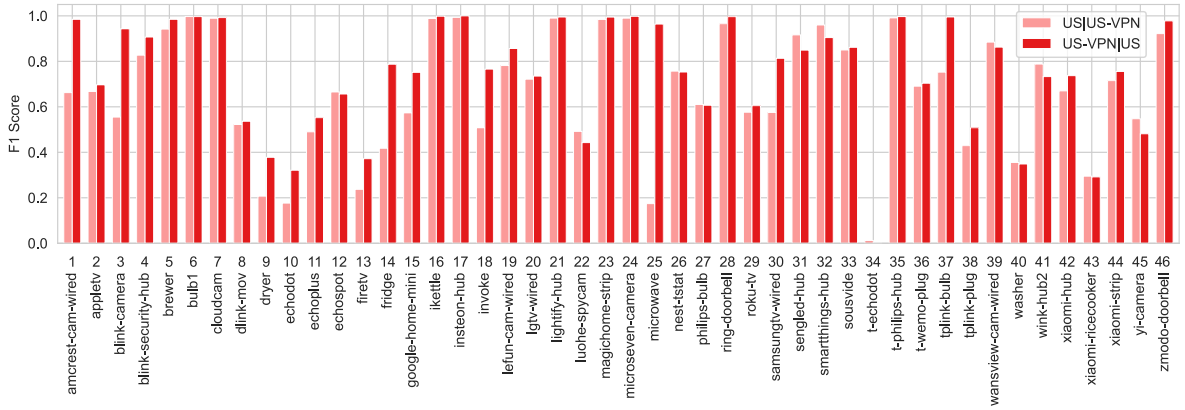


Figure 6.4: VPN and Normal data comparison with data collected from MonIoTr US lab

In both laboratories, sessions trained with VPN data and tested with normal data are more successful. However, it is essential to note that this higher score is not systematic. The reason why VPN-trained models perform better than normal models lies in the variation in success rates for specific devices, rather than a general rule. In the UK and US data, 25 devices overlap. Analysing the results of these devices in all four cases will provide deeper insights, as depicted in Figure 6.5, showing the distribution for the 25 devices in all four scenarios. Among the intersecting devices, only three devices (2, 8, 22) perform better in both laboratories when VPN is used as training data and normal data is used as test data. In most other cases, either the opposite is true (17, 25), and for other devices do not have a generalizable pattern.

The examination of three devices in Figure 6.4 reveals intriguing disparities. *xiaomi-hub, sengled-hub*, and *echodot* devices exhibit noteworthy performance differences between laboratories, with positive outcomes in the UK and suboptimal results in the US, and vice versa.

Specifically, the *sengled-hub* device demonstrates results exceeding 80 in the US labo-

ratory but falling below 50 in the UK laboratory. Table 6.2 illustrates the sample counts in the sessions, and the sample count is identified as a potential explanatory factor for these anomalies. Upon scrutinizing the sample counts for the *sengled-hub* device, we observe close proximity in counts within the US laboratory, while in the UK laboratory, normal and VPN sample counts diverge significantly, with the latter being double the former. We posit that this imbalance in sample counts is the primary factor contributing to the observed differences in laboratory results.

Turning attention to the *echodot* device, the UK laboratory outperform the US visibly. A closer inspection of sample counts reveals a relatively balanced distribution of VPN-normal samples in the UK laboratory (Normal/VPN ratio of 1.3). In contrast, the US laboratory exhibits a stark contrast, with the number of VPN samples surpassing the number of normal samples by over fivefold. We attribute this anomalous distribution as the key factor behind the subpar results.

For the *xiaomi-hub* device, the VPN-normal sample counts are comparable between both laboratories, yet the intriguing distinction lies in the overall sample counts. In the superior-performing UK laboratory, the sample count is approximately three times that of the US laboratory.

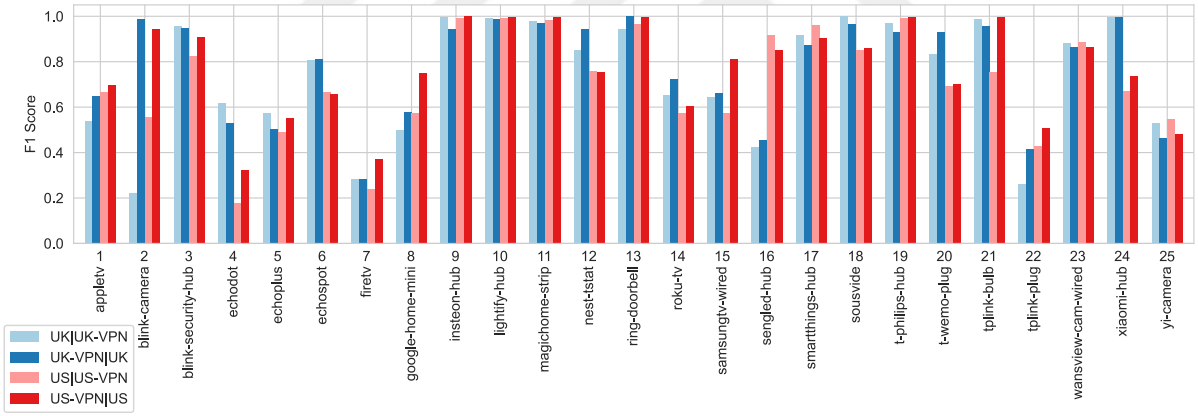


Figure 6.5: VPN and Normal data comparison in the MonIoTr dataset with data from both UK and USA laboratories

In the comparison of VPN vs. Normal data, we did not compare data collected in different laboratories each other. We found it more appropriate to present this comparison under model generalizability. In this regard, you can find more examples of comparisons of VPN vs. Normal data in the next section as a part of model generalizability.

6.3.2 Measurement of Model Generalizability

Device identification aims to ensure that a model developed for a device can effectively identify that device in diverse environments. In IoTDevID, our focus was on acquiring generalizable models by mitigating overfitting during feature and model selection, and by developing methods suitable for various environments. However, datasets available for device identification studies often lack the necessary device intersections required to measure model generalizability. Consequently, testing a model trained on one dataset (A) on another dataset (B) to demonstrate generalizability is highly limited. This represents a significant challenge in device identification research, with no existing work testing model generalizability on different datasets.

To address this challenge, we have previously explored model generalizability by employing data collected in different sessions (data collected on different timeframes) within the same datasets for training and testing. However, the MonIoTr dataset offers a valuable opportunity to overcome this limitation. It comprises a diverse collection of devices across two distinct laboratories, featuring variations in interacting devices, physical environments, and users. With over 20 intersecting devices, the MonIoTr dataset closely aligns in terms of device number with benchmark datasets commonly used in device identification research. Consequently, testing these intersecting devices using the isolated data from each laboratory as separate datasets will provide valuable insights into model generalizability.

In order to measure the generalizability of the model, we made the following approach. We filtered the intersecting devices in each dataset and obtained sub-datasets consisting of 25 devices. In this way, we have four different datasets containing 25 devices: UK, UK-VPN, US, US-VPN. Using these datasets, we used each of the country-specific datasets for training and testing. For example, we trained our model with UK laboratory data and tested it with US data. In this context, this gave us eight different cases. These cases and the results obtained from these cases are given in Table 6.4.

Table 6.4: Results obtained with data collected from different country laboratories

	Train | Test	Accuracy	Precision	Recall	F1 Score	Train-t	Test-t	Alg-t
Individual	UK|US	0.788±0.007	0.723±0.017	0.720±0.018	0.694±0.022	12.935	0.964	0.000
	UK|US-VPN	0.688±0.007	0.669±0.015	0.626±0.011	0.595±0.018	15.891	0.979	0.000
	UK-VPN|US	0.765±0.004	0.703±0.016	0.669±0.015	0.651±0.020	19.657	0.975	0.000
	UK-VPN|US-VPN	0.727±0.002	0.723±0.014	0.709±0.007	0.681±0.008	20.010	0.883	0.000
	US|UK	0.835±0.004	0.782±0.025	0.783±0.013	0.760±0.022	9.826	1.024	0.000
	US|UK-VPN	0.758±0.004	0.731±0.011	0.682±0.015	0.679±0.014	14.685	1.365	0.000
	US-VPN|UK	0.796±0.002	0.778±0.015	0.737±0.007	0.723±0.014	12.538	1.034	0.000
	US-VPN|UK-VPN	0.760±0.001	0.801±0.007	0.802±0.001	0.783±0.004	16.584	1.029	0.000
Aggregated	UK|US	0.852±0.006	0.904±0.031	0.856±0.023	0.849±0.036	11.570	1.044	36.588
	UK|US-VPN	0.783±0.033	0.786±0.026	0.723±0.011	0.711±0.018	15.254	1.093	37.362
	UK-VPN|US	0.836±0.003	0.866±0.029	0.786±0.026	0.782±0.030	19.485	1.027	41.394
	UK-VPN|US-VPN	0.808±0.003	0.844±0.017	0.806±0.018	0.812±0.020	18.481	0.901	40.390
	US|UK	0.952±0.002	0.965±0.012	0.930±0.017	0.935±0.018	9.470	0.997	40.672
	US|UK-VPN	0.890±0.007	0.883±0.013	0.782±0.016	0.774±0.017	13.410	1.116	40.599
	US-VPN|UK	0.927±0.003	0.929±0.002	0.874±0.005	0.857±0.005	10.854	0.961	39.509
	US-VPN|UK-VPN	0.876±0.002	0.938±0.002	0.903±0.002	0.890±0.003	14.008	0.998	42.084

Table 6.4 presents the results obtained from various versions of the datasets for the 25 intersecting devices, focusing on F1 scores in 8 different cases. The highest score (78.30) is achieved with US-VPN|UK-VPN, followed closely by US|UK (76.00), while the lowest score is observed with UK|US-VPN (59.50).

In the case of individual packets, low performance is to be expected. We have already introduced the aggregation algorithm (see Algorithm 1) to cope with this situation. Analysing the aggregated results, it can be seen that it leads to a substantial increase of 9 to 17 points in all eight cases. With the application of the algorithm, the lowest F1 score increased to 71.1% (UK|US-VPN) and the highest increased to 93.5% (US|UK).

A more comprehensive insight can be obtained through the analysis of the class-based results presented in Table 6.5. In this table, aggregated results have been utilized, as they effectively emphasize the distinctions and underperforming devices by optimizing where feasible. For individual class-based results, refer to the Appendix Table A.12.

We will scrutinize the underperforming devices highlighted in red shades in Table 6.5. Notably, the *blink-camera* device exhibits suboptimal performance in both the UK|US-VPN and US|UK-VPN scenarios, despite achieving satisfactory results (85 F1 Score and above) in the remaining six cases. Analysis of the confusion matrices (see Figure A.7c & A.8c) for this device reveals consistent mislabelling of *blink-camera* samples as *ring-doorbell* in both instances. This misclassification may be attributed to the shared functionality of both devices as cameras, exacerbated by the anomaly resulting from the

Table 6.5: Aggregated class-based results of the evaluation using data from two different laboratories

Device	Train→ / Test→	UK / US	UK / US-VPN	US / UK	US / UK-VPN	UK-VPN / US	UK-VPN / US-VPN	US-VPN / UK	US-VPN / UK-VPN
appletv		0.942	0.888	0.964	0.888	0.977	0.998	0.923	0.869
blink-camera		0.973	0.067	0.994	0.168	0.858	0.890	0.995	0.999
blink-security-hub		0.999	0.999	0.998	0.998	0.999	0.999	0.998	0.998
echodot		0.036	0.000	0.868	0.006	0.018	0.000	0.635	0.370
echoplus		0.838	0.559	0.801	0.721	0.796	0.459	0.799	0.785
echospot		0.718	0.435	0.947	0.972	0.699	0.341	0.986	0.991
firetv		0.773	0.259	0.880	0.365	0.818	0.890	0.586	0.933
google-home-mini		0.974	0.984	1.000	0.980	0.990	0.999	0.995	0.991
insteon-hub		0.486	0.080	0.939	1.000	0.120	0.080	1.000	1.000
lightify-hub		0.999	0.999	0.956	0.805	0.997	0.998	0.996	0.971
magichome-strip		0.980	1.000	0.965	0.996	0.999	1.000	0.996	0.996
nest-tstat		0.990	0.986	1.000	0.998	0.996	1.000	0.989	1.000
ring-doorbell		1.000	0.932	1.000	0.946	1.000	1.000	1.000	1.000
roku-tv		0.910	0.925	0.974	0.730	0.891	0.909	0.844	0.854
samsungtv-wired		0.990	0.836	0.966	0.687	0.947	0.976	0.909	0.836
sengled-hub		0.866	0.563	0.462	0.233	0.608	0.816	0.034	0.193
smartthings-hub		1.000	1.000	1.000	0.999	1.000	1.000	1.000	1.000
sousvide		0.815	0.966	0.987	0.990	0.852	0.944	0.979	1.000
t-philips-hub		1.000	0.999	0.998	1.000	0.999	0.999	0.997	1.000
t-wemo-plug		0.991	0.896	0.963	0.976	0.976	0.980	0.936	0.982
tplink-bulb		0.934	0.976	0.997	0.999	0.966	0.994	0.999	0.999
tplink-plug		0.999	0.154	0.998	0.176	0.180	1.000	0.156	0.998
wansview-cam-wired		0.867	0.819	0.944	0.937	0.858	0.813	0.910	0.834
xiaomi-hub		0.994	0.996	0.976	1.000	0.987	0.987	0.999	0.999
yi-camera		0.146	0.464	0.789	0.769	0.021	0.239	0.766	0.651
Accuracy		0.852	0.783	0.952	0.890	0.836	0.808	0.927	0.876
F1 Score		0.849	0.711	0.935	0.774	0.782	0.812	0.857	0.890

Table 6.6: Intersecting devices in UK and US laboratories and the number of packets they contain.

Device Names	Packet (Sample) Number			
	US	US-VPN	UK	UK-VPN
appletv	13319	14685	17977	13775
blink-camera	30560	62151	83855	92434
blink-security-hub	2983	3771	3352	3863
echodot	11955	63302	31471	23483
echoplus	43742	51061	27440	35799
echospot	43504	53605	65150	83536
firetv	19479	20274	10487	25935
google-home-mini	13992	16102	19209	26735
insteon-hub	15067	17724	33	33
lightify-hub	7837	7916	3274	5245
magichome-strip	3667	3336	1138	1521
nest-tstat	13438	13508	25651	26854
ring-doorbell	441388	540796	820444	723017
roku-tv	10606	10095	5876	8900
samsungtv-wired	82579	34790	27148	37328
sengled-hub	4464	4967	3370	1642
smartthings-hub	25970	33576	21271	16856
sousvide	73	69	15842	14271
t-philips-hub	167841	136289	71515	29364
t-wemo-plug	40616	40155	32546	40069
tplink-bulb	19753	49492	14580	6715
tplink-plug	6812	16968	5965	5588
wansview-cam-wired	996276	823173	802384	907544
yi-camera	325660	277292	261276	281210
xiaomi-hub	2897	2588	6394	9298
Average	93779	91907	95106	96841

exceptionally high packet count of *ring-doorbell* (among the devices with the highest sample count in the dataset, refer to Table 6.6). This observation aligns closely with the findings in the US|US-VPN and UK|UK-VPN cases examined in the preceding section.

Another device exhibiting notably low success rates is the *echodot* device, except in two cases (US|UK: 86.8 and US|UK-VPN: 63.5) where results are comparatively better. The poor performance manifests in two distinct patterns based on the training data source. In instances where UK data is employed for training, a confusion arises between *echodot* and *wansview-cam-wired*. Notably, *echodot* samples are inaccurately classified as *wansview-cam-wired*, as illustrated in confusion matrices (see Figure A.7). Across all four cases, the *wansview-cam-wired* device boasts the highest sample count, contributing to the confusion. Conversely, when US data serves as the training set, the rationale behind the diminished success differs. Analysis of the confusion matrices, as depicted in Figure A.8, reveals that the *echodot* device is frequently misidentified with other devices of the same brand, namely *echodot* and *echoplus*. This deficiency in success arises from the classification of these three echo devices as a single entity due to their high similarity in terms of usage purpose (*echospot, echodot*, and *echoplus*). However, as discussed in Chapter 4 and 5, this phenomenon presents a challenging and probable issue in device identification.

The *firetv* device is among those presenting classification challenges. Despite seemingly normal packet counts, a substantial issue is evident in the data collection phase for this device. Notably, no usage data was collected in the UK lab, leading to the reliance on power state data for all representations of this device in that particular lab. We attribute the disparity in data collection methodologies between the UK and US laboratories as a significant contributor to the device's classification failure.

Similarly, the *insteon-hub* device encounters a comparable challenge, lacking usage data in the UK lab. Unlike *firetv*, this deficiency is reflected in the packet counts, with the US lab accumulating over 15,000 samples in a session, whereas the UK laboratory possesses only 33 samples (see Table 6.6). This skewed data distribution significantly diminishes the success rate, particularly when employing UK data for training.

In the case of the *sengled-hub*, consistently low success levels are observed in all but two cases (UK|US, UK-VPN|US-VPN). The primary factor contributing to this poor performance is the misclassification of numerous *sengled-hub* packets as *t-wemo-plug* (see Figure A.7 and A.8). However, owing to the substantial difference in packet counts (*t-wemo-plug* being approximately eight times larger), *sengled-hub* is more severely impacted by this complexity.

The *tplink-plug* device exhibits notably low success in four sessions, particularly characterized by the failure of VPN vs Normal versions. No issues are identified in Normal vs Normal or VPN vs VPN scenarios. Upon inspecting the confusion matrices, it becomes apparent that packets attributed to this device are predominantly misclassified as belonging to the *firetv* device. We postulate that the anomaly observed in the *firetv* device significantly influences this misclassification, thereby impacting the success of the *tplink-plug* device.

Lastly, the *yi-camera* device is also identified as one of the unsuccessful devices. Confusion matrices analysis indicates that a substantial number of packages from this device are erroneously classified as *wansview-cam-wired*. This misclassification can be attributed to the similarity in tasks performed by both devices and their unusually high packet counts. Notably, *wansview-cam-wired* less affected by this misclassification, primarily due to its approximately threefold higher sample count compared to the *yi-camera device*.

The MonIoTr dataset is advantageous in that it provides the possibility to test model generalisability with another IoT device. However, as the name suggests, this dataset was not originally created for device identification, but is focused on information exposure. Nevertheless, we have chosen to favour this dataset because of its attractive features. However, some features of the dataset make it difficult to use for device identification purposes. For example, this dataset consists of automated data collected over a relatively short period of time, rather than over a long period of device use. In this context, there is not enough or qualified data for some devices. During the data collection phase, specific devices encounter various issues. For instance, in the UK laboratory, data collection for certain devices (such as *firetv, xiaomi-cleaner*, and *insteon-hub*) is limited to the power phase, indicating personal errors rather than inherent device nature, as the same devices exhibit complete usage data in the US laboratory. Moreover, devices like Ring-doorbell, *wansview-cam-wired*, and *yi-camera* present an exceptionally high volume of data. IoT devices may be characterised by heterogeneity, with some devices tending to generate much more data than others, but the fact that these devices contain excessive amounts of data that undermine the identification process cannot be ignored. For example, the top three devices with the highest sample counts are *ring-doorbell* (2,525,645), *wansview-cam-wired* (3,529,377), and *yi-camera* (1,145,438). Conversely, the remaining 23 devices collectively contribute 2,240,366 samples. Notably, individual devices such as *ring-doorbell* or *wansview-cam-wired* alone generate more data than the

rest of the dataset, significantly complicating device identification. Addressing these anomalies can be achieved through data organization, specifically downsampling these devices.

6.4 Conclusions

This study represents our final work on device identification. So far we have analysed the literature in Chapter 3, developed and tested our device identification methodology in Chapter 4, and validated our methodology extensively by investigating the effects of data quality on device identification in Chapter 5. In this final chapter of the device identification part, we investigated the impact of our methodology on encrypted data (R1.6, C1.6), as well as checking model generalizability (R1.7, R1.8, C1.5, C1.7), i.e., the effectiveness of a previously created model on a new/different device.

The dataset includes data from different sessions (normal and VPN) from the same laboratory. These 2 different types of data also reduce concerns about the generalizability and representativeness of the feature set. The model achieves 88.3% F1 score on US data and 86.6% F1 score on UK data when using different sessions for training and testing.

The second approach centred on model generalizability. By leveraging the bipartite structure of the MonIoTr dataset and training on US laboratory data while testing on UK laboratory data, we achieved an F1 score of 91.90% for 25 overlapping devices. Thus, we demonstrate our method's model generalizability, affirming its capability to detect different devices of the same make and model. In addition, this dataset contains a large amount of encrypted data. In this context, the high success achieved here shows that our identification method works very well on encrypted payloads.

Table 6.7 presents the datasets used in each of these procedures and the corresponding best results. Remarkably, these results closely align with our previous findings, reaffirming the high success achieved with the IoTDevID methodology. Furthermore, this success demonstrates its generalizability and capability to handle encrypted data effectively.

Table 6.7: The datasets used in the studies with the IoTDevID method and the results obtained.

Dataset	Device No	Accuracy	F1 Score
Aalto University - IoT Devices Captures (Aalto)	27	0.833±0.002	0.861±0.004
UNSW Merged (UNSWM)	32	0.941±0.012	0.935±0.017
Locality-Sensitive IoT Fingerprinting (LSIF)	22	0.994±0.000	0.931±0.001
CICIoT22	32	0.943±0.001	0.925±0.007
$\text{MonIoTr}_{\text{VPN\|UK}}$	34	0.928±0.001	0.866±0.014
$\text{MonIoTr}_{\text{VPN\|US}}$	46	0.942±0.002	0.883±0.006
$\text{MonIoTr}_{\text{US\|UK}}$	25	0.952±0.000	0.935±0.001

Part II

IoT Attack Detection

Chapter 7

Literature Review for IoT Attack Detection

In the first part of our thesis, we focused on device identification. However, the secure operation of the device identification system can only be ensured if the data it works with is benign. Consequently, the implementation of an attack detection system becomes imperative to safeguard the network against potential attacks and to ensure the efficacy of the device identification process.

This chapter presents a literature review for IoT attack detection. In this context, the extensive literature on attack detection in IoT networks is reviewed based on datasets, features, machine learning methods and results. The latter chapters of this part can be summarised as follows: Chapter 8 explores the drawbacks of using individual packets for attack detection, a common practice in existing literature. Chapter 9 presents an extensible attack detection methodology, emphasising the importance of achieving generalizability and improved performance in IoT network-based attack detection. The following draft publication has been utilised in the preparation of this section.

- K. Kostas, M. Just, and M. Lones. IoTGeM: Generalisable Models for Behaviour-Based IoT Attack Detection :github.com/kahramankostas/IoTGeM

7.1 Overview

This section contains a literature review on attack detection, which is the second of the two themes of our study. With this review, we aim to provide a comprehensive overview of the research landscape on IoT attack detection. In this way, we aim to familiarise our-

selves with the work in the literature, the emerging issues and their possible solutions. In addition, we provide information on the tools, methods, algorithms and results obtained in attack detection. This analysis aims to provide a comprehensive understanding of the performance, accuracy and limitations of IoT attack detection techniques.

7.2 Related Work

7.2.1 Literature Review Criteria

This section is the output of a literature review we conducted in mid-March 2023. The initial stage of this literature review centred on obtaining a thorough and moderately detailed grasp of the research environment in the selected domain.

We used some constraints while searching the literature. In our search, we paid attention to the articles having the following conditions.

- Published in 2019 and later,
- IoT focused,
- Including anomaly detection or intrusion detection,
- Used supervised machine learning methods.

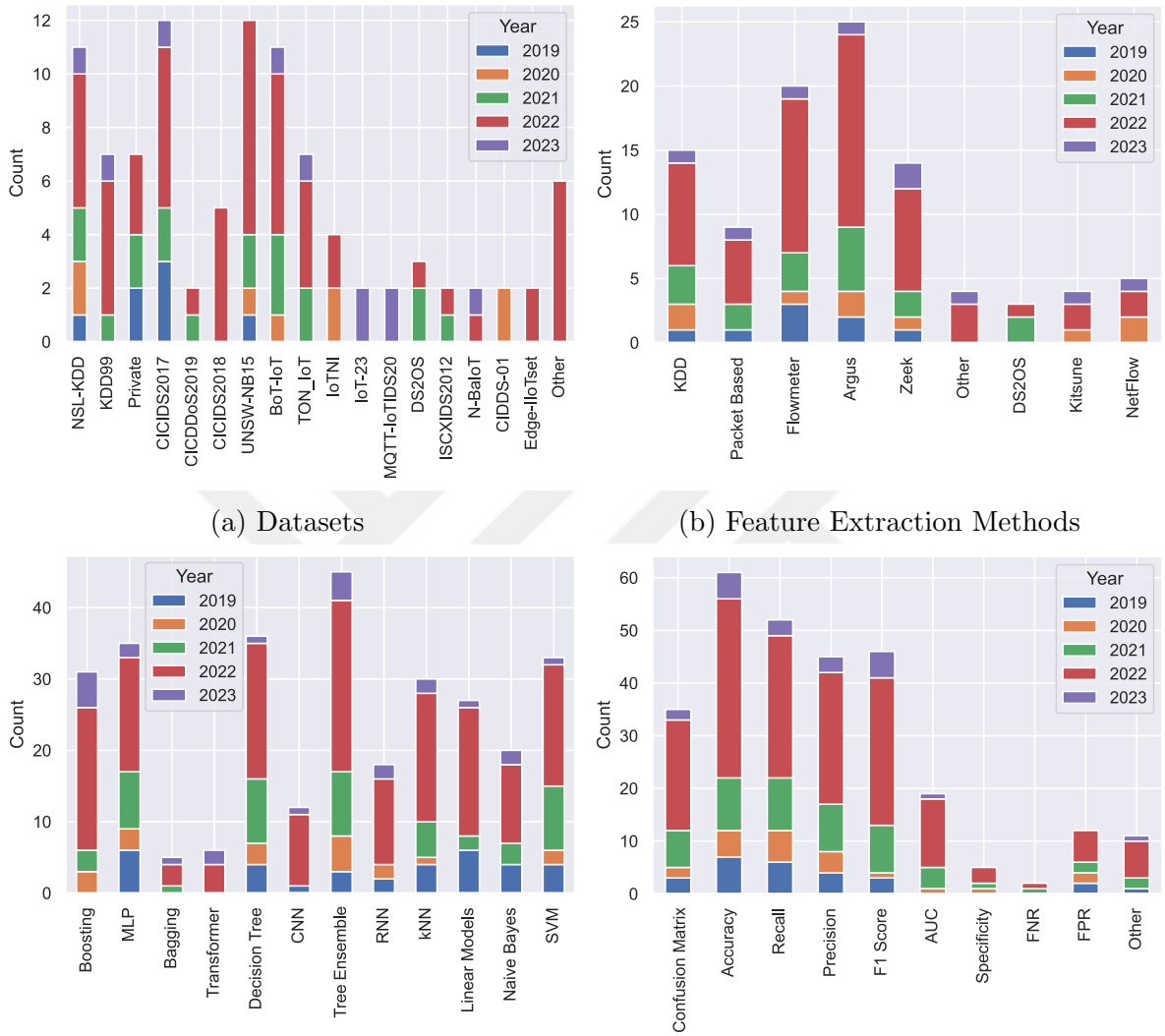
We did not include Wireless Sensor Network (WSN), which is included under IoT in many studies. This is because WSN studies generally focus on radio frequency-based attacks at the physical layer, which is beyond our scope of research. Firstly, we created a search in Google Scholar using the conditions mentioned above. Our Boolean search strategy used keywords as follows: "(IoT OR internet of things) AND ("anomaly detection" OR "Intrusion detection" OR IDS) AND (machine learning OR deep learning) AND (Supervised)". At the end of this process, we found about 350 articles. As a result of our review of these articles, we eliminated unrelated ones (the articles do not include IoT devices, intrusion/attack detection, or ML methods). As a result of this elimination, 68 studies with 99 experiments remained. Table 7.1 provide a comprehensive listing of this literature on ML approaches within IoT-based attack detection. Here, we consider four components of these previous approaches—their use of data, features, ML algorithm(s), and evaluation metrics as summarised in Figure 7.1—and use this to frame our discussion of the limitations of previous studies.

Table 7.1: Comparison of related work on IoT attack detection. Only the best scores are included in the evaluation section.

		Dataset																			Machine Learning														Feature Type									Evaluation					
Year	Study	KDD99	NSL-KDD	ISCXIDS2012	CICIDS2017	CICIDS2018	CICDDoS2019	UNSW-NB15	BoT-IoT	TON_IoT	IoTNI	IoT-23	DS2OS	N-BaIoT	Private	CIDDS-01	Edge-IIoTset	MQTT-IoTIDS20	Other	Bagging	Boosting	CNN	Decision Tree	Tree Ensemble	kNN	Linear Models	MLP	Naive Bayes	RNN	SVM	Transformer	Argus	DS2OS	CICFlowmeter	KDD	Kitsune	NetFlow	Other	Packet Based	Zeek	Confusion Matrix	Accuracy	Recall	Precision	F1 Score	AUC
2019	[161]		•																					•						•				•							93	92	100	96	
	[192]				•																	•		•		•	•	•					•								97	99	98		
	[97]				•																		•		•		•			•			•							•	98				
	[14]				•																		•		•	•		•					•								100	99	99	99	
	[84]							•																		•					•									•		84				
	[247]														•								•	•	•	•	•	•		•		•								•	100	97			
	[19]														•								•	•	•		•		•										•			98	99	99	
2020	[25]		•																											•											100	99			
	[31]								•																		•		•			•									•	100	100	100		
	[146]										•													•									•									100	100		
	[54]										•												•	•						•					•						•	100	99	99		
	[33]															•							•	•	•		•									•						100	100	100	100	
	[233]		•					•								•					•		•	•			•					•					•			•		95	100			99
2021	[194]		•																	•	•		•												•					•	100	99			
	[145]	•																					•		•		•	•		•					•						95	91	96	93	
	[73]						•														•		•	•										•								98	98	98	98	
	[5]							•																•			•			•		•								•	•	99				
	[44]								•														•	•	•	•	•	•		•		•									•	97	97	97	97	
	[1]								•	•																	•			•		•										100	100	100	100	100
	[2]									•											•		•									•									•	100	96	99		100
	[187]												•										•	•	•								•								•	100	100	100	100	100
	[157]												•										•	•			•			•			•	•								99	98	98	98	
					•																		•	•			•			•			•	•								99	98	99	99	
	[210]								•															•			•			•		•		•						•	•	99	99	98	98	
				•																				•			•			•		•		•						•	•	100	100	99	100	
			•																					•			•			•		•		•						•	•	96	94	94	94	
			•																					•			•			•		•		•						•	•	100	100	100	100	
								•																•			•			•		•		•						•	•	77	68	55	57	
	[51]																		•		•		•	•	•	•	•	•		•									•					100		
	[18]														•								•	•			•			•									•					100	100	100	
2022	[184]		•																					•		•				•				•							98					
	[6]		•																				•	•	•	•	•	•		•				•						•	79	79	84	78	96
	[245]	•																			•	•	•	•			•	•		•				•							93	92	100	95	98
			•																		•	•	•	•			•	•		•				•							81	81	97	82	92
	[166]														•						•		•	•	•	•		•											•		100	100	100	100	
	[60]	•																					•	•	•	•								•							•	99	99	100	100	
	[190]					•																•				•	•			•				•							•	96			95	98
	[49]					•															•			•		•		•						•							•					100
	[177]				•																•		•	•										•							•	100	100	100	100	100
						•															•		•	•										•							•	100	100	100	100	100
	[213]				•																		•	•	•									•							•	100	100	100	100	100
	[195]							•													•				•	•		•		•				•							•	100		100	100	
	[163]							•													•			•										•							•	•	90	90	90	90	
	[29]							•												•	•			•						•				•							•	•	96	96			
	[227]							•													•		•	•	•	•	•		•					•							•	•	98	98	98	98	
	[27]							•	•													•												•							•		99	99			
	[159]								•														•	•	•		•		•					•								•	100				
	[147]								•															•										•									98				
	[107]									•													•	•	•	•		•		•				•											100	100	
	[95]								•													•		•	•				•					•								•	99	99	99	99	
	[71]								•														•			•			•					•								•	99	99	100	100	100
	[193]	•																					•											•								•	100				

Table 7.1: Comparison of related work on IoT attack detection. Only the best scores are included in the evaluation section (continued).

Year	Study	Dataset																			Machine Learning												Feature Type									Evaluation					
		KDD99	NSL-KDD	ISCXIDS2012	CICIDS2017	CICIDS2018	CICDDoS2019	UNSW-NB15	BoT-IoT	TON_IoT	IoTNI	IoT-23	DS2OS	N-BaIoT	Private	CIDDS-01	Edge-IIoTset	MQTT-IoTIDS20	Other	Bagging	Boosting	CNN	Decision Tree	Ensemble Trees	kNN	Linear Models	MLP	Naive Bayes	RNN	SVM	Transformer	Argus	DS2OS	Flowmeter	KDD	Kitsune	NetFlow	Other	Packet Based	Zeek	Confusion Matrix	Accuracy	Recall	Precision	F1 Score	AUC
2022	[203]					●		●														●	●			●	●	●	●			●		●					●		98	95		94	97
																						●	●			●	●	●	●			●		●					●		99	92		92	96
										●												●	●			●	●	●	●			●		●					●		98	100		99	72
	[12]										●									●			●		●		●			●			●							●	100				
	[39]																		●			●					●						●			●				●	100	100	100	100	
					●																	●											●			●				●	100	100	100	100	
	[186]												●								●											●								●	99	96	100	98	100
	[78]														●								●	●	●	●		●		●							●				99	99	97	98	
	[10]										●									●			●		●		●							●	●					●	100	100	100	100	100
			●																	●			●		●		●							●	●					●	100	100	100	100	100
	[69]					●																	●				●		●					●					●		94	92	82	84	
																			●				●				●		●					●					●		91	91	77	80	
																			●				●				●		●					●					●		99	85	92	86	
	[82]	●																	●				●	●		●	●	●		●				●	●							100	100	100	
	[131]	●																						●	●		●			●			●	●					●	100				
			●																					●	●		●			●			●	●					●	100				
								●																●	●		●			●			●	●					●	97				
					●																			●	●		●			●			●	●					●	100				
	[66]													●							●			●											●					●	100	100	100	100	
	[4]														●									●	●		●			●							●				91	94	88	88	
	[164]									●											●				●		●										●				100	100			100
	[68]																●				●			●	●		●			●							●		●		100	95	89	87	
	[75]									●							●				●								●		●							●			100	100	100	100	100
	[13]							●													●		●	●	●	●	●			●	●									●	98	98	96	97	
	[83]			●																	●	●	●		●	●			●	●	●								●		●	99	99	99	99	
					●																●	●	●		●	●			●	●	●			●					●		●	98	98	98	98	
	[238]				●																						●		●	●	●			●								99	99	99	99	
							●																				●		●	●	●			●								99	99	99	98	
2023	[234]								●			●									●			●							●								●	99				
	[3]													●						●	●															●					100	100	100	100	
	[202]																	●					●	●	●	●		●		●								●			100	100	100	100	
	[183]											●															●											●	●			100	100	100	
																		●									●											●	●			100	100	100	
	[241]	●																					●	●				●	●		●			●	●					●	98			73	
			●																				●	●				●	●		●			●	●					●	92			82	
					●																		●	●				●	●		●			●	●					●	91			76	
	[235]									●											●			●			●				●					●					●	98			98	100



(a) Datasets (b) Feature Extraction Methods

(c) ML Algorithms (d) Evaluation Metrics

Figure 7.1: Usage of datasets, machine learning (ML) algorithms, feature types, and evaluation metrics in the literature over time.

Table 7.2: The datasets used in the literature and some properties of them. P: Packet-based, F: Flow-based, IPF: Individual Packet Feature, CFM: CICFlowMeter.

Dataset	year	IoT Data	Label	Raw Data	Feature Number	Feature Type	Attack number	Attacks								
								BoTnet	Brute-force	DDoS	DoS	Exploitation	Malware	MitM	Scanning	Web	Wi-FiAttack
AB-TRAP [51]	2021	no	P	yes	41	IPF	22								•		
AWID2 [112]	2015	no	P	no	154	IPF	7							•			•
BoT-IoT [113]	2018	yes	F	yes	29	Argus	10	•		•	•		•		•		
CICDDoS2019 [207]	2019	no	F	yes	83	CFM	13			•	•	•					
CICIDS2017 [205]	2017	no	F	yes	83	CFM	14	•	•	•	•	•			•	•	
CICIDS2018 [206]	2018	no	F	yes	83	CFM	16	•	•		•				•	•	
CIDDS-01 [191]	2017	no	F	no	15	NetFlow	4		•		•				•		
DS2OS [23]	2020	yes	F	no	12	Other	7				•		•		•		
ECU-IoHT [7]	2021	yes	P	no	7	Wireshark	3						•				•
Edge-IIoTset [68]	2021	yes	P	yes	61	IPF	14		•	•			•	•	•	•	
EMBER [17]	2018	no	P	no	51	Other	11				•		•				
InSDN [61]	2020	no	F	yes	83	CFM	7	•	•		•				•		
IoT-23 [70]	2018	yes	F	yes	22	ZEEK	11	•	•	•	•	•			•	•	
IoTNID [104]	2019	yes	P	yes	NA	NA	10	•					•				
ISCXIDS2012 [208]	2012	no	F	no	204	Other	4	•	•		•			•	•		
KDD99 [199]	1999	no	F	no	41	KDD	22		•		•	•	•		•		
Kitsune [160]	2017	yes	P	yes	115	Kitsune	9	•			•			•	•		
MQTT-IoT-IDS2020 [88]	2020	yes	P	yes	44	IPF,Other	4		•						•		
MQTTset [231]	2020	yes	P	yes	60	IPF	5		•		•	•					
N-BaIoT [153]	2018	yes	P	no	115	Kitsune	9	•			•				•		
NSL-KDD [223]	2009	no	F	no	41	KDD	22		•		•	•	•		•		
TON_IoT [169]	2020	yes	F	yes	12	ZEEK	9		•	•	•		•	•	•	•	
UNSW-NB15 [170]	2015	no	F	yes	47	Argus,Zeek	9				•	•	•		•		

7.2.2 Data

The use of different datasets in related work published from 2019 is summarised in Figure 7.1a and in Table 7.1 *Dataset* column. Furthermore, comprehensive information about the datasets employed in the literature is provided in Table 7.2. This table presents details such as the dataset names, publication year, the number of attacks included, the presence of IoT devices and raw data, the type and quantity of features, as well as the types of attacks encompassed. In parallel with Table 7.2, we can examine the datasets used in the studies in more detail.

When analysing the datasets over the years, it becomes evident that the majority of them were produced between 2015 and the present, with only a few predating 2015. Notably, older datasets show less frequent usage of IoT devices, while newer ones reflect a more common inclusion of IoT device data, aligning with their proliferation.

Many datasets consist of features derived from various feature extraction methods, elaborated further in the Features Section 7.2.3. These datasets are typically provided in CSV or JSON formats, representing previously extracted data using these methods. However, some studies additionally offer raw data alongside the pre-extracted features, enabling the extraction of new features through different approaches. The IoT-NID dataset stands as an exception, providing only raw files without any prior extraction.

Another crucial aspect of the data is its labelling, which indicates whether the data is benign or malicious. These labels facilitate the application of machine learning methods. Dataset creators offer labels directly or provide guidelines for labelling. Alternatively, files may be categorised/isolated into only malicious and only benign files (e.g., CIC IoT Dataset 2022 (CICIoT22) dataset). The labelling process can be either packet-based (shown with P), where each network packet is labelled, or flow-based (shown with F), where only the flows receive labels, not individual packets.

While the cumulative count of distinct attacks across various datasets exceeds 100, this table categorises them into 10 groups for the sake of presentation. Figure 7.2 shows which attacks are included in which dataset in expanded form. For example, Kitsune dataset, which contains 12 attacks (OS Scan, Fuzzing; Video Injection, ARP MitM, Active Wiretap; SSDP Flood, SYN DoS, SSL Renegotiation; Mirai Telnet), is grouped into 4 attack types (Scanning, MitM, DoS, Botnet). In making this grouping, we used similar categorisation found in the literature [11,36,144,174,239], as well as the groupings of some of the datasets [104, 160, 206] themselves. In this context, the 10 attack groups and their brief descriptions are as follows.

Scanning Attack [32] In a Scanning attack, the attacker aims to gather information about the victim's vulnerabilities. This attack is a passive attack that does not cause any damage on its own. However, the information obtained by the attacker can be used for other attacks.

Man in the Middle (MitM) Attack [79, 198] In a MitM attack, the attacker who enters between the parties in legitimate communication can receive and change the messages sent by both sides. A message sent by User A is directed to the attacker using methods such as ARP Poisoning, DNS Spoofing, DHCP Spoofing. The attacker sends the message received (with or without change) to user B. A similar process takes place in the opposite direction, too.

Denial-of-Service (DoS) attack [77] In Denial of Service attacks, the attacker exploits Internet protocols and consumes the resources of the service provider. Thus, it is aimed so that the legitimate user cannot receive service. If it is done by more than one source at the same time, it is called a Distributed Denial of Service (DDoS).

Malware (Software) attack [239] Malware attacks are malicious software which aim at information theft or to harm (such as economic loss, power depletion etc.). Examples of such attacks are viruses, malware, worms, or trojans.

Botnet [34] A bot is an automated software program that acts as an agent for a user or another program, resembling human behaviour in response to certain inputs. A botnet, on the other hand, consists of multiple computers with the same bot installed, enabling communication and control among them. Typically, botnets are exploited for malicious purposes, such as sending spam emails, propagating harmful software, DDoS attacks and carrying out cyber attacks.

In addition, we would like to highlight the Mirai botnet, a type of botnet attack that is prominent on IoT devices. Mirai attack has been observed since August 2016. In Mirai attack, the attacker exploits neglected security vulnerabilities of IoT devices (such as open ports, default usernames and passwords) to infect them with malicious code. These devices are then used by the attacker to orchestrate large-scale DDoS attacks [20].

Brute-force [196] Brute-force is a technique used by attackers to gain unauthorised access to a system or network by systematically attempting all possible combinations

of passwords or encryption keys until the correct one is discovered. These attacks require exhaustive trials of each potential password or key, making them time-consuming and computationally demanding. Nonetheless, as modern computers and distributed networks like botnets have increased computational power, brute-force attacks have become more widespread.

Exploitation [137] An exploitation attack is a form of assault in which the attacker capitalises on vulnerabilities or weaknesses within a system to gain unauthorised access, control, or acquire sensitive information. These attacks can focus on different aspects of the system, such as software or hardware vulnerabilities.

Web Attack [50] A web attack encompasses various types of assaults directed at websites and web applications, aiming to achieve objectives like stealing sensitive information, disrupting web-based services, or gaining control over web systems. These attacks can result in severe consequences, such as bypassing authentication systems, making unauthorised modifications to web databases and content, extracting data, stealing information, and taking control of servers.

Wi-Fi Attack [232] A Wireless Fidelity (WiFi) Attack is a harmful action that specifically targets wireless networks, taking advantage of weaknesses in the WiFi protocol or connected devices. These attacks can be executed through various methods and aim to achieve different goals, including unauthorised network access, compromising sensitive data, or disrupting network functionality.

Up to this point, we have examined the properties of the datasets. Hereafter, we examine the relationship of the datasets with the studies in the literature.

Although numerous studies have been published in recent years, it is still a common practice to rely on old datasets. For instance, in the studies referenced [6, 25, 60, 145, 161, 184, 194, 245], the KDD99 and NSL-KDD datasets were frequently used. The KDD99 dataset dates back to 1999, and NSL-KDD is an error-corrected version of KDD99. These datasets are widely trusted and have been employed in thousands of studies, making them valuable benchmarks. However, their age presents a significant drawback, as they lack up-to-date attacks and do not represent modern devices. Consequently, their applicability in contemporary network security studies is seriously limited. Using these datasets offers the advantage of comparing results with previous studies due to their extensive use in security research. Therefore, we support the notion of using these

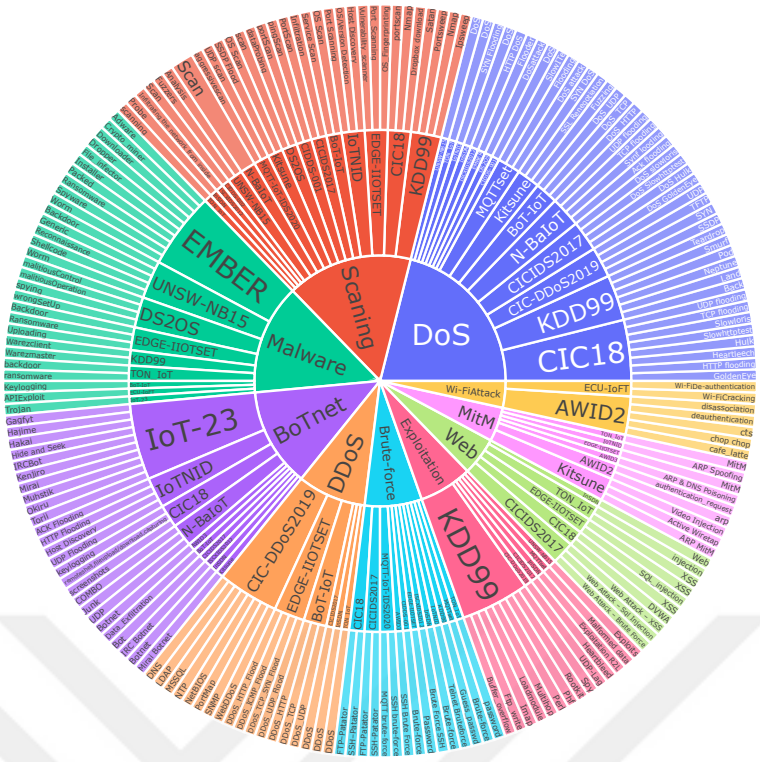


Figure 7.2: Distribution of attacks in the literature by dataset and attack groups. The sub-graphs, broken down by attack group, are given in the Appendix Figure A.10.

datasets as complementary sources in security studies. However, we strongly caution against using them as the sole dataset in network security studies concerning modern issues. Relying solely on these outdated datasets can lead to serious limitations and inaccuracies in the context of current network security problems.

Furthermore, these datasets do not contain any data related to IoT devices. In fact, it is common practice for IoT-oriented studies of network security to use datasets that do not contain IoT data such as the AB-TRAP, CICDDoS2019, CICIDS2017, CICIDS2018, UNSW-NB15 datasets. For example, [5,14,29,49,73,84,97,163,177,190,192,195,213,227] all use recent datasets, but these datasets do not contain any IoT data. Consequently, it is unclear how these studies yield results that are specific to IoT-based networks or attacks or to the heterogeneous network data flow originating from IoT. According to our perspective, at least one IoT-based or IoT-related dataset should be included in the evaluation step of IoT-oriented studies.

It is also common for studies to use private datasets. Although the datasets used in [4, 18,19,51,78,166,247] do contain IoT devices, as far as we are aware they are not publicly accessible, and consequently, it is difficult to confirm their data quality. Whilst it is

understandable that some datasets cannot be shared for privacy reasons, reproducibility is a serious problem in ML-based studies [106], and it is therefore important that studies that use private data are externally validated on a publicly accessible dataset. Since there are many open-access datasets today, it is very easy to verify the studies with a second open-access dataset in such cases. When the results are analysed, it is seen that only 13 [10,39,69,83,131,157,177,183,203,211,238,241,245] of the 68 studies used more than one dataset. In order to demonstrate the realism and generalizability of the results, it is very important to validate the study on more than one dataset. However, most of the studies in the literature are far from this.

Another issue with datasets is that many (such as KDD99, NSL-KDD, CIDDS-01, DS2OS, MQTTset, N-BaIoT, and InSDN) contain only pre-extracted features, and do not make the raw network data such as Packet Capture (pcap) file available. Without raw data, it is not possible to investigate new features or extract features using different tools. So, this approach, which severely restricts the creation of new features, changes and modifications of features, severely affects the comparability of data or studies with others.

On the other hand, many datasets with IoT devices with labelled and raw data for attack detection, both simulation-based datasets such as BoT-IoT [113], Edge-IIoT [68], TON-IoT [169], MQTT-IoT-IDS2020 [88], and real device-based dataset such as CICIoT2022 [48], IoT-ENV [93], IoT-NID [104], Kitsune [160], IoT-23 [70] are widely used in the literature. These datasets are analysed as part of our attack detection model in Section 9.2.1.

