Machine Learning-Enabled Security in Internet of Things and Cyber-Physical Systems

by

Jinxin Liu

Thesis Supervisor : Dr. Burak Kantarci

A thesis
presented to the University of Ottawa
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Vojislav Misic Professor, Department of Computer Science, Toronto Metropolitan University
External Examiner:	Ahmed Refaey Hussein Professor, School of Engineering, University of Guelph
Supervisor:	Burak Kantarci Professor, School of Electrical Engineering and Computer Science, University of Ottawa
Internal Member:	Guy-Vincent Jourdan Professor, School of Electrical Engineering and Computer Science, University of Ottawa

Internal Member:	Melike Erol-Kantarci Professor, School of Electrical Engineering and Computer Science, University of Ottawa
Internal Member:	Hussein Mouftah Professor, School of Electrical Engineering and Computer Science, University of Ottawa

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Internet of Things (IoT) is a promising and thriving technology that incorporates a variety of smart devices that provide enhanced services for remote communication and interaction between humans and physical items. The number of deployed IoT devices will increase to 41.6 billion in 2025, as predicted by International Data Corporation. With such a large population, assaults on IoT networks will harm a vast number of users and IoT devices. In light of this, we explore security from physical and network viewpoints in this thesis. To preserve privacy in IoT environment, this thesis begins by proposing RASA, a context-sensitive access authorization approach. We evaluate the promise of RASA-generated policies against a heuristic rule-based policy. The decisions of the RASA and that of the policy are more than 99% consistent.

Furthermore, not only physical attacks but also cybercrimes will threaten IoT networks; consequently, this thesis proposes various Network Intrusion Detection System (NIDS) to identify network intrusions. In this thesis, we firstly examine traditional attacks in the NSL-KDD dataset that can impact sensor networks. Furthermore, in order to detect the introduced attacks, we study eleven machine learning algorithms, among which, XGBoost ranks the first with 97% accuracy. As attack tactics continue to evolve, Advanced Persistent Threat (APT) poses a greater risk to IoT networks than traditional incursions. This thesis presents SCVIC-APT-2021 to define a APT benchmark. Following upon this, an ML-based Attack Centric Method (ACM) is introduced achieving 9.4% improvement with respect to the baseline performance.

This thesis proposes a Combined Intrusion Detection System (CIDS) that takes network and host information into consideration to reduce data noise and improve the performance of IDS. Two new CIDS datasets, SCVIC-CIDS-2021 and SCVIC-CIDS-2022, are generated. We further propose CIDS-Net to incorporate network and host related data. CIDS-Net boost the macro F1 score of the best baseline by 5.8% (up to 99.95%) and 5.1% (up to 91.3%), respectively on the two datasets.

Besides of detection performance, timely response is considered as a critical metric of NIDS. This thesis introduces Multivariate Time Series (MTS) early detection into NIDS . We form TS-CICIDS2017 which is a time series based NIDS dataset and a new deep learning-based early detection model called Multi-Domain Transformer (MDT) is proposed, resulting in a 84.1% macro F-score with only few of the initial packets. To reduce the size of NIDS inputs, this work proposes a deep learning-based lossy time series compressor (Deep Dict) to achieve a high compression ratio while limiting the decompression error within a desired range. As demonstrated by the results, Deep Dict outperforms the compression ratio of the state-of-the-art lossy compression methods by up to 53.66%.

Acknowledgements

I would like to express my sincere gratitude to my supervisor Dr. Burak Kantarci for his continuous support throughout my Ph.D. study. He opened the door for me to a novel research field, helped me to find a suitable research topic, and guided me successfully. The honest, careful and hard working spirit of Dr. Burak Kantarci will always inspire me. I would also like to thank Dr. Murat Simsek for his help and contributions to our work and all the Smart Connected Vehicles Innovation Centre (SCVIC) members for their helping and accompanying.

In addition, this thesis cannot be completed without the successful collaborations with Ciena Corp. and BlackBerry Ltd., therefore I'd like to thank Dr. Petar Djukic, Dr. Mehran Bagheri, Dr. Andrew Walenstein, and Dr. Andrew Malton for offering a variety of resources and academic ideas.

Finally, I must express my very profound gratitude to my parents for supporting me in every step of my life and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

6LoWPAN: IPv6 over Low -Power Wireless Personal Area Networks

A/RPE: Absolute/Relative Positional Encoding

AB: AdaBoost

Acc: Accuracy

ACFS: Attack Centric Feature Selection;

ACM: Attack Centric Method

APT: Advanced Persistent Threat

AUC: Area Under the Curve

BN: Bayesian Network

BTAE: Bernoulli Transformer AutoEncoder

CIDS-DFF: CIDS Dataset Formation Framework

CIDS: Combined IDS

CNN: Convolutional neural network

CoAP: Constrained Application Protocol

CR: Compression Ratio

CRV: Cluster Risk Value

DBSCAN: Density-Based Spatial Clustering of Applications with Noise

DC: Domain Controller

DDoS: Distributed Denial of Service

DoS: Denial of Service attack

DSR: Diagonal Softmax Regression

DT: Decision Tree

EM: Expectation Maximization

FFN: Feed Forward Network

FN: False Negative

FNN: Feed-Forward Neuron Network

FP: False Positive

HIDS: Host-based IDS

IAT: Inter-Arrival Time

IDS: Intrusion Detection System

IoT: Internet of things

MaAE: Maximum Absolute Error

MCC: Matthews Correlation Coefficient

MD-MHA: Multi-Domain Multi-Head Attention

MDT: Multi-Domain Transformer

NB: Naive Bayes

NIDS: Network-based IDS

OS: Operating System

PCAP: Packet Capture

PoD: Ping of Death

QEL: Quantized Entropy Loss

R2L: Remote to Local

RASA: Risk-Aware Smart Access Control

RF: Random Forest

RNN: Recurrent Neuron Network

RPL: Routing Protocol for Low-Power and Lossy Networks

SVM: Support Vector Machine

TN: True Negative

TP: True Positive

TS-NFM: Time Series Network Flow Meter

U/MTS: Uni/Multi-variate Time Series

U2R: User to Root

VPN: Virtual Private Network

WSN: Wireless Sensor Network

XGB: XGBoost

List of Symbols

$\mathbf{C}$: The coefficient matrix used for generating synthetic dataset

$\mathbf{C_{Loss}}$: CIDS-Net's loss function

$\mathbf{C^{Dur}_{a_i,b_j}}$: Duration-based Normalized Coupling Values

$\mathbf{C^{Freq}_{a_i,b_j}}$: Frequency-based Normalized Coupling Values

$\mathbf{D}$: A distance matrix where $d_{.,j} = r_. - s_j$

$\mathbf{Dur_{a_i,b_j}}$: Duration of the occurrence of a_i, and b_j

$\mathbf{E_k}$: The k^{th} event which is defined as a collection of the elements from various set

$\mathbf{F^{A,B}_n}$: The coupling Feature

$\mathbf{Freq_{a_i,b_j}}$ Number of times a_i, and b_j occur together

$\mathbf{H}$: The objective function which calculates r's quantized entropy

$\mathbf{P}$: A differentiable function used to replace p

$\mathbf{Ppl}, \mathbf{Dev}, \mathbf{Doc}, \mathbf{Loc}$: Users, Devices, Documents, Locations

$\mathbf{R^{Dur}_{a_i,b_j}}$: Risk of Duration-based Normalized Coupling Values

$\mathbf{R^{Freq}_{a_i,b_j}}$: Risk of Frequency-based Normalized Coupling Values

$\mathbf{S}$: A collection of unique values of r_q

$\delta_{\mathbf{a_i,b_j}}(\mathbf{k})$: Coupling distribution

ϵ : User specified Maximum Absolute Error

$\hat{\mathbf{x}}$: The time series predicted by BTAE model

$\hat{\mathbf{y_e f}}$: The output of host feature encoder

$\hat{\mathbf{y_e m}}$: The output of host message encoder

$\hat{\mathbf{y_n}}$: The output of network feature encoder

$\hat{\mathbf{y}}$: The output of prediction

$\mathbf{c}$: The Bernoulli distributed latent states

$\mathbf{f_{binarization}}$: Binarization funtion used for transforming real-value latent state to Bernoulli latent states

$\mathbf{g}_\epsilon, \mathbf{g}'_\epsilon$: A rectangular function and its approximation

$\mathbf{h_{SR/DSR}}$: The hypothesis function of softmax regression or diagnal softmax regression