7.2.3 Features

The use of different feature extraction methods in related work published from 2019 is summarised in Figure 7.1b and Table 7.2. Most of the datasets used for attack detection are obtained with common methods and tools. Wireshark [237], Zeek (formerly Bro) [181], Argus [135], and FlowMeter (CICFlowMeter, formerly ISCXFlowMeter) [128] are examples of commonly used network monitoring and analysis tools using for feature extraction (creating features from raw data). For example, the UNSW-NB15 dataset used in [5, 29, 84, 163, 195, 227] was generated using Zeek and Argus and contains 49 features. ToN_IoT and Bot-IoT datasets were generated using Argus and contain 42 and 29 features respectively. The CICIDS2017-18-19 datasets were all generated by CICFlowMeter and contain 83 features. The common characteristics of these tools are that all of the generated features are flow-based. They focus on analysing network

traffic between the source and destination port/IP pair. Network statistics are calculated during this flow. The size of the flow is determined by the timeout or the agreement between the source and destination. Unlike these tools, Kitsune, used in [3,54,66,82] uses a rolling window approach, converting pcap files into a dataset with 115 features. The rolling window approach does not consider the flow between the source and destination but instead focuses on packets within the window size. When the window size is set to n, network statistics are calculated based on n packets between the source and destination.

In the use of individual packet-based features, individual features obtained from network packets are used. This approach has been used in various studies [18,19,51,68,166, 202]. We can also list the following datasets that use individual packet features: AWID2, MQTTset, Edge-IIoTset and MQTT-IoT-IDS20 (also contains flow-based). Apart from this, some studies [52, 150] detect attacks by training deep learning models on raw network data without first converting it into features. However, since the raw data (pcap file) consists of network packets, these studies face similar challenges to those that use individual packet features, such as the unintentional use of identifying features. The disadvantages of using individual packets and the studies using them are analysed in depth in Chapter 8. An analysis of the characteristics, advantages and disadvantages of the flow, window, and packet methods can be found in Section 9.2.3.

Many of the studies we have mentioned use pre-rendered versions of the datasets, but it is also possible to create new feature sets by using different tools if the dataset includes a pcap file. For example, the IoTID20 [10] dataset was created using FlowMeter from IoT-NID pcap files. So, both flow [10] and rolling window [54] features were extracted from the IoT-NID dataset. On the other hand, KDD99, NSL-KDD, CIDDS-01, DS2OS, MQTTset, N-BaIoT and InSDN datasets do not include pcap files. In this respect, feature extraction and modification studies that can be done with these datasets are quite limited.

7.2.4 Machine Learning and Evaluation Metrics

The use of different ML algorithms and metrics in related work published from 2019 is summarised in Figure 7.1c–7.1d and in Table 7.1 *Machine Learning–Evaluation* columns. The most widely used algorithms in the literature are tree ensembles (especially random forest), decision trees, MLP, SVM, boosting and kNN. In addition to classical ML algorithms, the use of deep learning methods has been increasing recently. For example, the use of CNN and RNN has increased, and transformers have started to be used intensively since 2022. Researchers have often evaluated their approach using more than

one ML model. In many studies with multiple comparisons, ensemble models (such as RF and XGB) gave the best results [3, 5, 29, 31, 49, 54, 60, 66, 73, 78, 161, 163, 177, 213, 227, 241, 245, 247].

Whilst deep learning (such as CNN, RNN and transformers) has become popular in recent years, classical ML algorithms remain widely used. In part, this can be attributed to the good performance of classical approaches in cases where feature extraction has already taken place (tabular data) [139], something that is true of most datasets and studies in the area of network security. The results of our preliminary studies also support this view (see Appendix A.2).

Accuracy is the most commonly used method of evaluation. In the majority of studies, accuracy is supported by other methods such as recall, precision, and F1 score, but there are also studies [131, 147, 184, 234] where only accuracy is reported. However, in an area such as attack detection, which suffers from unbalanced data distribution, accuracy alone is not a reliable metric [85]. Although the accuracies reported vary from 0.77 to 1, many studies report a success rate of greater than 0.99 (see Table 7.1). In cases of unbalanced data distribution and where only accuracy is reported, these scores may not accurately reflect model success. More generally, the reliability of results reported in the literature is questionable due to common errors made in network security and ML-based studies (e.g., information leakage, overfitting) [22]. In this context, it is essential that studies are not only focused on high metric scores but are also free from common errors, transparent and reproducible.

7.3 Conclusions

Within this chapter, an examination of the literature pertaining to IoT attack detection has been conducted, encompassing datasets, feature classifications, and machine learning techniques employed in these investigations (R2.1, R2.2, C2.1). Critical gaps and noteworthy considerations within the literature have been highlighted (R2.3, C2.1).

Chapter 8 elaborates on the limitations of utilizing individual packet features, central to device identification, for attack detection, supported by comprehensive explanations and experimental findings (R2.3, C2.2).

In Chapter 9, a novel attack detection methodology is introduced, with a focus on enhancing generalizability and bolstering IoT security through improved attack detection mechanisms (R2.4, R2.6, C2.3, C2.4, C2.5).

Chapter 8

The Risks of Using Only Individual Packet Features

> The dose makes the poison.
>
> Paracelsus

In this chapter, our focus revolves around a thorough examination of the limitations associated with using individual packets for attack detection in the existing literature. Within this context, we meticulously review the relevant literature, carefully assessing both the strengths and weaknesses inherent in this approach. The following draft publication has been utilised in the preparation of this section.

- K. Kostas, M. Just, and M. Lones. IoTGeM: Generalisable Models for Behaviour-Based IoT Attack Detection :github.com/kahramankostas/IoTGeM

8.1 Overview

In a secure IoT network, two essential elements are device identification and attack detection. Attack detection is a crucial prerequisite for device identification as device identification can run solely on benign networks. Conversely, device identification plays a vital role in streamlining attack detection efforts by identifying vulnerable devices and enabling the implementation of necessary precautions.

In our device identification phase, we have achieved considerable success by utilising features extracted from individual packets. While it may appear reasonable to employ the same models and features utilised for device identification in the context of attack

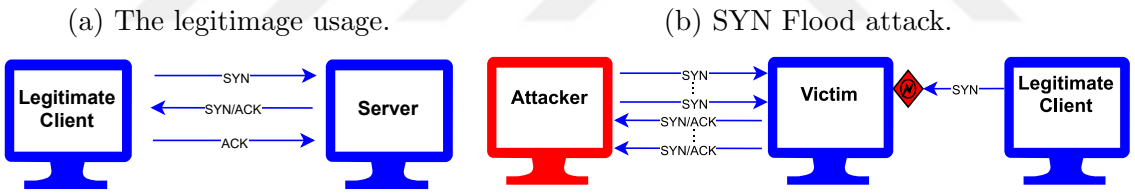
detection, we have made a deliberate choice not to adopt such an approach. Our decision stems from the firm belief that Individual Packet Feature (IPF)s are inadequate and ill-suited for effective attack detection. Therefore, in this chapter, we will explore the reasons why IPFs are unsuitable for effective attack detection.

8.2 Why Should IPF not be Used?

Detecting an attack using individual packet features alone is challenging. In this section, we'll elaborate on the reasons behind this difficulty. In the subsequent sections, we support this notion by presenting examples from existing literature and our own experimental findings.

This situation can be explained with a simple example: Synchronize (SYN) Flood [77], which is a Denial of Service (DoS) attack based on the exploit of 3-handshake in the Transmission Control Protocol (TCP) protocol. During the 3-way handshake process, the client that wants to establish a connection sends a SYN packet to the server. The server receiving this packet sends SYN-Acknowledgement (ACK) packet to the client. Finally, the client responds with the ACK packet, and the TCP connection is established (see Figure 8.1).

Figure 8.1: Legitimate and malicious use of 3-way handshake



In the case of a SYN Flood attack, the adversary inundates the server with an extensive stream of SYN packets. In response, the server sends SYN-ACK packets to establish connections, awaiting ACK packets from the other side to complete the handshake. However, the attacker fails to send these expected ACK packets, leading to a proliferation of incomplete handshakes. Consequently, the server's resources become depleted, rendering it unable to serve legitimate users.

When scrutinizing these individual SYN packets in isolation, we may not discern any distinguishing features among them, apart from identification information such as Internet Protocol (IP) or Media Access Control (MAC) addresses. However, these identifying features are inadequate for effective attack detection, as they cannot be predicted

in advance. As a result, we are unable to determine whether individual SYN packets should be considered benign or malicious.

Nevertheless, when examining the larger picture, we observe a significant pattern: despite receiving thousands of SYN packets from the same source, the server receives few or no corresponding ACK packets. This holistic perspective reveals the abnormal behaviour indicative of a SYN Flood attack. Thus, while analysing individual packets in isolation proves futile for detection, assessing the entire sequence enables the identification of such attacks.

This phenomenon is not limited to SYN Flood attacks; many other attack scenarios exhibit similar characteristics. To effectively detect and mitigate a range of attacks, it's essential to embrace a comprehensive approach that takes into account the context and interaction of packets, rather than relying solely on the features of individual packets.

However, like with any generalisations, there are exceptions. One example is single-packet attacks that exploit malformed packets. These attacks typically involve sending malformed network packets that the device cannot handle, leading to malfunctions or crashes in the receiving device [87]. However, addressing these attacks usually involves packet filtering or firewalls, not machine learning behavioural analysis, since they can be effectively managed using signature/rule-based approaches.

8.3 IPF Usage in the Literature

The use of individual packets in attack detection is quite common. Our literature review shows that 10 out of 68 studies use individual packet features (see Table 8.1). However, when we look at the literature, it seems that many studies have achieved very good results. So how can we explain these results if we claim that the methods they use to detect attacks are invalid? In this context, we investigate techniques that exploit individual characteristics and try to elucidate the mechanisms behind their documented exceptional performance.

When Table 8.1 is analysed, we see that most of the studies use open access datasets. The datasets used in [51,68,69,75,202] had a similar nature. All of these datasets contain raw data as well as individual packet specifications in pre-extracted Comma-separated values (CSV) format. All of the studies used these pre-extracted datasets. Another feature of these datasets is that they contain a single file for each attack, no sessions or other splitting. Therefore, in all these studies, the training and test data were split from a single file. Splitting the training and testing data from a single file results in

Table 8.1: Studies in the literature review using individual packet features, their datasets and results. The dataset specified as Private* is the same in both studies.

Study	Dataset	Accuracy	Recall	Precision	F1 Score
[75]	Edge-IIoTset(CSV)	100.00	100.00	100.00	100.00
[68]	Edge-IIoTset(CSV)	100.00	89.00	95.00	87.00
[202]	MQTT-IoT-IDS2020(CSV)	99.98	99.98	99.98	99.98
[69]	MQTTset(CSV)	91.00	91.00	77.00	80.00
[51]	AB-TRAP (CSV)				100.00
[39]	AWID2 (RAW)	99.96	99.99	99.95	99.97
[83]	ISCXIDS2012 (RAW)	99.42	99.41	99.34	99.37
	CICIDS2017 (RAW)	97.87	98.16	97.59	97.83
[166]	Private	100.00	100.00	100.00	100.00
[19]	Private*		98.00	99.00	99.00
[18]	Private*		100.00	100.00	100.00

the created files containing many identifying features, such as IP addresses, which may cause information leakage. This situation is exemplified in Figure 8.2. In [69, 75, 202], although the features used in their studies are not clear, there is no evidence that these identifying features have been deleted or censored. In this respect, it is quite possible that identifying features were used in these works.

In [68], IP addresses have been removed. However, this study suffer from a different type of information leakage. Upon examining these datasets, it becomes evident that they encompass various features such as ports, IDs, synchronisation, and acknowledgement numbers. Despite these features being extracted from individual packets, they provide insights into the interrelation of packets. For instance, when a flow is established between two ports, all individual packets within that flow bear the corresponding port numbers (see Figure 8.2 as an example, focusing on port and ID numbers). In a split that does not consider the flows, even with methods like cross-validation which is widely accepted, it is highly likely that some packets representing the same flow may end up in the training data while others in the test data. Moreover, the presence of a single file representing the attack in the mentioned studies, and the absence of distinct sessions in the datasets, make it exceedingly challenging to segregate them in a manner that prevents information leakage.

In the [68] study, it can be seen that the features that stand out for many attacks are sequence and acknowledgement numbers, these numbers are session-based identifiers and it is quite possible that they are subject to information leakage. In [51], although

target/source-based and mostly session-based identifiers were removed, information leakage occurred between the two datasets due to the way the training and test sets were separated. In none of these studies [51,68,69,75,202] did we see any indication that flows were taken into account in the training and test data partition, or that these session-based identifiers were removed. Therefore in these studies [51,68,69,75,202], information leakage between the training and test data is distinctly possible. While the test results published in these studies may appear promising, models relying on identifying features due to information leakage are prone to leading to less robust test results when applied to other datasets.

Table 8.1, two studies using open access data [39, 83] use pcap files which are raw data of datasets, not pre-extracted features. Chen et al. [39]'s study extract features from the pcap files of the AWID2 dataset. However, the use of raw data can be quite dangerous because these files also contain identifiers such as IP and MAC addresses. In this work, there is no indication that these attributes were deleted/discarded. Our opinion is that these identifying features are used. In the other study [83], the features are again extracted from the raw data of two datasets (ISCXIDS2012, CICIDS2017). However, in their study, while extracting the features, identifying features such as IP addresses and frame number are not used which is good for generalisation of results. Despite these precautions, we suspect that some identifying information might have inadvertently leaked. For example, in their study, the Internet checksum has been used, which is an identifying feature as it stores a hashed version of the source and destination IP addresses. In addition, both studies do not distinguish between training and testing by session, and it is possible that too many attributes about sessions may leak between training and testing data.

The other three studies [18, 19, 166] from Table 8.1 did not use publicly available data. Among these, many of the features used in [166] are identifying features (such as Frame number, Frame time, eth source, eth destination, Source IP, and destination IP), so the results obtained are unrealistic and inaccurate.

In the other two studies [18, 19], it is obvious that identifying features such as IP, MAC, and frame number have been removed. However, since session-based segregation has not been done in these datasets, it is quite possible that the characteristics of the sessions have leaked between the training and test sets. On the other hand, [19] claims that the testing phase is isolated from the training data, so the model is also successful on unseen data. Although the process of isolating this unseen data is vague, this claim is interesting. The claim that these individual packet features can detect unseen data in

attack detection is quite ambitious. In the next section, we will examine how this claim can be realised.

8.4 How does IPF Succeed?

In our research, we asserted that individual packet features alone are insufficient for the effective detection of network attacks. However, it has come to our attention that several studies in Table 8.2 have reported high scores while utilising individual packet features. We propose two possible explanations for this phenomenon.

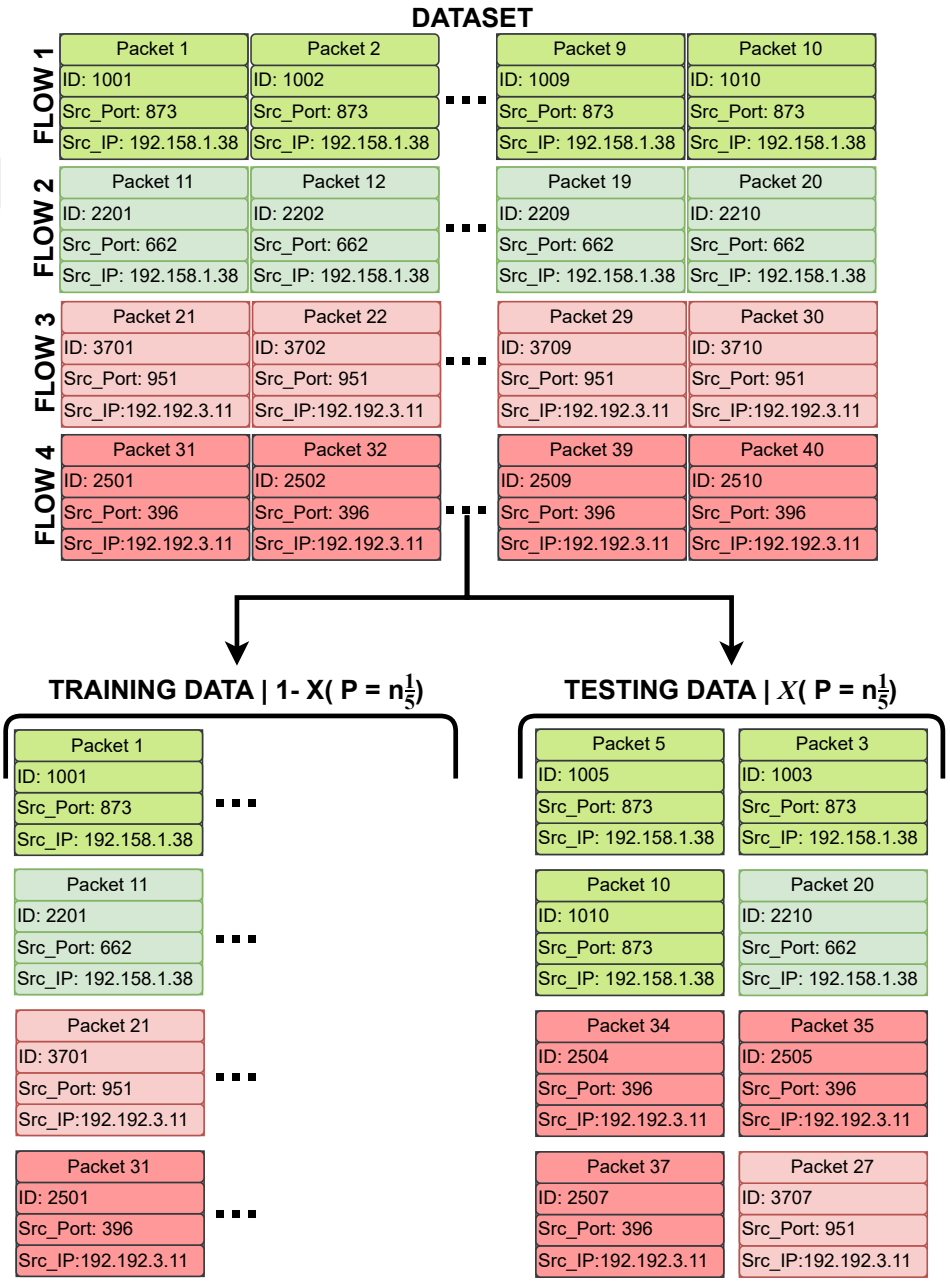
8.4.1 Identifying Features

The utilisation of identifying features may be a contributing factor. Identifying features are those that directly or indirectly reveal information about the target or source of the packets. For instance, the usage of MAC and IP addresses represents a direct identification method. Additionally, the Internet checksum, while not directly containing the IP address, encapsulates a compressed version of the IP header, thus acting as an identifier. However, it is worth noting that the inclusion of source and destination-based identifiers is generally considered a significant flaw, and most studies tend to remove such data from their analyses.

Furthermore, we wish to draw attention to the concept of session-based identifiers. Despite flow-based features being prevalent in attack detection studies, we believe that session-based identifiers have not received sufficient attention. The lack of exploration in this area is partly attributed to these individual features not being used in flow-based approaches. When dealing with individual features, incorporating session-based identifying features into the training and testing datasets without proper consideration can lead to information leakage issues.

To substantiate our claim, we conducted experiments using the IoT-NID dataset, which offers multiple sessions for each attack. This dataset records each attack execution as a separate session, providing an opportunity to explore the identifiability of session-based features. We utilised examples from this dataset to demonstrate the effectiveness of session-based identifiers. Our approach involved merging the first and second sessions of the attacks to create a new dataset. We then performed a 5-times 5-fold cross-validation on this merged dataset, employing one of the session-based identifiers at each iteration to distinguish between benign and attack samples. In the subsequent step, we

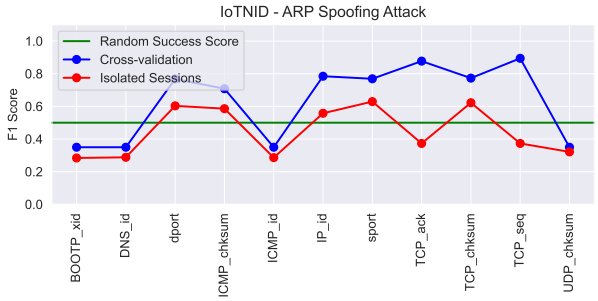
Figure 8.2: An 80/20 split for a hypothetical dataset containing 40 samples (4 session, 40 network packets), showcasing the division into 80% training data and 20% testing data for model development and evaluation, respectively. This example illustrates how information leakage takes place. In the scenario involving the source IP attribute, information leakage occurs because this address uniquely identifies both the attacker and the benign source, remaining constant across both the training and test sets. If we examine other features such as source port numbers and ID numbers, they vary across different sessions but exhibit correlations within each session (port numbers remain constant, ID numbers increment sequentially). Consequently, if packets from the same session are found in both the training and test sets, it can result in information leakage. Benign: 🟢 🟢 Malicious: 🔴 🔴



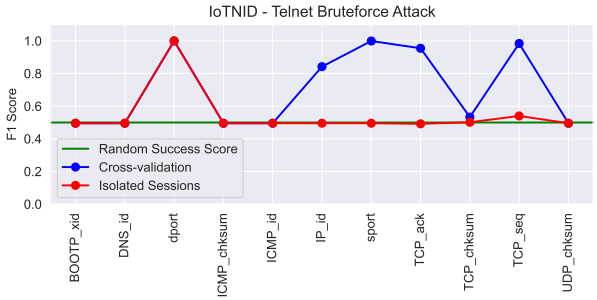
executed the separation by utilising the first session as the training set and the second session as the testing set. Figure 8.3 illustrates the distribution of these features for four attacks.

Figure 8.3: Comparison of Cross-Validation (CV) and isolated data performance of features on some attacks in the IoT-NID dataset

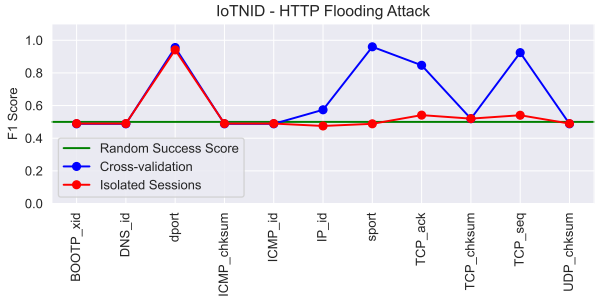
(a) IoT-NID dataset ARP Spoofing attack.



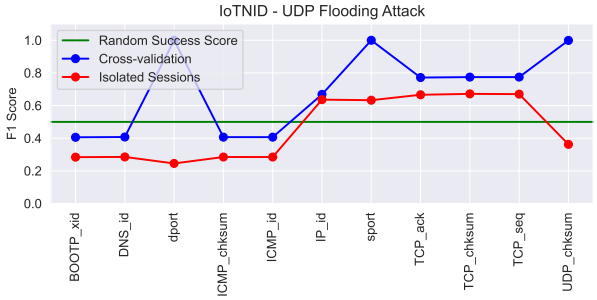
(b) IoT-NID dataset Telnet Bruteforce attack.



(c) IoT-NID dataset HTTP flooding attack.



(d) IoT-NID dataset UDP Flooding attack.

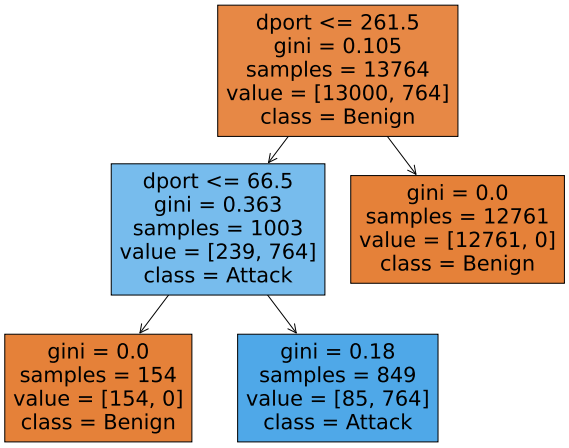


The results of this analysis reveal a notable pattern: in the majority of cases, the descriptive features demonstrate better success in the cross-validation scenario, while in isolation, they exhibit limited effectiveness. Figure 8.3 visually depicts this trend. In this context, these results support the hypothesis that these features lead to information leakage.

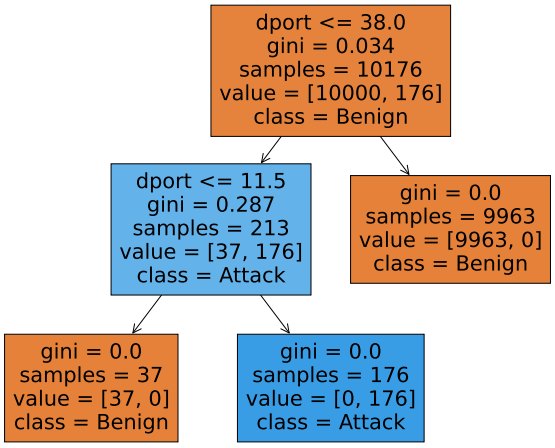
However, upon a more comprehensive examination of the figures, we observed that certain features consistently display exceptional performance in both scenarios. Dport feature in Telnet Brute-force (Figure 8.3b) and HTTP Flooding (Figure 8.3d) attacks are prime examples of this consistent success across the two scenarios. If we examine the models associated with the features that achieve this extraordinary success, as shown in Figure 8.4, we see that these models have an extremely simple structure. This brings us to the second explanation for achieving high performance with individual packet features: low data complexity.

Figure 8.4: Visualization of high-achieving decision tree models.

(a) HTTP Flood model using dport feature.



(b) Brute-Force model using dport feature.



8.4.2 Low Data Complexity

Most attacks follow a pattern and produce uniform outputs. For example, the distribution of the size of the attack packets in the HTTP flood attack on four different datasets is shown in Figure 8.5. Because of this uniform nature of the attacks, if the complexity level of the dataset is low in studies using individual packets, even very basic characteristics, such as size or time, can become identifying features.

Figure 8.5: The distribution of packet size malicious data (HTTP Flood) in four different datasets.

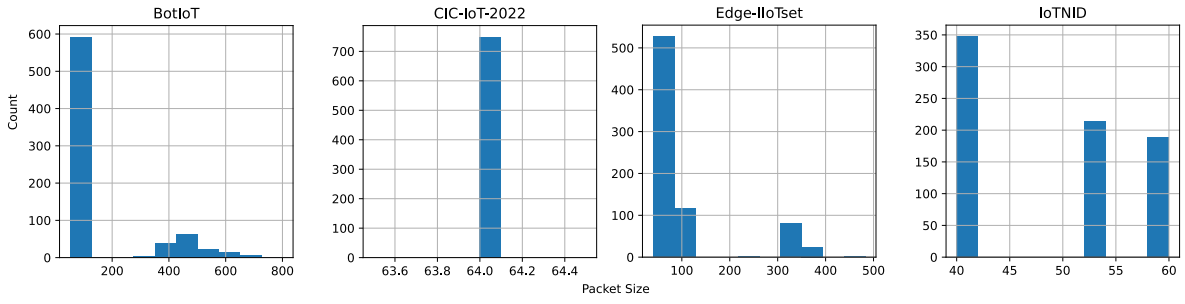


Figure 8.6 presents compelling empirical evidence to support our assertions. Specifically, in Figures 8.6a-d, we depict the User Datagram Protocol (UDP) attacks observed in four distinct sessions. A detailed analysis of these figures reveals remarkable similarities between sessions 1 (8.6a) and 3 (8.6c), as well as between sessions 2 (8.6b) and 4 (8.6d). Moreover, it is noteworthy that the size of attack packets remains constant even

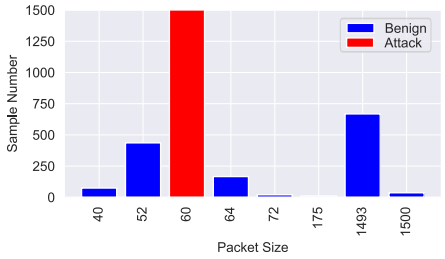
across these different sessions.

When incorporating only packet size as a feature in this example, we achieve perfect detection performance (see Table 8.1). Even when testing the model trained with the first session on the third session, where the model has not encountered this data previously, we observe a notably high success rate. The same holds true for sessions 2 and 4, indicating strong consistency in their detection capabilities. However, this superior success is only a consequence of the information leakage problem. The packet size in a UDP attack may change in another attack, and this model will fail to catch these attacks, or it will detect benign packets of this specific size as attacks.

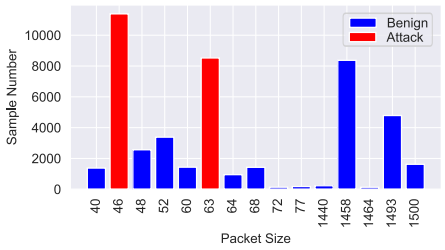
Figure 8.6e provides another insightful example, showcasing the change of labels over time in the Kitsune dataset, Mirai attack. In this graph, packets before a specific time exhibit all benign characteristics, while those occurring after the designated time are identified as attacks.

Figure 8.6: Distribution of labels in Kitsune and IoT-NID datasets.

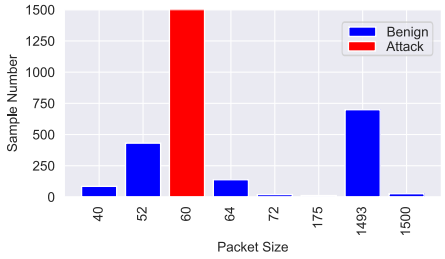
(a) IoT-NID UDP-Session-1 size histogram.



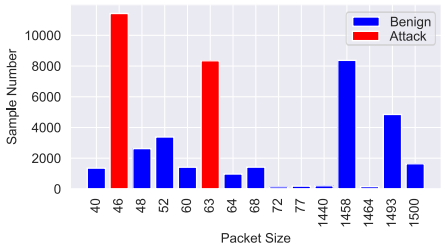
(b) IoT-NID UDP-Session-2 size histogram.



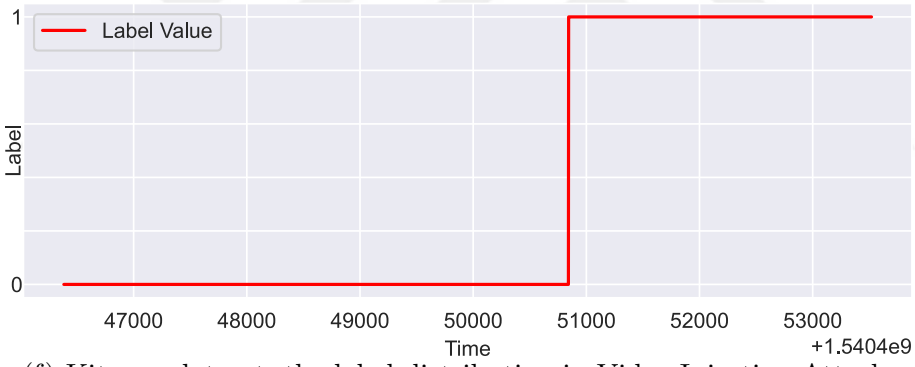
(c) IoT-NID UDP-Session-3 size histogram.



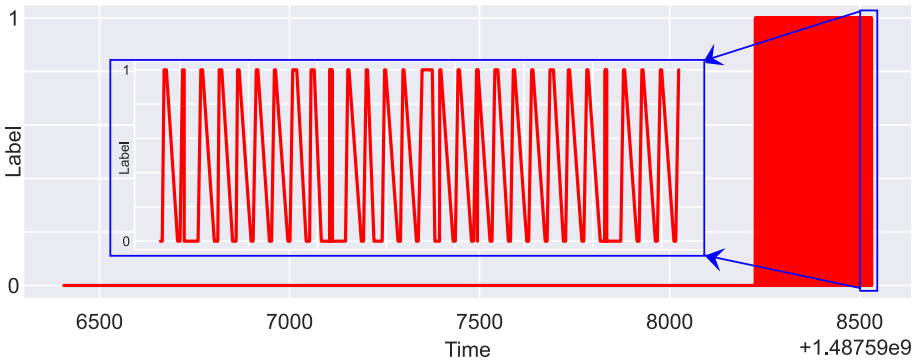
(d) IoT-NID UDP-Session-4 size histogram.



(e) Kitsune dataset, the label distribution in Mirai attack.



(f) Kitsune dataset, the label distribution in Video Injection Attack.

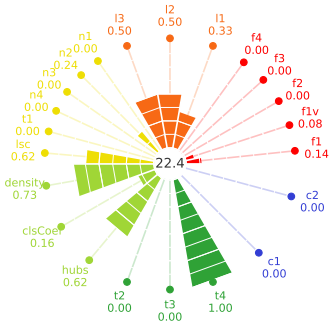


Additionally, we present yet another example, Figure 8.6f, in which all packets up

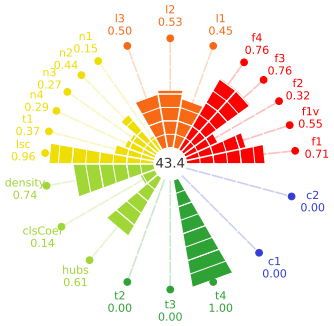
to a certain time demonstrate benign behaviour, while thereafter, a mixture of attack and benign packets is evident. The graph illustrates a dense concentration of data over a brief temporal window. However, focusing on a smaller portion of this mixture (the last 2000 samples) reveals a rhythmic pattern between attack and benign traffic over time. The models trained using the time feature for these two attacks perform well on these datasets, but are not capable of detecting the same attack on another dataset or in real-life.

Figure 8.7: Complexity analysis of UDP, Mirai, and Video injection attacks according to time and size characteristics. The central number depicted in the figures represents the comprehensive complexity score, which is derived as the average of 22 distinct analytical methods (The complexity score ranges from 0 to 100, where higher values indicate increasing complexity). These 22 methods stem from six different approaches to assessing complexity, each distinctly colour-coded for clarity of categorisation [138]: red for feature-based, orange for linearity-based, yellow for neighbourhood-based, light green for network-based, dark green for dimensionality-based, and blue for class imbalance-based.

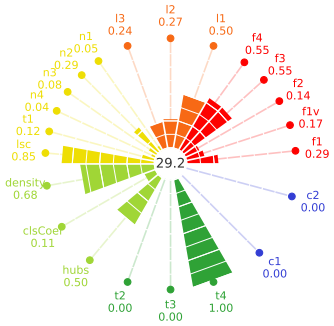
(a) Mirai Time Complexity



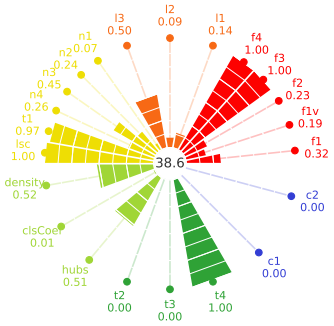
(b) UPD Flood Time Complexity



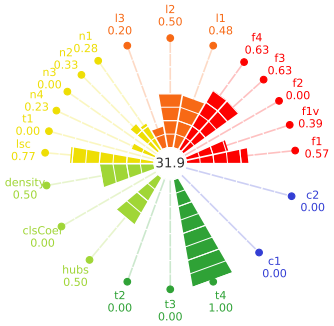
(c) Video injection Time Complexity



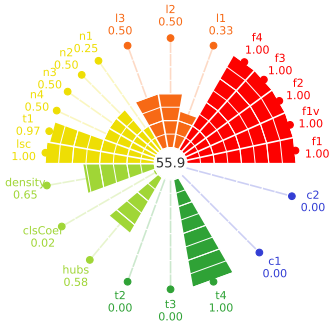
(d) Mirai Size Complexity.



(e) UPD Flood Size Complexity.



(f) Video injection Size Complexity.



To get a better understanding of the complexity of the data, we calculated the complexity score [138] of the attack files. Figure 8.7 shows the computation of the

complexity scores of 3 different data using time and size characteristics. When these figures are analysed, it can be seen that the UDP flood attack has low size complexity and high time complexity. On the other hand, Mirai and video injection attacks show low size complexity and high time complexity.

Table 8.2 shows the results of the attack detection process using only one feature for each of the three attacks with shared complexity values. Upon examination the cross-validation (10 times, 10 fold) results in Table 8.2, we observe that the UDP attack, characterised by very low complexity in terms of size, achieves flawless detection. Similarly, in the Mirai attack, exhibiting very low time complexity attains nearly perfect detection rates. Conversely, the video injection attack, which possesses a relatively higher size complexity but lower time complexity, did not achieve substantial success in terms of size-based detection but demonstrated significant success when considering time-based detection.

In our second experiment, isolated part in Table 8.2, capitalising on the multi-session nature of the UDP attack, we conducted tests involving the second session with a fourth session that closely resembled it, and third sessions that were notably different. The results of these tests indicate exceptional performance in the fourth sessions that were similar, while not achieving significant results in the third sessions exhibited substantial dissimilarities from the second session.

What all of these examples have in common is that high levels of success can be achieved by using individual features that are not suitable for attack detection. Although identifying features are not used here, the uniformity of attack data enables even basic individual characteristics to serve as identifying features.

We have expanded this sample a little more. Figure 8.8 shows the results of another experiment on complexity. In this experiment, a basic feature (packet size) was used to separate all attack types in IoT-NID dataset. From this graph, the inverse correlation between complexity and success can be observed. Especially if the complexity is below the critical level (45), highly successful discrimination is achieved. On the other hand, as complexity increases, success seems to decrease.

Figure 8.8: Reflection of the change in the complexity value due to a single feature on the model success

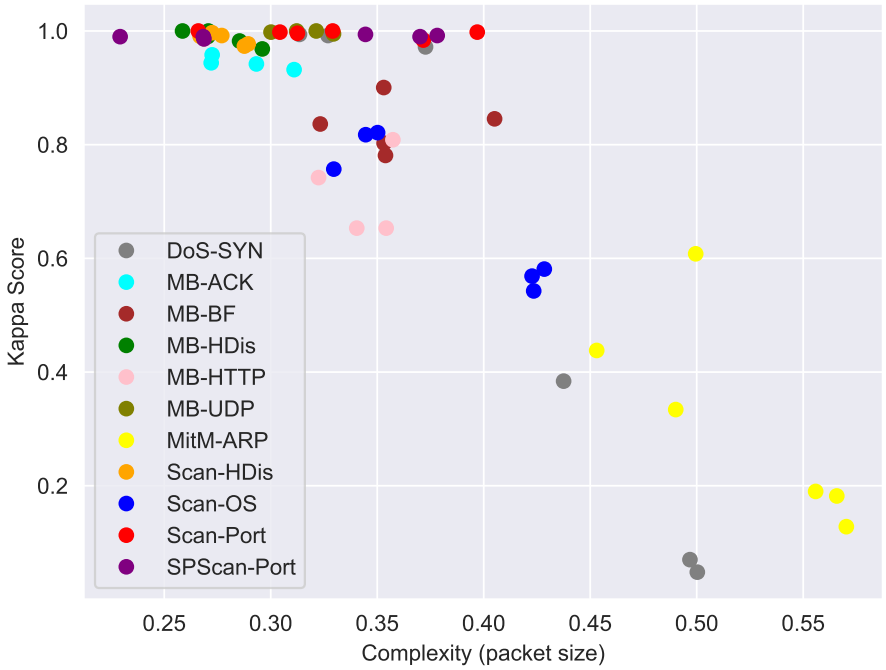


Table 8.2: Near-perfect classification of 2 datasets afflicted by low data complexity. In Kitsune Mirai and Video injection data, the labels follow a certain time pattern. In the IoT-NID UDP data, attack packets have a specific size. In the Cross-Validation part of the table, the results depict the mean and standard deviation based on 10 repetitions of 10-fold cross-validation. In the isolated part, the mean and standard deviation are presented from 100 repetitions. Temporal information is not used in order to keep the features at the simplest level. The "time" feature here refers to a timestamp, not a temporal feature.

	Dataset	Subdataset	Feature	ML	Accuracy	Precsion	Recall	F1 Score	Kappa
Cross-validated	Kitsune	Video Inj	Size	DT	0.959±0.000	0.479±0.000	0.500±0.000	0.489±0.000	0.000±0.000
	Kitsune	Mirai	Size	DT	0.899±0.001	0.804±0.002	0.918±0.001	0.842±0.002	0.688±0.003
	IoT-NID	UDP-S2	Size	DT	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	0.999±0.001
	Kitsune	Video Inj	Time	DT	0.988±0.000	0.933±0.002	0.919±0.002	0.926±0.001	0.852±0.003
	Kitsune	Mirai	Time	DT	0.997±0.000	0.998±0.00	0.990±0.001	0.994±0.000	0.988±0.001
	IoT-NID	UDP-S2	Time	DT	0.841±0.007	0.833±0.008	0.839±0.008	0.835±0.008	0.671±0.016
Isolated	IoT-NID	UDP-S2vsS3	Size	DT	0.032±0.000	0.016±0.000	0.500±0.000	0.031±0.000	0.000±0.000
	IoT-NID	UDP-S2vsS4	Size	DT	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	0.999±0.000
	IoT-NID	UDP-S2vsS3	Time	DT	0.526±0.000	0.508±0.000	0.566±0.000	0.379±0.000	0.017±0.000
	IoT-NID	UDP-S2vsS4	Time	DT	0.866±0.000	0.858±0.000	0.865±0.000	0.861±0.000	0.722±0.000

The attacks discussed in this chapter are not IoT specific, which requires clarification of the decision to exclude more sophisticated Network Intrusion Detection Systems

(NIDS). The focus is not on detecting attacks, but on highlighting the inadequacy of detecting attacks using IPF. This research specifically addresses the inadequacy of IPF features for this purpose and examines the rationale underlying the successes reported in the literature using these features.

In our analysis, we adopt Decision Tree (DT) as a fast and interpretable machine learning algorithm. This choice is motivated by the intention to illuminate the limitations of the system rather than to achieve optimal detection. For example, in notable examples of success, the visualisation of DT proved invaluable for analysing the factors contributing to such results.

8.5 Conclusion

In this chapter, our focus revolves around a thorough examination of the limitations associated with using individual packets for attack detection in the existing literature. Within this context, we meticulously review the relevant literature, carefully assessing both the strengths and weaknesses inherent in this approach (RQ2.1, RQ2.2, RQ2.3, C2.1).

As part of our analysis, we delve into the primary challenges encountered in attack detection when relying solely on individual packet features. We investigate potential factors contributing to the observed successes in the use of individual features. Key considerations include the utilization of identifying features, which may inadvertently introduce information leakage between the training and test datasets, thus affecting the model's performance. Additionally, we explore the dataset features with exceptionally low complexity, which can lead to information leakage and hamper the model's generalization capacity (RQ2.5, RQ2.6, C2.2, C2.4).

By addressing these issues, our objective is to foster a deeper understanding of the shortcomings of utilizing individual packets for attack detection. Furthermore, we seek to stimulate a more comprehensive exploration of alternative strategies and feature combinations, ultimately enhancing the robustness and accuracy of attack detection systems.

Chapter 9

Generalisable Models for IoT Attack Detection

This chapter introduces an approach to model Internet of Things (IoT) network attack detection, prioritising generalizability and improved performance. We enhance feature extraction through a rolling (sliding) window technique and introduce a multi-step feature selection process to mitigate overfitting. Our methodology involves constructing and testing models on isolated training and testing datasets, evading information leakage that hindered prior models' generality. Rigorous evaluations encompass diverse machine learning models, metrics, and datasets. We enhance model trustworthiness through Explainable Artificial Intelligence (XAI) techniques, pinpointing features crucial for precise attack detection. The following draft publication has been utilised in the preparation of this section.

- K. Kostas, M. Just, and M. Lones. IoTGeM: Generalisable Models for Behaviour-Based IoT Attack Detection :github.com/kahramankostas/IoTGeM

9.1 Overview

Behaviour-based attack detection using Machine Learning (ML) is a common approach to securing IoT devices by detecting anomalous network behaviours. However, common mistakes in ML studies can raise serious doubts about the reliability of results [22, 106, 136], with examples such as information leakage and feature overfitting in previous IoT attack detection studies. In this chapter we focus on automating the behaviour-based discrimination of benign and malicious network data with ML algorithms while avoiding

such pitfalls. To do this, we introduce an approach (*IoTGeM*) for creating generalisable models for behaviour-based network attack detection for IoT devices, with the following contributions:

1. A rolling window method for feature extraction that achieves earlier detection and improved performance compared to conventional approaches.
2. A multi-step feature selection process based on a genetic algorithm with exogenous feedback, which is effective in eliminating features that can lead to overfitting.
3. A detailed examination of feature effectiveness by analysing the relationship between features and attacks using XAI techniques.

We further adopt a strict methodology for ensuring the generalisability of ML models, and the ease of validating, replicating, or extending our approach [136]:

- We build and test ML models using isolated train and test datasets to avoid common data leaks.
- We rigorously evaluate our methodology with a diverse portfolio of machine learning models, evaluation metrics and datasets, and we avoid using inappropriate metrics for certain situations, e.g., erroneously using only accuracy with unbalanced datasets.
- We use publicly available datasets and make our scripts available to the public[1].

This chapter is organised as follows. Section 9.2 describes the materials and methods we use for data selection, feature extraction and feature selection. Section 9.3 evaluates the results of the models, compares them with other methods, and provides a detailed summary analysis relating feature effectiveness to different attacks. Conclusions are given in Section 9.4.

9.2 Materials and Methods

To build a generalisable ML model that detects attacks we first analyse and decide on the most suitable datasets for our study. Then, we introduce our features by analysing the advantages and disadvantages of feature extraction methods. After feature extraction, we perform feature selection by analysing and eliminating some extracted features

[1]Materials available at: github.com/kahramankostas/IoTGeM

by analysing these features under different conditions. From the remaining features, we create the appropriate feature combination using a Genetic Algorithm (GA) and exogenous feedback. In the next stage, we discover the most suitable parameters by hyper-parameter optimisation. Finally, we train ML models with selected features and parameters, and after that test these models with previously unseen data and obtain our final results. This process is visualised in Figure 9.1

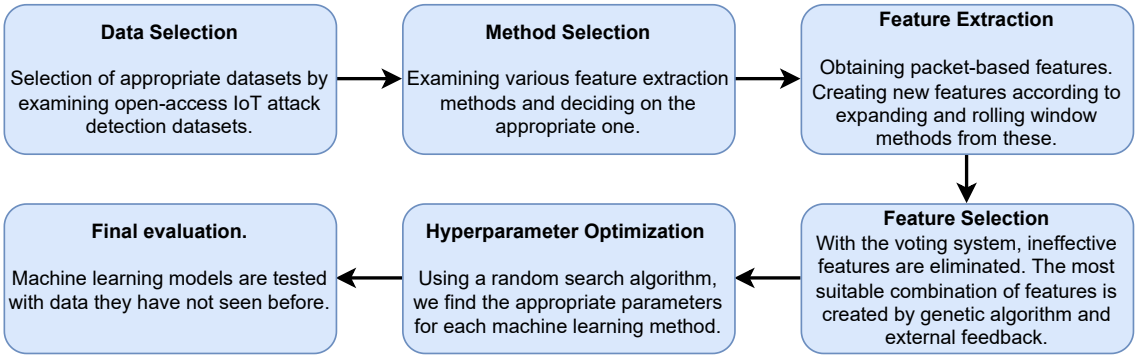


Figure 9.1: The visualisation of steps in system implementation.

9.2.1 Data Selection

We examined the datasets used in IoT attack detection studies. We searched the ones that are publicly available, contain raw data (pcap files) and are labelled. The list of these is given in Table 9.1, where we also measure them in terms of the number of devices and attack types they contain and classify them in terms of whether they contain real IoT data and multiple sessions. IoT devices differ from non-IoT devices in particular by their heterogeneous nature, in that IoT devices often contain unique software and hardware. This diversity is very difficult to simulate, and consequently it is preferable that datasets include real rather than simulated IoT devices. Repeated attacks in different sessions are also desirable since it allows comparison of each attack across sessions and a deeper analysis of the nature of the attacks. It can also be used to prevent overfitting of session-specific features.

The IoT-NID and CIC IoT Dataset 2022 (CICIoT22) datasets are the only ones which contain real, multi-session IoT data. While CICIoT22 has a large number of devices, it contains relatively few attack types. Furthermore, attack situations are isolated in this dataset. Only one device is involved in each session in which an attack occurs. Therefore, the number of devices does not make any difference. IoT-NID, on the other hand, contains few devices but many attacks, which makes it particularly useful for

Table 9.1: Datasets that can be used for IoT attack detection

Dataset	Year	Real IoT	Session	Device No.	Attack No.
BoT-IoT [113]	2018	×	×	5	10
CICIoT22 [48]	2022	✓	✓	40	4
Edge-IIoT [68]	2022	×	×	9	13
IoT-ENV [93]	2021	✓	×	5	3
IoT-NID [104]	2019	✓	✓	2	10
Kitsune [160]	2018	✓	×	9	9
TON-IoT [169]	2019	×	×	7	9
IoT-23 [70]	2020	✓	×	3	8
MQTT-IoT-IDS2020 [88]	2020	×	×	13	4

training different attack detection models. For this reason, IoT-NID is used as the main dataset for training models.

It is important to completely isolate training and testing data. To ensure this, we always measure the success of a model on data that the model has never seen before. Figure 9.2 depicts how data is used in this study. The same colour occurring in different columns for the same attack represents different sessions from the IoT-NID dataset, i.e. where an attack was carried out more than once, in different sessions. The data in the first column are used for training the models, data in the Train/CV, HPO and Validation columns are then used to refine and select models, and data in the Session and Dataset Test columns are used to measure the generality of models. More specifically, data in the HPO column is used both for feature reduction and hyperparameter optimisation, and data in the validation column are used in the feature selection step.

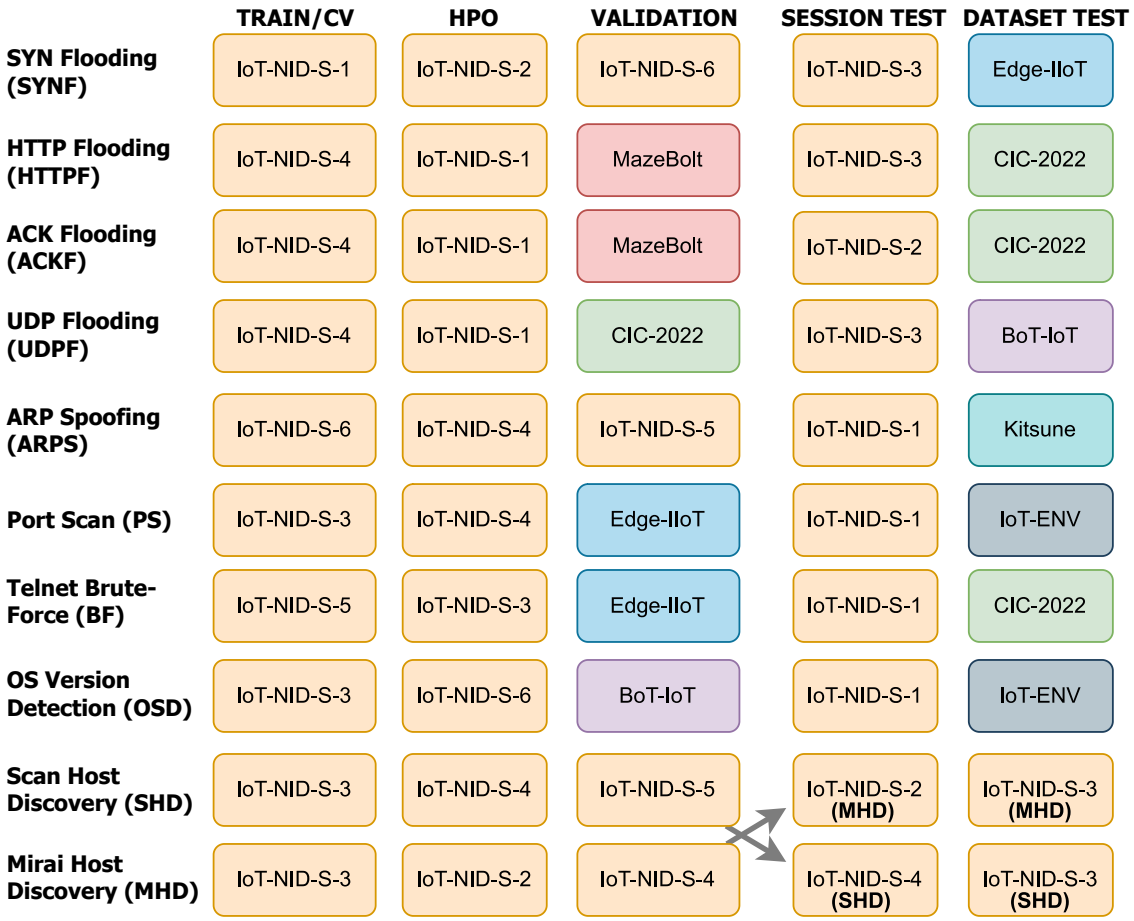


Figure 9.2: Classification of datasets according to where they will be used. Train/CV=Training and cross-validation. HPO=Feature reduction and hyperparameter selection. Validation=Data validation and feature selection. S-n= Session n.

Data is quite limited for some of the attacks. In the case of Hypertext Transfer Protocol (HTTP)Flooding (F) and Acknowledgement (ACK)F attacks, there was insufficient data for us to use a separate IoT-NID session for validation, and we addressed this by sourcing extra data from kb.mazebolt.com for these attacks. We selected data from this source because the structure of these two attacks closely resembled their versions in IoT-NID. For Bruteforce (BF), we addressed the paucity of data for specific attacks by using slightly different variants of the attack in the training, validation and test datasets. Specifically, the training data involves telnet BF, the validation data involves password attacks and the test data involves a Real Time Streaming Protocol (RTSP) BF attack. For Scanning Host Discovery (SHD) and Mirai Host Discovery (MHD), there were no suitable equivalent data in other datasets. So that we could still measure the generality of these attack models, we used an MHD session to test SHD models, and vice versa.

This is a feasible approach because SHD and MHD are very similar variants of the same attack (host discovery) made using different devices and tools (scan and Mirai).

9.2.2 ML Algorithm Selection

Since there is no one-size-fits-all approach for every problem, it is important to consider performance across a range of ML models [136]. We selected one model from each of the following model categories [76]; linear model: Logistic (Logit) Regression (LR), tree-based: Decision Tree (DT), Bayesian-based: Naïve Bayes (NB), kernel-based: Support Vector Machine (SVM), ensemble-based: Random Forest (RF), instance-based: k-Nearest Neighbours (KNN)), and artificial neural networks (ANN): MultiLayer Perceptron (MLP). This set of models provides a good intersection with those used in previous studies. Additionally, we also included XGBoost, eXtreme Gradient Boosting (XGB), which, although infrequently encountered in the attack detection literature, is known to work well on tabular datasets [209].

We have made the code for our work public and we have used a modular code structure so that only minor code changes are required to support other algorithms.

9.2.3 Feature Extraction Methods

In the literature, it is common to see flow-based or individual packet-based approaches to feature extraction. In this section, we describe these two approaches and propose a window-based approach as an alternative.

9.2.3.1 Individual Packet-Based Features

Individual packet features can be obtained from the packet headers or packet payloads of the individual network packets. As discussed in Chapter 8, whilst this approach is commonly used [18, 51, 68, 166, 183, 202], it has a number of issues that can impair generalisability. For this reason, we do not use it in our main study. However, given its prevalence in the literature, in Chapter 8 we further discuss the limitations of this approach and experimentally show how even simple features such as packet size or timestamp can become identifiers and lead to models with apparently high accuracy but very poor generality. For this, individual features were extracted from raw data using Python, Scapy, and Wireshark tools, mostly from headers, but we also extracted payload-based features such as payload-entropy or payload-bytes.

9.2.3.2 Flow-Based Features

Unlike features from individual packets, flow features [5, 6, 14, 25, 29, 49, 60, 73, 84, 97, 145, 161, 163, 177, 184, 190, 192, 194, 195, 213, 227, 245] are sufficient for attack detection, but they have some drawbacks. For example, in order to generate features in a flow system, the flow must end with a termination flag or timeout. In this respect, in order for a flow to be identified as an attack, it is necessary to wait for it to be terminated first. Only after the flow ends can the statistics required for attack detection be calculated. In this study, flow-based features are extracted from the raw data using the CICFlowMeter[2] tool. We chose CICFlowMeter for this, since it extracts significantly more features than alternative tools, and also specifically focuses on features that can be used in attack detection. Note that the IoTID20 [229] dataset also contains features extracted using CICFlowMeter from IoT-NID; however, our preliminary analysis of IoTID20 (see Appendix A.7.1) identified anomalies, prompting us to do our own feature extraction.

9.2.3.3 Window-Based Features

In this approach, we focus on the changes in the data flow between the source and destination. However, in contrast to the flow-based approach, instead of generating an aggregate statistic for each flow, we utilise the changes in the network data that occur as each packet arrives. While similar approaches have been used in the literature [3, 54, 66], we present a different approach in terms of both methodology and features. This uses both rolling (RW) and expanding windows (EW) that extract features from information carried by packets between the source and destination (Media Access Control (MAC)/Internet Protocol (IP) address). In the rolling window approach, we observe the change between packets within a certain window size. The expanding window starts with a small window covering the initial data and gradually expands to include new data. Each time a new packet is added, the window becomes larger and statistics are recalculated. We use four different sources: size, time, destination-source and Transmission Control Protocol (TCP) flags, and also calculate the mean and standard deviation of the window values for some individual features. The process is visualised in Figure 9.3.

[2] https://github.com/ahlashkari/CICFlowMeter

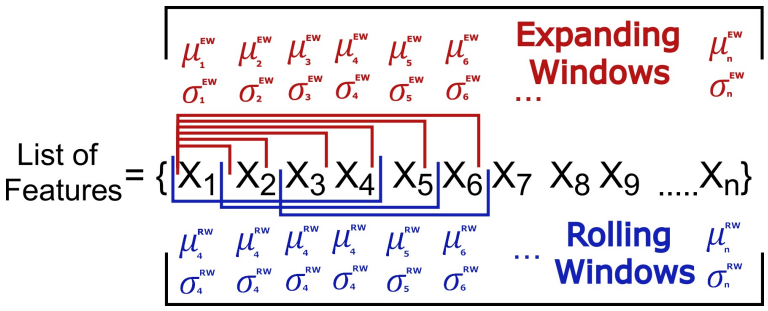


Figure 9.3: The process of extracting window-based features. μ is the mean, σ is the standard deviation

When using the rolling window method, we also had to decide on the window size we would use. In this context, we conducted preliminary experiments that determined the window size with two MitM (Video Injection and Active Wiretap) attacks that we did not use in our other experiments. We chose these attacks because MitM attacks are sensitive to packet time changes rather than flags, packet content, and destination-source features. Therefore, statistical features such as window features play an important role in detecting these attacks. As a preliminary study, we tried to detect these attacks using only window features. In these experiments, we tested window sizes ranging from two to 20, using a combination of EW features and EW-RW features for *packet size, time, TCP window size, payload bytes, payload entropy* features. The experiment results, depicted in Figure 9.4, represent the averages from 10 repeats obtained DT. In the context of MitM attacks, our emphasis has been on the selection of an appropriate window size due to the sensitivity of these attacks to packet time changes. The primary determinant in achieving effective results lies in the distinctive features chosen rather than the specific ML algorithms employed. It is anticipated that similar outcomes would be attained with various ML methods, given the consistent use of relevant features. Using outcomes of this experiment, we added the RW features with two, six, and nine window sizes, which provided the highest success, to our feature set. Considering the limited data generation propensity of numerous IoT devices, the impracticality of employing excessively large window sizes becomes evident. In this context, we did not exceed 10 in the rolling window size. We also added the EW features which also increased the success considerably. In addition, we applied the expanding window approach to TCP flags and source-destination elements. Our feature list is given in the Appendix, Table A.9, including an explanation of each feature type mentioned in this chapter.

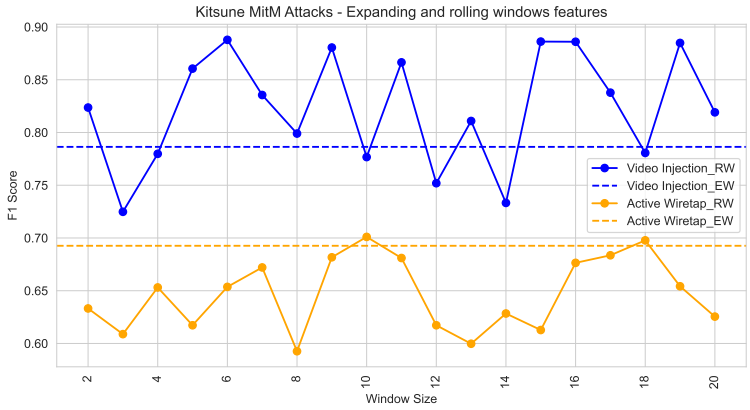


Figure 9.4: Utilizing diverse window sizes for Kitsune MitM attack detection and the change of performance according to the window size.

In Section 9.3.1, we compare our window-based approach against a flow-based implementation in terms of its ability to support attack detection and generalisability.

9.2.4 Feature Selection

It is important to identify stable features that are capable of supporting generalisable models. We do this using a three-step method. In the first step, we remove features that are clearly inappropriate, i.e. device or session-based identifiers such as IP-checksum, ID, synchronisation, and port numbers (see Chapter 8). In the second step, we employ a feature elimination process that considers the value of each feature in turn, and in the third step we use a GA to further refine the remaining features, taking into account their potential interactions.

9.2.4.1 Feature elimination

The aim of this stage is to identify and eliminate features that are a risk to generalisability. For three different validation scenarios, we consider each feature in turn, and measure its association with the target variable when used to train an ML model which is then evaluated using Cohen's kappa. We use ExtraTree (extremely randomised tree) for its speed, interpretability, and visualization capabilities. It is deliberately chosen to prevent bias against other machine learning algorithms used in the final evaluation. A kappa value less than or equal to zero indicates an unreliable feature.

In the first scenario, cross-validation is used within a single session (Train/CV column in Figure 9.2). In the second scenario, two independent sessions from the same dataset

are used, one (Train/CV column) for training, the other (HPO column) for validation. This helps to eliminate features that do not generalise beyond a single session, a common problem when using cross-validation alone. In the third scenario, we use a validation session from a different dataset (Validation column).

Features are assessed in each scenario, and if a feature surpasses a kappa score of 0, it is awarded one vote for that scenario. Features with at least two votes, including one from the final scenario, are included in the feature set for the GA. This approach prioritises features selected in the last, most stringent, scenario whilst also taking into account their utility in the other scenarios to discourage spurious correlations. Figure 9.5 depicts the voting process for a particular attack.

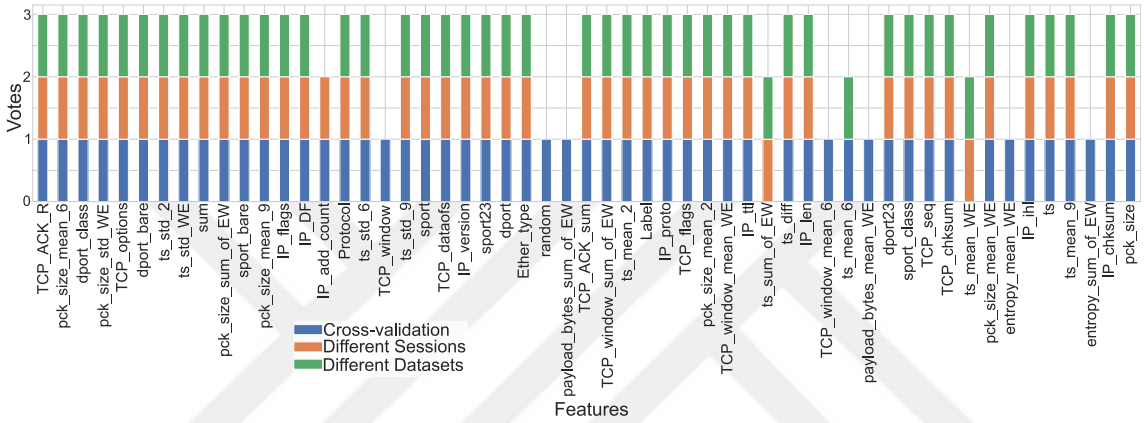


Figure 9.5: Voting process during the feature elimination step for the Host Discovery attack.

9.2.4.2 Feature Selection with GA and External Feedback

We employ a GA to further optimise the feature list, taking into account the potential interactions between features. GAs are a popular approach to feature selection, since they allow the space of feature combinations to be explored in a relatively efficient manner, with a good chance of finding an optimal, or at least near-optimal, combination of features. The GA process is depicted in Figure 9.6. The validation results, in terms of F1 score, are used as feedback for the GA, guiding the creation of new feature combinations. This iterative process continues for a specified number of generations (e.g., 25), with the best combination of features observed during this period chosen as the final feature set. In this procedure, the GA obtains an F1 score as external performance feedback from a distinct dataset (see Figure 9.2). Notably, this approach diverges by employing disparate data for fitness evaluation. This external feedback fosters the selection of feature

combinations exemplifying generalizability, thereby addressing overfitting concerns.

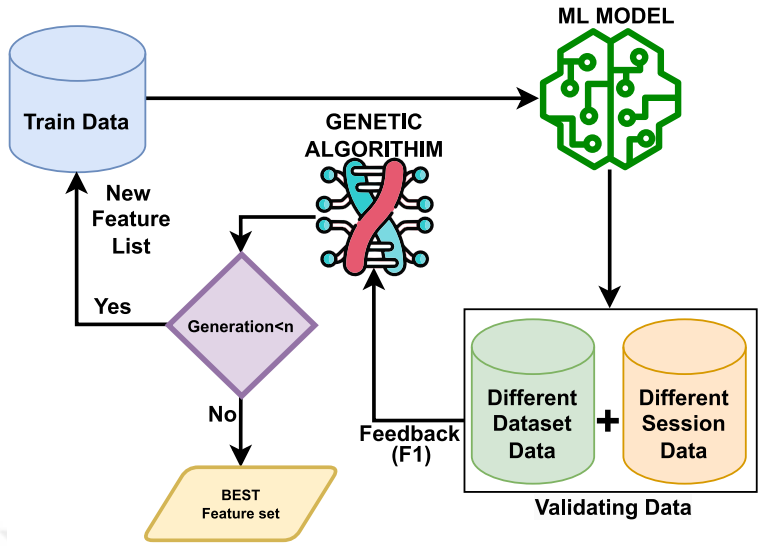


Figure 9.6: Visualisation of feature selection with Genetic Algorithm. The GA outputs the best feature set from n generations with the feedback it receives from the validation dataset.

9.3 Performance Evaluation

In this section, we examine the performance of attack detection models trained with the feature sets we have created. Table 9.2 presents the results for each attack when using window-based features. In each case, F1 scores are shown for when the model is evaluated using (a) cross-validation, (b) an independent session from the same dataset it was trained on, and (c) a dataset different to the one it was trained on. This is intended to give insight into each model's generality, with each subsequent evaluation scenario being a more stringent test of the model's ability to generalise to unseen (and therefore more realistic) conditions. F1 scores are presented in order to mask the effect of unbalanced data distributions. Other evaluation measures, such as accuracy, recall, and precision can be found in Appendix A.7.2.

For cross-validation, near-perfect discrimination is achieved for all attacks using at least one ML model. When an isolated session is used for evaluation, high levels of discrimination are observed for all but the Address Resolution Protocol (ARP)Spoofing (S) attack. When a different dataset is used for evaluation, high levels of discrimination are achieved for all but the ARPS, BF and Operating System Version Discovery (OSD) attacks. It is promising that the majority of attacks are still detected with good levels of

discrimination even in the most realistic evaluation condition. However, the success rate clearly drops as the evaluation conditions become more stringent, and this highlights the fact that cross-validation—an approach widely used in the literature—can lead to overly-confident measures of generality, even when steps are taken to remove features that cause overfitting in the feature selection stage.

Notably, the two decision tree ensemble approaches, XGB and RF, lead to the most successful models, with only a few exceptions. For BF and User Datagram Protocol (UDP)F attacks, NB demonstrated the highest success, while LR performed best for OSD attacks.

9.3.1 Comparison with Flow-Based Features

Next, we compare models trained using our window-based features against those trained using flow-based features, the latter being more typical of those found in the existing attack detection literature. In order to ensure a fair comparison, we use the same raw data and apply the same feature selection steps.

Table 9.3 shows the results. For cross-validation, both approaches achieve near-perfect scores in at least one ML model for UDP, Port Scanning (PS), and Synchronize (SYN) attacks. However, the window-based approach outperforms the flow-based approach, especially for the ARPS, ACK and BF attacks.

When evaluated using an independent session, both methods achieve near-perfect scores in at least one model for UDPF, PS, and SYNF attacks. The window-based approach exhibits superior performance for ACKF, BF, and HTTPF attacks, with approximately 3, 16, and 6 percentage point differences, respectively. For the ARPS attack, XGB model scores exhibit nearly identical performance between the two methods, while the flow method outperforms the window method across all other models.

When evaluated using an independent dataset, the flow-based approach fails to detect ACKF, SYNF, and UDPF attacks, while the window-based approach achieves scores above 0.94 for at least one ML model for all three attacks. Both approaches demonstrate similar performance in PS attacks, while the window-based approach outperforms the flow-based approach for ACKF, SYNF, and UDPF attacks.

9.3.2 Feature Effectiveness

These results strongly suggest that a window-based approach is beneficial in terms of both model performance and generality. To give more insight into why this is the case,

Table 9.2: F1 scores of attack detection models when evaluated using cross-validation, isolated sessions and isolated datasets.

Attack	ML	CV	Session Test	Dataset Test
ACK	DT	1.000	0.999	0.912
	KNN	1.000	1.000	0.212
	LR	1.000	1.000	0.986
	MLP	1.000	0.999	0.333
	NB	1.000	1.000	0.212
	RF	1.000	0.999	0.989
	SVM	0.694	0.235	0.333
	XGB	1.000	1.000	0.989
ARP	DT	0.994	0.583	0.454
	KNN	0.925	0.607	0.341
	LR	0.808	0.631	0.567
	MLP	0.642	0.563	0.373
	NB	0.494	0.411	0.329
	RF	0.997	0.664	0.345
	SVM	0.447	0.432	0.333
	XGB	0.997	0.733	0.349
BF	DT	0.999	0.481	0.351
	KNN	0.979	0.925	0.520
	LR	0.996	0.516	0.402
	MLP	0.938	0.969	0.486
	NB	0.602	0.601	0.738
	RF	1.000	0.974	0.317
	SVM	0.973	0.971	0.313
	XGB	0.998	0.898	0.313
HTTP	DT	1.000	0.993	1.000
	KNN	0.998	0.893	0.355
	LR	0.994	0.993	0.310
	MLP	0.985	0.930	0.388
	NB	0.632	0.450	0.356
	RF	1.000	0.996	1.000
	SVM	0.490	0.486	0.333
	XGB	1.000	0.996	1.000
MHD	DT	0.994	0.808	0.795
	KNN	0.981	0.992	0.993
	LR	0.984	0.994	0.998
	MLP	0.995	0.994	0.995
	NB	0.562	0.795	0.833
	RF	0.996	0.980	0.985
	SVM	0.557	0.581	0.587
	XGB	0.996	0.793	0.811
OS	DT	0.999	0.890	0.542
	KNN	0.944	0.493	0.457
	LR	0.998	0.885	0.691
	MLP	0.485	0.490	0.332
	NB	0.590	0.247	0.453
	RF	0.975	0.784	0.618
	SVM	0.485	0.499	0.333
	XGB	0.998	0.890	0.551
SHD	DT	0.999	0.890	0.889
	KNN	0.998	0.740	0.756
	LR	0.998	0.611	0.639
	MLP	0.999	0.560	0.781
	NB	0.490	0.687	0.726
	RF	0.999	0.816	0.684
	SVM	1.000	0.497	0.495
	XGB	0.999	0.954	0.912
PS	DT	1.000	1.000	0.896
	KNN	1.000	1.000	0.997
	LR	0.999	1.000	0.997
	MLP	1.000	1.000	0.997
	NB	0.972	0.963	0.971
	RF	0.999	1.000	0.862
	SVM	0.472	0.608	0.461
	XGB	0.999	1.000	0.945
SYN	DT	1.000	0.999	0.930
	KNN	0.999	0.997	0.298
	LR	1.000	0.993	0.981
	MLP	0.999	0.997	0.333
	NB	0.993	0.993	0.998
	RF	1.000	0.999	1.000
	SVM	0.945	0.951	0.333
	XGB	1.000	0.999	1.000
UDP	DT	1.000	0.884	0.359
	KNN	1.000	1.000	0.218
	LR	1.000	1.000	0.218
	MLP	0.999	1.000	0.812
	NB	0.669	0.983	0.941
	RF	1.000	1.000	0.377
	SVM	1.000	1.000	0.556
	XGB	1.000	1.000	0.218

Table 9.3: F1 score comparison of window and flow-based approaches.

		CV		Session Test		Dataset Test	
Attack	ML	Window	Flow	Window	Flow	Window	Flow
ACK	DT	1.000	0.893	0.999	0.966	0.912	0.113
	KNN	1.000	0.370	1.000	0.022	0.212	0.093
	LR	1.000	0.894	1.000	0.966	0.986	0.113
	MLP	1.000	0.895	0.999	0.966	0.333	0.113
	NB	1.000	0.799	1.000	0.886	0.212	0.115
	RF	1.000	0.893	0.999	0.966	0.989	0.113
	SVM	0.694	0.894	0.235	0.966	0.333	0.113
	XGB	1.000	0.893	1.000	0.967	0.989	0.113
ARP	DT	0.994	0.659	0.583	0.698	0.454	0.401
	KNN	0.925	0.636	0.607	0.710	0.341	0.599
	LR	0.808	0.661	0.631	0.674	0.567	0.336
	MLP	0.642	0.641	0.563	0.686	0.373	0.759
	NB	0.494	0.463	0.411	0.428	0.329	0.730
	RF	0.997	0.682	0.664	0.727	0.345	0.421
	SVM	0.447	0.375	0.432	0.424	0.333	0.211
	XGB	0.997	0.681	0.733	0.731	0.349	0.389
BF	DT	0.999	0.851	0.481	0.767	0.351	0.433
	KNN	0.979	0.781	0.925	0.726	0.520	0.359
	LR	0.996	0.456	0.516	0.458	0.402	0.316
	MLP	0.938	0.640	0.969	0.564	0.486	0.443
	NB	0.602	0.486	0.601	0.506	0.738	0.686
	RF	1.000	0.883	0.974	0.810	0.317	0.343
	SVM	0.973	0.474	0.971	0.480	0.313	0.316
	XGB	0.998	0.892	0.898	0.803	0.313	0.503
HTTP	DT	1.000	0.962	0.993	0.933	1.000	0.681
	KNN	0.998	0.890	0.893	0.840	0.355	0.566
	LR	0.994	0.934	0.993	0.919	0.310	0.920
	MLP	0.985	0.761	0.930	0.791	0.388	0.359
	NB	0.632	0.804	0.450	0.747	0.356	0.971
	RF	1.000	0.971	0.996	0.962	1.000	0.596
	SVM	0.490	0.392	0.486	0.369	0.333	0.462
	XGB	1.000	0.968	0.996	0.941	1.000	0.652
PS	DT	1.000	0.995	1.000	0.994	0.896	0.949
	KNN	1.000	0.966	1.000	0.912	0.997	0.696
	LR	0.999	0.990	1.000	0.999	0.997	0.994
	MLP	1.000	0.924	1.000	0.891	0.997	0.685
	NB	0.972	0.959	0.963	0.966	0.971	0.739
	RF	0.999	1.000	1.000	0.998	0.862	0.980
	SVM	0.472	0.516	0.608	0.475	0.461	0.546
	XGB	0.999	1.000	1.000	0.997	0.945	0.980
SYN	DT	1.000	0.977	0.999	0.988	0.930	0.127
	KNN	0.999	0.932	0.997	0.913	0.298	0.367
	LR	1.000	0.960	0.993	0.901	0.981	0.422
	MLP	0.999	0.975	0.997	0.913	0.333	0.208
	NB	0.993	0.746	0.993	0.832	0.998	0.372
	RF	1.000	0.992	0.999	0.992	1.000	0.000
	SVM	0.945	0.499	0.951	0.498	0.333	0.500
	XGB	1.000	0.977	0.999	0.986	1.000	0.110
UDP	DT	1.000	1.000	0.884	1.000	0.359	0.001
	KNN	1.000	0.926	1.000	0.441	0.218	0.001
	LR	1.000	0.975	1.000	1.000	0.218	0.001
	MLP	0.999	0.945	1.000	0.437	0.812	0.001
	NB	0.669	0.628	0.983	0.687	0.941	0.001
	RF	1.000	1.000	1.000	1.000	0.377	0.001
	SVM	1.000	0.493	1.000	0.289	0.556	0.001
	XGB	1.000	0.990	1.000	1.000	0.218	0.001

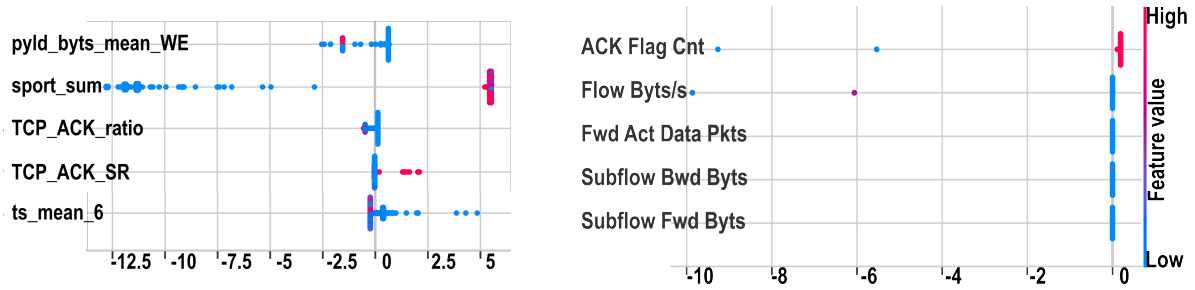
in this section we analyse the trained models using an XAI technique, SHapley Additive exPlanations (SHAP) [140, 165], and consider how features are actually used for attack detection.

9.3.2.1 Flood attacks

First we consider the four flood attacks: ACK, HTTP, SYN, and UDP floods.

ACK flood (ACKF) For this attack, we see that both types of feature support useful models for the first two evaluation scenarios, but only window-based features support models that generalise to an independent dataset. Figure 9.7a shows that the most important feature in the window approach is the *sport_sum*. While this value is low in benign samples, it is usually high in attack data. In the case of ACKF attack, this value gives the idea that the source port diversity is high. We see that *payload_bytes_mean_WE* and *ts_mean_6* features also play an active role due to the relatively small size of the packet flow in a short time in the attack situation. *TCP_ACK_SR* and *TCP_ACK_ratio* are also important, because they give statistics about the ACK flag. In the flow approach, two prominent features stand out: *Flow Byts/s*[3] and *ACK Flag Cnt*. This is expected as an ACKF attack involves an abundance of ACK flags and relatively smaller data flow in terms of size. However, considering the number of packets with ACK flags or the size of the flow may not provide sufficient information for determining the presence of an attack, and this may explain why the use of these features limits the generality of the model when evaluated on a different dataset.

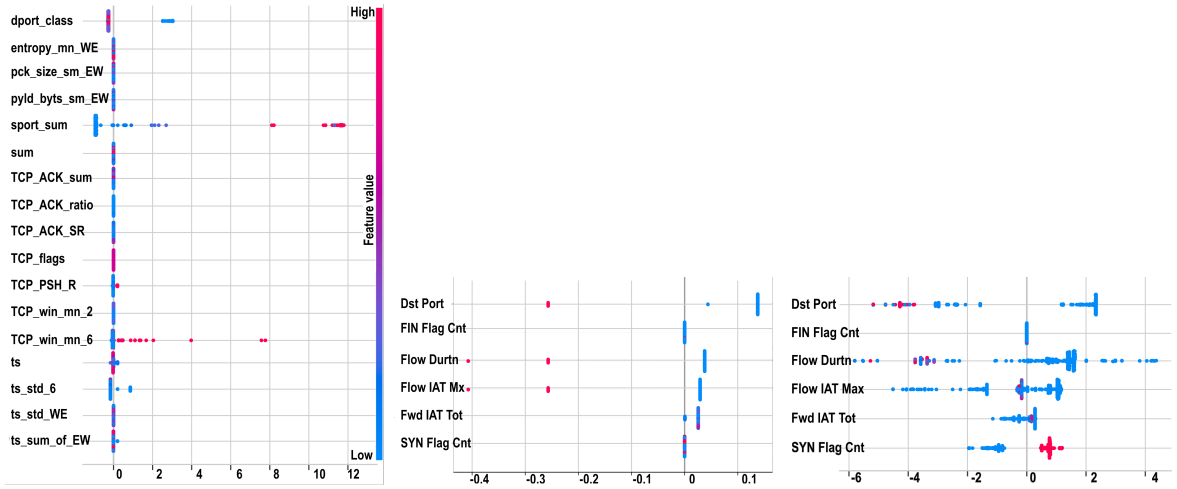
[3]Feature list: github.com/ahlashkari/CICFlowMeter/blob/master/ReadMe.txt



(a) Window-Based XGB. (b) Flow-Based XGB.

Figure 9.7: Comparison of window and flow-based features for the ACK attack, showing the effect of different features with their importance. Each point in this SHAP plot represents a sample in the dataset. On the y-axis, the attributes are listed in descending order of importance from top to bottom (with a maximum of 20 features shown per plot if the number of features is more than 20, otherwise all features). The x-axis shows the SHAP value: a high value indicates that it contributes to the attack class, while a low value contributes to the benign class. The colour transition indicates whether the value of the feature is high (red) or low (blue).

HTTP flood (HTTPF) Both feature approaches support the training of discriminative models, though the models build from window-based features are slightly better. Interestingly, ML model type is important here: for window features, decision tree models generalise well to an independent dataset, whereas for flow features, it is the the NB and LR models that generalise. Taking this into account, Figure 9.8 shows the SHAP analysis of the XGB and NB models. For window features (Figure 9.8a), the two most important features are related to port numbers. The *sport_sum* feature shows a high port diversity in the attack data, while the *dest_port_class* is varied for benign and uniform for attacks. Apart from this, another distinguishing feature is related to TCP-window size. This value appears to be higher in the attack data. For flow features (Figure 9.8c), it is similarly observed that the most important characteristic is the destination port. While the normal data shows variation, the port number is stable in the attack data. Another important feature is the flow time, presumably because HTTPF attack-generating flows tend to be shorter. When we compare with Figure 9.8b, we can see that similar features are also prominent there. However, while the flag-related feature creates a clear distinction in XGB, it is not noticeable in the NB model. When evaluated on an independent dataset, we think the main reason for the failure of the flow-based XGB model is that this flag feature causes overfitting.

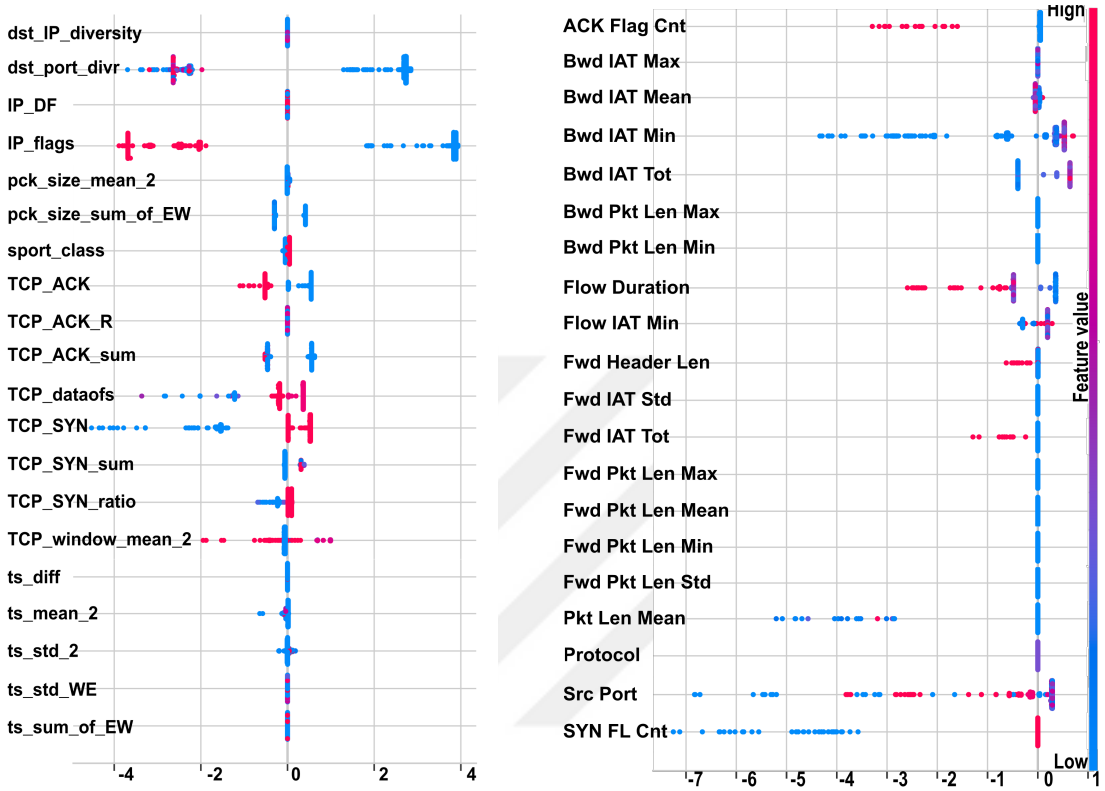


(a) WB XGB. (b) FB NB. (c) FB XGB.

Figure 9.8: Comparison of window (WB) and flow (FB) models for HTTP Flood attack.

SYN flood (SYNF) For the SYNF attack both approaches are successful in the first two evaluation conditions, but only the window approach generalises to an independent dataset. In Figure 9.9 the XGB model for both approaches is analysed with the SHAP plot. In Figure 9.9a, we can see that the variety of ports provides good discrimination. This is likely due to the fact that this attack targets a specific port. On the other hand, it can be seen that flag statistics features are concentrated. One of the biggest effects of SYN attacks on the network is the anomaly in flags. In this context, it is quite reasonable that features related to SYN and ACK flags are particularly important. While the *IP_flag* feature is not expected to make a significant contribution to the classification between the two groups, it was observed as a distinguishing feature in the model. When Figure 9.9b is analysed, we can see that the Inter-Arrival Times (IAT) features stand out and make a significant distinction. Although they have a lower importance score, flow time, port number, flag and size statistics (such as *Flow Duration, Src Port, SYN Flag Cnt, ACK Flag Cnt, Fwd Pkt Len Mean, Fwd Pkt Len Std, Bwd Pkt Len Max* etc.) are important discriminating features. Considering the attack characteristics, we expect significant success by utilizing these features alone. However, the overfitting of the model due to the prominence of IAT features overshadows the contributions of these features. The potential drawbacks associated with IAT features are outlined below. These features can lead to high-dimensional datasets, increasing model complexity and allowing patterns in the noise to cause overfitting. In addition, not all IAT-based features contain the same information, potentially introducing redundancy and irrelevance, further confounding

the model. Sparse data distributions can result, with more features than meaningful data instances, leading to poor performance on unseen data. Complex temporal patterns captured by IAT-based features may struggle to generalise effectively, and insufficient data can exacerbate the risk of overfitting by preventing accurate pattern learning and signal/noise discrimination.



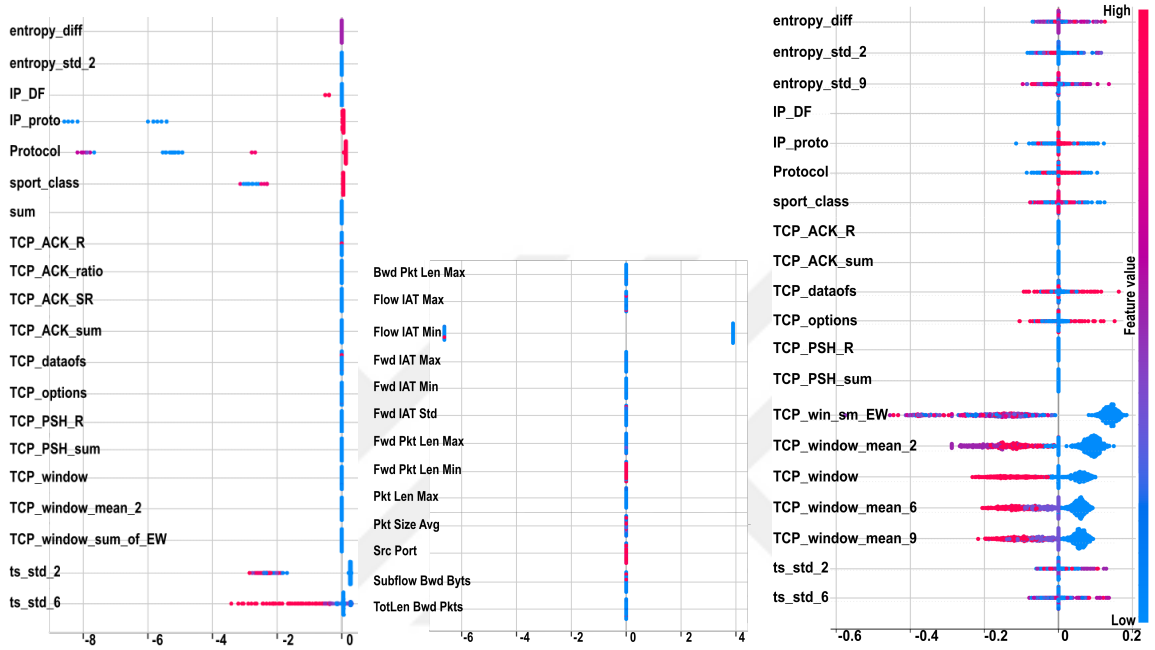
(a) Window-Based XGB.

(b) Flow-Based XGB.

Figure 9.9: Comparison of window (WB) and flow (FB) models for SYN attack.

UDP flood (UDPF) Only the NB and MLP models in the window approach achieved significant success when evaluated on an independent dataset. The SHAP plots for these models are presented in Figure 9.10. For the flow models, Figure 9.10b illustrates the prominence of a solitary feature, denoted as *Flow IAT Min* which pertains to the minimum time interval between packets within a flow. The clear dominance of this particular feature triggers concerns about overfitting which implies an excessive dependence on this attribute, hindering the model's ability to generalize effectively. Examining Figure 9.10a and Figure 9.10c, it can be seen that features that focus on the time between packets cause separation in both models (*ts_std_6, ts_std_2*). However, protocol- and port-centric features (*IP_proto, Protocol, sport_class*) are also prominent. These features are quite

reasonable for this attack's detection. Apart from these, the use of TCP-based features in the model is significant. The TCP header has a much more complex structure than the UDP header and therefore contributed more to the feature pool. However, since all attacks in this dataset use the UDP protocol, even the fact that a packet is TCP is a serious discriminator (if it is TCP, it is benign). In an experiment using a dataset with a predominance of UDP-based benign packets, TCP-based characteristics would not be expected to be so dominant.



(a) WB MLP. (b) FB XGB. (c) WB NB.

Figure 9.10: Comparison of window (WB) and flow (FB) models for UDP attack.

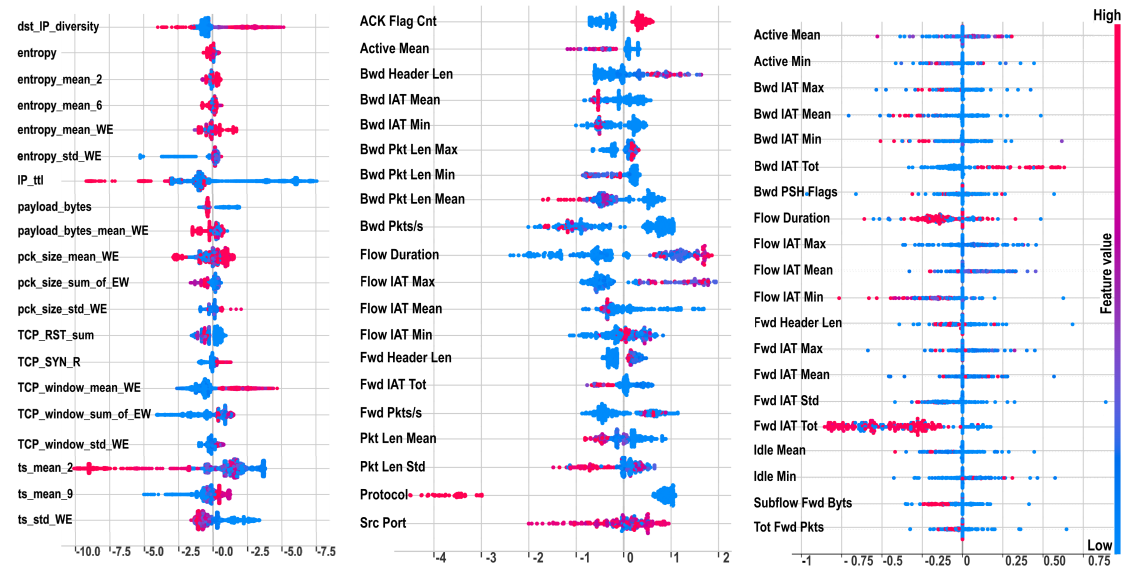
9.3.2.2 ARPS, BF, and PS Attacks

This section examines the feature effectiveness of three other attacks.

ARP Spoofing (ARPS) This attack is notable for being the only one where the flow based approach outperforms the window-based approach when evaluated on an independent dataset. It is also notable in that the attack packets are very similar to benign packets. The only difference is the time fluctuations in packet transport due to the involvement of the attacker. Therefore, it is very difficult to detect. As a consequence, the detection rate of this attack is generally low. Figure 9.11 shows the SHAP analysis of the XGB models, which are the most successful for the frist two

evaluation conditions, and the flow based MLP models, which generalise best to the independent dataset. When we examine Figure 9.11a, we can see that only 3 of the 10 most important features are able to give information about the time between packets. *ts_mean_2* takes the first place, but *ts_std_WE, ts_mean_9* are far behind. *IP_ttl* seems to be a useful feature for distinguishing IP addresses that look the same, rather than being an individual feature. The prominence of the *dst_IP_diversity* feature, which provides information about the IP-MAC relationship, is also quite understandable. Even if size, TCP-window or flags related features were highlighted, we do not think they would be useful in detecting this attack. Examination of Figure 9.11b shows that protocol, size, and flag based features stand out. We do not consider these features to be of any importance for an ARPF attack. In particular, although the *Protocol* and *ACK Flag Cnt* features provide a clear distinction, we do not consider this to be specific to the dataset and therefore not generalisable. On the other hand, the IAT features (*Fwd IAT Tot, Bwd IAT Mean, Flow IAT Min, Bwd IAT Min, Flow IAT Mean, Flow IAT Max*), which express the time statistics between packets, have the potential to provide discrimination, although they are suppressed by other features. When Figure 9.11c is examined, it is seen that with a few exceptions (*Fwd Header Len, Subflow Fwd Byts, Tot Fwd Pkts*), all features are time-dependent, especially IAT based. In this respect, the overfitting problem caused by irrelevant features in the first two models is not observed in this model. Thanks to this characteristic, this model is relatively successful in comparison to the others. The distinguishing characteristic of this attack—setting it apart from benign scenarios—is the temporal fluctuations between packets. Given the extensive inclusion of inter-packet time features within the flow method, the heightened efficacy of this method in detecting the attack, compared to our approach, is comprehensible. Given the thorough integration of inter-packet time features in the flow method, it is logical that this approach excels in detecting the attack compared to our method. Nevertheless, except for this specific attack, our approach performs equally or superiorly in all other attack instances. Moreover, the window-based method possesses the potential for early attack detection owing to its feature extraction capability, which is independent of the end of the flow. To achieve a more comprehensive solution, resolving the issue could entail enhancing the inter-packet time properties of the window method or adopting a hybrid method that incorporates the flow method.