$\mathbf{l}, \mathbf{d}$: The length and the number of variables of time series

$\mathbf{n}(\mathbf{s_j})$: The number of s_j occurs in r_q

$\mathbf{p}(\mathbf{s_j})$: The possibility of s_j appears in r_q

$\mathbf{r}$: The residual between original time series and the predicted time series

$\mathbf{r_q}$: The uniformedly quantized r

$\mathbf{r_{encoded}}$: The encoded r_q compressed by entropy coder

$\mathbf{t}, \mathbf{t_p}$: Timestamp, and polynomial timestamp

$\mathbf{x}$: Uni/multi-variate time series

$\mathbf{x_{recon}}$: The reconstructed time series

$\mathbf{y}$: Real-value latent states generated by the BTAE's encoder

List of Metrics

$\mathbf{Accuracy} = \frac{\mathbf{TP+TN}}{\mathbf{TP+TN+FP+FN}}$: The ratio of correctly predicted samples to all samples

$\mathbf{CompressionRatio(CR)} = \frac{|\mathbf{original\_file}|}{|\mathbf{compressed\_file}|}$: Evaluating the performance of compressor

$\mathbf{F1} = \mathbf{2} * \frac{\mathbf{precision*recall}}{\mathbf{precision+recall}}$: The harmonic mean of precision and recall

$\mathbf{FalseNegativeRate(FNR)} = \frac{\mathbf{N_{FN}}}{\mathbf{N_{FN}+N_{TP}}}$: The ratio of FN to all positive class

$\mathbf{FalsePositiveRate(FPR)} = \frac{\mathbf{N_{FP}}}{\mathbf{N_{FP}+N_{TN}}}$: The ratio of FP to all negative class

$\mathbf{N_{All}}$: The number of samples

$\mathbf{N_{FN}}$: False Negative

$\mathbf{N_{FP}}$: False Positive

$\mathbf{N_{TN}}$: True Negative

$\mathbf{N_{TP}}$: True Poitive

$\mathbf{Precision} = \frac{\mathbf{N_{TP}}}{\mathbf{N_{TP}+N_{FP}}}$: The ratio of TP to TP and NP

$\mathbf{Recall} = \frac{\mathbf{N_{TP}}}{\mathbf{N_{TP}+N_{FN}}}$: Same as true positive rate

$\mathbf{TrueNegativeRate(TNR)} = \frac{\mathbf{N_{TN}}}{\mathbf{N_{TN}+N_{FP}}}$: The ratio of TN to all negative class

Publications

The following list includes the publications published, accepted, under revision, or under submission.

- Patents
  - A. Walenstein, A. Malton, J. Liu, M. Simsek, B. Kantarci, M. ErolKantarci, "Risk-Aware Access Control System and Related Methods" (patent filed in Dec. 2020)
  - Z. Chen, M. Simsek, B. Kantarci, P. Djukic, J. Carnes, M. Bagheri, J. Liu, Y. Shen, "Machine learning detection of network attacks using traffic and log information," patent filed, 17/571,342 Jan 7, 2022
  - P. Djukic, I. Radovic, M. Amiri, J. Liu, B. Kantarci, "Compressing network data using Deep Neural Network (DNN) deployment," 17/880,400, utility patent filed on August 4th, 2022.
- Conference
  - J. Liu, B. Kantarci, C. Adams, "Machine Learning-Driven Intrusion Detection for Contiki-NG-Based IoT Networks Exposed to NSL-KDD Dataset" ACM Workshop on Wireless Security and Machine Learning (WiseML) , Linz, Austria (Virtual Event), July 2020.
  - J. Liu, M. Simsek, B. Kantarci, M. Bagheri, P. Djukic, "Collaborative Feature Maps of Networks and Hosts for AI-driven Intrusion Detection," IEEE Global Communications Conference (Globecom), Rio de Janeiro, Brazil, December 2022.
  - J. Liu, M. Simsek, B. Kantarci, "Multidomain transformer-based deep learning for early detection of network intrusion," IEEE International Conference on Communications (ICC), 2023, [Submitted]
- Journals
  - J. Liu, M. Simsek, B. Kantarci, M. Erol-Kantarci, A. Malton, A. Walenstein, "Risk-Aware Fine-Grained Access Control in Cyber-Physical Contexts", ACM Digital Threats: Research and Practice, 2021.