Brute-Force (BF) The window based approach performs best across the three evaluation scenarios. Model type is a significant factor: RF and XGB perform well in

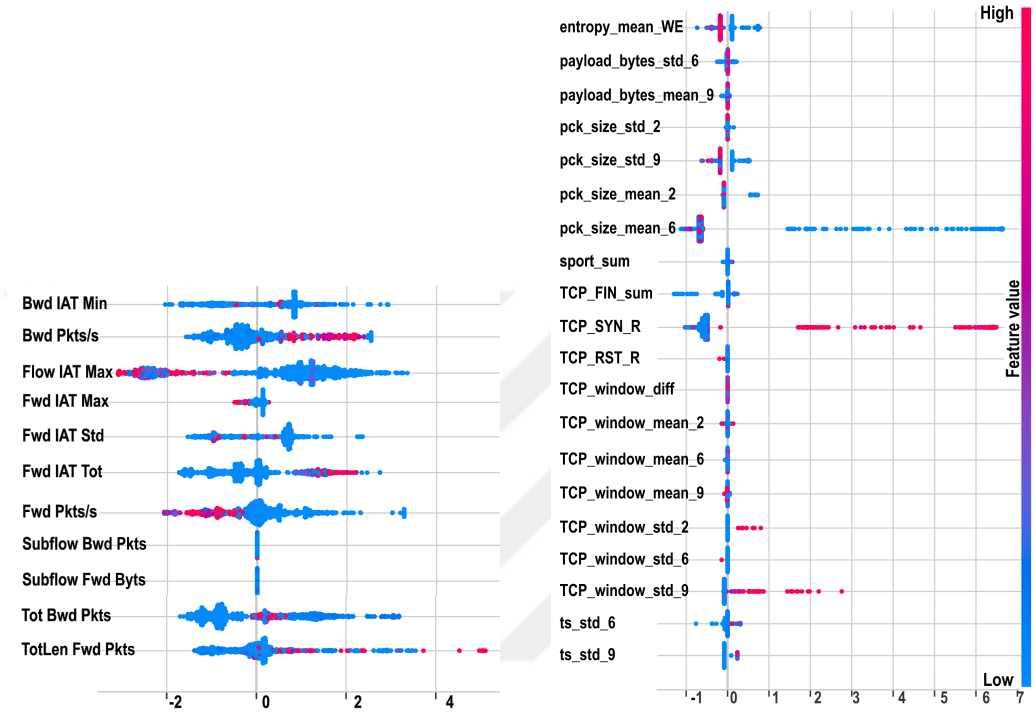


(a) WB XGB. (b) FB XGB. (c) FB MLP.

Figure 9.11: Comparison of window (WB) and flow (FB) models for ARPS attack.

the first two scenarios, but only NB achieves reasonable generality on the independent dataset. The telnet BF attack uses the TCP protocol to target specific ports and aims to obtain the username/password by trial and error. In this context, this attack is characterised by a large number of TCP packets targeting specific ports (23,2323 - telnet ports) in a short period of time. The fact that these packets contain passwords and usernames makes the payload-based characteristics important, and the size is expected to be in a certain range. Figure 9.12 shows the SHAP analysis of the XGB models. When Figure 9.12b is examined, it can be seen that size dependent properties such as *pck_size_mean6, pck_size_mean2, pck_size_std9 and entropy_mean_WE, payload_bytes_std6, payload_bytes_mean9* are used, and these provide information about the payload. In addition, TCP window size-related properties also provide good discrimination; this may be due to the fact that BF tools use a predefined window size [188]. Hence, window-based models seem to leverage suitable features. Additionally, the BF attack is comparatively straightforward to identify. Nonetheless, it is disconcerting that this attack exhibits relatively low performance in the third step. We attribute this suboptimal performance to the substantial dissimilarity between the training and test sets. In the initial two steps, both the training and testing datasets involve telnet BF attacks. However, during the third test phase, due to the unavailability of a matching attack type, we employed the Real-Time Streaming Protocol (RTSP) BF attack, which operates on a distinct protocol. Despite both being brute force attacks, there exist notable

differences in the tools, protocol, and approaches employed for each [99], and this may limit the ability of telnet BF models to generalise to RTSP BF attacks. For flow-based models, although features related to the number of outgoing and incoming packets (*Tot Bwd Pkts, Fwd Pkts/s, Bwd Pkts/s*) and packet size features (*TotLen Fwd Pkts*) are used, Figure 9.12a indicates that the IAT features (*Fwd IAT std, Bwd IAT Min, Tot Bwd Pkts* etc.) are more prominent, and this may explain their poorer performance.

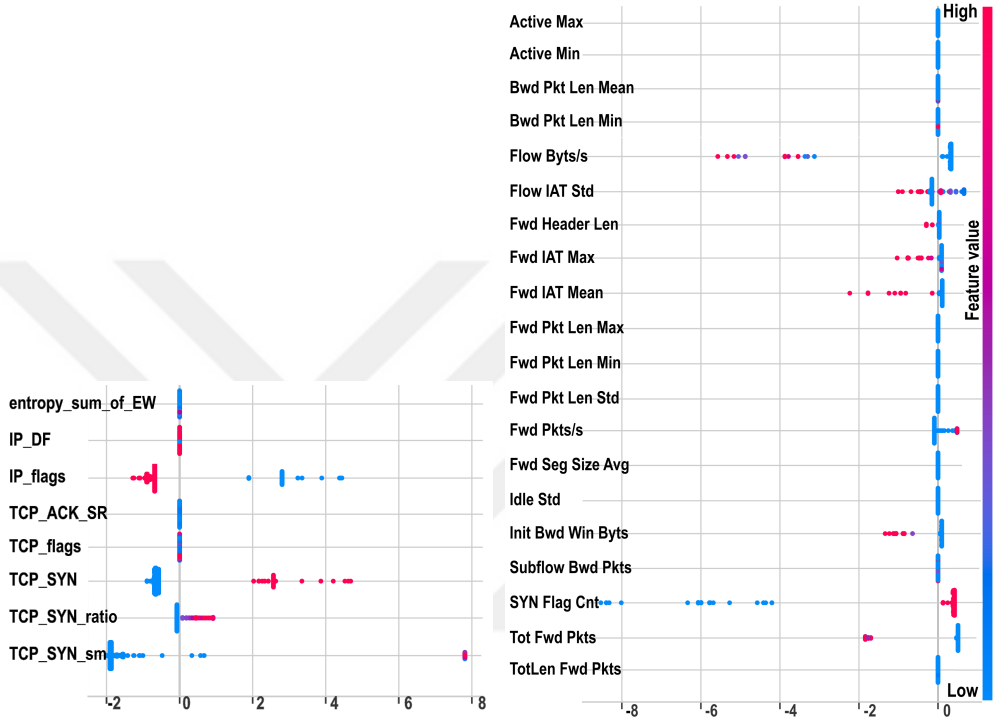


(a) Flow-Based XGB. (b) Window-Based XGB.

Figure 9.12: Comparison of window and flow-based models for BF attack.

Port Scanning (PS) For this attack, generalisable models could be trained using both window and flow-based features. In the port scanning attack scenario, the attacker sends many TCP packets to the ports with SYN flags set. In this context, flag-based features are important. However, features that provide information about the ports can also be distinctive. When Figure 9.13a is analysed, it can be seen that the features related to SYN flag statistics (*TCP_ACK_SR, TCP_SYN_ratio, TCP_ACK_sum, TCP_SYN, TCP_SYN_sum*) stand out in the window approach. In addition, the effect of payload-based properties (*entropy_sum_of_EW*) is also used, presumably since attack packets typically lack payload. Unexpected, however, is the use of an IP flag-based feature. This appears to be due to a spurious correlation in the training data, showing that even

a robust approach to feature selection can not always prevent models from picking up on irrelevant features. For the flow models, Figure 9.13b shows that the most important feature is the SYN packet count. Features related to packet count and packet size (*Init Bwd Win Byts,Fwd Pkts/s,Flow Byts/s,Tot Fwd Pkts* etc.) are prominent, and IAT features are also used, which seems reasonable for the analysis of the time between packets, due to heavy packet flow during attacks. However, it is quite interesting that the port-related features are not ranked highly in either SHAP analysis.



(a) Window-Based XGB. (b) Flow-Based XGB.

Figure 9.13: Comparison of window and flow-based models for the PS attack.

Both BF and PS attacks share a common objective of unauthorized access to systems or networks, albeit employing distinct methods. While BF attacks systematically attempt all credential combinations to breach security, PS attacks concentrate on identifying open ports to exploit potential vulnerabilities. In terms of feature importance for detection models, BF attacks, particularly those targeting Telnet protocols, prioritize payload-based features such as size-dependent characteristics and features related to TCP window size. This focus aligns with the nature of BF attacks, which aim for unauthorized access through repetitive login attempts. Conversely, PS attacks emphasize flag-based features, particularly those associated with SYN packet statistics, alongside payload-based features. A potential similarity lies in time-based characteristics, such as

the concentration of activities during specific intervals, though such features were not distinctly evident in either attack.

9.3.2.3 Flow-less Attacks

Table 9.3 excludes SHD, MHD, and OSD attacks because CICFlowMeter cannot extract features for these attacks. Host discovery attacks use MAC addresses for source and destination, making it incompatible with IP-based methods like CICFlowMeter. The OSD attack relies on individual packet properties like TCP window size and TCP flags, which are not accessible through CICFlowMeter. In this context, one of the advantages of our approach is that it makes it possible to obtain a wider range of layer features and thus potentially detect many more types of attacks. Although it is not possible to compare with the flow-based approach, we have performed feature analysis for these attacks using SHAP. Below, we analyse the most successful models for OSD, MHD, and SHD attacks: LR, LR, and XGB, respectively.

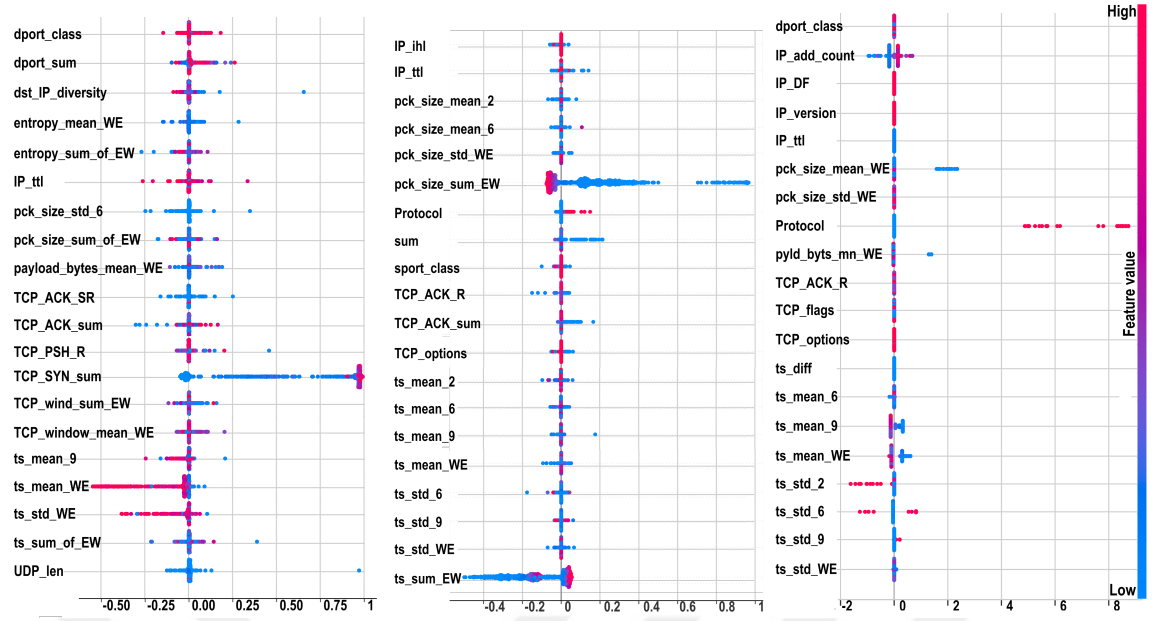
Figure 9.14a shows the characteristics given for the OSD attack. Since this attack consists of TCP packets with active SYN flags, TCP-based features are prominent. In addition, depending on the intensity of the attack, time-based features are also important. Even LR, which is the most successful model for detecting the OSD attack in our study, has not achieved significant success. This is likely due to the overfitting that occurred because the training dataset for this attack was very limited (see Appendix Table A.21 for sample numbers per attack).

For MHD attack, Figure 9.14b reveals that the Protocol, IP and TCP-related features are most significant, presumably because this attack does not have IP or TCP headers because it is composed of ARP packets. Considering that all network packets in the ARP protocol have the same size, we can say that time characteristics are effective in distinguishing between benign-attack packets.

Figure 9.14c shows the XGB model analysis for the SHD attack. Since SHD is a version of MHD with a different tool, similar characteristics are evident.

9.3.2.4 Summary

With the notable exception of ARPS attacks, the window approach is generally ahead in the detection of all attacks. The success of the flow approach is particularly low in cases where data from another dataset is used to evaluate the models. Based on the analysis of these attack models, one reason for this appears to be the extensive use of network-specific statistics in the features of the flow approach. Although models trained



(a) OSD LR. (b) MHD LR. (c) SHD XGB.

Figure 9.14: SHAP graphs of models for MHD, SHD, OSD attacks.

with these features may perform well in a specific network environment, the model is unable to sustain this performance when evaluated using data collected from a different network environment. In addition, the fact that flow feature extraction only works at the IP level makes it impossible to adapt to some types of attacks, such as MHD and SHD.

To augment the robustness of our comparative analysis, we utilize SHAP plots to highlight influential features in individual decisions made by the detection methods. While SHAP does not inherently unveil the inner workings of a model, it serves to identify features with more pronounced influence on specific decisions. The insights derived from SHAP analyses provide a critical perspective on the models' decision-making processes, offering a deeper understanding of how they interpret attacks. This validation contributes to our confidence in the models, emphasizing that they are anchored in meaningful interpretations of attack patterns and thereby reinforcing the reliability of our comparisons.

9.4 Conclusions

Previous research on behaviour-based attack detection on networks of IoT devices tends to produce attack detection models whose ability to adapt to unseen data is limited,

and often not demonstrated. In this chapter we present an approach for modelling IoT network attacks that focuses on generalizability and also leads to improved detection and performance.

In this context, we have analysed different feature sets, proposed a window-based approach to feature extraction that can provide packet-level attack detection, and compared it with the more conventional flow-based approach (RQ2.2, RQ2.3, C2.1, C2.3). We used a multi-step feature selection method, including a GA, to discover sets of features which are both predictive and generalizable. Since the feature selection method is based on external feedback from an independent dataset, it is very successful in eliminating features that are caused by information leakage and that lead to overfitting, which is a common problem in attack detection (RQ2.3, RQ2.6, C2.1, C2.4). The resulting feature sets were then used by eight different machine learning algorithms to detect ten attack types. The resulting attack detection models were evaluated using three different scenarios, testing their ability to generalise beyond the training distribution. Particularly notable was the superior ability of window-based models to generalise to previously unseen datasets, when compared to flow-based models. SHAP analysis (C2.4) of the most important features used by models showed that highly specific features such as IAT in the flow characteristics are prominent in many attacks. However, since these features provide information about the dynamics of the network rather than the nature of the attacks, they are far from generalisable and lead to models that suffer from overfitting. On the other hand, we find that the features of our approach are more in line with the nature of the attack, thus producing generalisable models that are successful even for previously unseen attack patterns.

Overall, our results showed success in detecting 10 different isolated (see Figure 9.2) attacks. In this context, it achieved $\geq$99% F1 score in ACK, HTTP, SYN, MHD, and PS attacks, 94% in UDP, 91% in SHD. Although no significant success was achieved in OSD, ARPS and BF attacks, the reasons for the failure of the models were analysed. In this context, upon analyzing the outcomes of the three models, it becomes apparent that our feature set may not be well-suited for the detection of ARPS attacks. The importance of IAT features becomes evident in effectively detecting this type of attack. In light of this, the flow-based approach, enriched with a more comprehensive array of IAT features, demonstrates greater efficacy in achieving successful detection. In the case of the OSD attack, we think that there are too few attacks on which to reliably train a model and that the model suffers from overfitting by focusing on features that reflect the features of the benign label. As a result, it is unable to cope with the test data, which is

much richer in terms of attacks. As for the BF attack, we believe that the failure in the last stage is because, although test data is also a brute force attack, the attack used in the test data is a sub-type of BF (RTSP BF) and quite different from the method used in the training data.

Chapter 10

Conclusions, Limitations and Future Work

> The end is another beginning.
>
> Rumi

10.1 Conclusions

The security of Internet of Things (IoT) devices is of paramount importance due to their widespread deployment and the potential for security oversights. In our endeavour to fortify IoT device security, our primary objective was to concentrate on attack detection, identifying it as a primary threat. Our research in this domain has revealed that IoT devices are susceptible to numerous vulnerabilities owing to their heterogeneous nature, resulting in instances of IoT-based attacks exploiting these weaknesses. Consequently, we chose to address the underlying vulnerabilities before tackling the specific attacks, akin to eliminating the breeding grounds before targeting the mosquitoes.

Our initial focus centred on device identification, representing the foundational stage of our IoT device security research due to its potential to offer comprehensive solutions for IoT device vulnerabilities. Subsequently, we conducted a thorough investigation into attack detection. Interlinking these two research themes, we drew inspiration from the Yin-Yang (☯) symbol's alliteration in our introduction (see Figure 1.3). It's crucial to highlight that the relationship between these thematic elements within our thesis is not strictly linear but rather symbiotic, reinforcing and enhancing each other. In this section, we shall elucidate the key insights gleaned from our thesis, aligning them with

these two pivotal research themes. Color codes according to themes: Device Identification (DI):🟡, Attack detection (AD):🔵.

🟡 **Using individual packets in device identification allows detecting devices using different communication protocols.**

In device identification, both flow statistics and header statistics are obtained by concatenating multiple packets. Even features extracted from individual packets are in some cases combined to produce larger fingerprints. However, it's important to note that in all the mentioned cases, the grouping of packets is based on IP or Media Access Control (MAC) addresses. This limitation means that these methods are not effective in scenarios where different communication protocols are used, as they rely heavily on IP and MAC addresses. To address this limitation, utilising individual packets for device identification becomes crucial. This approach helps solve the problem by allowing device identification across a broader spectrum of situations, where various communication protocols may be in use.

🔵🟡 **Differences between datasets make it difficult to compare datasets.**

One measure of success in device identification and attack detection is comparing results with other studies. There are many open-access datasets for device identification. In the case of benchmarking, different datasets are likely to have different device distributions and different data types. For example, Aalto, a widely used dataset, contains only installation data, hosts too many similar devices, and contains devices using different connection protocols that can cause transfer problems. These disadvantages make the Aalto very challenging compared to other datasets on the market. In this context, the structure of the datasets should be taken into account when analysing the results obtained from the datasets.

This is very similar for attack detection datasets. When attack detection databases are analysed, it is evident that they contain a wide range of attacks. Due to the differences in these attacks, the detection successes are quite different from each other. In this context, it is not possible to compare the results obtained from different datasets.

Even in studies using the same dataset, comparisons can be very difficult due to factors such as the evaluation criteria, the exclusion of some of the devices/attacks from the study, etc.

🔵🟡 **Utilising multiple datasets is crucial for demonstrating the generalizability of the research.**

When only a single dataset is utilised, the methodology might inadvertently develop

a bias toward that specific dataset. Without testing it on alternative datasets, it's impossible to ascertain the presence of this bias. Therefore, in any research endeavour, it is highly advantageous to evaluate these methodologies across multiple datasets, showcasing their broader applicability. Such evaluations can assess the generalizability of the feature set, algorithm, or method.

We are aware that finding these intersections is quite difficult, as testing generalisation of the model sample requires using the same devices for device identification and the same attacks for attack detection. A pragmatic approach to this kind of data scarcity would be to ensure that data collection is done more than once by different people.

Data collection processes need to be standardised.

For the device identification datasets, benign data is acquired from a test bed employing various data collection approaches. One method involves having someone utilise these devices a couple times per day to capture normal usage data. Alternatively, data can be collected at specific intervals through automated processes. While automation ensures the collection of a substantial volume of data in a short timeframe, it may introduce an artificial pattern that does not follow the normal data distribution. Therefore, we contend that data collected by real people over an extended duration on the test bed provides a more accurate representation of the benign/normal pattern.

It is a common practice to segment data into sessions for device detection, yet uncommon to do so for attack detection. Segmenting data into sessions when gathering information about attacks has its merits. Repeating the data collection over multiple sessions expands the dataset and offers ample opportunities to separate training and test data. Moreover, altering the attacker device in some sessions during the repetition of the same attack diversifies the dataset and yields appropriate validation and test data.

information leakage represents a significant challenge in security work based on machine learning and can manifest in various dimensions.

In both device identification and attack detection, the inadvertent inclusion of information about the training set in the test set is a potential issue. To mitigate this leakage, it is advisable to carefully select test and training data from distinct sessions. However, in some scenarios, some features may mistakenly turn into identifying features, as the features used may reflect patterns specific to the dataset, regardless of device or session. That also suggests IP identifiers are poor features. For instance, in the primary experiments of device identification [116], we used the "IP add count" feature inherited from IoT Sentinel [158]. This feature gave the number of other devices that a device communicated with on the dataset, which for many devices is a unique identifier regardless of

sessions.

This usually occurs in very low-complexity datasets, using features such as size or time that reveal the pattern of the dataset. In this type of information leakage, one of the labels (usually the "attack" due to its uniform nature) is represented by data that has been stacked in a certain area in terms of time or size.

Need to be aware of and avoid pitfalls specific to IoT security.

To avoid common pitfalls, we recommend better adherence to the standard ML practices set out in checklists such as REFORMS [105]. However, IoT network security, particularly device identification and attack detection, presents its own unique challenges. Some of the key pitfalls in IoT security that we have encountered during our work are summarised below:

Labelling Issues

- Device identification commonly employs MAC addresses for labelling. However, relying solely on MAC addresses for grouping or merging in real-life applications can prove ineffective as one MAC address may correspond to multiple devices, posing challenges in accurate identification.
- It is advisable to utilise benign data for device identification studies. While the scarcity of benign data often leads to using parts of attack detection datasets for this purpose, caution must be exercised. For instance, datasets like Yourthings [11] may appear suitable due to the vast number of involved devices, but caution is warranted due to vulnerability tests conducted during data collection, making it closely resemble attack data.

Data Extraction Issues

- Despite the prevalent use of feature extraction in deep learning without explicit feature engineering, caution must be exercised when dealing with Packet Capture (pcap) files (raw network data). Feature extraction without adequately handling identifying information present in pcap files can result in information leakage.
- During network feature extraction, attention should be given to checksums, serving as potential identifiers due to their compressed representation of the header.
- Avoidance of feature sets focusing on a singular piece of information is advisable. Studies concentrating solely on DNS addresses or payload may face limitations in scenarios with multiple devices of the same brand or encrypted payloads.

- Valuable insights can be derived by analysing feature effects as part of the model output. Assessing overfitting through measures like feature importance scores and reflecting on prominent features are crucial steps in creating more generalizable models.

Data Distribution Issues

- Heterogeneity in IoT datasets introduces challenges, including data scarcity and imbalance. Some devices inherently generate limited data, can necessitating the use of data augmentation techniques. However, ensuring the test data reflects the real-life distribution is vital when employing augmentation or reduction techniques. Keeping the test data separate from these processes is recommended.

- Evaluation in datasets with imbalanced data should meticulously consider not only overall results but also class-wise results, mitigating biases from labels with excessive examples. Analysing the confusion matrix is recommended to gain a comprehensive understanding of the result.

Detecting an attack using only individual packet features is challenging.

For device identification, we have used features extracted from individual packets with very good results. However, attack detection presents a distinctive challenge compared to device identification in terms of using individual features. Regrettably, the features that prove efficacious in device identification often fall short in effectively addressing numerous attack scenarios. While extant research endeavours have demonstrated notable accomplishments employing individual features, we have substantiated, both theoretically and empirically through our experiments, that these achievements are primarily a result of overfitting owing to inadvertent information leakage.

How you combine individual packets matters.

We implemented an aggregation algorithm to improve device identification success. Unlike traditional approaches to MAC address merging, which have been criticised, our algorithm differs in the merging process. Traditional methods perform merging based on the MAC address only. Our method, on the other hand, is able to detect MAC addresses with different behavioural patterns by examining the model outputs and perform the merging process based on this detection. Thus, it is able to recognise devices that suffer from transfer problems, while at the same time achieving significantly improved performance over using only individual packets.

For attack detection rather than relying on features derived from individual packets for attack detection, opting for aggregate methodologies that emphasise the interplay

among network packets (e.g., rolling window or flow) stands as the basic approach for attaining results that demonstrate generalizability. Among these two methods, we chose the rolling window method. The window method has the potential for earlier detection than its alternative, the flow method. To create a feature in a flow system, the flow must be terminated with a termination flag or timeout. In this respect, a flow must wait until it is terminated before it can be identified as an attack, since feature extraction can only be done upon termination. The window method, however, performs feature creation by detecting changes at the packet level.

Our methodology, which has many intersections between our two themes, is open to development and application to other areas.

While we have had to adopt different feature sets for our efforts in device identification and attack detection, there is significant parallelism in many areas of our research. Both methodologies have exhibited a symbiotic relationship, significantly reinforcing each other.

For example, there is a significant convergence in our literature review approach in both studies. In addition, the use of session-based datasets, a standard practice in device identification, proves to be equally useful in the attack detection domain, underlining its versatile utility. Moreover, the commonalities extend to multi-layered feature selection techniques and the use of isolated data in the testing phase, creating a consistent methodological thread across both domains.

An opportune moment arose to extend the applicability of our methodology, presenting another context in which our approach has been used effectively. In an ancillary work [119], which is beyond the scope of this thesis, our goal was to estimate the spatial separation between two points by analysing Received Signal Strength Indicator (RSSI) signals. The feature selection process used in this work and its validation using isolated data closely mirrors the procedural steps followed in device identification and attack detection. This foray into a different field allowed us to observe and validate the effectiveness of our methodology in different applications.

Integrating Device Identification and attack Detection into an IoT System

While IoTDevID excels at identifying devices, it falls short in providing a solution to detect and act against vulnerable devices. On the other hand, IoTGEM effectively detects attacks but lacks the capability to take appropriate actions against them. The primary objective of both approaches is focused on "detection". Figure 10.1 outlines our proposed system to enhance cyber protection within a hypothetical network by

integrating device identification and attack detection.

The initial step in this five-step system is core security, addressing both physical security and user errors. Although this step is challenging to externally interfere with, it's included here to emphasise its paramount importance.

The data collection gateway is situated in Tier 3. At this gateway, the collected data undergoes analysis through device identification and attack detection systems. Both methodologies are linked to an Software-Defined Networking (SDN) system capable of network management by interpreting their results.

The device identification system pinpoints the brand and model of the devices, relaying this information to the SDN. Subsequently, the SDN assesses the vulnerability of these devices, employing a CVE-based database for this evaluation. Based on the vulnerability analysis, Internet of Things (IoT) devices in Tier 1 are categorised into zones (sub-networks) with distinct connection authorisations within Tier 2.

IoTGEM, also positioned in Tier 3, scrutinises the network for potential attacks, promptly notifying the SDN upon detection. This allows the SDN to initiate necessary measures, such as alarms, notifications, access blocking, or isolation, as a response to the detected attack.

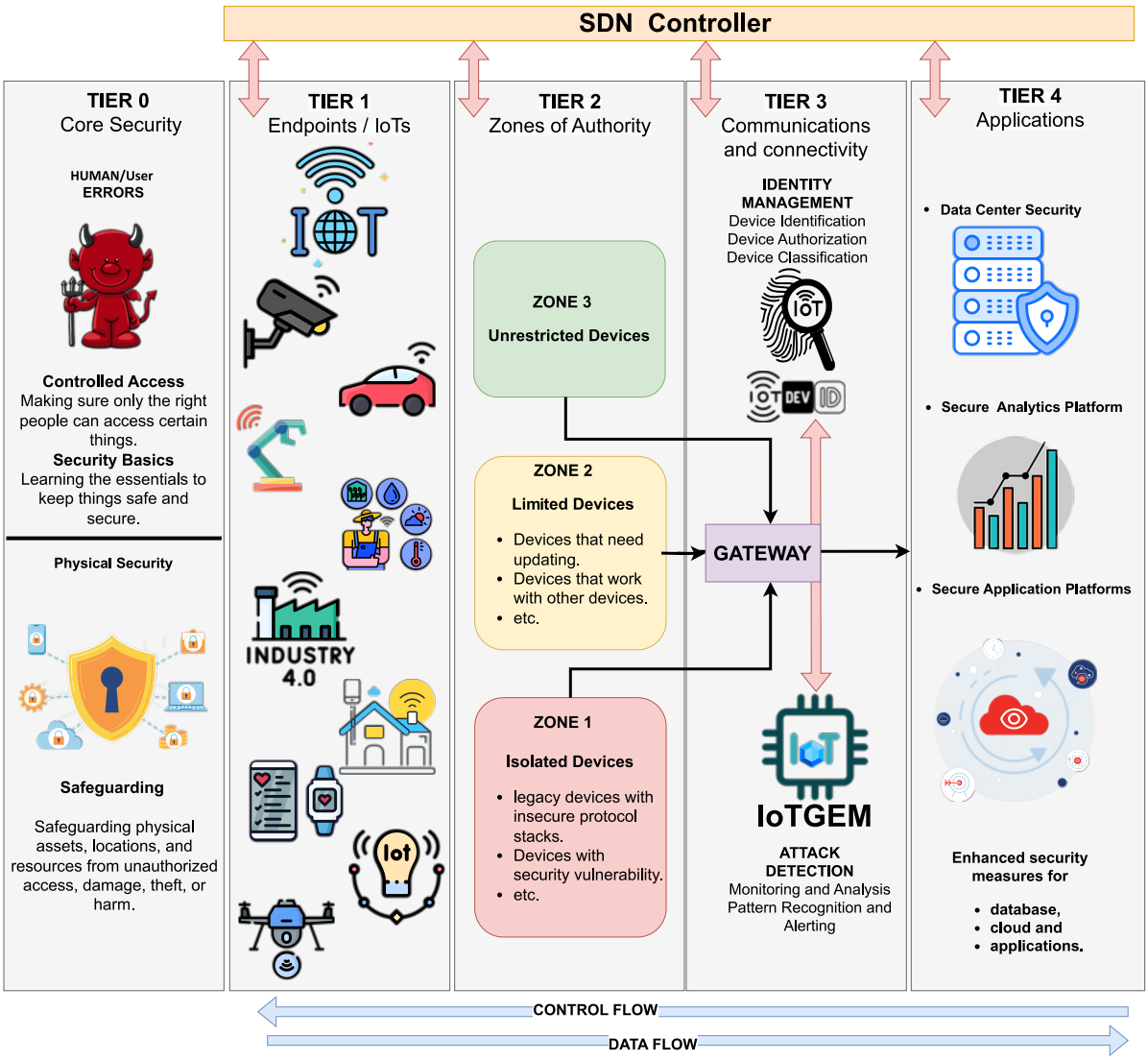


Figure 10.1: Proposed Integration of Device Identification and attack Detection for Enhanced IoT Cyber Protection.

10.2 Summary

In this thesis, we aim to provide a holistic understanding of the challenges and solutions of the field by bringing together comprehensive insights into IoT device identification and attack detection. The chapters that comprise this thesis and their relationship to the thesis objectives can be listed as follows:

Chapter 1 summarises the main motivations, objectives, research questions, and contributions that form the foundation of our work. Its purpose is to provide the reader with a navigational guide covering the main outlines of our thesis.

Chapter 2 delves into the foundational concepts bridging IoT device identification and IoT attack detection. It explores network models, machine learning algorithms, and evaluation methods, laying the groundwork for subsequent exploration. This comprehensive understanding sets the stage for our investigations.

Chapter 3 conducts a comprehensive literature review for IoT device identification. By examining the existing literature, it aims to provide a nuanced understanding of the datasets, features, and machine learning techniques used in IoT device identification (RQ1.1, RQ1.2, C1.1). It also serves as an important guide for innovative methodologies by illuminating potential pitfalls and limitations (RQ1.3, C1.1). The foundation established here forms the basis for subsequent chapters and serves as a catalyst for the creation of innovative solutions.

Chapter 4 introduces a novel device identification methodology. This chapter presents a machine learning-based approach to distinguish IoT devices in networks. The systematic approach addresses limitations identified in previous studies by striving for robustness, transparency, and real-world applicability (RQ1.2, RQ1.3, C1.1). The multi-stage feature selection represents a breakthrough, generating a set of features that surpass limitations and demonstrate significant performance (RQ1.3, C1.3). Solving challenges related to non-IP devices further enhances the adaptability and flexibility of this methodology (RQ1.4, C1.2). The developed methodology is tested on multiple datasets and compared with alternative methods (RQ1.7, C1.4).

Chapter 5 deepens the device identification research by validating feature sets and aggregation algorithms on an independent dataset (RQ1.4, RQ1.7, C2.2, C1.5). This chapter validates the importance of data diversity in model performance, particularly in the context of active and idle usage scenarios (RQ1.5, C2.6). Verification studies emphasise the need to evaluate model generalizability, a recurring theme carried forward to later chapters.

Chapter 6 represents the final work of device identification study. This section further tests the device identification methodology on a different dataset (RQ1.4, RQ1.7, C2.2, C1.5) while also analysing its effectiveness on encrypted and unencrypted networks (RQ1.6, C2.6). Leveraging data from different laboratories, this chapter demonstrates the model's ability to detect devices beyond training environments (RQ1.8, C1.7).

Chapters 7 to 9 shift the focus to IoT attack detection. Chapter 7 encompasses a literature review of IoT attack detection. Examining the literature on IoT attack detection provides critical insights into datasets, feature classifications, and machine learning techniques (RQ2.1, RQ2.2, RQ2.3, RQ2.4, C2.1). These insights contextualise

our efforts to renew and improve attack detection mechanisms in subsequent chapters.

Chapter 8 explores the limitations of using individual packets for attack detection, a common practice in the existing literature (RQ2.1, RQ2.2, RQ2.3, C2.1). By examining factors that can mislead model performance, including information leakage, this chapter aims to evaluate the strengths and weaknesses of relying solely on individual packet features. Thus, it encourages a better understanding of the disadvantages of using individual packets for attack detection and promotes the exploration of alternative strategies and feature combinations to enhance the reliability and accuracy of attack detection (RQ2.3, RQ2.5, C2.2, C2.4).

Chapter 9 highlights the importance of generalizability and improved performance in IoT network-based behaviour attack detection. Comparing a window-based feature extraction approach to the traditional flow-based method, this chapter describes identifying and generalizable features. It includes an advanced rolling window feature extraction method, multi-stage feature selection to reduce overfitting, and a careful evaluation of various machine learning models, metrics, and datasets (RQ2.2, RQ2.3, RQ2.4, RQ2.6, C2.3, C2.4). Additionally, the use of explainable artificial intelligence techniques assists in identifying critical features for accurate attack detection (C2.5).

Our research serves as a bridge between device identification and attack detection. We present a cohesive approach that not only identifies IoT devices but also detects malicious network activities. By aligning these capabilities, we establish a secure environment for benign data and thwart potential attacks. The journey we've undertaken highlights the significance of diverse datasets, robust feature selection, and model generalizability in building effective security solutions for IoT ecosystems.

In conclusion, this thesis represents a comprehensive exploration into the realms of IoT device identification and attack detection, contributing to the field's understanding and paving the way for enhanced security measures in the rapidly evolving landscape of IoT technology.

10.3 Limitations

In the quest to advance the field of IoT security, both our Device Identification and Attack Detection approaches make diverse contributions. However, as with any research endeavour, our methodologies are not without limitations, which we describe below. In this context, we list the limitations of our study under two headings in line with our two themes.

10.3.1 Device Identification Limitations

Dataset Selection: Our study incorporated all available publicly accessible IoT Device Identification (IoT DI) datasets as of the time of our research. Nevertheless, the rapid progress in IoT technologies likely led to the release of new data even during the course of our work. We encourage fellow researchers to utilise additional datasets to further assess the generalizability of our findings, and we provide scripts to facilitate this exploration.

Algorithm Exploration: While we employed a diverse set of commonly used machine learning algorithms, our selection was not exhaustive, and there may be alternative algorithms with superior performance. For example, this work highlights traditional Machine Learning (ML) approaches rather than deep learning. However, given the rapidly evolving deep learning environment and its potential, the study is not comprehensive or conclusive. It is also necessary to highlight ongoing research in deep learning due to its versatility and potential for innovation. Future research may uncover advances that challenge current results, highlighting the need for continued comparison of traditional machine learning and deep learning approaches.

Device Vulnerabilities: While IoTDevID effectively identifies devices, its scope does not extend to addressing vulnerabilities or implementing necessary security measures. Introducing a Software-Defined Networking (SDN) based network management solution offers a potential remedy to this limitation. Notably, IoTSentinel has proposed an SDN-based approach to audit device vulnerabilities using the Common Vulnerabilities and Exposures (CVE) database [45]. However, depending solely on the CVE database presents several risks. The database may lack comprehensive coverage of IoT device vulnerabilities, leading to undetected threats, delayed updates, false positives, and over-reliance on a single source. Moreover, it lacks real-time monitoring capabilities, hindering its effectiveness in identifying evolving vulnerabilities and device behavior. In tandem, RFC 9124 standards provide an alternative framework for device identification, incorporating attributes such as device type, manufacturer, software version, and vulnerabilities. The SDN controller can leverage these attributes to determine appropriate authorization levels, enabling actions like quarantining devices with known vulnerabilities and granting unrestricted access to those with up-to-date software and no identified vulnerabilities.

Network Context: IoTDevID is currently designed for deployment in benign networks, indicating a requirement for an attack detection system to safeguard against potential attacks. Integrating IoTDevID with attack detection mechanisms would enhance network security comprehensively.

Deployment Flexibility: The deployment, training, and maintenance of a Device Identification (DI) model in a specific network present considerations beyond our study's scope. While we primarily focused on multi-class classifiers for device discrimination, the scenario of frequent device additions and removals might necessitate an ensemble of single-class classifiers. This architectural shift would obviate the need for complete model retraining.

10.3.2 Attack Detection Limitations

Dataset Availability: Our approach's effectiveness is constrained by the availability of publicly accessible datasets. This limitation restricts the scope of our method to attacks that are covered by these existing datasets. As the landscape of IoT security is rapidly evolving, we anticipate the emergence of new datasets over time. Applying our approach to these new datasets would not only enhance its practicality but also contribute to the evolving literature in the field.

Algorithm Selection: In the machine learning phase, our emphasis was placed on widely recognised algorithms to align with established research trends. Nonetheless, the realm of machine learning encompasses a plethora of algorithms that we did not incorporate into our approach. While it would be intriguing to explore alternative machine learning algorithms, the feasibility of considering all possible algorithms is impractical due to their sheer number.

Efficiency: Our approach processes pcap files without incorporating timeouts. However, during the extraction of features using sliding window techniques, there's a possibility that the feature extraction process might exceed the required runtime, especially in scenarios involving Distributed Denial-of-Service (DDoS) attacks. This extended processing duration is attributed to the process continuing until the completion of the window size. To enhance the efficiency of the method, integrating timeouts in real-time applications would be a valuable improvement.

Feature Extraction Performance: Feature extraction, performed using Python and the Scapy library, is functional but relatively slow. While our approach generates accurate results, the speed of execution could be enhanced. Exploring the development of faster versions of our method using alternative languages like C++ or Go holds the potential to address this performance limitation.

10.4 Future Work

The research undertaken in this thesis has not only shed light on the current landscape of device identification and attack detection within the IoT security domain but has also established a foundation for forthcoming research initiatives. The ever-evolving nature of IoT security introduces new challenges and opportunities, compelling further investigation and innovation. This section delineates potential avenues for future research, building upon the insights derived from the present study.

The comprehensive literature reviews conducted on device identification and attack detection in this research are poised to serve as foundational resources for practitioners in the field. By meticulously identifying and scrutinizing common errors in device identification studies, we have provided a valuable guide for future research, helping steer investigations clear of potential pitfalls.

Our innovative work on cross-validation in device identification, specifically testing the trained model on different devices of the same type (refer to Chapter 6), represents a novel approach aimed at refining the precision and reliability of results in device type detection. We anticipate and encourage the broader adoption of this methodology in both IoT data generation and device detection studies.

We articulated a significant issue regarding the use of MAC addresses in identifying non-IP devices, terming it a "transfer problem" and suggested a solution to this problem. We believe that we have paved the way for further addressing this real-world problem in the future, both in the creation of new datasets and in device identification studies.

The examination of individual packets in attack detection revealed the weaknesses and potential problems associated with this method in real-world applications. Given its continued active use, we believe our study serves as a valuable cautionary reference for researchers considering this approach.

To ensure transparency and reproducibility, we have made our method accessible to other researchers for utilization and development. Researchers can freely leverage our open-access method, feature set, and codes, and they are encouraged to replicate our

experiments with different data while expanding and refining our method.

In summary, this chapter outlines a roadmap for future research, leveraging the foundational work presented in this thesis. From systematic reviews to the exploration of device states and data types, enhancing model generalizability, and scrutinizing the intricacies of individual packets in attack detection, our objective is to not only advance the boundaries of knowledge in IoT security but also to address critical gaps in the existing discourse.

Appendix A

Appendices

A.1 Feature Comparison: Individual Packet Headers vs. Flow-Based Features

This section focuses on comparing features derived from packet headers and flow-based features. The aggregated-IoTDevID (see Chapter 4) method is employed for packet header features, while flow-based features are analysed using the CICFlowMeter method. To ensure equitable comparison, a GA-based feature selection process akin to the IoTDevID method is implemented in the CICFlowMeter method. The selected dataset is the dataset for "information exposure for consumer IoT devices" (IMC'19) (MonIoTr) dataset (see Chapter 6), chosen for its VPN feature, facilitating comparisons under diverse conditions within the same laboratory setting, characterized by changing IP addresses and time values. Additionally, the dataset covers 25 intersecting devices across two laboratories, offering the opportunity to test methods on the same devices under various network conditions.

The experiment consists of three stages. In the initial stage, data collected from four different scenarios/laboratories (UK, UKVPN, US, USVPN) undergo cross-validation. In the second stage, data from each laboratory are tested under different states (normal vs. VPN and vice versa). In the final stage, models trained with intersecting devices in one laboratory are tested with data collected from another laboratory.

In the first stage, data from four scenarios/laboratories are subjected to the cross-validating process. Table A.1 presents the results for the four datasets obtained with IoTDevID and CICFlowMeter methods. These results represent the average of 10 times 2-fold cross-validation, chosen due to the presence of classes with few examples in the

datasets. Both methods exhibit considerable success, with the flow method achieving an F1 score of 80 or above for all four datasets, while IoTDevID exceeds 90 in this regard, establishing its superiority.

Table A.1: Comparison of four different cases with cross-validation on a MonIoTr dataset. Results include the mean and standard deviation of 10 times 2 fold cross validation.

Dataset	Flow Based		Packet Based	
	Accuracy	F1 Score	Accuracy	F1 Score
UK	0.908±0.002	0.808±0.011	0.962±0.001	0.940±0.018
UK-VPN	0.857±0.003	0.808±0.009	0.968±0.001	0.910±0.002
US	0.952±0.001	0.827±0.009	0.957±0.001	0.918±0.012
US-VPN	0.945±0.001	0.834±0.008	0.969±0.001	0.926±0.020

In the second stage, data from each laboratory are tested with the dataset collected in a different state. For instance, the UK laboratory data is used as training data, and the VPN version of the UK laboratory is used as test data. These data, differing in timestamps and IP addresses, are considered as collected in the same laboratory but in a different session/conditions. Examining Table A.2 reveals a decrease in both methods according to CV results. However, IoTDevID maintains a high level of success (F1 score of 81 or above for all four scenarios), while the Flow method experiences a notable decrease (ranging between 51 and 62).

Table A.2: MonIoTr dataset, comparison of VPN and normal data from the same laboratory. Results include the mean and standard deviation of 25 repeats.

Datasets		Flow Based		Packet Based	
Train	Test	Accuracy	F1 Score	Accuracy	F1 Score
UK	UK-VPN	0.513±0.004	0.620±0.005	0.921±0.000	0.814±0.014
UK-VPN	UK	0.447±0.003	0.529±0.004	0.935±0.000	0.863±0.002
US	US-VPN	0.667±0.003	0.511±0.005	0.900±0.001	0.814±0.002
US-VPN	US	0.657±0.002	0.532±0.011	0.915±0.001	0.828±0.006

In the final stage, models utilizing the intersecting devices feature of the MonIoTr dataset in one laboratory are tested with data from another laboratory. Table A.3 displays the results, where the IoTDevID approach maintains a high level of success (F1 score ranging between 72 and 87), while the Flow method exhibits lower scores (ranging between 37 and 50 F1 score).

Table A.3: MonIoTr dataset, comparison of data collected from two different laboratories. Results include the mean and standard deviation of 25 repeats.

Datasets		Flow Based		Packet Based	
Train	Test	Accuracy	F1 Score	Accuracy	F1 Score
US	UK	0.553±0.136	0.446±0.027	0.922±0.000	0.873±0.001
US-VPN	UK	0.661±0.005	0.484±0.012	0.932±0.001	0.813±0.011
US	UK-VPN	0.671±0.023	0.464±0.013	0.896±0.001	0.753±0.008
US-VPN	UK-VPN	0.389±0.004	0.393±0.008	0.895±0.002	0.832±0.004
UK	US	0.501±0.012	0.418±0.014	0.846±0.000	0.823±0.002
UK-VPN	US	0.613±0.005	0.372±0.008	0.837±0.000	0.793±0.009
UK	US-VPN	0.710±0.009	0.498±0.012	0.820±0.001	0.726±0.002
UK-VPN	US-VPN	0.391±0.003	0.376±0.008	0.827±0.003	0.835±0.002

In conclusion, while the flow method excels under consistent conditions, it struggles to maintain this success with changing conditions. In contrast, the IoTDevID method demonstrates greater resilience to network changes. Given the ultimate goal of device type identification—enabling a model trained for a specific device to identify that device in different networks—these results underscore the superior efficacy of features obtained from packet headers over flow-based features.

A.2 Deep Learning for IoT Device Identification and Attack Detection

This section expands the work of Chapters 3 and 7. In this section, we present some of the results of our preliminary experiments on the use of deep learning for device identification and attack detection.

A.2.1 CNN-based IoT Device Identification [114]

Table A.4 presents three distinct result sets concerning the identification of the Aalto dataset. In the initial step, Convolutional Neural Network (CNN)$_{with\ Payload}$, only the raw bytes of individual packet payloads, without any feature extraction, were transformed into 784 - 28X28 images and employed for device identification. In the subsequent step, CNN$_{with\ Fingerprint}$, the same dataset was converted into a 25-feature fingerprint

extracted from individual packets. These fingerprints are in the form of a 5X5 image and are used for device identification through CNN. Lastly, in the Decision Tree $(DT)_{with\ Fingerprint}$ step, device identification was again performed, this time using DT with the same 25-feature fingerprints.

Table A.4: The comparative analysis of device identification methods applied to the Aalto dataset using CNN and DT

Method	Accuracy	Time (seconds)
$CNN_{with\ Payload}$	0.631	209.926
$CNN_{with\ Fingerprint}$	0.625	20.809
$DT_{with\ Fingerprint}$	0.770	0.260

Upon analysing the results, it becomes evident that the outcomes of the two distinct CNN approaches closely resemble each other. However, the method utilising fingerprints demonstrates significantly improved speed compared to the method relying on raw payload bytes. It is also essential to highlight the limitation of purely payload-based approaches, which are ineffective when dealing with encrypted data. On the other hand, the approach involving fingerprinting and the DT algorithm proves to be markedly superior to both CNN-based approaches.

A.2.2 LSTM-based IoT Device Identification [115]

In this section, we compare various Long short-term memory (LSTM) methods (Vanilla LSTM, Stacked LSTM, CNN-LSTM, Encoder-Decoder LSTM) and Decision Trees (DT) using 25 unique fingerprints extracted from individual packets. The results of this comparison are presented in Table A.5. Notably, the results indicate that even the LSTM method with the highest accuracy (ED-LSTM - 0.621) lags far behind DT (0.77) in terms of performance. Additionally, it's worth highlighting the significant speed advantage of DT, which outpaces all other methods by a substantial margin.

Table A.5: The comparative analysis of device identification methods applied to the Aalto dataset using LSTM and DT

Method	Accuracy	Time (seconds)
CNN-LSTM	0.590	71
ED-LSTM	0.621	61
Stacked-LSTM	0.490	19
Vanilla-LSTM	0.516	24
DT	0.770	0.26

These results emphasise the superior performance and efficiency of the Decision Trees method in the context of device identification.

A.2.3 CNN-based IoT Attack Detection

In this application, we created graphs that observe the change of feature data over time and classified them via CNN. Although attack detection using CNNs is not a new approach, we have taken a different approach from previous studies. Preliminary studies [91, 134] focused on differentiating these graphs by converting attack and healthy separated data into graphs. In this case, it is tried to distinguish between images consisting only of harmful data and images consisting only of normal. In our approach, after all packets are plotted on the timeline, this graph is converted into packet-sized pieces. CNN is fed with graphical fragments representing packet, not with whole graphs.

For this application, we chose MitM - ARP Spoofing, which is the most difficult attack to detect in the dataset. When we use the real-time (packet time in milliseconds), there were gaps on the X-axis. Thus, in the chart creation step, we used the packets' actual sequence numbers instead of real-time.

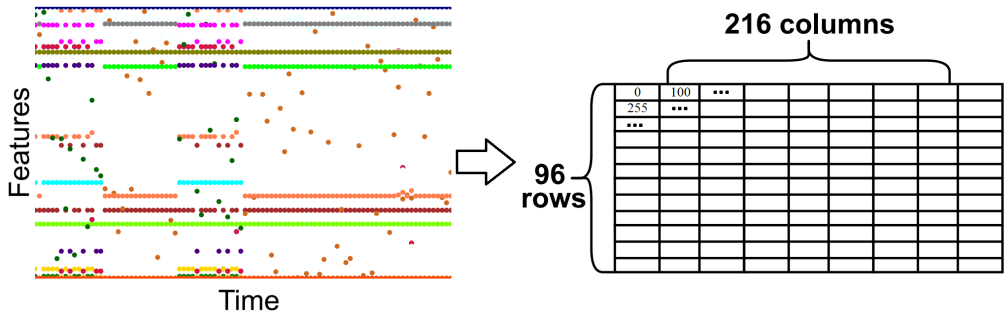


Figure A.1: Creating a dataset from a time series graph. The representation of the distribution of packet features on the graph and the conversion of this to a matrix.

As can be seen from the graph in Figure A.1, the X-axis shows the time, that is, 96 consecutive packets. The Y-axis shows the features (each of the 32 features has been used with a different colour). Parts of the graph, such as the axes and margin, have been removed.

Later, we converted this graph to numerical data. As a result of this process, we obtained a tensor of (72,96,3). Then we convert this tensor to a 216,96 size matrix and get the transposition. As a result of processing a graph, we obtained a dataset of 216 columns, 96 rows (see Figure A.1. The results we have obtained using this dataset and its comparison other methods are given in Table A.6 below.

Table A.6: Comparison of time series graphs' results

Dataset	ML	Acc	Prec	Rec	F1	Kap	AUC
Kitsune - MitM	CNN	0.680	0.690	0.680	0.670	0.350	0.780
	ETC	0.698	0.698	0.698	0.698	0.395	0.785
	DT	0.705	0.705	0.705	0.705	0.413	0.705
	RF	0.700	0.700	0.700	0.700	0.397	0.794

A.2.4 BERT-based Attack Detection

We have applied the transfer learning method, which is a prominent method especially in NLP (Natural language processing) and the analysis of sequential data, to attack detection. For this, we used the NSL-KDD dataset, which is one of the classical attack detection datasets, and the pre-trained BERT [56] (Bidirectional Encoder Representations from Transformers) model.

By converting the training and testing datasets to string values, we used the training dataset for the fine-tuning step. We repeated this process for binary and multiple classification. In order to compare the results, we made the same classification to the dataset with a classical machine learning method, the decision tree. The binary classification results of both works are given in Table A.7 and Figure A.2. The multi class classification results are given in Table A.8 and Figure A.3. The code is available at attack detection using BERT[1].

When these results are examined, it is seen that the performances of both studies are quite similar. However, the computational cost of even the fine-tuning phase of the BERT model is quite high. On the contrary, a transfer learning model exclusively trained on network data may potentially yield superior results.

A.2.5 Conclusion

While the findings of this study suggest the superiority of traditional Machine Learning (ML) approaches over deep learning, it is essential to recognise the expansive and rapidly evolving landscape of deep learning architectures and potential encodings/representations. This study should not be perceived as exhaustive or conclusive, as the realm of deep learning continues to evolve with ongoing advancements and novel methodologies. The versatility and adaptability of deep learning models, coupled with their potential for innovation and refinement, emphasise the importance of continued explo-

[1] https://github.com/kahramankostas/NSL-KDD-dataset-attack-detection-using-BERT

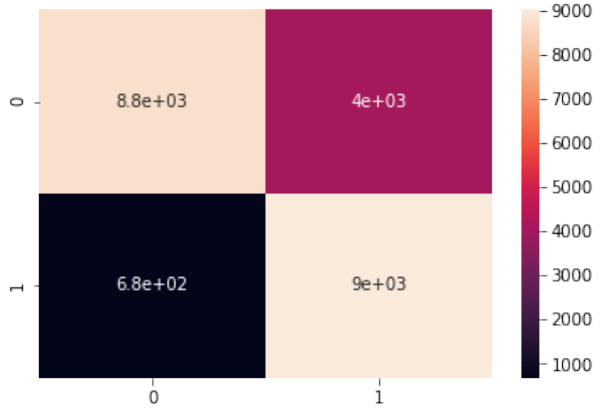
Table A.7: Results of BERT and DT binary classification models.

	DT -Binary Classification			
	Precision	Recall	F1-score	Support
0	0.68	0.96	0.8	9710
1	0.96	0.66	0.78	12833
accuracy		0.79		
macro	0.82	0.81	0.79	22543
weighted	0.84	0.79	0.79	22543

	BERT -Binary Classification			
	Precision	Recall	F1-score	Support
0	0.93	0.68	0.79	12833
1	0.69	0.93	0.79	9710
accuracy		0.79		
macro	0.81	0.81	0.79	22543
weighted	0.83	0.79	0.79	22543

Figure A.2: Confusion matrix of BERT and DT binary classification models.

(a) BERT -Binary Classification



(b) DT -Binary Classification

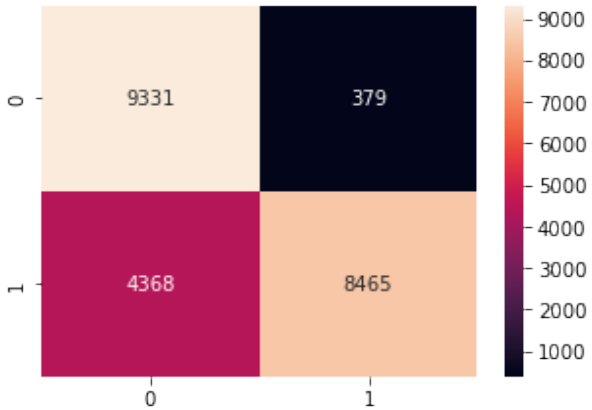


Table A.8: Results of BERT and DT multiclass classification models.

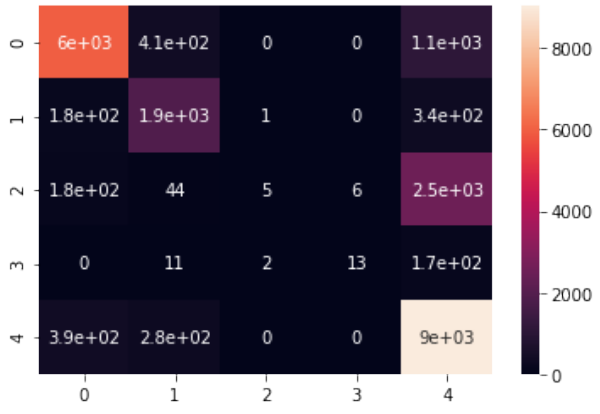
	DT -Multiclass Classification			
	Precision	Recall	F1-score	Support
0	0.96	0.82	0.89	7458
1	0.83	0.64	0.72	2421
2	0.98	0.08	0.16	2754
3	0.04	0.07	0.05	200
4	0.69	0.97	0.81	9710
accuracy		0.77		
macro	0.7	0.52	0.52	22543
weighted	0.82	0.77	0.74	22543

	BERT - Multiclass Classification			
	Precision	Recall	F1-score	Support
0	0.89	0.80	0.84	7458
1	0.72	0.78	0.75	2421
2	0.63	0.00	0.00	2754
3	0.68	0.07	0.12	200
4	0.69	0.93	0.79	9710
accuracy		0.75		
macro	0.72	0.52	0.50	22543
weighted	0.75	0.75	0.70	22543

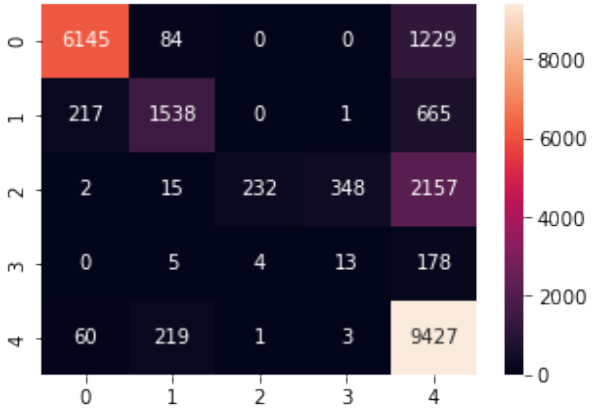
ration and investigation into this dynamic field. Future research may uncover variations and improvements within deep learning that could potentially challenge the conclusions drawn in this study, underscoring the necessity for ongoing assessment and comparison of traditional ML and deep learning approaches.

Figure A.3: Confusion matrix of BERT and DT multiclass classification models.

(a) BERT - Multiclass Classification



(b) DT - Multiclass Classification



A.3 IoTDevID

This section expands the work of Chapter 4.

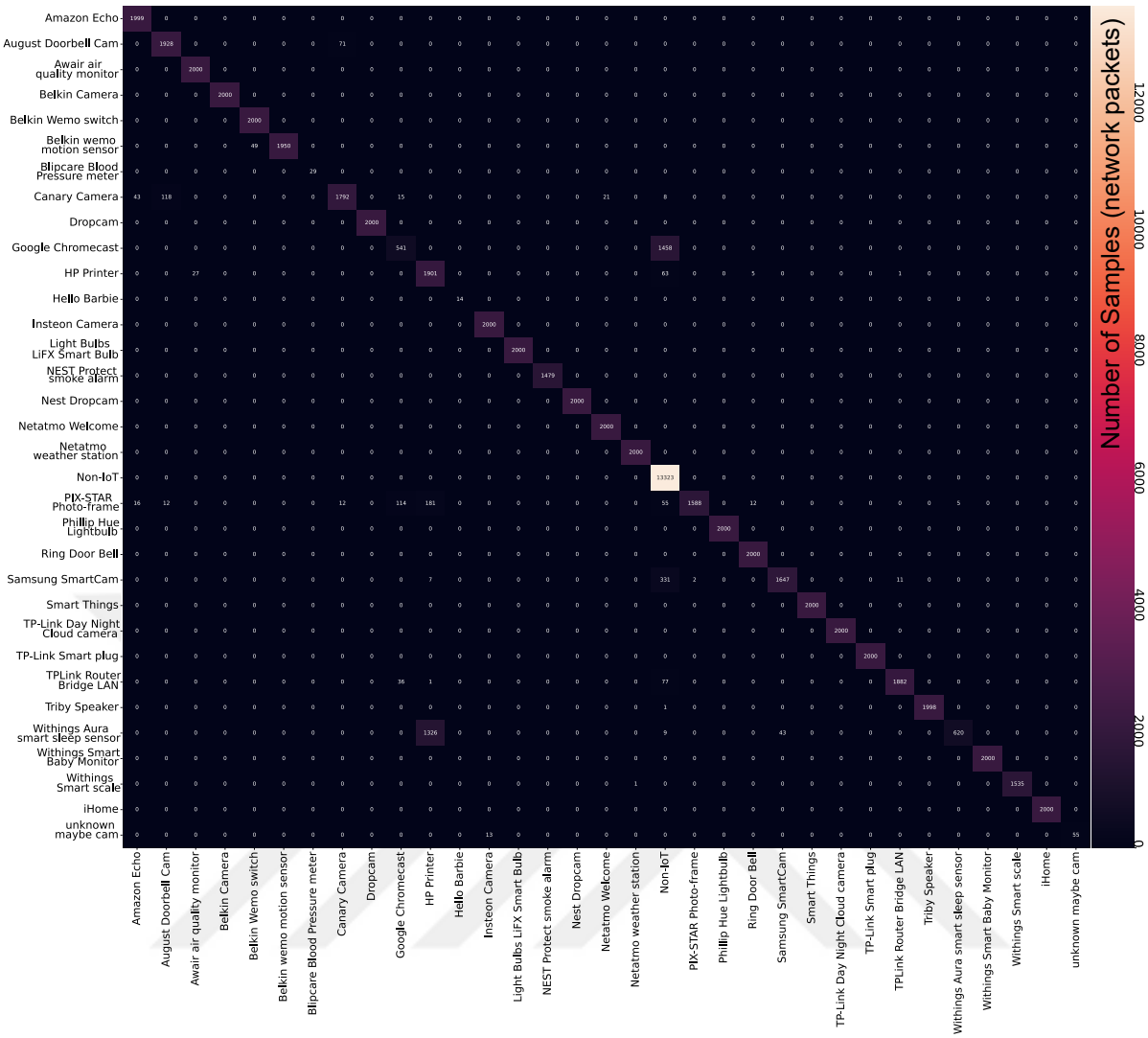


Figure A.4: Confusion matrix for UNSW Merged (UNSWM) dataset aggregated method, showing means of 100 results using DT.

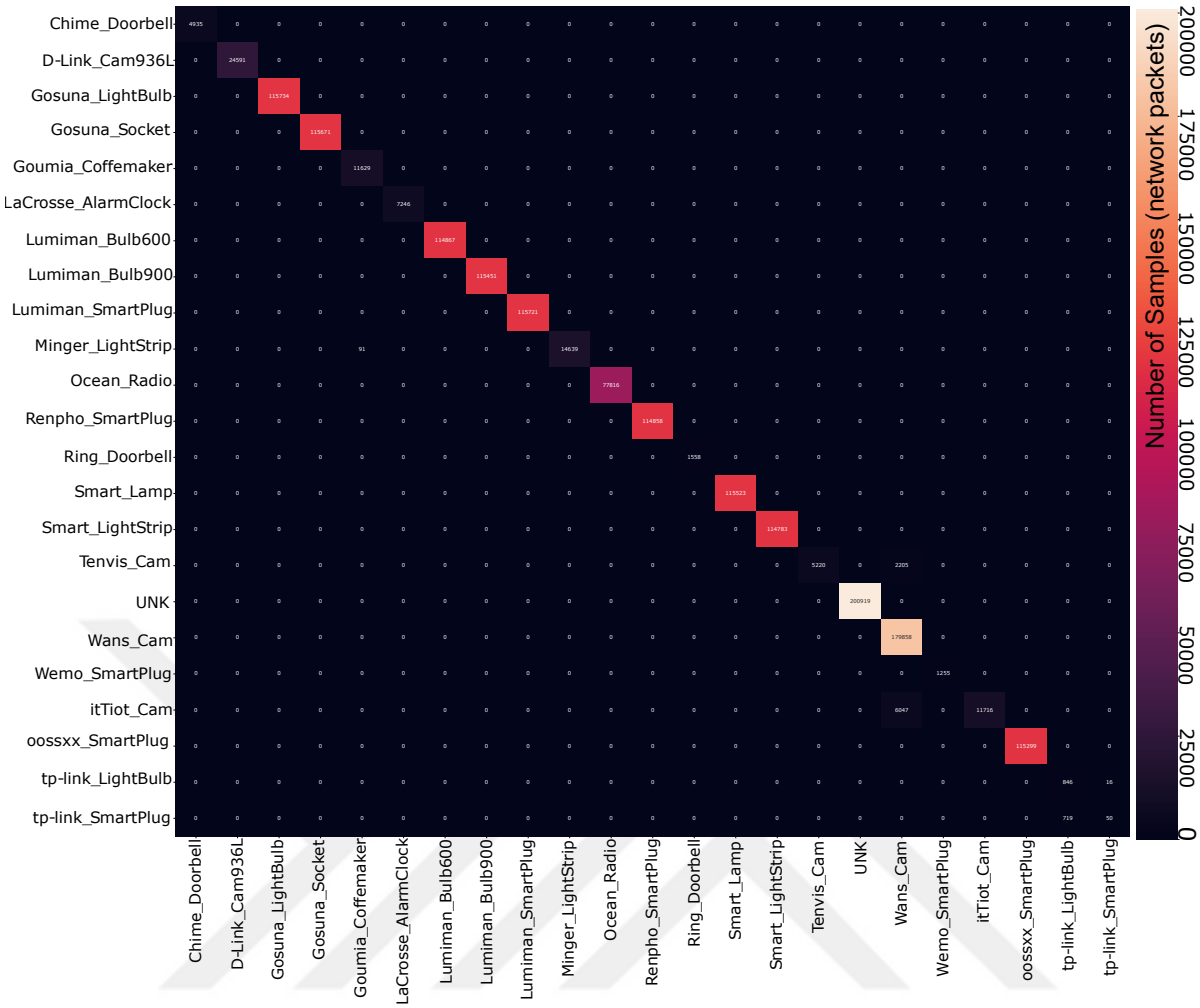


Figure A.5: Confusion matrix for Locality-Sensitive IoT Fingerprinting (LSIF) dataset aggregated method, showing means of 100 results using DT.

Table A.9: Feature list and descriptions (1/4)

No	Feature	Description	Feature Type		
			Ind.	RW	EW
1	BOOTP_chaddr	BOOTP client's hardware address	•		
2	BOOTP_ciaddr	BOOTP client's IP address	•		
3	BOOTP_file	BOOTP Boot Filename	•		
4	BOOTP_flags	BOOTP Flags	•		
5	BOOTP_giaddr	BOOTP gateway IP address	•		
6	BOOTP_hlen	BOOTP Hardware Length	•		
7	BOOTP_hops	BOOTP Hardware Options	•		
8	BOOTP_htype	BOOTP Hardware Len	•		
9	BOOTP_op	Bootstrap Protocol (BOOTP) Options	•		
10	BOOTP_options	BOOTP Options	•		
11	BOOTP_secs	BOOTP Seconds	•		
12	BOOTP_siaddr	BOOTP server IP address	•		
13	BOOTP_sname	BOOTP Server Name	•		
14	BOOTP_xid	BOOTP Transaction Identifier	•		
15	BOOTP_yiaddr	BOOTP your IP address	•		
16	DHCP_options	Dynamic Host Configuration Protocol (DHCP) Options	•		
17	DNS_aa	DNS Authoritative Answer	•		
18	DNS_ad	DNS Authentic Data	•		
19	DNS_an	DNS Answer section	•		
20	DNS_ancount	DNS Answer Count	•		
21	DNS_ar	DNS additional information section	•		
22	DNS_arcount	DNS Additional Information Count	•		
23	DNS_cd	DNS Checking Disabled	•		
24	DNS_id	DNS Identifier	•		
25	DNS_length	Domain Name System (DNS) Length	•		
26	DNS_ns	DNS authoritary section	•		
27	DNS_nscount	DNS Authority Count	•		
28	DNS_opcode	DNS Operation Code	•		
29	DNS_qd	DNS question section	•		
30	DNS_qdcount	DNS The unsigned fields query count	•		
31	DNS_qr	DNS Query-Response	•		
32	DNS_ra	DNS Recursion Available	•		
33	DNS_rcode	DNS Response Code	•		
34	DNS_rd	DNS Recursion Desired	•		
35	DNS_tc	DNS TrunCation	•		
36	DNS_z	DNS Reserved for future use	•		
37	dport_class	Destination Port Class (IoTDevID classing)	•		
38	dport23	Destination Port Class (keep wellknown ports between 0-1023)	•		
39	dport	Destination Port Number	•		
40	EAPOL_len	Extensible Authentication Protocol (EAPOL) Length	•		
41	EAPOL_type	Extensible Authentication Protocol (EAPOL) type	•		
42	EAPOL_version	Extensible Authentication Protocol (EAPOL) version	•		
43	Ether_dst	Destination Media Access Control (MAC) Address	•		
44	Ether_src	Source MAC Address	•		
45	Ether_type	Ethernet Type	•		
46	ICMP_addr_mask	ICMP Address Mask	•		
47	ICMP_chksum	ICMP Checksum	•		
48	ICMP_code	ICMP Code	•		
49	ICMP_gw	ICMP gateway	•		
50	ICMP_id	ICMP identifier	•		
51	ICMP_length	ICMP length	•		

Table A.9: Feature list and descriptions (2/4)

No	Feature	Description	Feature Type		
			Ind.	RW	EW
52	ICMP_nexthopmtu	ICMP Next Hop Maximum Transmission Unit (MTU)	•		
53	ICMP_ptr	ICMP ConditionalField	•		
54	ICMP_reserved	ICMP ConditionalField	•		
55	ICMP_seq	ICMP Sequence Number	•		
56	ICMP_ts_ori	ICMP ConditionalField	•		
57	ICMP_ts_rx	ICMP ConditionalField	•		
58	ICMP_ts_tx	ICMP ConditionalField	•		
59	ICMP_type	Internet Control Message Protocol (ICMP) Type	•		
60	ICMP_unused	ICMP ConditionalField	•		
61	IP_chksum	IP Checksum	•		
62	IP_DF	IP Don't Fragment	•		
63	IP_dst	Destination IP Address	•		
64	IP_flags	IP Flags	•		
65	IP_frag	IP fragmentation	•		
66	IP_id	IP identifier	•		
67	IP_ihl	IP Internet Header Length	•		
68	IP_len	IP Length	•		
69	IP_MF	IP More Fragments	•		
70	IP_options	IP Options	•		
71	IP_proto	IP Protocols	•		
72	IP_src	Source Internet Protocol (IP) Address	•		
73	IP_tos	IP type of service	•		
74	IP_ttl	IP Time To Live	•		
75	IP_version	IP version	•		
76	IP_Z	IP Zero	•		
77	LLC_ctrl	Logical Link Control - Control	•		
78	LLC_dsap	Logical Link Control - Destination Service Access Point	•		
79	LLC_ssap	Logical Link Control - Source Service Access Point	•		
80	MAC_dst	Destination MAC Address	•		
81	MAC_src	Source MAC Address	•		
82	entropy	Payload Entropy	•		
83	payload_bytes	Payload size in Bytes	•		
84	pck_size	Packet (Frame) Size	•		
85	Protocol	WireShark Protocol	•		
86	sport_class	Source Port Class (IoTDevID classing)	•		
87	sport23	Source Port Class (keep wellknown ports between 0-1023)	•		
88	sport_bare	Source Port Number	•		
89	TCP_ACK	Acknowledgment Flag	•		
90	TCP_ack	TCP Acknowledgment Number	•		
91	TCP_chksum	TCP Checksum	•		
92	TCP_CWR	CWR Flag	•		
93	TCP_dataofs	TCP data ofset	•		
94	TCP_dport	TCP Destination Ports	•		
95	TCP_ECE	ECE Flag	•		
96	TCP_FIN	FINished Flag	•		
97	TCP_flags	TCP Flags	•		
98	TCP_options	TCP Options	•		
99	TCP_PSH	Push Flag	•		
100	TCP_reserved	TCP Reserved	•		
101	TCP_RST	Reset Flag	•		
102	TCP_seq	TCP Sequence Number	•		

Table A.9: Feature list and descriptions (3/4)

No	Feature	Description	Feature Type		
			Ind.	RW	EW
103	TCP_sport	TCP Source Ports	•		
104	TCP_SYN	Sync Flag	•		
105	TCP_URG	Urgent Flag	•		
106	TCP_urgptr	TCP Urgent Pointer	•		
107	TCP_window	TCP Window Size	•		
108	ts	Time Stamp	•		
109	UDP_chksum	UDP Checksum	•		
110	UDP_dport	UDP Destination Ports	•		
111	UDP_len	User datagram protocol (UDP) Length	•		
112	UDP_sport	UDP Source Ports	•		
113	WS_dst	WireShark Destination Address	•		
114	WS_src	WireShark Source Address	•		
115	ID	WS_src=¿WS_dst	•		
116	pck_size_mean_6	RW (Rolling windows) packet size mean - Window size =6		•	
117	pck_size_std_6	RW packet size Std. - Window size =6		•	
118	ts_mean_6	RW packet time mean - Window size =6		•	
119	ts_std_6	RW packet time Std.- Window size =6		•	
120	TCP_window_mean_6	RW TCP Windows size mean - Window size =6		•	
121	TCP_window_std_6	RW TCP Windows size Std. - Window size =6		•	
122	payload_bytes_mean_6	RW Payload bytes mean - Window size =6		•	
123	payload_bytes_std_6	RW Payload bytes Std. - Window size =6		•	
124	entropy_mean_6	RW Entropy mean - Window size =6		•	
125	entropy_std_6	RW Entropy Std. - Window size =6		•	
126	pck_size_mean_9	RW (Rolling windows) packet size mean - Window size =9		•	
127	pck_size_std_9	RW packet size Std. - Window size =9		•	
128	ts_mean_9	RW packet time mean - Window size =9		•	
129	ts_std_9	RW packet time Std.- Window size =9		•	
130	TCP_window_mean_9	RW TCP Windows size mean - Window size =9		•	
131	TCP_window_std_9	RW TCP Windows size Std. - Window size =9		•	
132	payload_bytes_mean_9	RW Payload bytes mean - Window size =9		•	
133	payload_bytes_std_9	RW Payload bytes Std. - Window size =9		•	
134	entropy_mean_9	RW Entropy mean - Window size =9		•	
135	entropy_std_9	RW Entropy Std. - Window size =9		•	
136	pck_size_mean_2	RW (Rolling windows) packet size mean - Window size =2		•	
137	pck_size_std_2	RW packet size Std. - Window size =2		•	
138	ts_mean_2	RW packet time mean - Window size =2		•	
139	ts_std_2	RW packet time Std.- Window size =2		•	
140	TCP_window_mean_2	RW TCP Windows size mean - Window size =2		•	
141	TCP_window_std_2	RW TCP Windows size Std. - Window size =2		•	
142	payload_bytes_mean_2	RW Payload bytes mean - Window size =2		•	
143	payload_bytes_std_2	RW Payload bytes Std. - Window size =2		•	
144	entropy_mean_2	RW Entropy mean - Window size =2		•	
145	entropy_std_2	RW Entropy Std. - Window size =2		•	
146	dst_IP_diversity	Number of Destination IP (by Source IP)			•
147	dst_port_diversity	Number of Destination Port (by Source IP)			•
148	src_IP_diversity	Number of Source IP (by destination IP)			•
149	IP_add_count	Number of Source IP (by destination MAC)			•
150	pck_size_diff	the size difference of consecutive packets			•
151	pck_size_mean_WE	EW (expanding windows) packet size mean			•
152	pck_size_std_WE	EW packet size Std.			•
153	pck_size_sum_of_EW	EW packet size total.			•

Table A.9: Feature list and descriptions (4/4)

No	Feature	Description	Feature Type		
			Ind.	RW	EW
154	ts_diff	the time difference of consecutive packets			•
155	ts_mean_WE	EW packet time mean			•
156	ts_std_WE	EW packet time Std.			•
157	ts_sum_of_EW	EW packet time total.			•
158	TCP_window_diff	TCP Windows size difference of consecutive packets			•
159	TCP_window_mean_WE	EW TCP Windows size mean			•
160	TCP_window_std_WE	EW TCP Windows size Std.			•
161	TCP_window_sum_of_EW	EW TCP Windows size total.			•
162	payload_bytes_diff	Payload bytes difference of consecutive packets			•
163	payload_bytes_mean_WE	EW Payload bytes mean			•
164	payload_bytes_std_WE	EW Payload bytes Std.			•
165	payload_bytes_sum_of_EW	EW Payload bytes total.			•
166	entropy_diff	Entropy difference of consecutive packets			•
167	entropy_mean_WE	EW Entropy mean			•
168	entropy_std_WE	EW Entropy Std.			•
169	entropy_sum_of_EW	EW Entropy total.			•
170	dport_sum	EW Unique destination port number			•
171	sport_sum	EW Unique source port number			•
172	TCP_FIN_sum	EW FINished Flag			•
173	TCP_SYN_sum	EW Sync Flag			•
174	TCP_RST_sum	EW Reset Flag			•
175	TCP_PSH_sum	EW Push Flag			•
176	TCP_ACK_sum	EW Acknowledgment Flag			•
177	TCP_URG_sum	EW Urgent Flag			•
178	TCP_ECE_sum	EW ECE Flag			•
179	TCP_CWR_sum	EW CWR Flag			•
180	TCP_FIN_ratio	TCP_FIN/TCP_FIN_sum			•
181	TCP_SYN_ratio	TCP_SYN/TCP_SYN_sum			•
182	TCP_RST_ratio	TCP_RST/TCP_RST_sum			•
183	TCP_PSH_ratio	TCP_PSH/TCP_PSH_sum			•
184	TCP_ACK_ratio	TCP_ACK/TCP_ACK_sum			•
185	TCP_URG_ratio	TCP_URG/TCP_URG_sum			•
186	TCP_ECE_ratio	TCP_ECE/TCP_ECE_sum			•
187	TCP_CWR_ratio	TCP_CWR/TCP_CWR_sum			•
188	sum	EW all TCP Flags			•
189	TCP_FIN_SR	TCP_FIN/sum			•
190	TCP_SYN_SR	TCP_SYN/sum			•
191	TCP_RST_SR	TCP_RST/sum			•
192	TCP_PSH_SR	TCP_PSH/sum			•
193	TCP_ACK_SR	TCP_ACK/sum			•
194	TCP_URG_SR	TCP_URG/sum			•
195	TCP_ECE_SR	TCP_ECE/sum			•
196	TCP_CWR_SR	TCP_CWR/sum			•
197	TCP_FIN_R	TCP_FIN_sum/sum			•
198	TCP_SYN_R	TCP_SYN_sum/sum			•
199	TCP_RST_R	TCP_RST_sum/sum			•
200	TCP_PSH_R	TCP_PSH_sum/sum			•
201	TCP_ACK_R	TCP_ACK_sum/sum			•
202	TCP_URG_R	TCP_URG_sum/sum			•
203	TCP_ECE_R	TCP_ECE_sum/sum			•
204	TCP_CWR_R	TCP_CWR_sum/sum			•
205	Label	Packet Level Label	•		

A.4 MonIoTr

This section expands the work of Chapter 6.

Table A.10: UK and UK-VPN Class based results

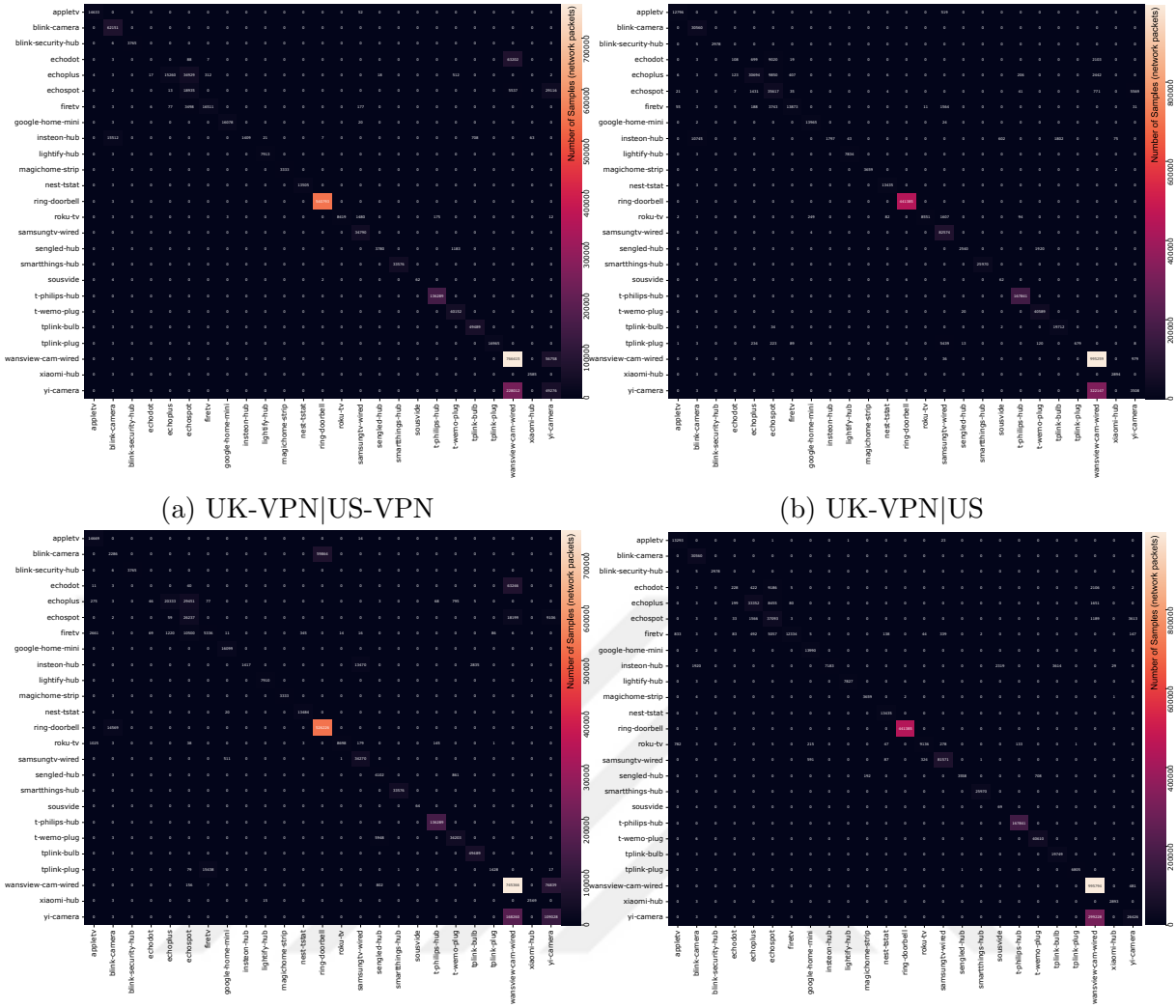
	UK-VPN	F1 Score		UK
	Packet No	UK|UK-VPN	UK-VPN|UK	Packet No
appletv	13775	0.53707	0.65026	17977
blink-camera	92434	0.22037	0.98661	83855
blink-security-hub	3863	0.9584	0.94654	3352
bosiwo-camera-wired	359580	0.92606	0.93555	698552
charger-camera	563	0.96865	0.94762	965
echodot	23483	0.61755	0.52887	31471
echoplus	35799	0.57573	0.50444	27440
echospot	83536	0.80621	0.81241	65150
firetv	25935	0.28347	0.2847	10487
google-home	25854	0.4853	0.44027	19382
google-home-mini	26735	0.49837	0.5766	19209
honeywell-thermostat	2779	0.93565	0.85604	2503
insteon-hub	33	0.99619	0.94471	33
lightify-hub	5245	0.99302	0.98554	3274
magichome-strip	1521	0.97829	0.9723	1138
nest-tstat	26854	0.84935	0.94552	25651
netatmo-weather-station	603	0.9072	0.89778	340
ring-doorbell	723017	0.94528	0.99904	820444
roku-tv	8900	0.65466	0.7227	5876
samsungtv-wired	37328	0.64545	0.66069	27148
sengled-hub	1642	0.42313	0.45643	3370
smarter-coffee-mach	1665	0.98292	0.96316	1356
smartthings-hub	16856	0.91746	0.8741	21271
sousvide	14271	0.99883	0.96453	15842
t-philips-hub	29364	0.96915	0.93048	71515
t-wemo-plug	40069	0.83404	0.92939	32546
tplink-bulb	6715	0.98985	0.95777	14580
tplink-plug	5588	0.26013	0.41428	5965
wansview-cam-wired	907544	0.88054	0.86273	802384
xiaomi-cam2	346227	0.90391	0.90329	456611
xiaomi-cleaner	243	0.31913	0.4496	295
xiaomi-hub	9298	0.99475	0.99468	6394
yi-camera	281210	0.52953	0.46447	261276
Average		0.75457	0.78291	

Table A.11: US and US-VPN Class based results

	US-VPN	F1 Score		US
	Packet No	US|US-VPN	US-VPN|US	Packet No
amcrest-cam-wired	252423	0.6625	0.98477	504375
appletv	14685	0.66749	0.69685	13319
blink-camera	62151	0.55498	0.94316	30560
blink-security-hub	3771	0.82702	0.90696	2983
brewer	651	0.94188	0.98463	650
bulb1	7122	0.99697	0.99673	6897
cloudcam	181797	0.9893	0.99242	144224
dlink-mov	411	0.52226	0.53657	394
dryer	8348	0.20735	0.37811	9064
echodot	63302	0.1767	0.32127	11955
echoplus	51061	0.49017	0.55245	43742
echospot	53605	0.66537	0.65639	43504
firetv	20274	0.23725	0.37191	19479
fridge	79603	0.41789	0.78725	93653
google-home-mini	16102	0.57349	0.75142	13992
ikettle	4328	0.98853	0.99833	4522
insteon-hub	17724	0.99327	1	15067
invoke	7929	0.50799	0.76577	7108
lefun-cam-wired	314145	0.78156	0.85653	378072
lgtv-wired	24676	0.72152	0.73509	56608
lightify-hub	7916	0.99021	0.99495	7837
luohe-spycam	115184	0.49209	0.44269	245945
magichome-strip	3336	0.98441	0.99463	3667
microseven-camera	45412	0.99002	0.99816	146908
microwave	557	0.17432	0.96392	554
nest-tstat	13508	0.75692	0.7531	13438
philips-bulb	3862	0.61011	0.60674	3860
ring-doorbell	540796	0.96563	0.99653	441388
roku-tv	10095	0.57607	0.60521	10606
samsungtv-wired	34790	0.57516	0.81346	82579
sengled-hub	4967	0.91637	0.84941	4464
smartthings-hub	33576	0.95993	0.90448	25970
sousvide	69	0.85	0.86154	73
t-echodot	19151	0.01248	0.00024	7186
t-philips-hub	136289	0.99116	0.99681	167841
t-wemo-plug	40155	0.69068	0.70367	40616
tplink-bulb	49492	0.7524	0.99513	19753
tplink-plug	16968	0.42938	0.50808	6812
wansview-cam-wired	823173	0.88447	0.86247	996276
washer	8362	0.35533	0.34865	9395
wink-hub2	15751	0.788	0.73296	12558
xiaomi-hub	2588	0.67018	0.73709	2897
xiaomi-ricecooker	279	0.29511	0.29188	242
xiaomi-strip	5932	0.71527	0.75554	5543
yi-camera	277292	0.54765	0.48151	325660
zmodo-doorbell	157752	0.92194	0.97799	164545
Average		0.66997	0.74768	

Table A.12: Class-based results of the evaluation using data from two different labs

Device	Train→ / Test→	UK US	UK US-VPN	US UK	US UK-VPN	UK-VPN US	UK-VPN US-VPN	US-VPN UK	US-VPN UK-VPN
appletv		0.693	0.557	0.627	0.623	0.695	0.802	0.540	0.676
blink-camera		0.834	0.409	0.964	0.415	0.803	0.880	0.980	0.967
blink-security-hub		0.948	0.954	0.885	0.871	0.934	0.959	0.889	0.942
echodot		0.133	0.041	0.569	0.096	0.111	0.062	0.475	0.429
echoplus		0.519	0.423	0.470	0.457	0.494	0.424	0.480	0.520
echospot		0.470	0.372	0.762	0.743	0.433	0.395	0.754	0.768
firetv		0.414	0.219	0.293	0.192	0.439	0.515	0.245	0.573
google-home-mini		0.705	0.739	0.893	0.853	0.737	0.853	0.861	0.884
insteon-hub		0.300	0.124	0.789	0.969	0.203	0.017	0.872	0.998
lightify-hub		0.881	0.903	0.838	0.735	0.912	0.913	0.878	0.813
magichome-strip		0.874	0.912	0.635	0.924	0.856	0.813	0.893	0.949
nest-tstat		0.802	0.705	0.929	0.808	0.805	0.859	0.805	0.928
ring-doorbell		0.993	0.957	0.996	0.951	0.993	0.997	0.998	0.996
roku-tv		0.471	0.567	0.544	0.406	0.525	0.593	0.405	0.570
samsungtv-wired		0.801	0.647	0.584	0.432	0.841	0.781	0.512	0.485
sengled-hub		0.636	0.255	0.493	0.367	0.515	0.443	0.263	0.360
smartthings-hub		0.939	0.954	0.873	0.854	0.809	0.890	0.899	0.894
sousvide		0.356	0.383	0.946	0.925	0.506	0.547	0.968	0.988
t-philips-hub		0.992	0.974	0.957	0.911	0.959	0.966	0.954	0.931
t-wemo-plug		0.814	0.629	0.958	0.919	0.776	0.784	0.915	0.980
tplink-bulb		0.897	0.942	0.971	0.966	0.818	0.949	0.981	0.968
tplink-plug		0.841	0.365	0.911	0.352	0.299	0.898	0.323	0.835
wansview-cam-wired		0.821	0.676	0.817	0.788	0.804	0.702	0.737	0.692
xiaomi-hub		0.863	0.801	0.827	0.977	0.745	0.669	0.990	0.993
yi-camera		0.356	0.369	0.475	0.439	0.258	0.320	0.466	0.437
Accuracy		0.788	0.688	0.835	0.758	0.765	0.727	0.796	0.760
F1 Score		0.694	0.595	0.760	0.679	0.651	0.681	0.723	0.783



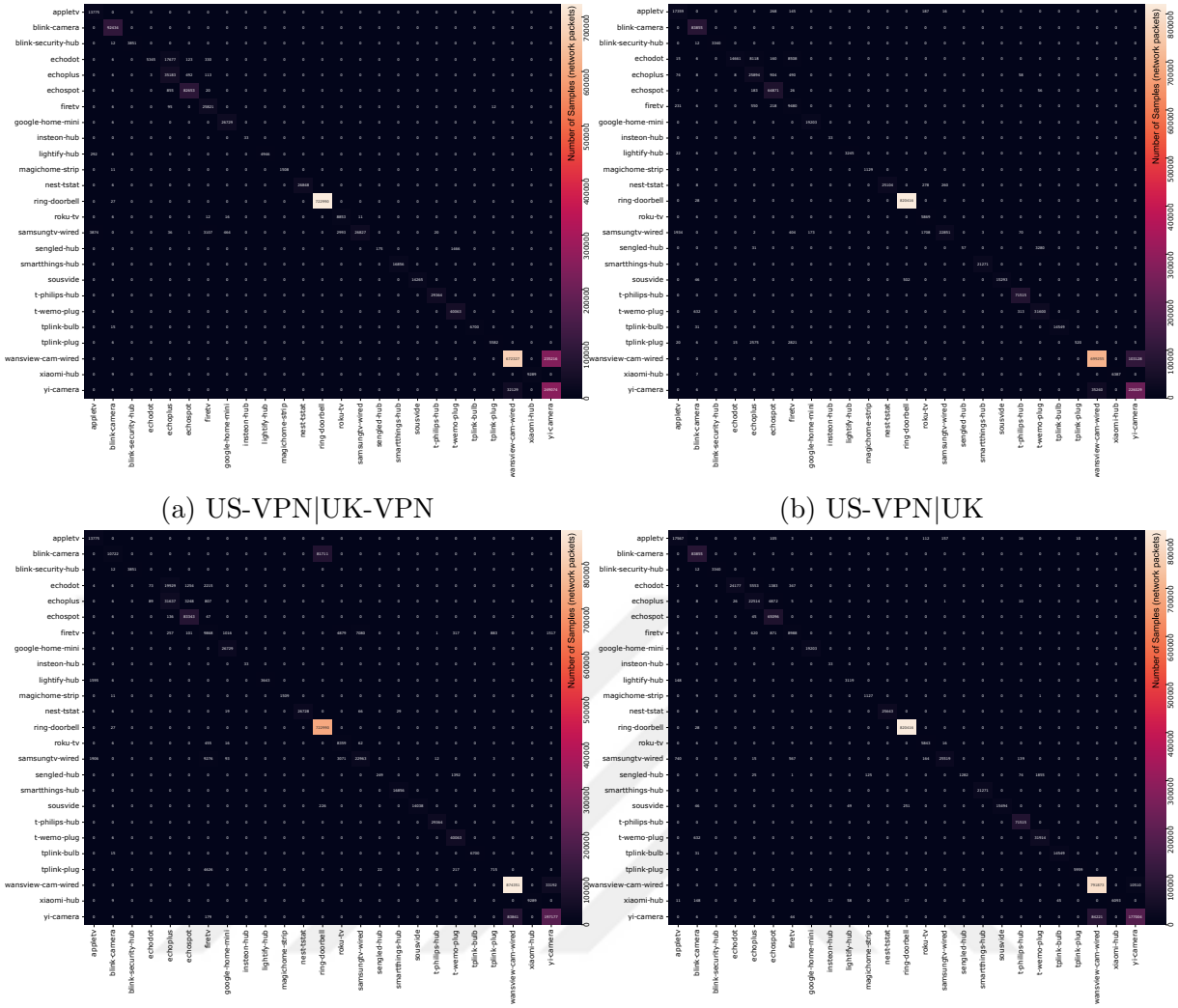
(a) UK-VPN|US-VPN

(b) UK-VPN|US

(c) UK|US-VPN

(d) UK|US

Figure A.7: Confusion matrices depict scenarios where distinct laboratories were employed in the MonIoTr dataset. UK data is utilized for training, while US data is utilized for testing.



(a) US-VPN|UK-VPN

(b) US-VPN|UK

(c) US|UK-VPN

(d) US|UK

Figure A.8: Confusion matrices depict scenarios where distinct laboratories were employed in the MonIoTr dataset. US data is utilized for training, while UK data is utilized for testing.

A.4.1 Comparison of active and idle states

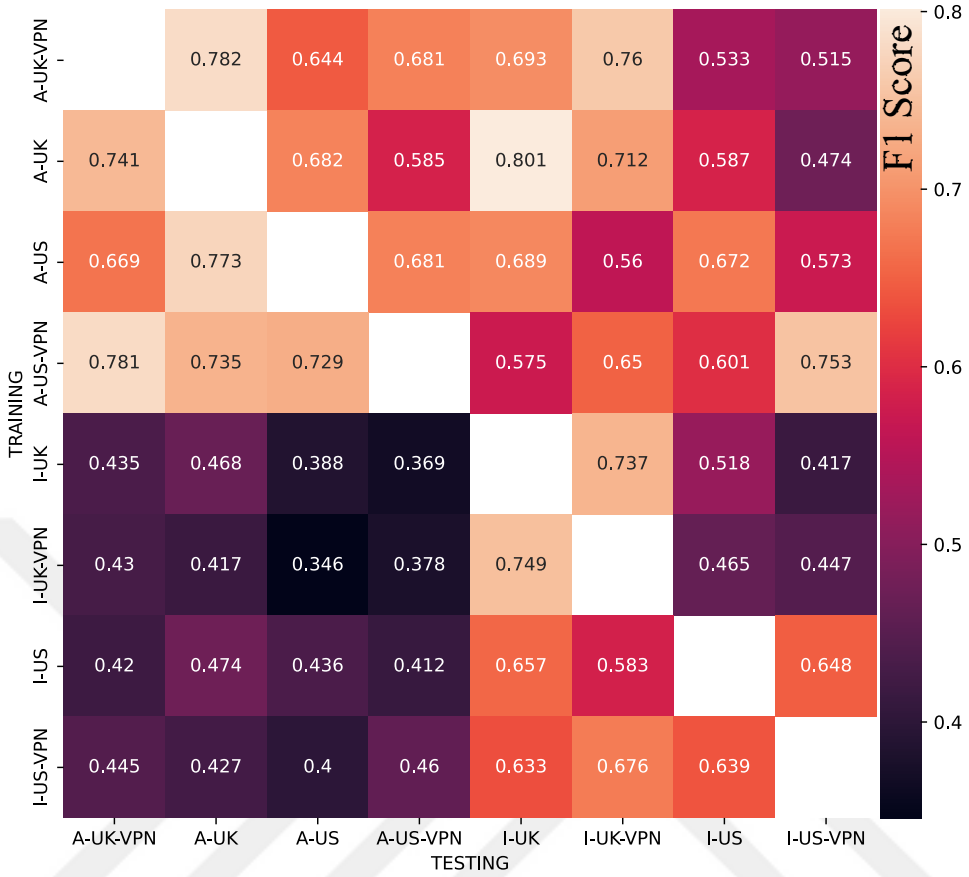
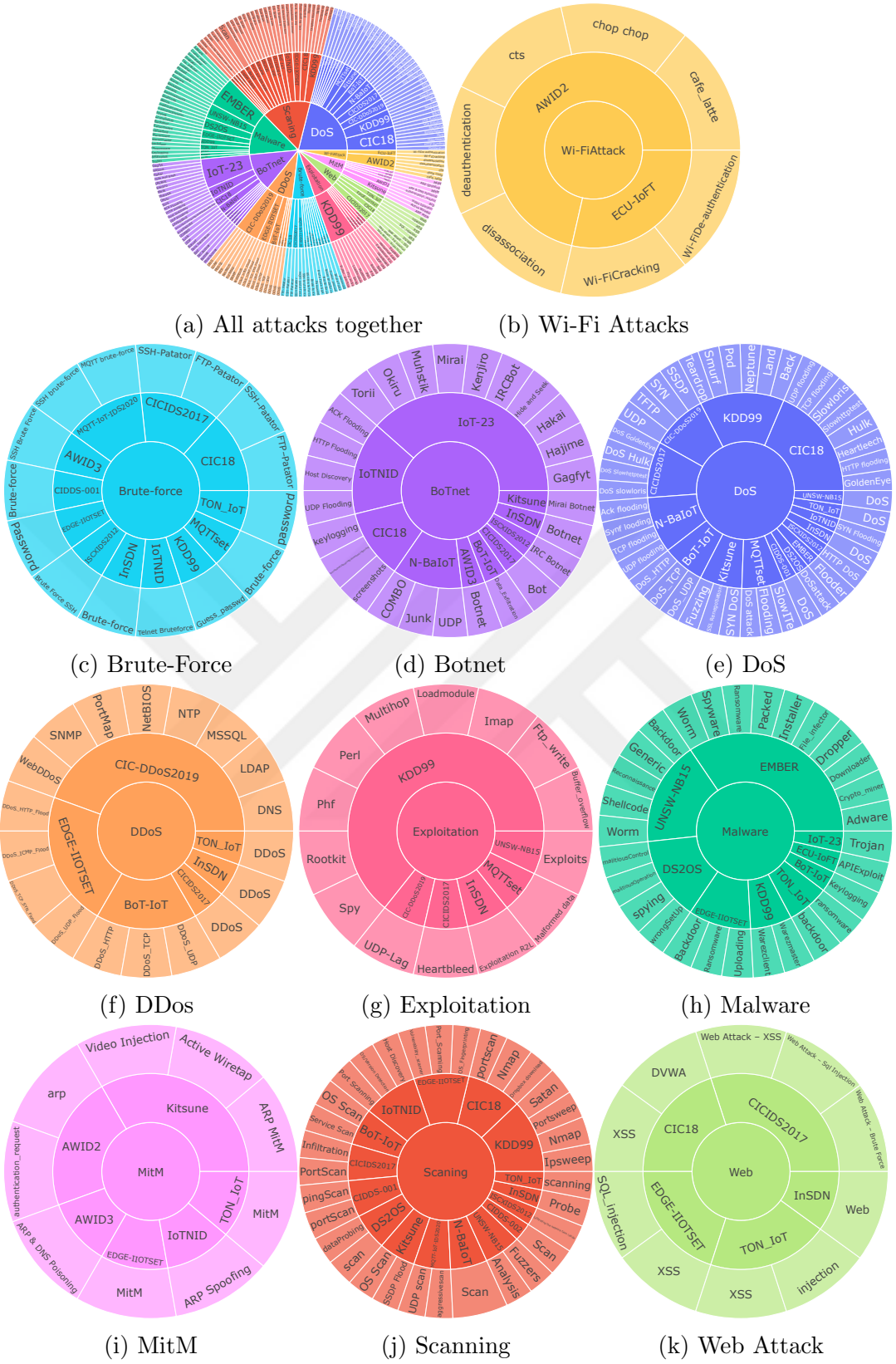


Figure A.9: Using eight different states as training and test data, the results of all possible situations in terms of F1 score in the MonIoTr dataset.