– J. Liu, M. Nogueira, J. Fernandes, B. Kantarci, "Adversarial Machine Learning: A Multi-Layer Review of the State-of-the-Art and Challenges for Wireless and Mobile Systems," IEEE Communications Surveys and Tutorials, vol. 24, no. 1, pp. 123-159, Firstquarter 2022.

– Z. Chen, J. Liu, Y. Shen, M. Simsek, B. Kantarci, H. T. Mouftah, P. Djukic, "Machine Learning-Enabled IoT Security: Open Issues and Challenges Under Advanced Persistent Threats," ACM Computing Surveys, 2022.

– J. Liu, Y. Shen, M. Simsek, B. Kantarci, H. T. Mouftah, M. Bagheri, P. Djukic, "A New Realistic Benchmark for Advanced Persistent Threats in Network Traffic," IEEE Networking Letters, 2022.

– J. Liu, M. Simsek, B. Kantarci, M. Bagheri, P. Djukic, "Bridging Networks and Hosts via Machine Learning-Based Intrusion Detection," IEEE Transactions on Dependable and Secure Computing, 2022. [Submitted]

– J. Liu, P. Djukic, B. Kantarci, "Deep Dict: Deep Learning-based Lossy Time Series Compression," IEEE Transactions on Knowledge and Data Engineering, 2022. [Submitted]

Chapter 1

Introduction

Due to rising cybercrime incidents (such as seizing control of industry hardware and smart lights, or recording videos of personal locations), security of Internet of Things (IoT) systems has gained attention in recent years [14]. In order to provide comprehensive defense, this thesis aims at protecting security issues in IoT environments from physical threats and network intrusions. Access to a resource may be given or denied based on the context in which the request is made, including the cyberphysical environment. Therefore, this thesis proposes an unsupervised method for identifying illicit medical access. As a key security mechanism against network attacks, Machine/deep learning-based Network Intrusion Detection Systems (NIDSs) have been thoroughly studied. We enhance NIDSs from several angles, including the construction of five NIDS datasets, detection performance, and quick response. This thesis begins by examining the impact of classic attack approaches (e.g., TCP/UDP flood) on modern IoT systems and networks; the findings indicate that some of the attacks are still valid. Therefore, we generate the NIDS dataset by executing NSL-KDD intrusions on IoT networks and identify them using various machine learning methods. As the strategies of attackers continue to evolve, not only are conventional intrusions probed, but Advanced Persistent Threats (APT) pose an even greater risk to IoT devices and networks. This thesis creates an APT NIDS dataset and proposes a new ensemble approach to enhance the detection performance against APT attacks. Even with sophisticated ML models, conventional and APT intrusions might be difficult to identify. Some benign and incursion traffic may share the same network characteristics, therefore extra information, such as host-based data, is necessary to increase detection accuracy and reduce data noise. This thesis proposes a Combined Intrusion Detection System (CIDS) to incorporate network- and host-based information.

In addition to detection performance, defence speed is an essential parameter for eval-

uating IDS. Conventional flow-based NIDSs gather network packets over a rather lengthy time window (i.e., 2 minutes for CIC-IDS-2018). To identify intrusions as quickly as feasible, we thus define network invasions as time series and detect intrusions using a small number of packets. In addition, rather than extracting arbitrary characteristics from packets, this thesis develops a model based on deep learning to automatically obtain features for improved detection performance. Although expressing network flows as time series can have multiple benefits, storing and sending such a large quantity of data can be expensive; hence, this thesis presents a lossy time series compressor to dramatically reduce data size.

As this thesis improves IoT security from various viewpoints, the sections that follow describe these enhancements in depth.

1.1 Machine Learning-based Authorization in Cyber Physical System

Whether access should be granted to a resource may depend upon the context in which the access is attempted, including the cyberphysical context. For example, a physician may have a legitimate need to access a patient's private health information (PHI) on a mobile display screen while in a private consultation room, but it might be important to lock out the access if the physician wheels the display into the hallway, or if a different patient is moved into the shared room. Failure to lock out could lead to leaking PHI to other people who the patient has not consented to allow access to. In such cases the identity of the accessing physician has not changed, nor has the physician's *role* changed with respect to the patient. The concern is thus unrelated to *authentication* and static user *roles*. Instead, it is an issue of context-dependent or context-sensitive *authorization*.