In Figure A.9, we present the 56 different results obtained by using all the cases in Table 6.1 as training and test data. Analysing these results, the cases where Idle data is used as training data and Active data is used as test data are noticeably less successful than the other distributions. This comparison confirms our data quality experiments comparing the Idle/Active cases with the CIC IoT Dataset 2022 (CICIoT22) data.

A.5 All Attacks Graphs

This section expands the work of Chapter 7. See Figure A.10



(a) All attacks together (b) Wi-Fi Attacks

(c) Brute-Force (d) Botnet (e) DoS

(f) DDos (g) Exploitation (h) Malware

(i) MitM (j) Scanning (k) Web Attack

Figure A.10: Distribution of attacks in the literature by dataset and attack groups.

A.6 Machine Learning Hyperparameters: Model Configuration Details

In this section, a comprehensive overview of the machine learning hyperparameters employed throughout the thesis is presented, offering detailed insights into the configurations that contributed to the model development and evaluation processes.

A.6.1 Genetic Algorithm Hyperparameters

These hyperparameters collectively define the configuration of the genetic algorithm, influencing how the algorithm explores the solution space and evolves a population of feature subsets over multiple generations to optimize the specified fitness function (F1 score in this case).

Population Size (size): It defines the number of individuals (chromosomes) in each generation of the genetic algorithm. In this case, the population consists of 200 individuals.

Number of Features (n_feat): It represents the total number of features in the dataset (the number of features varies in device identification and attack detection tasks). The genetic algorithm aims to find the subset of features that maximizes the fitness score.

Number of Parents Selected for Reproduction (n_parents): During the selection phase of the genetic algorithm, a certain number of individuals with higher fitness scores (parents) are chosen to create the next generation. Here, 120 parents are selected.

Mutation Rate (mutation_rate): It determines the probability that a bit (feature) in the chromosome will be flipped (mutated). A lower mutation rate, such as 0.005, indicates a relatively low probability of mutation.

Number of Generations (n_gen): It specifies how many generations of individuals will be created and evaluated. In this case, the genetic algorithm runs for 100 generations.

A.6.2 Device Identification Hyperparameters

Aalto Dataset

```
"NB":CategoricalNB(alpha=1e-09)

"DT" :DecisionTreeClassifier(criterion='gini', max_depth=26.0,
                             max_features=26,min_samples_split= 6)

"RF":RandomForestClassifier(bootstrap= True, criterion= "gini",
max_depth= 18.0, max_features= 8, min_samples_split= 9, n_estimators= 96)

"KNN":KNeighborsClassifier(algorithm= 'brute', leaf_size= 41,
                           n_neighbors= 48, weights= 'distance')

"MLP":MLPClassifier(solver= 'adam', learning_rate= 'constant',
hidden_layer_sizes= (1220, 1965), alpha= 0.1, activation= 'relu')

"SVM":SVC(C=10,gamma=1)

"GB":GradientBoostingClassifier(learning_rate=0.001,
                                subsample=0.1,n_estimators=500,max_depth= 10)
```

UNSW Dataset

```
"DT" :DecisionTreeClassifier(criterion='entropy', max_depth=27.0,
                        max_features=19,min_samples_split= 8)
```

LSIF Dataset

```
"DT" :DecisionTreeClassifier(criterion='entropy', max_depth=19,
                        max_features=4,min_samples_split= 2)
```

Figure A.11: Hyperparameters utilized for the three datasets (Aalto University - IoT Devices Captures (Aalto), UNSWM, LSIF) in Chapter 4.

CIC Dataset

```
"DT" :DecisionTreeClassifier( criterion= "entropy", max_depth= 31,
                             max_features= 18, min_samples_split= 3)
```

MonIoTr Dataset

```
'UKVPN|UK': DecisionTreeClassifier(max_depth=25, max_features=27,
                                   min_samples_split=4),
'UKVPN|USVPN': DecisionTreeClassifier(criterion='entropy', max_depth=23,␣
 ↪max_features=29,
                                      min_samples_split=6),
'UKVPN|US': DecisionTreeClassifier(criterion='entropy', max_depth=32,
                                   max_features=26),
'UK|USVPN': DecisionTreeClassifier(max_depth=31, max_features=28,
                                   min_samples_split=4),
'UK|US': DecisionTreeClassifier(criterion='entropy', max_depth=24,␣
 ↪max_features=16,
                                 min_samples_split=3),
'USVPN|US' DecisionTreeClassifier(criterion='entropy', max_depth=25,␣
 ↪max_features=29,
                                   min_samples_split=4)}
```

Figure A.12: Hyperparameters utilized for the two datasets (CICIoT22 and MonIoTr) in Chapter 5 & 6.

A.6.3 Attack Detection Hyperparameters

Table A.13: Hyperparameter configurations for DT across two different methods (FLOW and WINDOW) applied to various attacks.

	criterion	max_depth	max_features	min_samples_split	Attack
FLOW	gini	5	35	6	SYN
	entropy	16	2	2	HTTP
	gini	5	2	8	ACK
	entropy	3	11	2	UDP
	entropy	28	27	2	ARP
	gini	4	16	9	SP
	gini	19	10	3	BF
WINDOW	gini	6	10	2	SYN
	gini	6	10	3	HTTP
	entropy	3	4	4	ACK
	gini	5	20	6	UDP
	gini	17	51	2	ARP
	entropy	6	6	2	SP
	entropy	6	21	3	BF
	entropy	1	53	4	OS
	entropy	4	26	9	SCHD
	gini	7	12	9	MHDis

Table A.14: Hyperparameter configurations for Random Forest (RF) across two different methods (FLOW and WINDOW) applied to various attacks.

	bootst	criter	max_depth	max_features	min_samp_split	n_estimators	Attack
FLOW	FALSE	gini	6	5	9	73	SYN
	FALSE	entropy	14	9	2	7	HTTP
	FALSE	entropy	4	9	7	183	ACK
	FALSE	gini	6	5	10	7	UDP
	FALSE	entropy	25	1	3	48	ARP
	FALSE	entropy	6	5	6	29	SP
	TRUE	entropy	18	2	2	125	BF
WINDOW	FALSE	gini	7	8	2	98	SYN
	TRUE	gini	7	9	2	73	HTTP
	TRUE	entropy	8	7	8	55	ACK
	FALSE	entropy	14	3	3	35	UDP
	FALSE	gini	17	9	2	140	ARP
	FALSE	gini	8	7	2	1	SP
	FALSE	gini	5	9	9	138	BF
	FALSE	entropy	5	2	6	3	OS
	FALSE	gini	13	9	7	73	SCHD
	TRUE	gini	11	3	5	75	MHDis

Table A.15: Hyperparameter configurations for XGBoost, eXtreme Gradient Boosting (XGB) across two different methods (FLOW and WINDOW) applied to various attacks.

	n_estimators	min_child_weight	max_depth	learning_rate	Attack
FLOW	100	1	2	0.15	SYN
	1500	1	2	0.2	HTTP
	1100	1	3	0.1	ACK
	500	4	2	0.05	UDP
	1100	3	3	0.2	ARP
	1500	1	2	0.1	SP
	900	2	10	0.15	BF
WINDOW	1500	4	2	0.2	SYN
	900	3	2	0.15	HTTP
	1100	1	2	0.15	ACK
	900	2	2	0.15	UDP
	1100	3	2	0.2	ARP
	1100	1	3	0.2	SP
	1100	4	2	0.05	BF
	100	3	2	0.15	OS
	100	2	2	0.1	SCHD
	1100	2	2	0.15	MHDis

Table A.16: Hyperparameter configurations for k-Nearest Neighbours (KNN) across two different methods (FLOW and WINDOW) applied to various attacks.

	algorithm	leaf_size	n_neighbors	weights	Attack
FLOW	auto	42	6	distance	SYN
	brute	7	4	distance	HTTP
	auto	11	4	distance	ACK
	auto	23	4	distance	UDP
	ball_tree	5	16	distance	ARP
	kd_tree	33	3	distance	SP
	kd_tree	38	2	distance	BF
WINDOW	kd_tree	27	11	distance	SYN
	ball_tree	40	2	distance	HTTP
	auto	13	1	distance	ACK
	ball_tree	45	9	uniform	UDP
	kd_tree	10	2	distance	ARP
	auto	33	1	distance	SP
	auto	10	9	distance	BF
	brute	7	10	distance	OS
	kd_tree	21	1	distance	SCHD
	ball_tree	46	1	uniform	MHDis

Table A.17: Hyperparameter configurations for Naïve Bayes (NB) across two different methods (FLOW and WINDOW) applied to various attacks.

	var_smoothing	Attack
FLOW	2.85E-06	SYN
	2.85E-09	HTTP
	1.00E-09	ACK
	1.00E-09	UDP
	1.00E-09	ARP
	2.31E-09	SP
	0.1	BF
WINDOW	1.23E-09	SYN
	1.00E-09	HTTP
	2.85E-07	ACK
	1.23E-09	UDP
	1.23E-08	ARP
	1.00E-09	SP
	0.053367	BF
	1.00E-09	OS
	0.00432876	SCHD
	1.00E-09	MHDis

Table A.18: Hyperparameter configurations for Logistic (Logit) Regression (LR) across two different methods (FLOW and WINDOW) applied to various attacks.

	C	penalty	Solver	Attack
FLOW	3.12528	none	lbfgs	SYN
	1.39267	l1	liblinear	HTTP
	0.000175	none	lbfgs	ACK
	0.000508	none	lbfgs	UDP
	7.21E-05	l2	newton-cg	ARP
	50.8013	l1	liblinear	SP
	5.71E-05	l2	liblinear	BF
WINDOW	59.8441	l1	liblinear	SYN
	0.046384	l2	lbfgs	HTTP
	0.204882	l2	lbfgs	ACK
	74.8807	none	lbfgs	UDP
	74.5065	l1	liblinear	ARP
	2.45721	none	lbfgs	SP
	27.4817	l1	liblinear	BF
	29.4052	l1	liblinear	OS
	0.820152	none	lbfgs	SCHD
	0.000453	none	lbfgs	MHDis

Table A.19: Hyperparameter configurations for MultiLayer Perceptron (MLP) across two different methods (FLOW and WINDOW) applied to various attacks.

	solver	learning_rate	hidden_layer_sizes	alpha	activation	Attack
FLOW	adam	constant	(901,)	0.1	relu	SYN
	adam	constant	(408,)	0.001	tanh	HTTP
	adam	adaptive	(95,)	0.01	relu	ACK
	lbfgs	constant	(1043,)	0.0001	tanh	UDP
	sgd	adaptive	(1637,)	0.0001	tanh	ARP
	adam	constant	(1533,)	0.05	relu	SP
	adam	adaptive	(892, 1585)	0.001	tanh	BF
WINDOW	adam	adaptive	(1184, 433)	0.05	relu	SYN
	adam	adaptive	(439,)	0.0001	relu	HTTP
	lbfgs	adaptive	(1193, 630)	0.05	tanh	ACK
	adam	adaptive	(1391, 1912)	0.0001	relu	UDP
	adam	constant	(1015,)	0.1	relu	ARP
	adam	constant	(851, 318)	0.0001	relu	SP
	adam	adaptive	(1129, 732)	0.001	relu	BF
	lbfgs	constant	(851, 318)	0.01	tanh	OS
	lbfgs	adaptive	(310, 1663)	0.1	tanh	SCHD
	adam	constant	(1568, 1928)	0.01	tanh	MHDis

Table A.20: Hyperparameter configurations for Support Vector Machine (SVM) across two different methods (FLOW and WINDOW) applied to various attacks.

	gamma	C	Attack
FLOW	0.001	1	SYN
	0.1	1	HTTP
	0.1	10	ACK
	1	10	UDP
	0.01	10	ARP
	0.01	10	SP
	1	10	BF
WINDOW	1	10	SYN
	1	1	HTTP
	1	10	ACK
	0.1	10	UDP
	0.01	1	ARP
	1	1	SP
	0.01	10	BF
	0.01	10	OS
	1	10	SCHD
	0.1	1	MHDis

A.7 IoTGeM

This section expands the work of Chapter 9.

A.7.1 Problems with the IoTID20 Dataset

We performed the feature extraction process by applying CICFlowMeter to all the pcap files we used. Although a dataset (IoTID20) has already been produced using CICFlowMeter from IoT-NID raw data, we preferred to redo the feature extraction process instead of using it. We used the CICFlowMeter[2] tool to extract features from the dataset and labeled them in parallel with the defined rules in IoT-NID dataset [104] using our flow-labeller script[3]. We believe that there may have been issues with the

[2] https://github.com/ahlashkari/CICFlowMeter

[3] https://github.com/kahramankostas/IoTGeM/blob/main/0001-Feature-Extraction-PCAP2CSV/000-FLOW-LABELLER.ipynb

feature extraction process of the IoTID20 dataset. Some clues leading to this conclusion are as follows. Although the raw data contains 10 different attacks, there are 8 attacks in the dataset. It is conceivable that some pcap files are labelled as if they contain a single attack, ignoring the fact that they contain more than one type of attack.

When examining the content generator directives in the IoT-NID dataset [104], distinct pcap files are apparent, each corresponding to a specific attack and session. However, these files encompass not only malicious packets but also benign ones within each attack file. This becomes most apparent when considering that the benign file contains 137,000 packets, whereas the entire dataset comprises over 15,000,000 benign packets. Notably, this dataset lacks pre-defined labels; solely raw data and filtering rules for packet-level labelling are provided. Employing the rules from content creators facilitates the separation of benign and malicious segments within all files. Yet, in the context of IoTID20, it appears that these rules were disregarded, resulting in the mislabelling of all pcap files as attacks. Substantiating this assertion, we observe the following: a flow comprises either a single packet or a combination of several. In this regard, the number of flows derived from a pcap file must consistently be less than or equal to the number of packets. However, a scrutiny of IoTID20 reveals numerous instances where this principle is violated, with the count of attacks often surpassing the expected count (refer to Table A.21). For many attacks such as HTTP, BF, OSD, PS, MHD, and SHD the number of samples provided in IoTID20 is much higher than the number of samples in the raw data.

Table A.21: The comparison of the number of packets in the IoT-NID raw data with the number of flows in the IoTDI20 dataset

Type	Label	Packets	Flows
Normal	Normal	1,756,276	40,073
Mirai Botnet	Host Discovery (MHD)	673	22,192
Scanning	Host Discovery (SHD)	2,454	
Scanning	Port Scanning (PS)	20,939	53,073
Scanning	OS/Version Detection (OSD)	1,817	
Man in the Middle (MITM)	ARP Spoofing (ARPS)	101,885	35,377
Denial of Service (DoS)	SYN Flooding (SYNF)	64,646	59,391
Mirai Botnet	Telnet Bruteforce (BFF)	1,924	121,181
Mirai Botnet	UDP Flooding (UDPF)	949,284	183,554
Mirai Botnet	ACK Flooding (ACKF)	75,632	55,124
Mirai Botnet	HTTP Flooding(HTTPF)	10,464	55,818

A.7.2 IoTGeM extended ML Results

Table A.22: Window approach CV results. T is time

Window approach CV results.

Attack	ML	Accuracy	BalancedAcc	Precision	Recall	F1-Score	kappa	ROC	Train-T	Test-T
ACK	DT	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	0.106	0.000
	KNN	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	0.999±0.000	1.000±0.000	0.130	0.128
	LR	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	0.437	0.000
	MLP	1.000±0.000	1.000±0.001	1.000±0.000	1.000±0.001	1.000±0.001	1.000±0.001	1.000±0.001	16.252	0.132
	NB	1.000±0.000	0.999±0.001	1.000±0.000	0.999±0.001	1.000±0.000	0.999±0.001	0.999±0.001	0.031	0.006
	RF	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	0.999±0.000	1.000±0.000	5.534	0.103
	SVM	0.948±0.005	0.631±0.011	0.974±0.002	0.631±0.011	0.694±0.014	0.398±0.027	0.631±0.011	16.365	1.069
	XGB	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	14.246	0.064
ARP	DT	0.995±0.001	0.994±0.002	0.993±0.002	0.994±0.002	0.994±0.002	0.987±0.004	0.994±0.002	0.440	0.003
	KNN	0.944±0.002	0.928±0.003	0.922±0.002	0.928±0.003	0.925±0.002	0.850±0.005	0.928±0.003	0.202	0.116
	LR	0.869±0.008	0.784±0.015	0.846±0.011	0.784±0.015	0.808±0.014	0.618±0.027	0.784±0.015	39.338	0.001
	MLP	0.760±0.059	0.651±0.063	0.758±0.103	0.651±0.063	0.642±0.053	0.315±0.089	0.651±0.063	16.574	0.015
	NB	0.759±0.007	0.529±0.004	0.705±0.050	0.529±0.004	0.494±0.009	0.082±0.012	0.529±0.004	0.027	0.013
	RF	0.998±0.001	0.997±0.001	0.997±0.001	0.997±0.001	0.997±0.001	0.994±0.002	0.997±0.001	11.375	0.091
	SVM	0.755±0.008	0.509±0.002	0.877±0.004	0.509±0.002	0.447±0.004	0.027±0.005	0.509±0.002	68.318	4.711
	XGB	0.998±0.001	0.997±0.001	0.997±0.001	0.997±0.001	0.997±0.001	0.994±0.002	0.997±0.001	14.762	0.042
BF	DT	1.000±0.000	0.999±0.002	0.999±0.002	0.999±0.002	0.999±0.001	0.999±0.002	0.999±0.002	0.159	0.003
	KNN	0.999±0.000	0.982±0.008	0.976±0.010	0.982±0.008	0.979±0.007	0.957±0.013	0.982±0.008	0.004	2.133
	LR	1.000±0.000	0.999±0.002	0.992±0.006	0.999±0.002	0.996±0.003	0.991±0.006	0.999±0.002	2.328	0.001
	MLP	0.989±0.005	0.929±0.060	0.951±0.022	0.929±0.060	0.938±0.039	0.875±0.078	0.929±0.060	53.638	0.033
	NB	0.972±0.001	0.659±0.031	0.577±0.019	0.659±0.031	0.602±0.023	0.205±0.046	0.659±0.031	0.025	0.008
	RF	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	8.303	0.108
	SVM	0.995±0.001	0.951±0.014	0.998±0.001	0.951±0.014	0.973±0.008	0.946±0.016	0.951±0.014	2.791	0.520
	XGB	1.000±0.000	0.996±0.004	1.000±0.000	0.996±0.004	0.998±0.002	0.995±0.004	0.996±0.004	9.904	0.039
HTTP	DT	1.000±0.000	0.999±0.001	1.000±0.000	0.999±0.001	1.000±0.000	0.999±0.001	0.999±0.001	0.227	0.001
	KNN	1.000±0.000	0.998±0.001	0.998±0.001	0.998±0.001	0.998±0.001	0.996±0.001	0.998±0.001	0.257	0.550
	LR	0.999±0.001	0.997±0.002	0.999±0.008	0.997±0.002	0.994±0.005	0.988±0.009	0.997±0.002	1.618	0.002
	MLP	0.998±0.003	0.990±0.012	0.982±0.030	0.999±0.012	0.985±0.016	0.971±0.032	0.999±0.012	4.185	0.006
	NB	0.849±0.003	0.922±0.001	0.606±0.003	0.922±0.001	0.632±0.004	0.303±0.007	0.922±0.001	0.043	0.014
	RF	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	9.788	0.129
	SVM	0.960±0.002	0.500±0.000	0.480±0.001	0.500±0.000	0.490±0.000	0.000±0.000	0.500±0.000	38.264	2.876
	XGB	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	14.758	0.052
MHD	DT	1.000±0.000	0.998±0.007	0.999±0.014	0.998±0.007	0.994±0.007	0.987±0.015	0.998±0.007	0.130	0.000
	KNN	1.000±0.000	0.999±0.004	0.965±0.019	0.999±0.004	0.981±0.011	0.963±0.021	0.999±0.004	0.207	1.006
	LR	1.000±0.000	1.000±0.000	0.969±0.019	1.000±0.000	0.984±0.010	0.967±0.021	1.000±0.000	1.259	0.000
	MLP	1.000±0.001	0.996±0.013	0.995±0.008	0.996±0.013	0.995±0.008	0.990±0.015	0.996±0.013	321.473	0.157
	NB	0.972±0.001	0.986±0.000	0.537±0.006	0.986±0.000	0.562±0.010	0.134±0.020	0.986±0.000	0.044	0.017
	RF	1.000±0.000	1.000±0.000	0.992±0.009	1.000±0.000	0.996±0.004	0.992±0.009	1.000±0.000	2.198	0.107
	SVM	0.981±0.002	0.533±0.018	0.990±0.001	0.533±0.018	0.557±0.032	0.121±0.062	0.533±0.018	5.476	0.788
	XGB	1.000±0.000	1.000±0.000	0.993±0.012	1.000±0.000	0.996±0.006	0.992±0.013	1.000±0.000	21.24	0.063
OS	DT	1.000±0.000	0.999±0.002	0.998±0.002	0.999±0.002	0.999±0.001	0.997±0.002	0.999±0.002	0.321	0.003
	KNN	0.998±0.000	0.952±0.015	0.937±0.015	0.952±0.015	0.944±0.010	0.888±0.021	0.952±0.015	0.013	5.711
	LR	1.000±0.000	0.998±0.004	0.998±0.003	0.998±0.004	0.998±0.003	0.996±0.005	0.998±0.004	0.326	0.002
	MLP	0.943±0.006	0.500±0.000	0.471±0.003	0.500±0.000	0.485±0.001	0.000±0.001	0.500±0.000	151.776	0.053
	NB	0.931±0.001	0.965±0.001	0.560±0.006	0.965±0.001	0.590±0.010	0.202±0.018	0.965±0.001	0.061	0.018
	RF	0.999±0.002	0.966±0.093	0.997±0.004	0.966±0.093	0.975±0.067	0.951±0.134	0.966±0.093	0.090	0.008
	SVM	0.943±0.005	0.500±0.000	0.471±0.003	0.500±0.000	0.485±0.001	0.000±0.000	0.500±0.000	10.279	1.134
	XGB	1.000±0.000	0.998±0.002	0.998±0.002	0.998±0.002	0.998±0.001	0.997±0.002	0.998±0.002	3.201	0.035
SHD	DT	1.000±0.000	0.999±0.003	0.998±0.003	0.999±0.003	0.999±0.002	0.997±0.004	0.999±0.003	0.040	0.000
	KNN	1.000±0.000	1.000±0.000	0.997±0.004	1.000±0.000	0.998±0.002	0.997±0.004	1.000±0.000	0.054	0.046
	LR	1.000±0.000	0.999±0.001	0.997±0.003	0.999±0.001	0.998±0.002	0.996±0.004	0.999±0.001	0.165	0.000
	MLP	1.000±0.000	1.000±0.000	0.999±0.002	1.000±0.000	0.999±0.001	0.999±0.002	1.000±0.000	129.221	0.053
	NB	0.960±0.004	0.500±0.000	0.480±0.002	0.500±0.000	0.490±0.001	0.000±0.000	0.500±0.000	0.006	0.001
	RF	1.000±0.000	1.000±0.000	0.999±0.002	1.000±0.000	0.999±0.001	0.999±0.002	1.000±0.000	0.790	0.027
	SVM	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	3.350	0.814
	XGB	1.000±0.000	1.000±0.000	0.998±0.004	1.000±0.000	0.999±0.002	0.998±0.004	1.000±0.000	0.735	0.008
PS	DT	1.000±0.000	0.999±0.001	1.000±0.000	0.999±0.001	1.000±0.001	0.999±0.001	0.999±0.001	0.008	0.000
	KNN	1.000±0.000	1.000±0.000	0.999±0.002	1.000±0.000	1.000±0.001	0.999±0.002	1.000±0.000	0.016	0.017
	LR	1.000±0.000	0.999±0.001	0.999±0.002	0.999±0.001	0.999±0.001	0.999±0.002	0.999±0.001	0.073	0.003
	MLP	1.000±0.000	1.000±0.001	1.000±0.000	1.000±0.001	1.000±0.000	1.000±0.001	1.000±0.001	7.311	0.016
	NB	0.999±0.001	0.994±0.001	0.953±0.006	0.994±0.001	0.972±0.004	0.945±0.008	0.994±0.001	0.003	0.002
	RF	1.000±0.001	0.999±0.003	0.999±0.002	0.999±0.003	0.999±0.002	0.998±0.003	0.999±0.003	0.013	0.000
	SVM	0.890±0.007	0.500±0.001	0.545±0.212	0.500±0.001	0.472±0.003	0.002±0.003	0.500±0.001	6.131	0.900
	XGB	1.000±0.000	0.999±0.001	1.000±0.001	0.999±0.001	0.999±0.001	0.999±0.001	0.999±0.001	2.479	0.012
SYN	DT	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	0.999±0.000	1.000±0.000	0.033	0.001
	KNN	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.998±0.001	0.999±0.000	0.180	3.301
	LR	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	0.999±0.001	1.000±0.000	0.079	0.001
	MLP	0.999±0.001	0.999±0.001	0.999±0.001	0.999±0.001	0.999±0.001	0.998±0.002	0.999±0.001	168.516	0.168
	NB	0.994±0.001	0.994±0.001	0.993±0.001	0.994±0.001	0.993±0.001	0.987±0.001	0.994±0.001	0.022	0.010
	RF	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	0.999±0.000	1.000±0.000	2.135	0.083
	SVM	0.946±0.004	0.941±0.005	0.954±0.004	0.941±0.005	0.945±0.004	0.890±0.009	0.941±0.005	27.904	2.189
	XGB	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	0.999±0.000	1.000±0.000	14.174	0.041
UDP	DT	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	0.278	0.006
	KNN	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	0.538	56.370
	LR	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	2.819	0.007
	MLP	0.999±0.001	0.999±0.000	0.999±0.001	0.999±0.000	0.999±0.001	0.999±0.001	0.999±0.000	236.567	0.176
	NB	0.674±0.004	0.730±0.002	0.775±0.002	0.730±0.002	0.669±0.004	0.403±0.005	0.730±0.002	0.091	0.028
	RF	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	3.728	0.127
	SVM	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	0.076	0.003
	XGB	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	38.689	0.121

Table A.23: Window approach Session results. T is time

Window approach Session results.

Attack	ML	Accuracy	BalancedAcc	Precision	Recall	F1-Score	kappa	ROC	Train-T	Test-T
ACK	DT	1.000±0.000	1.000±0.000	0.999±0.000	1.000±0.000	0.999±0.000	0.999±0.001	1.000±0.000	0.021	0.002
	KNN	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	0.999±0.000	1.000±0.000	0.032	0.190
	LR	1.000±0.000	1.000±0.000	0.999±0.000	1.000±0.000	1.000±0.000	0.999±0.000	1.000±0.000	0.098	0.003
	MLP	1.000±0.000	1.000±0.000	0.999±0.000	1.000±0.000	0.999±0.000	0.999±0.000	1.000±0.000	26.160	0.814
	NB	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	0.999±0.000	1.000±0.000	0.003	0.012
	RF	0.999±0.000	1.000±0.000	0.999±0.000	1.000±0.000	0.999±0.000	0.998±0.000	1.000±0.000	1.113	0.169
	SVM	0.306±0.000	0.501±0.000	0.653±0.000	0.501±0.000	0.235±0.000	0.001±0.000	0.501±0.000	29.912	6.743
	XGB	1.000±0.000	1.000±0.000	0.999±0.000	1.000±0.000	1.000±0.000	0.999±0.000	1.000±0.000	2.940	0.065
ARP	DT	0.653±0.024	0.583±0.034	0.597±0.036	0.583±0.034	0.583±0.037	0.175±0.069	0.583±0.034	0.551	0.005
	KNN	0.688±0.000	0.604±0.000	0.641±0.000	0.604±0.000	0.607±0.000	0.23±0.000	0.604±0.000	0.271	0.664
	LR	0.661±0.011	0.636±0.011	0.629±0.011	0.636±0.011	0.631±0.011	0.264±0.022	0.636±0.011	57.483	0.003
	MLP	0.655±0.064	0.571±0.051	0.615±0.068	0.571±0.051	0.563±0.006	0.157±0.108	0.571±0.051	29.770	0.125
	NB	0.662±0.000	0.502±0.000	0.545±0.000	0.502±0.000	0.411±0.000	0.006±0.000	0.502±0.000	0.024	0.039
	RF	0.731±0.019	0.655±0.023	0.704±0.026	0.655±0.023	0.664±0.026	0.340±0.049	0.655±0.023	13.298	0.331
	SVM	0.674±0.000	0.516±0.000	0.835±0.000	0.516±0.000	0.432±0.000	0.041±0.000	0.516±0.000	184.752	36.351
	XGB	0.785±0.000	0.718±0.000	0.779±0.000	0.718±0.000	0.733±0.000	0.475±0.000	0.718±0.000	15.000	0.106
BF	DT	0.831±0.006	0.876±0.002	0.514±0.000	0.876±0.002	0.481±0.002	0.045±0.002	0.876±0.002	0.041	0.016
	KNN	0.998±0.000	0.987±0.000	0.877±0.000	0.987±0.000	0.925±0.000	0.850±0.000	0.987±0.000	0.002	2.146
	LR	0.896±0.002	0.911±0.001	0.523±0.000	0.911±0.001	0.516±0.001	0.078±0.001	0.911±0.001	0.270	0.005
	MLP	0.994±0.001	0.969±0.016	0.968±0.006	0.969±0.016	0.969±0.008	0.937±0.016	0.969±0.016	137.646	0.283
	NB	0.980±0.000	0.747±0.000	0.566±0.000	0.747±0.000	0.601±0.000	0.206±0.000	0.747±0.000	0.009	0.086
	RF	0.999±0.000	0.963±0.002	0.984±0.001	0.963±0.002	0.974±0.001	0.947±0.002	0.963±0.002	2.456	1.030
	SVM	0.995±0.000	0.948±0.000	0.997±0.000	0.948±0.000	0.971±0.000	0.943±0.000	0.948±0.000	7.031	3.603
	XGB	0.997±0.000	0.960±0.000	0.851±0.000	0.960±0.000	0.898±0.000	0.796±0.000	0.960±0.000	2.895	0.159
HTTP	DT	0.999±0.002	0.988±0.014	0.999±0.001	0.988±0.014	0.993±0.008	0.987±0.015	0.988±0.014	0.035	0.003
	KNN	0.981±0.000	0.834±0.000	0.985±0.000	0.834±0.000	0.893±0.000	0.787±0.000	0.834±0.000	0.053	0.463
	LR	0.999±0.000	0.996±0.000	0.991±0.000	0.996±0.000	0.993±0.000	0.986±0.000	0.996±0.000	0.388	0.002
	MLP	0.987±0.006	0.901±0.061	0.974±0.026	0.901±0.061	0.930±0.038	0.860±0.076	0.901±0.061	5.933	0.035
	NB	0.562±0.000	0.768±0.000	0.556±0.000	0.768±0.000	0.450±0.000	0.114±0.000	0.768±0.000	0.012	0.008
	RF	0.999±0.000	0.992±0.000	1.000±0.000	0.992±0.000	0.996±0.000	0.992±0.000	0.992±0.000	1.955	0.078
	SVM	0.944±0.000	0.500±0.000	0.472±0.000	0.500±0.000	0.486±0.000	0.000±0.000	0.500±0.000	77.398	14.51
	XGB	0.999±0.000	0.992±0.000	1.000±0.000	0.992±0.000	0.996±0.000	0.992±0.000	0.992±0.000	3.645	0.029
MHD	DT	0.970±0.032	0.794±0.221	0.975±0.029	0.794±0.221	0.808±0.225	0.626±0.436	0.794±0.221	0.017	0.000
	KNN	0.998±0.000	0.999±0.000	0.986±0.000	0.999±0.000	0.992±0.000	0.984±0.000	0.999±0.000	0.020	0.355
	LR	0.998±0.000	0.999±0.000	0.989±0.000	0.999±0.000	0.994±0.000	0.988±0.000	0.999±0.000	0.141	0.003
	MLP	0.998±0.000	0.999±0.000	0.988±0.000	0.999±0.000	0.994±0.000	0.987±0.001	0.999±0.000	311.831	0.520
	NB	0.917±0.000	0.955±0.000	0.734±0.000	0.955±0.000	0.795±0.000	0.597±0.000	0.955±0.000	0.006	0.005
	RF	0.995±0.000	0.964±0.001	0.996±0.000	0.964±0.001	0.980±0.001	0.959±0.001	0.964±0.001	0.365	0.043
	SVM	0.934±0.000	0.555±0.000	0.888±0.000	0.555±0.000	0.581±0.000	0.182±0.000	0.555±0.000	14.811	3.674
	XGB	0.959±0.000	0.719±0.000	0.977±0.000	0.719±0.000	0.793±0.000	0.59±0.000	0.719±0.000	3.092	0.024
OS	DT	0.997±0.000	0.988±0.000	0.824±0.000	0.988±0.000	0.890±0.000	0.779±0.000	0.988±0.000	0.06	0.007
	KNN	0.863±0.000	0.840±0.000	0.515±0.000	0.840±0.000	0.493±0.000	0.050±0.000	0.840±0.000	0.002	1.207
	LR	0.997±0.000	0.996±0.004	0.815±0.007	0.996±0.004	0.885±0.006	0.770±0.011	0.996±0.004	0.050	0.003
	MLP	0.966±0.086	0.486±0.043	0.497±0.000	0.486±0.043	0.490±0.025	0.003±0.003	0.486±0.043	211.455	0.197
	NB	0.325±0.000	0.326±0.000	0.496±0.000	0.326±0.000	0.247±0.000	0.005±0.000	0.326±0.000	0.016	0.020
	RF	0.994±0.004	0.867±0.202	0.747±0.150	0.867±0.202	0.784±0.157	0.568±0.314	0.867±0.202	0.020	0.002
	SVM	0.995±0.000	0.500±0.000	0.497±0.000	0.500±0.000	0.499±0.000	0.000±0.000	0.500±0.000	39.534	8.556
	XGB	0.997±0.000	0.988±0.000	0.824±0.000	0.988±0.000	0.890±0.000	0.779±0.000	0.988±0.000	0.712	0.025
SHD	DT	1.000±0.000	0.877±0.153	0.908±0.147	0.877±0.153	0.890±0.150	0.780±0.299	0.877±0.153	0.044	0.016
	KNN	0.999±0.000	0.658±0.000	1.000±0.000	0.658±0.000	0.740±0.000	0.481±0.000	0.658±0.000	0.055	2.050
	LR	0.997±0.000	0.677±0.000	0.581±0.000	0.677±0.000	0.611±0.000	0.222±0.000	0.677±0.000	0.180	0.005
	MLP	0.991±0.003	0.556±0.150	0.592±0.205	0.556±0.150	0.560±0.152	0.124±0.302	0.556±0.150	149.257	0.230
	NB	0.996±0.000	0.998±0.000	0.616±0.000	0.998±0.000	0.687±0.000	0.375±0.000	0.998±0.000	0.009	0.086
	RF	0.999±0.000	0.735±0.052	1.000±0.000	0.735±0.052	0.816±0.047	0.633±0.094	0.735±0.052	0.868	0.440
	SVM	0.990±0.000	0.500±0.000	0.495±0.000	0.500±0.000	0.497±0.000	0.000±0.000	0.500±0.000	9.767	6.008
	XGB	1.000±0.000	0.946±0.000	0.964±0.000	0.946±0.000	0.954±0.000	0.909±0.000	0.946±0.000	0.595	0.104
PS	DT	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	0.012	0.002
	KNN	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	0.025	0.053
	LR	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	0.066	0.000
	MLP	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	13.383	0.095
	NB	0.978±0.000	0.987±0.000	0.942±0.000	0.987±0.000	0.963±0.000	0.925±0.000	0.987±0.000	0.003	0.002
	RF	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	0.013	0.003
	SVM	0.861±0.000	0.586±0.000	0.928±0.000	0.586±0.000	0.608±0.000	0.256±0.000	0.586±0.000	13.930	4.091
	XGB	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	0.999±0.000	1.000±0.000	1.850	0.016
SYN	DT	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.998±0.000	0.999±0.000	0.018	0.002
	KNN	0.997±0.000	0.997±0.000	0.996±0.000	0.997±0.000	0.997±0.000	0.993±0.000	0.997±0.000	0.113	9.999
	LR	0.993±0.008	0.994±0.007	0.993±0.009	0.994±0.007	0.993±0.009	0.987±0.017	0.994±0.007	0.053	0.004
	MLP	0.997±0.001	0.997±0.001	0.996±0.001	0.997±0.001	0.997±0.001	0.993±0.001	0.997±0.001	369.283	1.311
	NB	0.993±0.000	0.994±0.000	0.992±0.000	0.994±0.000	0.993±0.000	0.986±0.000	0.994±0.000	0.017	0.045
	RF	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.998±0.000	0.999±0.000	1.196	0.333
	SVM	0.953±0.000	0.945±0.000	0.962±0.000	0.945±0.000	0.951±0.000	0.902±0.000	0.945±0.000	66.502	15.793
	XGB	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.999±0.000	0.998±0.000	0.999±0.000	9.864	0.146
UDP	DT	0.913±0.183	0.955±0.094	0.907±0.197	0.955±0.094	0.884±0.245	0.814±0.392	0.955±0.094	0.036	0.056
	KNN	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	0.062	196.800
	LR	1.000±0.000	0.999±0.000	1.000±0.000	0.999±0.000	1.000±0.000	0.999±0.000	0.999±0.000	0.399	0.022
	MLP	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	478.953	1.236
	NB	0.998±0.000	0.969±0.000	0.999±0.000	0.969±0.000	0.983±0.000	0.967±0.000	0.969±0.000	0.013	0.340
	RF	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	0.454	1.509
	SVM	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	0.149	0.025
	XGB	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	5.069	0.547

Table A.24: Window approach Test results. T is time

Window approach Test results.

Attack	ML	Accuracy	BalancedAcc	Precision	Recall	F1-Score	kappa	ROC	Train-T	Test-T
ACK	DT	0.920±0.229	0.945±0.156	0.900±0.269	0.945±0.156	0.912±0.246	0.882±0.310	0.945±0.156	0.018	0.006
	KNN	0.269±0.000	0.500±0.000	0.134±0.000	0.500±0.000	0.212±0.000	0.000±0.000	0.500±0.000	0.029	1.555
	LR	0.989±0.000	0.992±0.000	0.980±0.000	0.992±0.000	0.986±0.000	0.972±0.000	0.992±0.000	0.084	0.001
	MLP	0.500±0.000	0.500±0.000	0.250±0.000	0.500±0.000	0.333±0.000	0.000±0.000	0.500±0.000	20.468	0.613
	NB	0.269±0.000	0.500±0.000	0.134±0.000	0.500±0.000	0.212±0.000	0.000±0.000	0.500±0.000	0.006	0.011
	RF	0.991±0.000	0.994±0.000	0.984±0.000	0.994±0.000	0.989±0.000	0.978±0.000	0.994±0.000	0.941	0.296
	SVM	0.500±0.000	0.500±0.000	0.250±0.000	0.500±0.000	0.333±0.000	0.000±0.000	0.500±0.000	30.356	6.798
	XGB	0.991±0.000	0.994±0.000	0.985±0.000	0.994±0.000	0.989±0.000	0.978±0.000	0.994±0.000	1.745	0.082
ARP	DT	0.521±0.021	0.503±0.034	0.444±0.127	0.503±0.034	0.454±0.084	0.004±0.071	0.503±0.034	0.680	1.131
	KNN	0.519±0.000	0.478±0.000	0.266±0.000	0.478±0.000	0.341±0.000	0.048±0.000	0.478±0.000	0.306	98.193
	LR	0.615±0.065	0.611±0.092	0.653±0.116	0.611±0.092	0.567±0.103	0.214±0.174	0.611±0.092	77.085	0.397
	MLP	0.513±0.019	0.513±0.019	0.708±0.065	0.513±0.019	0.373±0.063	0.026±0.038	0.513±0.019	24.572	0.091
	NB	0.490±0.000	0.451±0.000	0.259±0.000	0.451±0.000	0.329±0.000	0.105±0.000	0.451±0.000	0.027	4.659
	RF	0.527±0.002	0.485±0.002	0.268±0.000	0.485±0.002	0.345±0.001	0.032±0.004	0.485±0.002	18.588	45.121
	SVM	0.500±0.000	0.500±0.000	0.250±0.000	0.500±0.000	0.333±0.000	0.000±0.000	0.500±0.000	194.755	41.650
	XGB	0.536±0.000	0.494±0.000	0.270±0.000	0.494±0.000	0.349±0.000	0.013±0.000	0.494±0.000	11.910	4.855
BF	DT	0.476±0.039	0.519±0.036	0.633±0.214	0.519±0.036	0.351±0.070	0.036±0.067	0.519±0.036	0.038	0.000
	KNN	0.574±0.000	0.609±0.000	0.758±0.000	0.609±0.000	0.520±0.000	0.202±0.000	0.609±0.000	0.005	0.042
	LR	0.502±0.004	0.543±0.004	0.739±0.001	0.543±0.004	0.402±0.007	0.079±0.007	0.543±0.004	0.242	0.000
	MLP	0.545±0.047	0.578±0.042	0.677±0.062	0.578±0.042	0.486±0.072	0.146±0.080	0.578±0.042	82.584	0.009
	NB	0.745±0.000	0.766±0.000	0.821±0.000	0.766±0.000	0.738±0.000	0.509±0.000	0.766±0.000	0.006	0.000
	RF	0.457±0.000	0.502±0.000	0.728±0.000	0.502±0.000	0.317±0.000	0.003±0.000	0.502±0.000	2.283	0.024
	SVM	0.455±0.000	0.500±0.000	0.228±0.000	0.500±0.000	0.313±0.000	0.000±0.000	0.500±0.000	7.066	0.193
	XGB	0.455±0.000	0.500±0.000	0.228±0.000	0.500±0.000	0.313±0.000	0.000±0.000	0.500±0.000	2.904	0.005
HTTP	DT	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	0.032	0.030
	KNN	0.473±0.000	0.520±0.000	0.730±0.000	0.520±0.000	0.355±0.000	0.037±0.000	0.520±0.000	0.046	6.081
	LR	0.450±0.000	0.500±0.000	0.225±0.000	0.500±0.000	0.310±0.000	0.000±0.000	0.500±0.000	0.335	0.013
	MLP	0.527±0.022	0.527±0.022	0.757±0.006	0.527±0.022	0.388±0.044	0.053±0.043	0.527±0.022	4.813	0.039
	NB	0.473±0.000	0.521±0.000	0.714±0.000	0.521±0.000	0.356±0.000	0.037±0.000	0.521±0.000	0.011	0.121
	RF	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.803	1.537
	SVM	0.500±0.000	0.500±0.000	0.250±0.000	0.500±0.000	0.333±0.000	0.000±0.000	0.500±0.000	78.257	21.362
	XGB	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	2.135	0.219
MHD	DT	0.982±0.017	0.770±0.218	0.986±0.013	0.770±0.218	0.795±0.204	0.595±0.403	0.770±0.218	0.020	0.003
	KNN	0.999±0.000	0.999±0.000	0.987±0.000	0.999±0.000	0.993±0.000	0.986±0.000	0.999±0.000	0.025	0.475
	LR	1.000±0.000	1.000±0.000	0.997±0.000	1.000±0.000	0.998±0.000	0.997±0.000	1.000±0.000	0.230	0.002
	MLP	0.999±0.000	1.000±0.000	0.990±0.004	1.000±0.000	0.995±0.002	0.990±0.005	1.000±0.000	316.916	0.821
	NB	0.964±0.000	0.981±0.000	0.760±0.000	0.981±0.000	0.833±0.000	0.668±0.000	0.981±0.000	0.007	0.008
	RF	0.998±0.001	0.973±0.007	0.998±0.000	0.973±0.007	0.985±0.004	0.970±0.008	0.973±0.007	0.480	0.075
	SVM	0.959±0.000	0.554±0.000	0.979±0.000	0.554±0.000	0.587±0.000	0.189±0.000	0.554±0.000	14.813	5.792
	XGB	0.979±0.000	0.732±0.000	0.987±0.000	0.732±0.000	0.811±0.000	0.623±0.000	0.732±0.000	4.638	0.067
OS	DT	0.597±0.012	0.757±0.001	0.616±0.000	0.757±0.001	0.542±0.008	0.223±0.006	0.757±0.001	0.087	0.031
	KNN	0.785±0.000	0.464±0.000	0.452±0.000	0.464±0.000	0.457±0.000	0.082±0.000	0.464±0.000	0.006	7.621
	LR	0.774±0.044	0.862±0.025	0.681±0.023	0.862±0.025	0.691±0.041	0.423±0.065	0.862±0.025	0.073	0.011
	MLP	0.486±0.055	0.486±0.055	0.348±0.208	0.486±0.055	0.332±0.033	0.028±0.111	0.486±0.055	183.464	0.379
	NB	0.724±0.000	0.449±0.000	0.459±0.000	0.449±0.000	0.453±0.000	0.090±0.000	0.449±0.000	0.017	0.185
	RF	0.826±0.111	0.639±0.146	0.698±0.127	0.639±0.146	0.618±0.119	0.266±0.227	0.639±0.146	0.031	0.066
	SVM	0.500±0.000	0.500±0.000	0.250±0.000	0.500±0.000	0.333±0.000	0.000±0.000	0.500±0.000	37.010	17.52
	XGB	0.610±0.000	0.758±0.000	0.616±0.000	0.758±0.000	0.551±0.000	0.230±0.000	0.758±0.000	1.039	0.259
SHD	DT	0.999±0.000	0.897±0.098	0.897±0.002	0.897±0.098	0.889±0.072	0.778±0.144	0.897±0.098	0.059	0.022
	KNN	0.998±0.000	0.676±0.000	0.968±0.000	0.676±0.000	0.756±0.000	0.512±0.000	0.676±0.000	0.078	1.901
	LR	0.989±0.000	0.971±0.000	0.583±0.000	0.971±0.000	0.639±0.000	0.281±0.000	0.971±0.000	0.288	0.007
	MLP	0.989±0.007	0.733±0.163	0.972±0.027	0.733±0.163	0.781±0.158	0.564±0.314	0.733±0.163	150.392	0.233
	NB	0.995±0.000	0.997±0.000	0.647±0.000	0.997±0.000	0.726±0.000	0.453±0.000	0.997±0.000	0.010	0.113
	RF	0.998±0.000	0.621±0.030	0.910±0.071	0.621±0.030	0.684±0.031	0.368±0.061	0.621±0.030	1.147	0.543
	SVM	0.979±0.000	0.500±0.000	0.490±0.000	0.500±0.000	0.495±0.000	0.000±0.000	0.500±0.000	8.656	5.830
	XGB	0.999±0.000	0.928±0.000	0.896±0.000	0.928±0.000	0.912±0.000	0.823±0.000	0.928±0.000	0.976	0.176
PS	DT	0.898±0.068	0.903±0.067	0.916±0.054	0.903±0.067	0.896±0.070	0.797±0.136	0.903±0.067	0.014	0.000
	KNN	0.997±0.000	0.997±0.000	0.997±0.000	0.997±0.000	0.997±0.000	0.994±0.000	0.997±0.000	0.020	0.042
	LR	0.997±0.000	0.997±0.000	0.997±0.000	0.997±0.000	0.997±0.000	0.994±0.000	0.997±0.000	0.069	0.000
	MLP	0.997±0.000	0.997±0.000	0.997±0.000	0.997±0.000	0.997±0.000	0.994±0.000	0.997±0.000	8.699	0.074
	NB	0.972±0.000	0.974±0.000	0.971±0.000	0.974±0.000	0.971±0.000	0.943±0.000	0.974±0.000	0.000	0.002
	RF	0.863±0.000	0.874±0.000	0.884±0.000	0.874±0.000	0.862±0.000	0.730±0.000	0.874±0.000	0.010	0.003
	SVM	0.592±0.000	0.554±0.000	0.783±0.000	0.554±0.000	0.461±0.000	0.116±0.000	0.554±0.000	13.858	4.933
	XGB	0.945±0.000	0.949±0.000	0.946±0.000	0.949±0.000	0.945±0.000	0.890±0.000	0.949±0.000	1.438	0.014
SYN	DT	0.943±0.181	0.950±0.158	0.971±0.091	0.950±0.158	0.930±0.221	0.900±0.315	0.950±0.158	0.016	0.180
	KNN	0.425±0.000	0.500±0.000	0.212±0.000	0.500±0.000	0.298±0.000	0.000±0.000	0.500±0.000	0.104	66.594
	LR	0.982±0.053	0.980±0.062	0.988±0.036	0.980±0.062	0.981±0.059	0.962±0.114	0.980±0.062	0.056	0.069
	MLP	0.500±0.000	0.500±0.000	0.250±0.000	0.500±0.000	0.333±0.000	0.000±0.000	0.500±0.000	172.875	0.613
	NB	0.998±0.000	0.998±0.000	0.998±0.000	0.998±0.000	0.998±0.000	0.996±0.000	0.998±0.000	0.012	0.838
	RF	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.170	8.297
	SVM	0.500±0.000	0.500±0.000	0.250±0.000	0.500±0.000	0.333±0.000	0.000±0.000	0.500±0.000	71.319	16.116
	XGB	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	5.706	1.489
UDP	DT	0.454±0.323	0.695±0.125	0.560±0.121	0.695±0.125	0.359±0.246	0.128±0.262	0.695±0.125	0.032	0.288
	KNN	0.255±0.000	0.620±0.000	0.512±0.000	0.620±0.000	0.218±0.000	0.012±0.000	0.620±0.000	0.057	944.879
	LR	0.255±0.000	0.620±0.000	0.512±0.000	0.620±0.000	0.218±0.000	0.012±0.000	0.620±0.000	0.380	0.142
	MLP	0.838±0.188	0.838±0.188	0.904±0.104	0.838±0.188	0.812±0.221	0.676±0.376	0.838±0.188	239.447	0.882
	NB	0.996±0.000	0.897±0.000	0.998±0.000	0.897±0.000	0.941±0.000	0.883±0.000	0.897±0.000	0.016	1.911
	RF	0.514±0.335	0.752±0.171	0.540±0.036	0.752±0.171	0.377±0.206	0.111±0.128	0.752±0.171	0.443	7.048
	SVM	0.620±0.000	0.620±0.000	0.784±0.000	0.620±0.000	0.556±0.000	0.240±0.000	0.620±0.000	0.140	0.019
	XGB	0.255±0.000	0.620±0.000	0.512±0.000	0.620±0.000	0.218±0.000	0.012±0.000	0.620±0.000	16.237	4.475