Recently, context-aware security approaches have gained significant attention, and there have been several studies around inferring and understanding contexts. But even state-of-the-art access control mechanisms for cyber-physical systems may lack the generality or flexibility desired to confine access authorization to dynamic cyberphysical context, or they otherwise burden the access policy administrators with having to tediously and continuously define every context, set the appropriate access restrictions, and understand the resulting policy well enough to assess the residual risks left by the policy. For example, methods for defining *geofences* may be used to define physical contexts for defining context-dependent access policy, but many geofencing access policies utilize geofences only for authorization decisions, and physical location is one contextual factor affecting authorization risks. For example, a phone may lock when moved out of its "home" geofence, but

unlocking restores all prior access, it is only used for context-dependent authentication.

A general framework and approach is required for effective context-dependent authorization policy management. Core concerns in any such context-dependent authorization regime is defining the contexts in which authorization decisions differ, identifying and quantifying risks for those contexts, and then using the derived risk model to create appropriate access control decisions. Given that contexts may change over time, an additional burden is tracking these changing contexts over time. A problem that is not yet well addressed are effective methods for automatically helping define and maintain context-dependent authorization policies, i.e., inferring the contexts, risk models, and access decision boundaries.

This thesis introduces a framework named Risk-Aware Smart Access (RASA) to learn authentication policies from examples of routine access. RASA begins with the observation that *if*, in normal practice, responsible actors access information they have ordinary access to only in those contexts in which the need access, and that they take care to guard against known risks. In such cases, even if an existing access control policy is not context-sensitive, responsible actors effectively fill in the missing contextual access conditions through their choices and actions, and these can be used to infer a baseline context-dependent access control policy. In a sense, they can program the context-dependent access conditions by demonstration. RASA uses coupling of access actions to identify patterns of access, clusters the actions, associates these coupling-defined features to risk scores, and then defines access boundaries based simply on these risk scores. To the best of our knowledge, no prior work exists for similar automated policy inference based on mapping inferred contexts onto risk levels.

The thesis evaluates the general promise of the approach in a simulated in-person healthcare environment. This use context is cyberphysical, and one where ensuring strong access boundaries–even for fully authenticated users–are critical for protecting sensitive PHI. Nearly 90% of participants, in a health care survey reported by the Ponemon Institute, have been affected by data leakage in the past three years [112]. Furthermore, inappropriate use of devices that handle sensitive document by internal employees has led to the breach of 4.5M patient records in 2015 [75]. Intelligent approaches to such security may be increasingly important [175], and not all risks are due to remote attackers–shoulder surfing in a shared patient room, for example. The context-dependent appropriateness of PHI access serves as a suitable test of cyberphysical context dependencies.

1.2 Intrusion Detection in IoT Networks

Several researchers have tackled the applicability of machine learning (ML) algorithms to detect security breaches and attacks on IoT networks. The study in [214] presents a comprehensive survey of ML algorithms for IoT security. Due to the lack of IoT intrusion datasets, a common strategy used in many studies is to use off-the-shelf datasets to inject malicious traffic into IoT networks [173] where the NSL-KDD dataset is directly used to test proposed Intrusion Detection System (IDS). Nonetheless, it is not viable to use a trained model to detect attack patterns in an IoT setting since regular traffic characteristics and attack patterns in IoT networks differ from those in Ethernet-based networks [12]. Researchers explore the available ML algorithms to detect network intrusion datasets in a non-IoT context, and implement routing attacks in IoT environments. However, well-known attacks like DoS and probe still threaten IoT networks.

Contiki-NG is an open source operating system for resource limited devices. It offers the cross-platform benefits, and supports low power communication standards, such as IPv6/ 6LoWPAN, 6TiSCH, RPL, and CoAP. Contiki-NG provides multithreading and optional preemptive multithreading based on protothreads to enhance resource allocation [67]. One of the major contributions of this thesis is to examine Contiki-NG when facing attacks from an open benchmark such as NSL-KDD. Thus, UDP, TCP/IP, and 6LOWPAN are carefully investigated to find the exact source that causes the vulnerabilities. The Contiki-NG operating system also provides a simulator called Cooja to help researchers simulate devices and networks, reducing the time and financial cost of experiments. Cooja supports simulation including MAC, network, and application layer protocols, and integration with external tools to provide additional information, such as battery consumption, network interference, and network topologies. This work utilizes Cooja analyzer to collect packet information and PCAP files on a Contiki-NG-based network. Based on the observations above, we investigate the vulnerabilities in Contiki-NG operating systems in IoT networks, and introduce attack types in the NSL-KDD dataset into a Contiki-NG-based IoT network.

In addition to classic intrusions, during the last decade, an increasing number of businesses and organisations have been attacked by a series of network assaults named Advanced Persistent Threat (APT) [45]. APT attacks aim to infiltrate the internal networks of governments and organisations in order to exfiltrate sensitive data and information. The APT initiators are well-organized and proficient in a variety of network attack techniques and resources. They aim for the confidential data of high-value targets such as governments and high-tech firms. APT's data leakage results in increased strategic errors and economic losses for victims. Although personal information may become one of the APT's rings, it is no longer the primary goal as it is in conventional network intrusions [97]. With this in

mind, we create a new APT NIDS dataset and propose an ensemble algorithm to detect it.

1.3 Combined Intrusion Detection System

With the increasing expansion of networks and communications, Intrusion Detection Systems (IDSs) have become an important research field to protect digital assets and critical infrastructures [99]. Extensive research has established the effectiveness of IDS under two categories based on the source of data: Network Intrusion Detection System (NIDS) and Host Intrusion Detection System (HIDS) [162].

NIDSs are devices or software distributed at tactical locations to monitor networks and detect hostile behaviors that raise security alerts [191]. NIDS workflows typically consist of three steps: sniffing network packets, extracting features from network traffic, and detecting intrusions [203]. Network traffic can be provided in two formats by NIDS: packet-based or flow-based [191]. Packet-based features summarize the status of individual packets, whereas flow-based features calculate the sample statistics of packets sharing a common property observed within a time window, such as the mean value of the number of bytes in a network flow [63]. The detection tool of an NIDS can be either human-written policies or machine/deep learning engines which can mine patterns from large volumes of network data. HIDSs, on the other hand, identify attacks via analyzing data from servers and end user devices such as system logs, system calls, registry keys and file systems [116]. Extensive research has been conducted on both NIDS and HIDS, and substantial machine learning models have been proposed to decrease bias and variation in order to improve detection performance.

When Machine Learning (ML)-based IDS is leveraged, even the best ML models cannot circumvent data noise, and that any model will ultimately reach its theoretical limit as a result of data noise. This can be analytically shown by decomposing ML errors [113]. Therefore, integrating the feature and data from various domains is one of the potential solutions to address this fundamental issue.

Nevertheless, since datasets including both the network and host-based data are scarce, few research concentrate on the integration of these dissimilar features [191]. This thesis mitigates the scarcity of CIDS dataset by proposing an framework which can integrates network and host based data. Furthermore, a deep learning based network is proposed to detect intrusion using both network and host-based features.