Table A.25: Flow approach CV results. T is time

Attack	ML	Accuracy	BalancedAcc	Precision	Recall	F1-Score	kappa	ROC	Train-T	Test-T
ACK	DT	0.894±0.015	0.910±0.013	0.897±0.015	0.910±0.013	0.893±0.016	0.789±0.030	0.910±0.013	0.000	0.000
	KNN	0.588±0.020	0.498±0.002	0.294±0.010	0.498±0.002	0.370±0.008	0.004±0.004	0.498±0.002	0.002	0.010
	LR	0.896±0.017	0.911±0.015	0.898±0.015	0.911±0.015	0.894±0.017	0.791±0.033	0.911±0.015	0.043	0.002
	MLP	0.896±0.012	0.912±0.009	0.898±0.012	0.912±0.009	0.895±0.012	0.792±0.024	0.912±0.009	1.808	0.003
	NB	0.799±0.028	0.830±0.021	0.836±0.020	0.830±0.021	0.799±0.028	0.613±0.050	0.830±0.021	0.001	0.002
	RF	0.894±0.011	0.910±0.007	0.897±0.014	0.910±0.007	0.893±0.011	0.788±0.022	0.910±0.007	0.330	0.025
	SVM	0.895±0.019	0.911±0.017	0.898±0.016	0.911±0.017	0.894±0.019	0.790±0.037	0.911±0.017	0.031	0.019
	XGB	0.894±0.027	0.910±0.023	0.898±0.023	0.910±0.023	0.893±0.027	0.789±0.053	0.910±0.023	0.982	0.001
ARP	DT	0.677±0.064	0.662±0.066	0.673±0.074	0.662±0.066	0.659±0.066	0.325±0.132	0.662±0.066	0.007	0.000
	KNN	0.673±0.047	0.639±0.058	0.669±0.061	0.639±0.058	0.636±0.062	0.290±0.116	0.639±0.058	0.002	0.002
	LR	0.678±0.059	0.660±0.060	0.667±0.065	0.660±0.060	0.661±0.062	0.325±0.124	0.660±0.060	1.347	0.000
	MLP	0.672±0.040	0.646±0.035	0.666±0.034	0.646±0.035	0.641±0.040	0.298±0.069	0.646±0.035	12.560	0.012
	NB	0.583±0.029	0.517±0.023	0.546±0.054	0.517±0.023	0.463±0.035	0.038±0.051	0.517±0.023	0.001	0.001
	RF	0.705±0.037	0.679±0.035	0.695±0.035	0.679±0.035	0.682±0.035	0.368±0.068	0.679±0.035	0.081	0.007
	SVM	0.584±0.057	0.498±0.009	0.369±0.127	0.498±0.009	0.375±0.027	0.006±0.023	0.498±0.009	0.014	0.006
	XGB	0.705±0.048	0.679±0.059	0.700±0.064	0.679±0.059	0.681±0.059	0.368±0.117	0.679±0.059	1.387	0.004
BF	DT	0.926±0.018	0.853±0.044	0.854±0.039	0.853±0.044	0.851±0.036	0.703±0.073	0.853±0.044	0.004	0.000
	KNN	0.890±0.030	0.782±0.050	0.788±0.074	0.782±0.050	0.781±0.062	0.564±0.122	0.782±0.050	0.002	0.002
	LR	0.838±0.037	0.494±0.007	0.423±0.016	0.494±0.007	0.456±0.011	-0.02±0.020	0.494±0.007	0.006	0.000
	MLP	0.858±0.026	0.621±0.052	0.732±0.114	0.621±0.052	0.640±0.062	0.295±0.117	0.621±0.052	57.888	0.066
	NB	0.817±0.040	0.501±0.029	0.555±0.124	0.501±0.029	0.486±0.034	0.008±0.080	0.501±0.029	0.001	0.000
	RF	0.943±0.019	0.863±0.035	0.908±0.034	0.863±0.035	0.883±0.032	0.766±0.064	0.863±0.035	0.299	0.021
	SVM	0.851±0.027	0.507±0.010	0.625±0.265	0.507±0.010	0.474±0.023	0.025±0.032	0.507±0.010	0.063	0.027
	XGB	0.948±0.013	0.878±0.044	0.915±0.041	0.878±0.044	0.892±0.030	0.785±0.059	0.878±0.044	1.580	0.006
HTTP	DT	0.964±0.010	0.961±0.009	0.962±0.012	0.961±0.009	0.962±0.010	0.923±0.021	0.961±0.009	0.004	0.000
	KNN	0.897±0.016	0.885±0.019	0.898±0.018	0.885±0.019	0.890±0.018	0.781±0.036	0.885±0.019	0.002	0.034
	LR	0.939±0.010	0.922±0.013	0.953±0.008	0.922±0.013	0.934±0.011	0.868±0.022	0.922±0.013	0.018	0.002
	MLP	0.796±0.017	0.749±0.015	0.838±0.017	0.749±0.015	0.761±0.017	0.538±0.031	0.749±0.015	2.353	0.007
	NB	0.831±0.017	0.788±0.019	0.879±0.014	0.788±0.019	0.804±0.020	0.619±0.038	0.788±0.019	0.002	0.000
	RF	0.973±0.008	0.970±0.009	0.973±0.008	0.970±0.009	0.971±0.009	0.942±0.017	0.970±0.009	0.048	0.003
	SVM	0.611±0.012	0.506±0.004	0.662±0.160	0.506±0.004	0.392±0.009	0.013±0.011	0.506±0.004	0.227	0.096
	XGB	0.970±0.007	0.966±0.009	0.971±0.006	0.966±0.009	0.968±0.007	0.937±0.014	0.966±0.009	1.859	0.009
PS	DT	0.999±0.002	0.990±0.021	0.999±0.001	0.990±0.021	0.995±0.012	0.989±0.024	0.990±0.021	0.001	0.000
	KNN	0.993±0.006	0.941±0.043	0.996±0.003	0.941±0.043	0.966±0.026	0.932±0.052	0.941±0.043	0.003	0.011
	LR	0.998±0.005	0.983±0.039	0.999±0.002	0.983±0.039	0.990±0.023	0.980±0.046	0.983±0.039	0.003	0.000
	MLP	0.981±0.026	0.932±0.053	0.924±0.083	0.932±0.053	0.924±0.064	0.849±0.128	0.932±0.053	3.001	0.008
	NB	0.993±0.004	0.940±0.054	0.986±0.035	0.940±0.054	0.959±0.039	0.918±0.079	0.940±0.054	0.001	0.000
	RF	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	0.053	0.006
	SVM	0.944±0.015	0.517±0.022	0.672±0.258	0.517±0.022	0.516±0.041	0.058±0.077	0.517±0.022	0.013	0.005
	XGB	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.854	0.005
SYN	DT	1.000±0.000	0.967±0.081	0.995±0.014	0.967±0.081	0.977±0.054	0.954±0.109	0.967±0.081	0.047	0.000
	KNN	0.999±0.000	0.885±0.068	1.000±0.000	0.885±0.068	0.932±0.044	0.863±0.088	0.885±0.068	0.001	0.469
	LR	1.000±0.000	0.954±0.063	0.973±0.058	0.954±0.063	0.960±0.050	0.920±0.100	0.954±0.063	0.446	0.000
	MLP	1.000±0.000	0.980±0.037	0.975±0.060	0.980±0.037	0.975±0.042	0.950±0.084	0.980±0.037	21.141	0.043
	NB	0.996±0.001	0.919±0.079	0.679±0.044	0.919±0.079	0.746±0.047	0.493±0.093	0.919±0.079	0.007	0.009
	RF	1.000±0.000	0.985±0.032	1.000±0.000	0.985±0.032	0.992±0.017	0.984±0.035	0.985±0.032	0.907	0.028
	SVM	0.998±0.001	0.500±0.000	0.499±0.001	0.500±0.000	0.499±0.000	0.000±0.000	0.500±0.000	42.601	10.340
	XGB	1.000±0.000	0.960±0.053	1.000±0.000	0.960±0.053	0.977±0.031	0.955±0.062	0.960±0.053	0.678	0.013
UDP	DT	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	0.005	0.000
	KNN	0.993±0.006	0.996±0.003	0.883±0.088	0.996±0.003	0.926±0.063	0.853±0.125	0.996±0.003	0.002	0.002
	LR	0.997±0.007	0.998±0.003	0.959±0.074	0.998±0.003	0.975±0.048	0.949±0.095	0.998±0.003	0.037	0.000
	MLP	0.995±0.008	0.966±0.071	0.949±0.121	0.966±0.071	0.945±0.104	0.890±0.207	0.966±0.071	25.329	0.011
	NB	0.889±0.026	0.943±0.013	0.598±0.053	0.943±0.013	0.628±0.075	0.288±0.134	0.943±0.013	0.001	0.000
	RF	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	0.012	0.002
	SVM	0.973±0.010	0.500±0.000	0.487±0.005	0.500±0.000	0.493±0.002	0.000±0.000	0.500±0.000	0.038	0.017
	XGB	0.999±0.003	0.999±0.001	0.983±0.037	0.999±0.001	0.990±0.021	0.981±0.042	0.999±0.001	0.414	0.004

Flow approach CV results.

Table A.26: Flow approach Session results. T is time

Flow approach Session results.

Attack	ML	Accuracy	BalancedAcc	Precision	Recall	F1-Score	kappa	ROC	Train-T	Test-T
ACK	DT	0.997±0.000	0.939±0.001	0.995±0.001	0.939±0.001	0.966±0.001	0.931±0.002	0.939±0.001	0.001	0.000
	KNN	0.022±0.000	0.500±0.000	0.511±0.000	0.500±0.000	0.022±0.000	0.000±0.000	0.500±0.000	0.002	0.295
	LR	0.997±0.000	0.940±0.000	0.995±0.000	0.940±0.000	0.966±0.000	0.931±0.000	0.940±0.000	0.060	0.001
	MLP	0.997±0.000	0.940±0.000	0.995±0.000	0.940±0.000	0.966±0.000	0.931±0.000	0.940±0.000	2.264	0.014
	NB	0.992±0.000	0.816±0.000	0.996±0.000	0.816±0.000	0.886±0.000	0.771±0.000	0.816±0.000	0.001	0.004
	RF	0.997±0.000	0.940±0.000	0.995±0.000	0.940±0.000	0.966±0.000	0.931±0.000	0.940±0.000	0.340	0.119
	SVM	0.997±0.000	0.940±0.000	0.995±0.000	0.940±0.000	0.966±0.000	0.931±0.000	0.940±0.000	0.035	0.407
	XGB	0.997±0.000	0.940±0.000	0.999±0.000	0.940±0.000	0.967±0.000	0.935±0.000	0.940±0.000	1.000	0.020
ARP	DT	0.702±0.018	0.721±0.017	0.707±0.016	0.721±0.017	0.698±0.017	0.409±0.032	0.721±0.017	0.011	0.000
	KNN	0.721±0.000	0.719±0.000	0.708±0.000	0.719±0.000	0.710±0.000	0.423±0.000	0.719±0.000	0.002	0.004
	LR	0.675±0.000	0.704±0.000	0.693±0.000	0.704±0.000	0.674±0.000	0.370±0.000	0.704±0.000	1.297	0.000
	MLP	0.694±0.040	0.698±0.033	0.688±0.034	0.698±0.033	0.686±0.037	0.379±0.070	0.698±0.033	16.508	0.035
	NB	0.554±0.000	0.464±0.000	0.437±0.000	0.464±0.000	0.428±0.000	-0.082±0.00	0.464±0.000	0.001	0.000
	RF	0.730±0.009	0.753±0.007	0.736±0.006	0.753±0.007	0.727±0.008	0.466±0.015	0.753±0.007	0.091	0.010
	SVM	0.643±0.000	0.518±0.000	0.819±0.000	0.518±0.000	0.424±0.000	0.044±0.000	0.518±0.000	0.022	0.025
	XGB	0.734±0.000	0.754±0.000	0.738±0.000	0.754±0.000	0.731±0.000	0.472±0.000	0.754±0.000	1.282	0.006
BF	DT	0.886±0.007	0.752±0.009	0.787±0.017	0.752±0.009	0.767±0.012	0.535±0.024	0.752±0.009	0.005	0.000
	KNN	0.869±0.000	0.709±0.000	0.749±0.000	0.709±0.000	0.726±0.000	0.452±0.000	0.709±0.000	0.002	0.008
	LR	0.844±0.000	0.498±0.000	0.423±0.000	0.498±0.000	0.458±0.000	-0.005±0.00	0.498±0.000	0.009	0.001
	MLP	0.846±0.015	0.558±0.035	0.682±0.073	0.558±0.035	0.564±0.052	0.159±0.092	0.558±0.035	58.708	0.267
	NB	0.829±0.000	0.517±0.000	0.562±0.000	0.517±0.000	0.506±0.000	0.050±0.000	0.517±0.000	0.001	0.000
	RF	0.916±0.001	0.767±0.003	0.884±0.004	0.767±0.003	0.810±0.003	0.622±0.006	0.767±0.003	0.340	0.038
	SVM	0.848±0.000	0.510±0.000	0.758±0.000	0.510±0.000	0.480±0.000	0.033±0.000	0.510±0.000	0.104	0.205
	XGB	0.912±0.000	0.762±0.000	0.873±0.000	0.762±0.000	0.803±0.000	0.608±0.000	0.762±0.000	4.756	0.024
HTTP	DT	0.934±0.075	0.936±0.068	0.936±0.063	0.936±0.068	0.933±0.076	0.868±0.144	0.936±0.068	0.005	0.000
	KNN	0.843±0.000	0.842±0.000	0.839±0.000	0.842±0.000	0.840±0.000	0.681±0.000	0.842±0.000	0.002	0.047
	LR	0.923±0.000	0.911±0.000	0.940±0.000	0.911±0.000	0.919±0.000	0.839±0.000	0.911±0.000	0.023	0.000
	MLP	0.810±0.007	0.784±0.008	0.859±0.010	0.784±0.008	0.791±0.009	0.594±0.016	0.784±0.008	2.555	0.011
	NB	0.779±0.000	0.745±0.000	0.860±0.000	0.745±0.000	0.747±0.000	0.522±0.000	0.745±0.000	0.000	0.000
	RF	0.963±0.001	0.964±0.001	0.961±0.002	0.964±0.001	0.962±0.001	0.924±0.003	0.964±0.001	0.062	0.001
	SVM	0.569±0.000	0.503±0.000	0.784±0.000	0.503±0.000	0.369±0.000	0.007±0.000	0.503±0.000	0.367	0.120
	XGB	0.942±0.000	0.942±0.000	0.940±0.000	0.942±0.000	0.941±0.000	0.882±0.000	0.942±0.000	2.052	0.010
PS	DT	0.998±0.003	0.992±0.010	0.996±0.010	0.992±0.010	0.994±0.010	0.988±0.020	0.992±0.010	0.002	0.000
	KNN	0.971±0.000	0.901±0.000	0.923±0.000	0.901±0.000	0.912±0.000	0.824±0.000	0.901±0.000	0.003	0.046
	LR	1.000±0.001	0.998±0.003	1.000±0.000	0.998±0.003	0.999±0.002	0.997±0.004	0.998±0.003	0.003	0.001
	MLP	0.949±0.050	0.961±0.030	0.863±0.121	0.961±0.030	0.891±0.095	0.787±0.184	0.961±0.030	3.235	0.020
	NB	0.989±0.000	0.942±0.000	0.994±0.000	0.942±0.000	0.966±0.000	0.932±0.000	0.942±0.000	0.002	0.001
	RF	0.999±0.000	0.996±0.002	1.000±0.000	0.996±0.002	0.998±0.001	0.995±0.002	0.996±0.002	0.043	0.008
	SVM	0.906±0.000	0.500±0.000	0.453±0.000	0.500±0.000	0.475±0.000	0.000±0.000	0.500±0.000	0.023	0.039
	XGB	0.999±0.000	0.995±0.000	0.999±0.000	0.995±0.000	0.997±0.000	0.995±0.000	0.995±0.000	1.632	0.009
SYN	DT	1.000±0.000	0.983±0.011	0.994±0.006	0.983±0.011	0.988±0.006	0.977±0.012	0.983±0.011	0.059	0.003
	KNN	0.998±0.000	0.879±0.000	0.953±0.000	0.879±0.000	0.913±0.000	0.825±0.000	0.879±0.000	0.003	1.996
	LR	0.998±0.000	0.879±0.000	0.925±0.000	0.879±0.000	0.901±0.000	0.801±0.000	0.879±0.000	0.530	0.002
	MLP	0.998±0.001	0.906±0.015	0.923±0.046	0.906±0.015	0.913±0.019	0.825±0.039	0.906±0.015	21.131	0.177
	NB	0.995±0.000	0.899±0.000	0.785±0.000	0.899±0.000	0.832±0.000	0.664±0.000	0.899±0.000	0.014	0.017
	RF	1.000±0.000	0.989±0.006	0.994±0.004	0.989±0.006	0.992±0.002	0.983±0.005	0.989±0.006	1.040	0.099
	SVM	0.994±0.000	0.500±0.000	0.497±0.000	0.500±0.000	0.498±0.000	0.000±0.000	0.500±0.000	65.403	50.148
	XGB	1.000±0.000	0.973±0.000	1.000±0.000	0.973±0.000	0.986±0.000	0.972±0.000	0.973±0.000	0.728	0.032
UDP	DT	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	0.001	0.000
	KNN	0.496±0.000	0.576±0.000	0.723±0.000	0.576±0.000	0.441±0.000	0.127±0.000	0.576±0.000	0.001	0.003
	LR	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	0.165	0.000
	MLP	0.497±0.058	0.577±0.049	0.673±0.166	0.577±0.049	0.437±0.084	0.130±0.086	0.577±0.049	36.424	0.027
	NB	0.687±0.000	0.718±0.000	0.723±0.000	0.718±0.000	0.687±0.000	0.402±0.000	0.718±0.000	0.000	0.002
	RF	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	0.012	0.002
	SVM	0.406±0.000	0.500±0.000	0.203±0.000	0.500±0.000	0.289±0.000	0.000±0.000	0.500±0.000	0.057	0.054
	XGB	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	0.397	0.003

Table A.27: Flow approach Test results. T is time

Attack	ML	Accuracy	BalancedAcc	Precision	Recall	F1-Score	kappa	ROC	Train-T	Test-T
ACK	DT	0.116±0.000	0.443±0.000	0.375±0.000	0.443±0.000	0.113±0.000	-0.024±0.00	0.443±0.000	0.000	0.000
	KNN	0.102±0.000	0.500±0.000	0.051±0.000	0.500±0.000	0.093±0.000	0.000±0.000	0.500±0.000	0.003	0.021
	LR	0.116±0.000	0.443±0.000	0.375±0.000	0.443±0.000	0.113±0.000	-0.024±0.00	0.443±0.000	0.044	0.000
	MLP	0.116±0.000	0.443±0.000	0.375±0.000	0.443±0.000	0.113±0.000	-0.024±0.00	0.443±0.000	2.851	0.002
	NB	0.118±0.000	0.444±0.000	0.383±0.000	0.444±0.000	0.115±0.000	-0.024±0.00	0.444±0.000	0.001	0.001
	RF	0.116±0.000	0.443±0.000	0.375±0.000	0.443±0.000	0.113±0.000	-0.024±0.00	0.443±0.000	0.357	0.036
	SVM	0.116±0.000	0.443±0.000	0.375±0.000	0.443±0.000	0.113±0.000	-0.024±0.00	0.443±0.000	0.024	0.022
	XGB	0.116±0.000	0.443±0.000	0.375±0.000	0.443±0.000	0.113±0.000	-0.024±0.00	0.443±0.000	0.670	0.005
ARP	DT	0.670±0.047	0.457±0.032	0.357±0.007	0.457±0.032	0.401±0.017	-0.108±0.08	0.457±0.032	0.009	0.000
	KNN	0.782±0.000	0.597±0.000	0.848±0.000	0.597±0.000	0.599±0.000	0.258±0.000	0.597±0.000	0.002	0.002
	LR	0.462±0.000	0.331±0.000	0.342±0.000	0.331±0.000	0.336±0.000	-0.326±0.00	0.331±0.000	1.491	0.000
	MLP	0.832±0.044	0.739±0.065	0.808±0.068	0.739±0.065	0.759±0.065	0.524±0.128	0.739±0.065	7.357	0.009
	NB	0.787±0.000	0.733±0.000	0.728±0.000	0.733±0.000	0.730±0.000	0.460±0.000	0.733±0.000	0.002	0.000
	RF	0.728±0.012	0.496±0.008	0.366±0.002	0.496±0.008	0.421±0.004	-0.01±0.023	0.496±0.008	0.087	0.010
	SVM	0.267±0.000	0.500±0.000	0.133±0.000	0.500±0.000	0.211±0.000	0.000±0.000	0.500±0.000	0.013	0.007
	XGB	0.636±0.000	0.433±0.000	0.352±0.000	0.433±0.000	0.389±0.000	-0.167±0.00	0.433±0.000	1.302	0.002
BF	DT	0.521±0.042	0.553±0.039	0.686±0.084	0.553±0.039	0.433±0.077	0.099±0.073	0.553±0.039	0.005	0.000
	KNN	0.463±0.000	0.496±0.000	0.480±0.000	0.496±0.000	0.359±0.000	-0.008±0.00	0.496±0.000	0.002	0.001
	LR	0.463±0.000	0.500±0.000	0.231±0.000	0.500±0.000	0.316±0.000	0.000±0.000	0.500±0.000	0.010	0.000
	MLP	0.548±0.164	0.579±0.152	0.566±0.298	0.579±0.152	0.443±0.209	0.155±0.305	0.579±0.152	29.691	0.013
	NB	0.701±0.000	0.722±0.000	0.804±0.000	0.722±0.000	0.686±0.000	0.425±0.000	0.722±0.000	0.000	0.000
	RF	0.464±0.037	0.499±0.035	0.349±0.200	0.499±0.035	0.343±0.063	-0.002±0.07	0.499±0.035	0.329	0.017
	SVM	0.463±0.000	0.500±0.000	0.231±0.000	0.500±0.000	0.316±0.000	0.000±0.000	0.500±0.000	0.053	0.003
	XGB	0.567±0.000	0.597±0.000	0.758±0.000	0.597±0.000	0.503±0.000	0.183±0.000	0.597±0.000	1.888	0.002
HTTP	DT	0.747±0.086	0.853±0.050	0.687±0.044	0.853±0.050	0.681±0.080	0.418±0.124	0.853±0.050	0.000	0.006
	KNN	0.614±0.000	0.775±0.000	0.633±0.000	0.775±0.000	0.566±0.000	0.256±0.000	0.775±0.000	0.000	0.192
	LR	0.957±0.000	0.975±0.000	0.882±0.000	0.975±0.000	0.920±0.000	0.841±0.000	0.975±0.000	0.020	0.003
	MLP	0.556±0.015	0.326±0.008	0.401±0.002	0.326±0.008	0.359±0.006	-0.23±0.005	0.326±0.008	1.241	0.309
	NB	0.985±0.000	0.991±0.000	0.952±0.000	0.991±0.000	0.971±0.000	0.941±0.000	0.991±0.000	0.000	0.014
	RF	0.651±0.003	0.797±0.002	0.643±0.001	0.797±0.002	0.596±0.003	0.291±0.003	0.797±0.002	0.060	0.044
	SVM	0.860±0.000	0.500±0.000	0.430±0.000	0.500±0.000	0.462±0.000	0.000±0.000	0.500±0.000	0.141	4.179
	XGB	0.719±0.000	0.837±0.000	0.666±0.000	0.837±0.000	0.652±0.000	0.366±0.000	0.837±0.000	1.869	0.176
PS	DT	0.989±0.021	0.962±0.015	0.948±0.101	0.962±0.015	0.949±0.077	0.898±0.151	0.962±0.015	0.000	0.002
	KNN	0.948±0.000	0.713±0.000	0.681±0.000	0.713±0.000	0.696±0.000	0.391±0.000	0.713±0.000	0.005	0.113
	LR	0.999±0.000	0.994±0.000	0.994±0.000	0.994±0.000	0.994±0.000	0.987±0.000	0.994±0.000	0.007	0.000
	MLP	0.904±0.112	0.753±0.092	0.672±0.076	0.753±0.092	0.685±0.110	0.394±0.170	0.753±0.092	3.587	0.041
	NB	0.964±0.000	0.710±0.000	0.780±0.000	0.710±0.000	0.739±0.000	0.478±0.000	0.710±0.000	0.002	0.002
	RF	0.997±0.000	0.969±0.000	0.992±0.000	0.969±0.000	0.980±0.000	0.960±0.000	0.969±0.000	0.044	0.012
	SVM	0.960±0.000	0.530±0.000	0.793±0.000	0.530±0.000	0.546±0.000	0.106±0.000	0.530±0.000	0.033	0.094
	XGB	0.997±0.000	0.969±0.000	0.992±0.000	0.969±0.000	0.980±0.000	0.960±0.000	0.969±0.000	1.531	0.004
SYN	DT	0.185±0.288	0.592±0.144	0.450±0.158	0.592±0.144	0.127±0.141	0.002±0.005	0.592±0.144	0.067	0.084
	KNN	0.580±0.000	0.510±0.000	0.500±0.000	0.510±0.000	0.367±0.000	0.000±0.000	0.510±0.000	0.005	18.839
	LR	0.731±0.000	0.765±0.000	0.500±0.000	0.765±0.000	0.422±0.000	0.000±0.000	0.765±0.000	0.918	0.054
	MLP	0.333±0.333	0.548±0.156	0.498±0.004	0.548±0.156	0.208±0.188	0.000±0.000	0.548±0.156	7.714	1.842
	NB	0.592±0.000	0.676±0.000	0.500±0.000	0.676±0.000	0.372±0.000	0.000±0.000	0.676±0.000	0.017	0.505
	RF	0.000±0.000	0.500±0.000	0.000±0.000	0.500±0.000	0.000±0.000	0.000±0.000	0.500±0.000	1.063	2.673
	SVM	1.000±0.000	0.500±0.000	0.500±0.000	0.500±0.000	0.500±0.000	0.000±0.000	0.500±0.000	30.999	404.215
	XGB	0.123±0.000	0.562±0.000	0.500±0.000	0.562±0.000	0.110±0.000	0.000±0.000	0.562±0.000	0.681	0.338
UDP	DT	0.001±0.000	0.363±0.000	0.000±0.000	0.363±0.000	0.001±0.000	-0.001±0.00	0.363±0.000	0.001	0.028
	KNN	0.001±0.000	0.500±0.000	0.001±0.000	0.500±0.000	0.001±0.000	0.000±0.000	0.500±0.000	0.000	1.290
	LR	0.001±0.000	0.500±0.000	0.001±0.000	0.500±0.000	0.001±0.000	0.000±0.000	0.500±0.000	0.140	0.009
	MLP	0.001±0.000	0.500±0.000	0.001±0.000	0.500±0.000	0.001±0.000	0.000±0.000	0.500±0.000	29.840	4.562
	NB	0.001±0.000	0.499±0.000	0.001±0.000	0.499±0.000	0.001±0.000	-0.00±0.000	0.499±0.000	0.002	0.126
	RF	0.001±0.000	0.418±0.071	0.001±0.000	0.418±0.071	0.001±0.000	-0.00±0.000	0.418±0.071	0.010	0.193
	SVM	0.001±0.000	0.500±0.000	0.001±0.000	0.500±0.000	0.001±0.000	0.000±0.000	0.500±0.000	0.027	12.958
	XGB	0.001±0.000	0.363±0.000	0.000±0.000	0.363±0.000	0.001±0.000	-0.001±0.00	0.363±0.000	0.392	0.207

Flow approach Test results.

Bibliography

[1] G. Abdelmoumin, D. B. Rawat, and A. Rahman. On the Performance of Machine Learning Models for Anomaly-Based Intelligent Intrusion Detection Systems for the Internet of Things. *IEEE Internet of Things Journal*, 4662(c):1–1, 2021. https://doi.org/10.1109/JIOT.2021.3103829.

[2] Q. Abu Al-Haija, A. Al Badawi, and G. R. Bojja. Boost-defence for resilient IoT networks: A head-to-toe approach. *Expert Systems*, 39(10):e12934, 2022. https://doi.org/10.1111/exsy.12934.

[3] Q. Abu Al-Haija and M. Al-Dala'ien. ELBA-IoT: an ensemble learning model for botnet attack detection in IoT networks. *Journal of Sensor and Actuator Networks*, 11(1):18, 2022. https://doi.org/10.3390/jsan11010018.

[4] S. Adeleye, C. Obruche, R. Idama, B. Iyamah, K. Oladele, and E. Aworonye. Security concept in internet of things using supervised & unsupervised machine learning algorithm. *GSJ*, 10(2), 2022. https://www.globalscientificjournal.com/researchpaper/SECURITY_CONCEPT_IN_INTERNET_OF_THINGS_USING_SUPERVISED_UNSUPERVISED_MACHINE_LEARNING_ALGORITHM.pdf.

[5] M. Ahmad, Q. Riaz, M. Zeeshan, H. Tahir, S. A. Haider, and M. S. Khan. Intrusion detection in internet of things using supervised machine learning based on application and transport layer features using unsw-nb15 data-set. *EURASIP Journal on Wireless Communications and Networking*, 2021(1):1–23, 2021. https://doi.org/10.1186/s13638-021-01893-8.

[6] M. S. Ahmad and S. M. Shah. Supervised machine learning approaches for attack detection in the IoT network. In *Internet of Things and Its Applications: Select Proceedings of ICIA 2020*, pages 247–260. Springer, 2022. https://doi.org/10.1007/978-981-16-7637-6_22.

[7] M. Ahmed, D. Cox, B. Simpson, and A. Aloufi. ECU-IoFT: A Dataset for Analysing Cyber-Attacks on Internet of Flying Things. *Applied Sciences*, 12(4):1990, 2022. https://doi.org/10.3390/app12041990.

[8] A. Aksoy and M. H. Gunes. Automated IoT device identification using network traffic. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pages 1–7. IEEE, 2019. https://doi.org/10.1109/ICC.2019.8761559.

[9] M. A. Al-Garadi, A. Mohamed, A. Al-Ali, X. Du, I. Ali, and M. Guizani. A survey of machine and deep learning methods for internet of things (IoT) security. *IEEE Communications Surveys & Tutorials*, 2020. https://doi.org/10.1109/COMST.2020.2988293.

[10] K. Albulayhi, Q. Abu Al-Haija, S. A. Alsuhibany, A. A. Jillepalli, M. Ashrafuzzaman, and F. T. Sheldon. IoT intrusion detection using machine learning with a novel high performing feature selection method. *Applied Sciences*, 12(10):5015, 2022. https://doi.org/10.3390/app12105015.

[11] O. Alrawi, C. Lever, M. Antonakakis, and F. Monrose. Sok: Security evaluation of home-based IoT deployments. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1362–1380. IEEE, 2019. https://doi.org/10.1109/SP.2019.00013.

[12] A. A. Alsulami, Q. Abu Al-Haija, A. Tayeb, and A. Alqahtani. An intrusion detection and classification system for IoT traffic with improved data engineering. *Applied Sciences*, 12(23):12336, 2022. https://doi.org/10.3390/app122312336.

[13] A. I. Alzahrani, A. Al-Rasheed, A. Ksibi, M. Ayadi, M. M. Asiri, and M. Zakariah. Anomaly detection in fog computing architectures using custom tab transformer for internet of things. *Electronics*, 11(23):4017, 2022. https://doi.org/10.3390/electronics11234017.

[14] P. Amangele, M. J. Reed, M. Al-Naday, N. Thomos, and M. Nowak. Hierarchical Machine Learning for IoT Anomaly Detection in SDN. *2019 International Conference on Information Technologies, InfoTech 2019 - Proceedings*, 2019. https://doi.org/10.1109/InfoTech.2019.8860878.

[15] N. Ammar, L. Noirie, and S. Tixeuil. Autonomous IoT device identification prototype. In *2019 Network Traffic Measurement and Analysis Conference (TMA)*, pages 195–196. IEEE, 2019. https://doi.org/10.23919/TMA.2019.8784517.

[16] N. Ammar, L. Noirie, and S. Tixeuil. Autonomous identification of IoT device types based on a supervised classification. In *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2020. https://doi.org/10.1109/ICC40277.2020.9148821.

[17] H. S. Anderson and P. Roth. EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models. *ArXiv e-prints*, Apr. 2018. http://adsabs.harvard.edu/abs/2018arXiv180404637A.

[18] E. Anthi, L. Williams, A. Javed, and P. Burnap. Hardening machine learning denial of service (DoS) defences against adversarial attacks in IoT smart home networks. *computers & security*, 108:102352, 2021. https://doi.org/10.1016/j.cose.2021.102352.

[19] E. Anthi, L. Williams, M. Słowińska, G. Theodorakopoulos, and P. Burnap. A supervised intrusion detection system for smart home IoT devices. *IEEE Internet of Things Journal*, 6(5):9042–9053, 2019. https://doi.org/10.1109/JIOT.2019.2926365.

[20] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, et al. Understanding the mirai botnet. In *26th USENIX security symposium (USENIX Security 17)*, pages 1093–1110, 2017. https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis.

[21] N. Aqil, F. Afifi, F. Zaki, and N. B. Anuar. Preliminary study on the effect of traffic representation on accuracy degradation in machine learning-based IoT device identification. In *2022 IEEE International Conference on Computing (ICOCO)*, pages 270–275. IEEE, 2022. https://doi.org/10.1109/ICOCO56118.2022.10031725.

[22] D. Arp, E. Quiring, F. Pendlebury, A. Warnecke, F. Pierazzi, C. Wressnegger, L. Cavallaro, and K. Rieck. Dos and don'ts of machine learning in computer security. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 3971–3988, 2022. https://www.usenix.org/system/files/sec22-arp.pdf.

[23] F. Aubet and M. Pahl. DS2OS traffic traces, 2018. Accessed: 2023-09-15, Available at https://www.kaggle.com/datasets/francoisxa/ds2ostraffictraces.

[24] L. Babun, H. Aksu, L. Ryan, K. Akkaya, E. S. Bentley, and A. S. Uluagac. Z-IoT: Passive Device-class Fingerprinting of ZigBee and Z-Wave IoT Devices. In *International Conference on Communications (ICC)*, pages 1–7. IEEE, 2020. https://doi.org/10.1109/ICC40277.2020.9149285.

[25] M. Bagaa, T. Taleb, J. B. Bernabe, and A. Skarmeta. A Machine Learning Security Framework for IoT Systems. *IEEE Access*, 8:114066–114077, 2020. https://doi.org/10.1109/ACCESS.2020.2996214.

[26] L. Bai, L. Yao, S. S. Kanhere, X. Wang, and Z. Yang. Automatic device classification from network traffic streams of internet of things. In *2018 IEEE 43rd conference on local computer networks (LCN)*, pages 1–9. IEEE, 2018. https://doi.org/10.1109/LCN.2018.8638232.

[27] S. Baniasadi, O. Rostami, D. Martín, and M. Kaveh. A novel deep supervised learning-based approach for intrusion detection in IoT systems. *Sensors*, 22(12):4459, 2022. https://doi.org/10.3390/s22124459.

[28] J. Bao, B. Hamdaoui, and W.-K. Wong. IoT device type identification using hybrid deep learning approach for increased IoT security. In *2020 International Wireless Communications and Mobile Computing (IWCMC)*, pages 565–570. IEEE, 2020. https://doi.org/10.1109/IWCMC48107.2020.9148110.

[29] A. Basharat, M. M. B. Mohamad, and A. Khan. Machine learning techniques for intrusion detection in smart healthcare systems: A comparative analysis. In *2022 4th International Conference on Smart Sensors and Application (ICSSA)*, pages 29–33. IEEE, 2022. https://doi.org/10.1109/ICSSA54161.2022.9870973.

[30] B. Bezawada, M. Bachani, J. Peterson, H. Shirazi, I. Ray, and I. Ray. Behavioral fingerprinting of IoT devices. In *Proceedings of the 2018 Workshop on Attacks and Solutions in Hardware Security*, pages 41–50, 2018. https://doi.org/10.1145/3266444.3266452.

[31] B. A. Bhuvaneswari and S. S. Anomaly detection framework for Internet of things traffic using vector convolutional deep learning approach in fog environment. *Future Generation Computer Systems*, 113:255–265, 2020. https://doi.org/10.1016/j.future.2020.07.020.

[32] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita. *Network Traffic Anomaly Detection and Prevention*. Springer International Publishing, 2017. https://doi.org/10.1007/978-3-319-65188-0.

[33] V. Bonandrini, J. F. Bercher, and N. Zangar. Machine Learning Methods for Anomaly Detection in IoT Networks, with Illustrations. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12081 LNCS:287–295, 2020. https://doi.org/10.1007/978-3-030-45778-5_19.

[34] E. Breece. Glossary of information security terms and definitions: Enterprise security office (ESO) guideline version 1.01. *State of Minnesota, Enterprise Office of Technology (OET). Retrieved February*, 2:2013, 2011. https://mn.gov/mnit/images/SEC_R_Glossary.pdf.

[35] A. Butterfield, G. E. Ngondi, and A. Kerr. *A Dictionary of Computer Science*. Oxford University Press, 7 edition, 2016. https://doi.org/10.1093/acref/9780199688975.001.0001.

[36] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki. Network intrusion detection for IoT security based on learning techniques. *IEEE Communications Surveys & Tutorials*, 21(3):2671–2701, 2019. https://doi.org/10.1109/COMST.2019.2896380.

[37] B. Charyyev and M. H. Gunes. IoT traffic flow identification using locality sensitive hashes. In *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2020. https://doi.org/10.1109/ICC40277.2020.9148743.

[38] Q. Chen, Y. Song, B. Jennings, F. Zhang, B. Xiao, and S. Gao. IoT-ID: robust IoT device identification based on feature drift adaptation. In *2021 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2021. https://doi.org/10.1109/GLOBECOM46510.2021.9685693.

[39] Y. Chen, Q. Lin, W. Wei, J. Ji, K.-C. Wong, and C. A. C. Coello. Intrusion detection using multi-objective evolutionary convolutional neural network for internet of things in fog computing. *Knowledge-Based Systems*, 244:108505, 2022. https://doi.org/10.1016/j.knosys.2022.108505.

[40] A. Choudhary. Genetic algorithm in machine learning using python, 2020. Accessed: 2023-03-16, Available at https://datascienceplus.com/genetic-algorithm-in-machine-learning-using-python/.

[41] R. R. Chowdhury, S. Aneja, N. Aneja, and E. Abas. Network traffic analysis based IoT device identification. In *Proceedings of the 2020 4th International Conference on Big Data and Internet of Things*, pages 79–89, 2020. https://doi.org/10.1145/3421537.3421545.

[42] R. R. Chowdhury, A. C. Idris, and P. E. Abas. Device identification using optimized digital footprints. *arXiv preprint arXiv:2212.04354*, 2022. https://doi.org/10.48550/arXiv.2212.04354.

[43] R. R. Chowdhury, A. C. Idris, and P. E. ABAS. Packet-level and IEEE 802.11 mac frame-level analysis for IoT device identification. *Turkish Journal of Electrical Engineering and Computer Sciences*, 30(5):1941–1961, 2022. https://doi.org/10.55730/1300-0632.3915.

[44] A. Churcher, R. Ullah, J. Ahmad, S. Ur Rehman, F. Masood, M. Gogate, F. Alqahtani, B. Nour, and W. J. Buchanan. An experimental analysis of attack classification using machine learning in IoT networks. *Sensors*, 21(2):446, 2021. https://doi.org/10.3390/s21020446.

[45] T. M. Corporation. CVE - Common vulnerabilities and exposures. https://cve.mitre.org/data/downloads/index.html.

[46] R.-A. Crăciun, R. N. Pietraru, and M. A. Moisescu. IoT device identification: A machine learning assessment. In *2023 13th International Symposium on Advanced Topics in Electrical Engineering (ATEE)*, pages 1–6. IEEE, 2023. https://doi.org/10.1109/ATEE58038.2023.10108170.

[47] I. Cvitić, D. Peraković, M. Periša, and B. Gupta. Ensemble machine learning approach for classification of IoT devices in smart home. *International Journal of Machine Learning and Cybernetics*, 12(11):3179–3202, 2021. https://doi.org/10.1007/s13042-020-01241-0.

[48] S. Dadkhah, H. Mahdikhani, P. K. Danso, A. Zohourian, K. A. Truong, and A. A. Ghorbani. Towards the development of a realistic multidimensional IoT profiling dataset. In *2022 19th Annual International Conference on Privacy, Security*

& Trust (PST), pages 1–11, 2022. https://doi.org/10.1109/PST55820.2022.9851966.

[49] Q.-V. Dang. Using machine learning for intrusion detection systems. *Computing and Informatics*, 41(1):12–33, 2022. https://doi.org/10.31577/cai_2022_1_12.

[50] H. X. Dau, N. T. T. Trang, and N. T. Hung. A survey of tools and techniques for web attack detection. *Journal of Science and Technology on Information security*, 1(15):109–118, 2022. https://doi.org/10.54654/isj.v1i15.852.

[51] G. De Carvalho Bertoli, L. A. Pereira Junior, O. Saotome, A. L. Dos Santos, F. A. N. Verri, C. A. C. Marcondes, S. Barbieri, M. S. Rodrigues, and J. M. Parente De Oliveira. An End-to-End Framework for Machine Learning-Based Network Intrusion Detection System. *IEEE Access*, 9:106790–106805, 2021. https://doi.org/10.1109/ACCESS.2021.3101188.

[52] M. J. De Lucia, P. E. Maxwell, N. D. Bastian, A. Swami, B. Jalaian, and N. Leslie. Machine learning raw network traffic detection. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications III*, volume 11746, pages 185–194. SPIE, 2021. https://doi.org/10.1117/12.2586114.

[53] B. A. Desai, D. M. Divakaran, I. Nevat, G. W. Peter, and M. Gurusamy. A feature-ranking framework for IoT device classification. In *2019 11th International conference on communication systems & networks (COMSNETS)*, pages 64–71. IEEE, 2019. https://doi.org/10.1109/COMSNETS.2019.8711210.

[54] M. G. Desai, Y. Shi, and K. Suo. IoT Bonet and Network Intrusion Detection using Dimensionality Reduction and Supervised Machine Learning. *2020 11th IEEE Annual Ubiquitous Computing, Electronics and Mobile Communication Conference, UEMCON 2020*, pages 0316–0322, 2020. https://doi.org/10.1109/UEMCON51285.2020.9298146.

[55] K. S. Desale and R. Ade. Genetic algorithm based feature selection approach for effective intrusion detection system. In *2015 International Conference on Computer Communication and Informatics (ICCCI)*, pages 1–6. IEEE, 2015. https://doi.org/10.1109/ICCCI.2015.7218109.

[56] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. https://doi.org/10.48550/arXiv.1810.04805.

[57] S. Dong, Z. Li, D. Tang, J. Chen, M. Sun, and K. Zhang. Your smart home can't keep a secret: Towards automated fingerprinting of IoT traffic. In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*, pages 47–59, 2020. https://doi.org/10.1145/3320269.3384732.

[58] R. Du, J. Wang, and S. Li. A lightweight flow feature-based IoT device identification scheme. *Security and Communication Networks*, 2022:1–10, 2022. https://doi.org/10.1155/2022/8486080.

[59] C. Duan, H. Gao, G. Song, J. Yang, and Z. Wang. ByteIoT: A practical IoT device identification system based on packet length distribution. *IEEE Transactions on Network and Service Management*, 19(2):1717–1728, 2021. https://doi.org/10.1109/TNSM.2021.3130312.

[60] G. El Baltaji. *Anomaly Detection at the Edge implementing Machine Learning Techniques*. PhD thesis, Politecnico di Torino, 2022. https://webthesis.biblio.polito.it/21937/.

[61] M. S. Elsayed, N.-A. Le-Khac, and A. D. Jurcut. InSDN: A novel SDN intrusion dataset. *IEEE Access*, 8:165263–165284, 2020. https://doi.org/10.1109/ACCESS.2020.3022633.

[62] M. Esmalifalak, L. Liu, N. Nguyen, R. Zheng, and Z. Han. Detecting stealthy false data injection using machine learning in smart grid. *IEEE Systems Journal*, 11(3):1644–1652, 2014. https://doi.org/10.1109/JSYST.2014.2341597.

[63] K. R. Fall and W. R. Stevens. *TCP/IP illustrated, volume 1: The protocols*. addison-Wesley, 2011.

[64] L. Fan, L. He, Y. Wu, S. Zhang, Z. Wang, J. Li, J. Yang, C. Xiang, and X. Ma. AutoIoT: Automatically updated IoT device identification with semi-supervised learning. *IEEE Transactions on Mobile Computing*, 2022. https://doi.org/10.1109/TMC.2022.3183118.

[65] L. Fan, S. Zhang, Y. Wu, Z. Wang, C. Duan, J. Li, and J. Yang. An IoT device identification method based on semi-supervised learning. In *2020 16th International Conference on Network and Service Management (CNSM)*, pages 1–7. IEEE, 2020. https://doi.org/10.23919/CNSM50824.2020.9269044.

[66] J. A. Faysal, S. T. Mostafa, J. S. Tamanna, K. M. Mumenin, M. M. Arifin, M. A. Awal, A. Shome, and S. S. Mostafa. XGB-RF: A hybrid machine learning approach for IoT intrusion detection. In *Telecom*, volume 3, pages 52–69. MDPI, 2022. https://doi.org/10.3390/telecom3010003.

[67] V. A. Ferman and M. A. Tawfeeq. Machine learning challenges for IoT device fingerprints identification. In *Journal of Physics: Conference Series*, volume 1963, page 012046. IOP Publishing, 2021. https://doi.org/10.1088/1742-6596/1963/1/012046.

[68] M. A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, and H. Janicke. Edge-IIoTset: A new comprehensive realistic cyber security dataset of IoT and IIoT applications for centralized and federated learning. *IEEE Access*, 10:40281–40306, 2022. https://doi.org/10.1109/ACCESS.2022.3165809.

[69] O. Friha, M. A. Ferrag, L. Shu, L. Maglaras, K.-K. R. Choo, and M. Nafaa. Felids: Federated learning-based intrusion detection system for agricultural internet of things. *Journal of Parallel and Distributed Computing*, 165:17–31, 2022. https://doi.org/10.1016/j.jpdc.2022.03.003.

[70] S. Garcia, A. Parmisano, and M. Erquiaga, Jose. IoT-23: A labeled dataset with malicious and benign IoT network traffic (version 1.0.0), 2020. Accessed: 2023-09-15, https://www.stratosphereips.org/datasets-iot23.

[71] U. Garg, H. Sivaraman, A. Bamola, and P. Kumari. To evaluate and analyze the performance of anomaly detection in cloud of things. In *2022 13th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pages 1–7. IEEE, 2022. https://doi.org/10.1109/ICCCNT54827.2022.9984316.

[72] S. Gash. The international dictionary of artificial intelligence. *Reference Reviews*, 14(6):28–29, 2000. https://doi.org/10.1108/rr.2000.14.6.28.295.

[73] V. Gaur and R. Kumar. Analysis of machine learning classifiers for early detection of DDoS attacks on IoT devices. *Arabian Journal for Science and Engineering*, 47(2):1353–1374, 2022. https://doi.org/10.1007/s13369-021-05947-3.

[74] A. Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2019.

[75] A. Ghourabi. A security model based on lightgbm and transformer to protect healthcare systems from cyberattacks. *IEEE Access*, 10:48890–48903, 2022. https://doi.org/10.1109/ACCESS.2022.3172432.

[76] S. Gollapudi. *Practical machine learning*. Packt Publishing Ltd, 2016.

[77] P. Gulihar and B. B. Gupta. Cooperative mechanisms for defending distributed denial of service (DDoS) attacks. In *Handbook of Computer Networks and Cyber Security*, pages 421–443. Springer, 2020. https://doi.org/10.1007/978-3-030-22277-2_16.

[78] B. Gupta, P. Chaudhary, X. Chang, and N. Nedjah. Smart defense against distributed denial of service attack in IoT networks using supervised learning classifiers. *Computers & Electrical Engineering*, 98:107726, 2022. https://doi.org/10.1016/j.compeleceng.2022.107726.

[79] B. B. Gupta, G. M. Perez, D. P. Agrawal, and D. Gupta. *Handbook of computer networks and cyber security*. Springer, 2020.

[80] S. A. Hamad, W. E. Zhang, Q. Z. Sheng, and S. Nepal. IoT device identification via network-flow based fingerprinting and learning. In *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pages 103–111. IEEE, 2019. https://doi.org/10.1109/TrustCom/BigDataSE.2019.00023.

[81] A. Hamza, H. H. Gharakheili, T. A. Benson, and V. Sivaraman. Detecting volumetric attacks on loT devices via SDN-based monitoring of MUD activity. In *Proceedings of the 2019 ACM Symposium on SDN Research*, pages 36–48, 2019. https://doi.org/10.1145/3314148.3314352.

[82] S. Han, Q. Wu, and Y. Yang. Machine learning for internet of things anomaly detection under low-quality data. *International Journal of Distributed Sensor Networks*, 18(10):15501329221133765, 2022. https://doi.org/10.1177/1550132922113376.

[83] X. Han, S. Cui, S. Liu, C. Zhang, B. Jiang, and Z. Lu. Network intrusion detection based on n-gram frequency and time-aware transformer. *Computers & Security*, 128:103171, 2023. https://doi.org/10.1016/j.cose.2023.103171.

[84] S. Hanif, T. Ilyas, and M. Zeeshan. Intrusion Detection in IoT Using Artificial Neural Networks on UNSW-15 Dataset. *HONET-ICT 2019 - IEEE 16th International Conference on Smart Cities: Improving Quality of Life using ICT, IoT and AI*, pages 152–156, 2019. https://doi.org/10.1109/HONET.2019.8908122.

[85] H. He and E. A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009. https://doi.org/10.1109/TKDE.2008.239.

[86] Z. He, J. Yin, Y. Wang, G. Gui, B. Adebisi, T. Ohtsuki, H. Gacanin, and H. Sari. Edge device identification based on federated learning and network traffic feature engineering. *IEEE Transactions on Cognitive Communications and Networking*, 8(4):1898–1909, 2021. https://doi.org/10.1109/TCCN.2021.3101239.

[87] Hewlett-Packard-Enterprise. Single-packet attacks. In *HPE FlexNetwork 7500 Switch Series Security Configuration Guide*. Hewlett Packard Enterprise Development LP, 2017. Accessed: 2023-09-17, Available at https://techhub.hpe.com/eginfolib/networking/docs/switches/7500/5200-1952a_security_cg/content/495505992.htm.

[88] H. Hindy, C. Tachtatzis, R. Atkinson, E. Bayne, and X. Bellekens. MQTT-IoT-IDS2020: MQTT internet of things intrusion detection dataset, 2020. Accessed: 2023-09-15, https://dx.doi.org/10.21227/bhxy-ep04.

[89] T. B. Hoang, L. Vu, and Q. U. Nguyen. A data sampling and two-stage convolution neural network for IoT devices identification. In *2022 RIVF International Conference on Computing and Communication Technologies (RIVF)*, pages 458–463. IEEE, 2022. https://doi.org/10.1109/RIVF55975.2022.10013866.

[90] X. Hu, H. Li, Z. Shi, N. Yu, H. Zhu, and L. Sun. A robust IoT device identification method with unknown traffic detection. In *Wireless Algorithms, Systems, and Applications: 16th International Conference, WASA 2021, Nanjing, China, June 25–27, 2021, Proceedings, Part I 16*, pages 190–202. Springer, 2021. https://doi.org/10.1007/978-3-030-85928-2_15.

[91] F. Hussain, S. G. Abbas, M. Husnain, U. U. Fayyaz, F. Shahzad, and G. A. Shah. IoT DoS and DDoS Attack Detection using ResNet. In *2020 IEEE 23rd International Multitopic Conference (INMIC)*, pages 1–6, 2020. https://doi.org/10.1109/INMIC50486.2020.9318216.

[92] F. Hussain, R. Hussain, S. A. Hassan, and E. Hossain. Machine learning in IoT security: Current solutions and future challenges. *IEEE Communications Surveys & Tutorials*, 22(3):1686–1721, 2020. https://doi.org/10.1109/COMST.2020.2986444.

[93] K. K. Huy. IoT environment dataset. Accessed: 2023-09-15, https://ocslab.hksecurity.net/Datasets/iot-environment-dataset.

[94] R.-H. Hwang, M.-C. Peng, V.-L. Nguyen, and Y.-L. Chang. An LSTM-based deep learning approach for classifying malicious traffic at the packet level. *Applied Sciences*, 9(16):3414, 2019. https://doi.org/10.3390/app9163414.

[95] K. Ibrahimi and H. Benaddi. Improving the IDS for BoT-IoT Dataset-Based Machine Learning Classifiers. In *2022 5th International Conference on Advanced Communication Technologies and Networking (CommNet)*, pages 1–6. IEEE, 2022. https://doi.org/10.1109/CommNet56067.2022.9993869.

[96] F. B. Insights. Internet of things market size, growth — IoT industry report 2026, 2019. Accessed: 2020-04-07, Available at https://www.fortunebusinessinsights.com/industry-reports/internet-of-things-iot-market-100307/.

[97] S. U. Jan, S. Ahmed, V. Shakhov, and I. Koo. Toward a Lightweight Intrusion Detection System for the Internet of Things. *IEEE Access*, 7:42450–42471, 2019. https://doi.org/10.1109/ACCESS.2019.2907965.

[98] R. Jiao, Z. Liu, L. Liu, C. Ge, and G. Hancke. Multi-level IoT device identification. In *2021 IEEE 27th International Conference on Parallel and Distributed*

Systems (ICPADS), pages 538–547. IEEE, 2021. https://doi.org/10.1109/ICPADS53394.2021.00073.

[99] Y. Jin. IoT remote access attack. Accessed: 2023-03-16, Available at http://cyberforensic.net/labs/remote.html.

[100] A. Jović, K. Brkić, and N. Bogunović. A review of feature selection methods with applications. In *2015 38th international convention on information and communication technology, electronics and microelectronics (MIPRO)*, pages 1200–1205. IEEE, 2015. https://doi.org/10.1109/MIPRO.2015.7160458.

[101] L. E. Juarez-Orozco, O. Martinez-Manzanera, S. V. Nesterov, S. Kajander, and J. Knuuti. The machine learning horizon in cardiac hybrid imaging. *European Journal of Hybrid Imaging*, 2(1):15, 2018. https://doi.org/10.1186/s41824-018-0033-3.

[102] Juniper-Netorks. Mac randomization and location analytics, Sep 2023. https://www.juniper.net/documentation/us/en/software/mist/location-services/topics/concept/mac-randomization.html.

[103] V. Jyothsna, R. Prasad, and K. M. Prasad. A review of anomaly based intrusion detection systems. *International Journal of Computer Applications*, 28(7):26–35, 2011. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=f4179593b4b47d8be8cd97e0caced755e16fefbb.

[104] H. Kang, D. H. Ahn, G. M. Lee, J. D. Yoo, K. H. Park, and H. K. Kim. IoT network intrusion dataset. Accessed: 2023-09-15, https://dx.doi.org/10.21227/q70p-q449.

[105] S. Kapoor, E. Cantrell, K. Peng, T. H. Pham, C. A. Bail, O. E. Gundersen, J. M. Hofman, J. Hullman, M. A. Lones, M. M. Malik, et al. Reforms: Reporting standards for machine learning based science. *arXiv preprint arXiv:2308.07832*, 2023. https://doi.org/10.48550/arXiv.2308.07832.

[106] S. Kapoor and A. Narayanan. Leakage and the reproducibility crisis in machine-learning-based science. *Patterns*, 2023. https://doi.org/10.1016/j.patter.2023.100804.

[107] N. Karmous, M. O.-E. Aoueileyine, M. Abdelkader, and N. Youssef. A proposed intrusion detection method based on machine learning used for internet of things systems. In *Advanced Information Networking and Applications: Proceedings of the 36th International Conference on Advanced Information Networking and Applications (AINA-2022), Volume 3*, pages 33–45. Springer, 2022. https://doi.org/10.1007/978-3-030-99619-2_4.

[108] H. Kawai, S. Ata, N. Nakamura, and I. Oka. Identification of communication devices from analysis of traffic patterns. In *2017 13th International Conference on Network and Service Management (CNSM)*, pages 1–5. IEEE, 2017. https://doi.org/10.23919/CNSM.2017.8256018.

[109] J. Y. Khan and M. R. Yuce. *Internet of Things (IoT): systems and applications*. CRC Press, 2019.

[110] D. Koike, S. Ishida, and Y. Arakawa. Called function identification of IoT devices by network traffic analysis. In *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, pages 737–743, 2021. https://doi.org/10.1145/3412841.3441951.

[111] R. Kolcun, D. A. Popescu, V. Safronov, P. Yadav, A. M. Mandalari, Y. Xie, R. Mortier, and H. Haddadi. The case for retraining of ml models for IoT device identification at the edge. *arXiv preprint arXiv:2011.08605*, 2020. https://doi.org/10.48550/arXiv.2107.07818.

[112] C. Kolias, G. Kambourakis, A. Stavrou, and S. Gritzalis. Intrusion detection in 802.11 networks: Empirical evaluation of threats and a public dataset. *IEEE Communications Surveys & Tutorials*, 18(1):184–208, 2015. https://doi.org/10.1109/COMST.2015.2402161.

[113] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-IoT dataset. *Future Generation Computer Systems*, 100:779–796, 2019. https://doi.org/10.1016/j.future.2019.05.041.

[114] K. Kostas. CNN based IoT Device Identification. *arXiv preprint arXiv:2304.13894*, 2023. https://doi.org/10.48550/arXiv.2304.13894.

[115] K. Kostas. LSTM based IoT Device Identification. *arXiv preprint arXiv:2304.13905*, 2023. https://doi.org/10.48550/arXiv.2304.13905.

[116] K. Kostas, M. Just, and M. Lones. IoTDevID: A Behaviour-Based Fingerprinting Method for Device Identification in the IoT. *arXiv preprint arXiv:2102.08866v1*, 2021. https://arxiv.org/pdf/2102.08866v1.pdf.

[117] K. Kostas, M. Just, and M. A. Lones. IoTDevID: A Behavior-Based Device Identification Method for the IoT. *IEEE Internet of Things Journal*, 9(23):23741–23749, 2022. https://doi.org/10.1109/JIOT.2022.3191951.

[118] K. Kostas, M. Just, and M. A. Lones. Iotdevid: A behavior-based device identification method for the iot. *IEEE IoT Journal*, 9(23):23741–23749, 2022. https://doi.org/10.1109/JIOT.2022.3191951.

[119] K. Kostas, R. Y. Kostas, F. Zampella, and F. Alsehly. Wifi based distance estimation using supervised machine learning. In *2022 IEEE 12th International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–8, 2022. https://doi.org/10.1109/IPIN54987.2022.9918128.

[120] J. Kotak and Y. Elovici. IoT device identification using deep learning. In *13th International Conference on Computational Intelligence in Security for Information Systems (CISIS 2020) 12*, pages 76–86. Springer, 2021. https://doi.org/10.1007/978-3-030-57805-3_8.

[121] J. Kotak and Y. Elovici. Adversarial attacks against IoT identification systems. *IEEE Internet of Things Journal*, 2022. https://doi.org/10.1109/JIOT.2022.3229906.

[122] J. Kotak and Y. Elovici. IoT device identification based on network communication analysis using deep learning. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–17, 2022. https://doi.org/10.1007/s12652-022-04415-6.

[123] D. E. Kouicem, A. Bouabdallah, and H. Lakhlef. Internet of things security: A top-down survey. *Computer Networks*, 141:199–221, 2018. https://doi.org/10.1016/j.comnet.2018.03.012.

[124] T. Kovanen, G. David, and T. Hämäläinen. Survey: Intrusion detection systems in encrypted traffic. In *Internet of Things, Smart Spaces, and Next Generation Networks and Systems*, pages 281–293. Springer, 2016. https://doi.org/10.1007/978-3-319-46301-8_23.

[125] C. M. Kozierok. *The TCP/IP guide: a comprehensive, illustrated Internet protocols reference*. No Starch Press, 2005.

[126] K. R. Kumar, C. Hemanth, C. A. Kumar, K. Sahith, and G. A. Prasanth. IoT device identification through network traffic analysis. *Int. Res. J. Modern. Eng. Technol. Sci*, 2(06), 2020. https://www.irjmets.com/uploadedfiles/paper/volume2/issue_6_june_2020/1702/1628083052.pdf.

[127] J. Kurmi and R. Matam. Device identification in IoT networks using network trace fingerprinting. In *2022 International Conference on Smart Applications, Communications and Networking (SmartNets)*, pages 1–6. IEEE, 2022. https://doi.org/10.1109/SmartNets55823.2022.9994003.

[128] A. H. Lashkari, G. Draper-Gil, M. S. I. Mamun, A. A. Ghorbani, et al. Characterization of tor traffic using time based features. In *ICISSp*, pages 253–262, 2017. https://doi.org/10.5220/0006105602530262.

[129] F. Le, J. Ortiz, D. Verma, and D. Kandlur. Policy-based identification of IoT devices' vendor and type by DNS traffic analysis. *Policy-Based Autonomic Data Governance*, pages 180–201, 2019. https://doi.org/10.1007/978-3-030-17277-0_10.

[130] T.-T.-H. Le, H. Kim, H. Kang, and H. Kim. Classification and explanation for intrusion detection system based on ensemble trees and shap method. *Sensors*, 22(3):1154, 2022. https://doi.org/10.3390/s22031154.

[131] M. Leon, T. Markovic, and S. Punnekkat. Comparative evaluation of machine learning algorithms for network intrusion detection and attack classification. In *2022 international joint conference on neural networks (IJCNN)*, pages 01–08. IEEE, 2022. https://doi.org/10.1109/IJCNN55064.2022.9892293.

[132] S. Liu, X. Xu, Y. Zhang, and Y. Wang. Autonomous anti-interference identification of IoT device traffic based on convolutional neural network. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2022. https://doi.org/10.1109/IJCNN55064.2022.9891943.

[133] X. Liu, Y. Han, and Y. Du. IoT device identification using directional packet length sequences and 1D-CNN. *Sensors*, 22(21):8337, 2022. https://doi.org/10.3390/s22218337.

[134] X. Liu, Z. Tang, and B. Yang. Predicting network attacks with CNN by constructing images from netflow data. In *2019 IEEE 5th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*, pages 61–66. IEEE, 2019. https://doi.org/10.1109/BigDataSecurity-HPSC-IDS.2019.00022.

[135] Q. LLC. Argus. *openargus*, 2020. https://openargus.org.

[136] M. A. Lones. How to avoid machine learning pitfalls: a guide for academic researchers. *arXiv preprint arXiv:2108.02497*, 2021. https://doi.org/10.48550/arXiv.2108.02497.

[137] A. Lopes, L. Reis, H. São Mamede, and A. Santos. Information security threat assessment using social engineering in the organizational context–literature review. In *World Conference on Information Systems and Technologies*, pages 233–242. Springer, 2022. https://doi.org/10.1007/978-3-031-04819-7_24.

[138] A. C. Lorena, L. P. Garcia, J. Lehmann, M. C. Souto, and T. K. Ho. How complex is your classification problem? a survey on measuring classification complexity. *ACM Computing Surveys (CSUR)*, 52(5):1–34, 2019. https://doi.org/10.1145/3347711.

[139] S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S.-I. Lee. From local explanations to global understanding with explainable AI for trees. *Nature machine intelligence*, 2(1):56–67, 2020. https://doi.org/10.1038/s42256-019-0138-9.

[140] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017. https://proceedings.neurips.cc/paper_files/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf.

[141] Y. Luo, X. Chen, N. Ge, W. Feng, and J. Lu. Transformer-based device type identification in heterogeneous IoT traffic. *IEEE Internet of Things Journal*, 2022. https://doi.org/10.1109/JIOT.2022.3221967.

[142] J. Ma, Y. Sang, Y. Zhang, X. Xu, B. Feng, and Y. Zeng. An adaptive ensembled neural network-based approach to IoT device identification. In *Collaborative*

Computing: Networking, Applications and Worksharing: 18th EAI International Conference, CollaborateCom 2022, Hangzhou, China, October 15-16, 2022, Proceedings, Part II, pages 214–230. Springer, 2023. https://doi.org/10.1007/978-3-031-24386-8_12.

[143] M. Mainuddin, Z. Duan, Y. Dong, S. Salman, and T. Taami. IoT device identification based on network traffic characteristics. In *GLOBECOM 2022-2022 IEEE Global Communications Conference*, pages 6067–6072. IEEE, 2022. https://doi.org/10.1109/GLOBECOM48099.2022.10001639.

[144] I. Makhdoom, M. Abolhasan, J. Lipman, R. P. Liu, and W. Ni. Anatomy of threats to the internet of things. *IEEE Communications Surveys & Tutorials*, 21(2):1636–1675, 2018. https://doi.org/10.1109/COMST.2018.2874978.

[145] J. Manhas and S. Kotwal. *Implementation of Intrusion Detection System for Internet of Things Using Machine Learning Techniques*, pages 217–237. Springer, Singapore, 2021. https://doi.org/10.1007/978-981-15-8711-5_11.

[146] P. Maniriho, E. Niyigaba, Z. Bizimana, V. Twiringiyimana, L. J. Mahoro, and T. Ahmad. Anomaly-based intrusion detection approach for iot networks using machine learning. In *2020 International Conference on Computer Engineering, Network, and Intelligent Multimedia (CENIM)*, pages 303–308, 2020. https://doi.org/10.1109/CENIM51130.2020.9297958.

[147] R. Manzano, N. Goel, M. Zaman, R. Joshi, and K. Naik. Design of a machine learning based intrusion detection framework and methodology for IoT networks. In *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0191–0198. IEEE, 2022. https://doi.org/10.1109/CCWC54503.2022.9720857.

[148] S. Marchal. IoT devices captures, aalto university, 2017. Accessed: 2023-06-09, https://research.aalto.fi/en/datasets/iot-devices-capture.

[149] S. Marchal, M. Miettinen, T. D. Nguyen, A.-R. Sadeghi, and N. Asokan. Audi: Toward autonomous IoT device-type identification using periodic communication. *IEEE Journal on Selected Areas in Communications*, 37(6):1402–1412, 2019. https://doi.org/10.1109/JSAC.2019.2904364.

[150] G. Marín, P. Casas, and G. Capdehourat. Rawpower: Deep learning based anomaly detection from raw network traffic measurements. In *Proceedings of the ACM SIGCOMM*, pages 75–77, 2018. https://doi.org/10.1145/3234200.3234238.

[151] A. Marzano, D. Alexander, O. Fonseca, E. Fazzion, C. Hoepers, K. Steding-Jessen, M. H. Chaves, Í. Cunha, D. Guedes, and W. Meira. The evolution of bashlite and mirai IoT botnets. In *2018 IEEE Symposium on Computers and Communications (ISCC)*, pages 00813–00818. IEEE, 2018. https://doi.org/10.1109/ISCC.2018.8538636.

[152] R. Maxion. Reproducibility: Buy low, sell high. *IEEE Security & Privacy*, 18(6):33–41, 2020. https://doi.org/10.1109/MSEC.2020.3005077.

[153] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici. N-BaIoT—network-based detection of IoT botnet attacks using deep autoencoders. *IEEE Pervasive Computing*, 17(3):12–22, 2018. https://doi.org/10.1109/MPRV.2018.03367731.

[154] Y. Meidan, M. Bohadana, A. Shabtai, J. D. Guarnizo, M. Ochoa, N. O. Tippenhauer, and Y. Elovici. ProfilIoT: a machine learning approach for IoT device identification based on network traffic analysis. In *Proceedings of the symposium on applied computing*, pages 506–509, 2017. https://doi.org/10.1145/3019612.3019878.

[155] Y. Meidan, M. Bohadana, A. Shabtai, M. Ochoa, N. O. Tippenhauer, J. D. Guarnizo, and Y. Elovici. Detection of unauthorized IoT devices using machine learning techniques. *arXiv preprint arXiv:1709.04647*, 2017. https://doi.org/10.48550/arXiv.1709.04647.

[156] Y. Meidan, V. Sachidananda, H. Peng, R. Sagron, Y. Elovici, and A. Shabtai. A novel approach for detecting vulnerable IoT devices connected behind a home NAT. *Computers & Security*, 97:101968, 2020. https://doi.org/10.1016/j.cose.2020.101968.

[157] R. V. Mendonça, J. C. Silva, R. L. Rosa, M. Saadi, D. Z. Rodriguez, and A. Farouk. A lightweight intelligent intrusion detection system for industrial internet of things using deep learning algorithms. *Expert Systems*, 39(5):e12917, 2022. https://doi.org/10.1111/exsy.12917.

[158] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A.-R. Sadeghi, and S. Tarkoma. IoT sentinel: Automated device-type identification for security enforcement in iot. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 2177–2184. IEEE, 2017. https://doi.org/10.1109/ICDCS.2017.283.

[159] A. Mihoub, O. B. Fredj, O. Cheikhrouhou, A. Derhab, and M. Krichen. Denial of service attack detection and mitigation for internet of things using looking-back-enabled machine learning techniques. *Computers & Electrical Engineering*, 98:107716, 2022. https://doi.org/10.1016/j.compeleceng.2022.107716.

[160] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai. Kitsune: an ensemble of autoencoders for online network intrusion detection. In *Network and Distributed Systems Security (NDSS) Symposium 2018*, 2018. http://dx.doi.org/10.14722/ndss.2018.23204.

[161] H. Mliki, A. H. Kaceam, and L. Chaari. Intrusion Detection Study and Enhancement Using Machine Learning. *Lecture Notes in Computer Science*, 12026 LNCS:263–278, 2020. https://doi.org/10.1007/978-3-030-41568-6_17.

[162] H. Modi. Dawn of the terrorbit era, Feb 2019. Accessed: 2023-05-16, Available at https://www.netscout.com/blog/dawn-terrorbit-era.

[163] M. I. Mohmand, H. Hussain, A. A. Khan, U. Ullah, M. Zakarya, A. Ahmed, M. Raza, I. U. Rahman, M. Haleem, et al. A machine learning-based classification and prediction technique for DDoS attacks. *IEEE Access*, 10:21443–21454, 2022. https://doi.org/10.1109/ACCESS.2022.3152577.

[164] P. S. Molcer, A. Pejić, K. Gulači, and R. Szalma. Machine learning based network intrusion detection system for internet of things cybersecurity. In *Security-Related Advanced Technologies in Critical Infrastructure Protection: Theoretical and Practical Approach*, pages 95–110. Springer, 2022. https://doi.org/10.1007/978-94-024-2174-3_8.

[165] C. Molnar. *Interpretable machine learning*. Lulu.com, 2020. https://christophm.github.io/interpretable-ml-book/.

[166] B. Mondal and S. K. Singh. A comparative analysis of network intrusion detection system for IoT using machine learning. In *Internet of Things and Its Applications:*

Select Proceedings of ICIA 2020, pages 211–221. Springer, 2022. https://doi.org/10.1007/978-981-16-7637-6_19.

[167] G. A. Morales, A. Bienek-Parrish, P. Jenkins, and R. Slavin. Protocol-agnostic iot device classification on encrypted traffic using link-level flows. CPS-IoT Week '23, page 19–24. Association for Computing Machinery, New York, NY, USA, 2023. https://doi.org/10.1145/3576914.3587487.

[168] B. Moran, H. Tschofenig, and H. Birkholz. A Manifest Information Model for Firmware Updates in Internet of Things (IoT) Devices. RFC 9124, Jan. 2022. https://doi.org/10.17487/RFC9124.

[169] N. Moustafa. A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets. *Sustainable Cities and Society*, 72:102994, 2021. https://doi.org/10.1016/j.scs.2021.102994.

[170] N. Moustafa and J. Slay. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In *2015 military communications and information systems conference (MilCIS)*, pages 1–6. IEEE, 2015. https://doi.org/10.1109/MilCIS.2015.7348942.

[171] N. Moustafa, B. Turnbull, and K.-K. R. Choo. An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things. *IEEE Internet of Things Journal*, 6(3):4815–4830, 2018. https://doi.org/10.1109/JIOT.2018.2871719.

[172] N. Najari, S. Berlemont, G. Lefebvre, S. Duffner, and C. Garcia. Network traffic modeling for iot-device re-identification. In *2020 International Conference on Omni-layer Intelligent Systems (COINS)*, pages 1–6. IEEE, 2020. https://doi.org/10.1109/COINS49042.2020.9191376.

[173] A. Namvar, C. Thapa, S. S. Kanhere, and S. Camtepe. Evaluating the security of machine learning based IoT device identification systems against adversarial examples. In *ICSOC 2021, Virtual Event, November 22–25, 2021, Proceedings 19*, pages 800–810. Springer, 2021. https://doi.org/10.1007/978-3-030-91431-8_57.

[174] M. Nawir, A. Amir, N. Yaakob, and O. B. Lynn. Internet of things (IoT): Taxonomy of security attacks. In *2016 3rd International Conference on Electronic*

Design (ICED), pages 321–326. IEEE, 2016. https://doi.org/10.1109/ICED.2016.7804660.

[175] NetFlow. NetFlow version 9 flow-record format, 2011. Accessed: 2023-07-14, Available at https://www.cisco.com/en/US/technologies/tk648/tk362/technologies_white_paper09186a00800a3db9.html.

[176] M. Nobakht, V. Sivaraman, and R. Boreli. A host-based intrusion detection and mitigation framework for smart home IoT using openflow. In *2016 11th International conference on availability, reliability and security (ARES)*, pages 147–156. IEEE, 2016. https://doi.org/10.1109/ARES.2016.64.

[177] O. D. Okey, S. S. Maidin, P. Adasme, R. Lopes Rosa, M. Saadi, D. Carrillo Melgarejo, and D. Zegarra Rodríguez. BoostedEnML: Efficient technique for detecting cyberattacks in IoT systems using boosted ensemble machine learning. *Sensors*, 22(19):7409, 2022. https://doi.org/10.3390/s22197409.

[178] J. Ortiz, C. Crawford, and F. Le. DeviceMien: network device behavior modeling for identifying unknown IoT devices. In *Proceedings of the International Conference on Internet of Things Design and Implementation*, pages 106–117, 2019. https://doi.org/10.1145/3302505.3310073.

[179] M. Ozay, I. Esnaola, F. T. Y. Vural, S. R. Kulkarni, and H. V. Poor. Machine learning methods for attack detection in the smart grid. *IEEE transactions on neural networks and learning systems*, 27(8):1773–1786, 2015. https://doi.org/10.1109/TNNLS.2015.2404803.

[180] E. Papadogiannaki and S. Ioannidis. Acceleration of intrusion detection in encrypted network traffic using heterogeneous hardware. *Sensors*, 21(4):1140, 2021. https://doi.org/10.3390/s21041140.

[181] V. Paxson. Bro: a System for Detecting Network Intruders in Real-Time. *Computer Networks*, 31(23-24):2435–2463, 1999. https://doi.org/10.1016/S1389-1286(99)00112-7.

[182] A. J. Pinheiro, J. d. M. Bezerra, C. A. Burgardt, and D. R. Campelo. Identifying IoT devices and events based on packet length from encrypted traffic. *Computer Communications*, 144:8–17, 2019. https://doi.org/10.1016/j.comcom.2019.05.012.

[183] N. Prazeres, R. L. C. Costa, L. Santos, and C. Rabadão. Engineering the application of machine learning in an ids based on IoT traffic flow. *Intelligent Systems with Applications*, page 200189, 2023. https://doi.org/10.1016/j.iswa.2023.200189.

[184] A. Raghuvanshi, U. K. Singh, G. S. Sajja, H. Pallathadka, E. Asenso, M. Kamal, A. Singh, and K. Phasinam. Intrusion detection using machine learning for risk mitigation in IoT-enabled smart irrigation in smart farming. *Journal of Food Quality*, 2022:1–8, 2022. https://doi.org/10.1155/2022/3955514.

[185] S. Raschka and V. Mirjalili. *Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2*. Packt Publishing Ltd, 2019.

[186] D. K. K. Reddy and H. Behera. Catboosting approach for anomaly detection in IoT-based smart home environment. In *Computational Intelligence in Data Mining: Proceedings of ICCIDM 2021*, pages 753–764. Springer, 2022. https://doi.org/10.1007/978-981-16-9447-9_56.

[187] D. K. K. Reddy, H. S. Behera, G. M. Pratyusha, and R. Karri. Ensemble Bagging Approach for IoT Sensor Based Anomaly Detection. *Lecture Notes in Electrical Engineering*, 702:647–665, 2021. https://doi.org/10.1007/978-981-15-8439-8_52.

[188] Regit. Investigation on an attack tool used in china, 2014. Accessed: 2023-03-16, Available at https://home.regit.org/2014/02/chinese-scanner.

[189] J. Ren, D. J. Dubois, D. Choffnes, A. M. Mandalari, R. Kolcun, and H. Haddadi. Information exposure for consumer IoT devices: A multidimensional, network-informed measurement approach. In *Proc. of the Internet Measurement Conference (IMC)*, 2019. https://doi.org/10.1145/3355369.3355577.

[190] K. Ren, Y. Zeng, Z. Cao, and Y. Zhang. ID-RDRL: a deep reinforcement learning-based feature selection intrusion detection model. *Scientific Reports*, 12(1):1–18, 2022. https://doi.org/10.1038/s41598-022-19366-3.

[191] M. Ring, S. Wunderlich, D. Grüdl, D. Landes, and A. Hotho. Flow-based benchmark data sets for intrusion detection. In *Proceedings of the 16th European Conference on Cyber Warfare and Security (ECCWS)*, pages 361–369.

ACPI, 2017. https://www.academia.edu/download/77273491/Flow-based_benchmark_data_sets_for_intrusion_detection.pdf.

[192] 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 2019*, pages 452–457, 2019. https://doi.org/10.1109/CCWC.2019.8666588.

[193] T. Saba, A. Rehman, T. Sadad, H. Kolivand, and S. A. Bahaj. Anomaly-based intrusion detection system for IoT networks through deep learning model. *Computers and Electrical Engineering*, 99:107810, 2022. https://doi.org/10.1016/j.compeleceng.2022.107810.

[194] T. Saba, T. Sadad, A. Rehman, Z. Mehmood, and Q. Javaid. Intrusion detection system through advance machine learning for the internet of things networks. *IT Professional*, 23(2):58–64, 2021. https://doi.org/10.1109/MITP.2020.2992710.

[195] Y. K. Saheed, A. I. Abiodun, S. Misra, M. K. Holone, and R. Colomo-Palacios. A machine learning-based intrusion detection for detecting internet of things network attacks. *Alexandria Engineering Journal*, 61(12):9395–9409, 2022. https://doi.org/10.1016/j.aej.2022.02.063.

[196] S. Salamatian, W. Huleihel, A. Beirami, A. Cohen, and M. Médard. Why botnets work: Distributed brute-force attacks need no synchronization. *IEEE Transactions on Information Forensics and Security*, 14(9):2288–2299, 2019. https://doi.org/10.1109/TIFS.2019.2895955.

[197] O. Salman, I. H. Elhajj, A. Chehab, and A. Kayssi. A machine learning based framework for IoT device identification and abnormal traffic detection. *Transactions on Emerging Telecommunications Technologies*, 33(3):e3743, 2022. https://doi.org/10.1002/ett.3743.

[198] A. Salmon. *Applied network security : master the art of detecting and averting advanced network security attacks and techniques*. Packt Publishing, Birmingham, UK, 2017.

[199] S. Salvatore, F. Wei, L. Wenke, P. Andreas, and C. Philip. KDD Cup 1999 Data, 1999. https://doi.org/10.24432/C51C7N.

[200] P. M. S. Sánchez, A. H. Celdrán, G. Bovet, and G. M. Pérez. Adversarial attacks and defenses on ML-and hardware-based IoT device fingerprinting and identification. *arXiv preprint arXiv:2212.14677*, 2022. https://doi.org/10.48550/arXiv.2212.14677.

[201] M. R. Santos, R. M. Andrade, D. G. Gomes, and A. C. Callado. An efficient approach for device identification and traffic classification in IoT ecosystems. In *2018 IEEE Symposium on Computers and Communications (ISCC)*, pages 00304–00309. IEEE, 2018. https://doi.org/10.1109/ISCC.2018.8538630.

[202] N. Saran and N. Kesswani. A comparative study of supervised machine learning classifiers for intrusion detection in internet of things. *Procedia Computer Science*, 218:2049–2057, 2023. https://doi.org/10.1016/j.procs.2023.01.181.

[203] M. Sarhan, S. Layeghy, N. Moustafa, M. Gallagher, and M. Portmann. Feature extraction for machine learning-based intrusion detection in IoT networks. *Digital Communications and Networks*, 2022. https://doi.org/10.1016/j.dcan.2022.08.012.

[204] M. R. Shahid, G. Blanc, Z. Zhang, and H. Debar. IoT devices recognition through network traffic analysis. In *2018 IEEE international conference on big data (big data)*, pages 5187–5192. IEEE, 2018. https://doi.org/10.1109/BigData.2018.8622243.

[205] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani. A detailed analysis of the CICIDS2017 data set. In *Information Systems Security and Privacy: 4th International Conference, ICISSP 2018, Funchal-Madeira, Portugal, January 22-24, 2018, Revised Selected Papers 4*, pages 172–188. Springer, 2019. https://doi.org/10.1007/978-3-030-25109-3_9.

[206] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp*, 1:108–116, 2018. https://doi.org/10.5220/0006639801080116.

[207] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani. Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy. In *2019 International Carnahan Conference on Security Technology (ICCST)*, pages 1–8. IEEE, 2019. https://doi.org/10.1109/CCST.2019.8888419.

[208] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *computers & security*, 31(3):357–374, 2012. https://doi.org/10.1016/j.cose.2011.12.012.

[209] R. Shwartz-Ziv and A. Armon. Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90, 2022. https://doi.org/10.1016/j.inffus.2021.11.011.

[210] M. A. Siddiqi and W. Pak. An Agile Approach to Identify Single and Hybrid Normalization for Enhancing Machine Learning-Based Network Intrusion Detection. *IEEE Access*, 9:137494–137513, 2021. https://doi.org/10.1109/ACCESS.2021.3118361.

[211] A. J. Siddiqui and A. Boukerche. TempoCode-IoT: temporal codebook-based encoding of flow features for intrusion detection in Internet of Things. *Cluster Computing*, 24(1):17–35, 2021. https://doi.org/10.1007/s10586-020-03153-8.

[212] A. Singh, N. Thakur, and A. Sharma. A review of supervised machine learning algorithms. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 1310–1315. IEEE, 2016. https://ieeexplore.ieee.org/abstract/document/7724478.

[213] A. P. Singh, S. Kumar, A. Kumar, and M. Usama. Machine learning based intrusion detection system for minority attacks classification. In *2022 International Conference on Computational Intelligence and Sustainable Engineering Solutions (CISES)*, pages 256–261. IEEE, 2022. https://doi.org/10.1109/CISES54857.2022.9844381.

[214] A. Sivanathan, H. H. Gharakheili, F. Loi, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman. Classifying IoT devices in smart environments using network traffic characteristics. *IEEE Transactions on Mobile Computing*, 18(8):1745–1759, 2018. https://doi.org/10.1109/TMC.2018.2866249.

[215] Y. Song, Q. Huang, J. Yang, M. Fan, A. Hu, and Y. Jiang. IoT device fingerprinting for relieving pressure in the access control. In *Proceedings of the ACM Turing Celebration Conference-China*, pages 1–8, 2019. https://doi.org/10.1145/3321408.3326671.

[216] V. L. Sujay. Number of internet of things (IoT) connected devices worldwide from 2019 to 2023, with forecasts from 2022 to 2030, 2023. Accessed: 2023-09-08, https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/.

[217] Sumitra and M. V. Shenoy. HFedDI: A novel privacy preserving horizontal federated learning based scheme for IoT device identification. *Journal of Network and Computer Applications*, 214:103616, 2023. https://doi.org/10.1016/j.jnca.2023.103616.

[218] Y. Sun, S. Fu, S. Zhang, H. Zhu, and Y. Li. Accurate IoT device identification from merely packet length. In *2020 16th International Conference on Mobility, Sensing and Networking (MSN)*, pages 774–778. IEEE, 2020. https://doi.org/10.1109/MSN50589.2020.00132.

[219] Y. Sun, J. Liu, A. K. Bashir, U. Tariq, W. Liu, K. Chen, and M. D. Alshehri. E-CIS: Edge-based classifier identification scheme in green & sustainable IoT smart city. *Sustainable Cities and Society*, 75:103312, 2021. https://doi.org/10.1016/j.scs.2021.103312.

[220] P. M. Sánchez Sánchez, J. M. Jorquera Valero, A. Huertas Celdrán, G. Bovet, M. Gil Pérez, and G. Martínez Pérez. Lwhbench dataset, 2022. Accessed: 2023-09-15, https://dx.doi.org/10.21227/8zmy-7z87.

[221] A. S. Tanenbaum and D. J. Wetherall. *Computer Networks, fifth edition*. Pearson, 2011.

[222] S. Tanveer, M. Husnain, H. Akram, S. G. Abbas, and G. A. Shah. Generic application layer features for IoT devices identification. In *2022 International Conference on Cyber Warfare and Security (ICCWS)*, pages 49–56. IEEE, 2022. https://doi.org/10.1109/ICCWS56285.2022.9998468.

[223] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani. A detailed analysis of the KDDCUP99 data set. In *2009 IEEE symposium on computational intelligence for security and defense applications*, pages 1–6. IEEE, 2009. https://doi.org/10.1109/CISDA.2009.5356528.

[224] V. Thangavelu, D. M. Divakaran, R. Sairam, S. S. Bhunia, and M. Gurusamy. DEFT: A distributed IoT fingerprinting technique. *IEEE Internet of Things Journal*, 6(1):940–952, 2018. https://doi.org/10.1109/JIOT.2018.2865604.

[225] J. Thom, N. Thom, S. Sengupta, and E. Hand. Smart recon: Network traffic fingerprinting for IoT device identification. In *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0072–0079. IEEE, 2022. https://doi.org/10.1109/CCWC54503.2022.9720739.

[226] O. Thompson, A. M. Mandalari, and H. Haddadi. Rapid IoT device identification at the edge. In *Proceedings of the 2nd ACM International Workshop on Distributed Machine Learning*, pages 22–28, 2021. https://doi.org/10.1145/3488659.3493777.

[227] G. Tiwari and R. Jain. Detecting and classifying incoming traffic in a secure cloud computing environment using machine learning and deep learning system. In *2022 IEEE 7th International Conference on Smart Cloud (SmartCloud)*, pages 16–21. IEEE, 2022. https://doi.org/10.1109/SmartCloud55982.2022.00010.

[228] F. Trad, A. Hussein, and A. Chehab. Using Siamese Neural Networks for Efficient and Accurate IoT Device Identification. In *2022 Seventh International Conference on Fog and Mobile Edge Computing (FMEC)*, pages 1–7. IEEE, 2022. https://doi.org/10.1109/FMEC57183.2022.10062771.

[229] I. Ullah and Q. H. Mahmoud. A scheme for generating a dataset for anomalous activity detection in IoT networks. In *Canadian conference on artificial intelligence*, pages 508–520. Springer, 2020. https://doi.org/10.1007/978-3-030-47358-7_52.

[230] I. Ullah and Q. H. Mahmoud. Network traffic flow based machine learning technique for IoT device identification. In *2021 IEEE International Systems Conference (SysCon)*, pages 1–8. IEEE, 2021. https://doi.org/10.1109/SysCon48628.2021.9447099.

[231] I. Vaccari, G. Chiola, M. Aiello, M. Mongelli, and E. Cambiaso. MQTTset, a new dataset for machine learning techniques on MQTT. *Sensors*, 20(22):6578, 2020. https://doi.org/10.3390/s20226578.

[232] M. Vanhoef and F. Piessens. Advanced Wi-Fi attacks using commodity hardware. In *Proceedings of the 30th Annual Computer Security Applications Conference*, pages 256–265, 2014. https://doi.org/10.1145/2664243.2664260.

[233] A. Verma and V. Ranga. Machine Learning Based Intrusion Detection Systems for IoT Applications. *Wireless Personal Communications*, 111(4):2287–2310, 2020. https://doi.org/10.1007/s11277-019-06986-8.

[234] J. Vitorino, I. Praça, and E. Maia. Towards adversarial realism and robust learning for iot intrusion detection and classification. *Annals of Telecommunications*, pages 1–12, 2023. https://doi.org/10.1007/s12243-023-00953-y.

[235] M. Wang, N. Yang, and N. Weng. Securing a smart home with a transformer-based IoT intrusion detection system. *Electronics*, 12(9):2100, 2023. https://doi.org/10.3390/electronics12092100.

[236] Y. Wang, B. P. Rimal, M. Elder, S. I. C. Maldonado, H. Chen, C. Koball, and K. Ragothaman. IoT device identification using supervised machine learning. In *2022 IEEE International Conference on Consumer Electronics (ICCE)*, pages 1–6. IEEE, 2022. https://doi.org/10.1109/ICCE53296.2022.9730354.

[237] WireShark. Wireshark: Network analyzer, 1997. www.wireshark.org.

[238] Z. Wu, H. Zhang, P. Wang, and Z. Sun. RTIDS: A robust transformer-based approach for intrusion detection system. *IEEE Access*, 10:64375–64387, 2022. https://doi.org/10.1109/ACCESS.2022.3182333.

[239] L. Xiao, X. Wan, X. Lu, Y. Zhang, and D. Wu. IoT security techniques based on machine learning: How do IoT devices use ai to enhance security? *IEEE Signal Processing Magazine*, 35(5):41–49, 2018. https://doi.org/10.1109/MSP.2018.2825478.

[240] P. Yadav, A. Feraudo, B. Arief, S. F. Shahandashti, and V. G. Vassilakis. Position paper: A systematic framework for categorising IoT device fingerprinting mechanisms. In *Proceedings of the 2nd international workshop on challenges in artificial intelligence and machine learning for internet of things*, pages 62–68, 2020. https://doi.org/10.1145/3417313.3429384.

[241] R. Yao, N. Wang, P. Chen, D. Ma, and X. Sheng. A cnn-transformer hybrid approach for an intrusion detection system in advanced metering infrastructure. *Multimedia Tools and Applications*, pages 1–24, 2022. https://doi.org/10.1007/s11042-022-14121-2.

[242] F. Yin, L. Yang, Y. Wang, and J. Dai. IoT ETEI: End-to-end IoT device identification method. In *2021 IEEE Conference on Dependable and Secure Computing (DSC)*, pages 1–8. IEEE, 2021. https://doi.org/10.1109/DSC49826.2021.9346251.

[243] S. Yin, W. Zhang, Y. Feng, Y. Xiang, and Y. Liu. Automatic IoT device identification: a deep learning based approach using graphic traffic characteristics. *Telecommunication Systems*, pages 1–14, 2023. https://doi.org/10.1007/s11235-023-01009-1.

[244] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga. A survey of intrusion detection in internet of things. *Journal of Network and Computer Applications*, 84:25–37, 2017. https://doi.org/10.1016/j.jnca.2017.02.009.

[245] C. Zhang, D. Jia, L. Wang, W. Wang, F. Liu, and A. Yang. Comparative research on network intrusion detection methods based on machine learning. *Computers & Security*, page 102861, 2022. https://doi.org/10.1016/j.cose.2022.102861.

[246] L. Zhang, L. Gong, and H. Qian. An effiective IoT device identification using machine learning algorithm. In *2020 IEEE 6th International Conference on Computer and Communications (ICCC)*, pages 874–877. IEEE, 2020. https://doi.org/10.1109/ICCC51575.2020.9345256.

[247] M. Zolanvari, M. A. Teixeira, L. Gupta, K. M. Khan, and R. Jain. Machine Learning-Based Network Vulnerability Analysis of Industrial Internet of Things. *IEEE Internet of Things Journal*, 6(4):6822–6834, 2019. https://doi.org/10.1109/JIOT.2019.2912022.

[248] Q. Zou, Q. Li, R. Li, Y. Huang, G. Tyson, J. Xiao, and Y. Jiang. Iotbeholder: A privacy snooping attack on user habitual behaviors from smart home wi-fi traffic. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 7(1):1–26, 2023. https://doi.org/10.1145/3580890.

[249] Ö. Zülfe. Osmanlı şiirinde tìġ-ı ġamġam tabiri. *Marmara Türkiyat Araştırmaları Dergisi*, 2(1):1–8, 2015. https://dergipark.org.tr/en/download/article-file/213463.