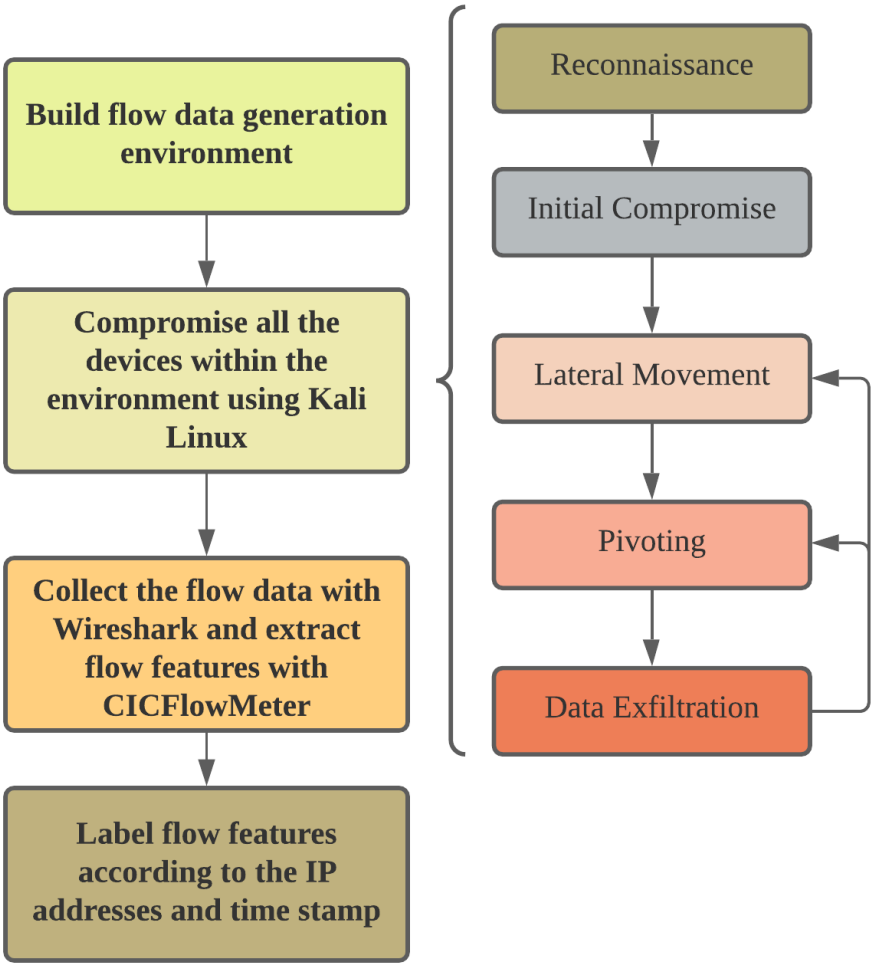


Figure 1.1: Dataset Generation Flow Chart

1.4 Early Detection for Network Intrusions

A NIDS typically builds on a three-step workflow: collecting network packets to construct flows, extracting features, and making decisions [152]. Numerous machine/deep learning models have been proposed, achieving remarkable detection performance with the rapid response; however, the process of flow formation and feature extraction requires relatively long time to accumulate distinguishable features; for example, flows are collected within a 2 minute and 1 minute time window on CICIDS2017 [202] and UNSW-NB15 [166], respectively. As a result, a substantial amount of malicious traffic (particularly Advanced Persistent Threat (APT) traffic) are detected after it reaches target systems [45]. APT attacks or penetration tests typically involve the execution of malicious payloads, which is one of the crucial steps of an APT attack. Table 1.1 summarizes the number of packets and time required to execute the common payloads. Due to the powerful functionalities of meterpreter payloads (a well-known payload for penetration tests), they need more packets and time to establish communication tunnels, and shell / PowerShell payloads require only 10 packets in 0.1s to execute. However, in conventional NIDS configurations, even if a meterpreter is utilized, attackers have sufficient time to conduct post-infiltration activities (such as data collection or lateral movement) before being discovered. Early detection, which is defined as categorising using the front subsequence of a time series, could be ideal to address timeliness in conventional NIDS. An essential metric is earliness, which is the length of the subsequence utilised for classification over the length of the time series. In the context of NIDS, earliness is the ratio of the number of packets used for intrusion detection to the total number of packets in a flow. As seen in Table 1.1, the duration of network intrusion executions is also crucial. Consequently, we propose a duration-based earliness, which is defined as the duration of packets throughout the duration of a flow.

This work introduces early detection into the NIDS field, allowing NIDSs to detect intrusions before they reach target systems completely, putting NIDSs one step ahead of attacks. With this in mind, this study begins by proposing a novel framework for representing network flows as multivariate time series named Time Series Network Flow Meter (TSNFM). Additionally, a new early detection model, Multi-Domain Transformer (MDT), is proposed with the goal of significantly improving NIDSs' earliness without losing detection performance. As illustrated in Fig. 1.2, a conventional NIDS detects intrusions after the entire flow has been formed, whereas our proposed Network Intrusion Early Detection System (NIDES) is capable of detecting intrusions with only the first few packets, providing sufficient time for other security mechanisms (e.g., firewalls) to cut off malicious traffic before they arrive and damage target systems.

Table 1.1: Transmission Duration and The Number of Packet of Different Type of Payloads. The Target Machine (Windows Server 2012 R2) is Exploited via PsExec from Metasploit. (PSExec needs extra 88 packets for 2.13s)

Payload Type	Connection Type	Num of packets	Duration (s)
meterpreter	bind_tcp	437	2.33
	reverse_tcp	454	2.34
	reverse_http	267	31.80
	reverse_https	563	31.41
shell	bind_tcp	12	0.78
	reverse_tcp	10	0.10
powershell	bind_tcp	11	1.70
	reverse_tcp	10	0.10

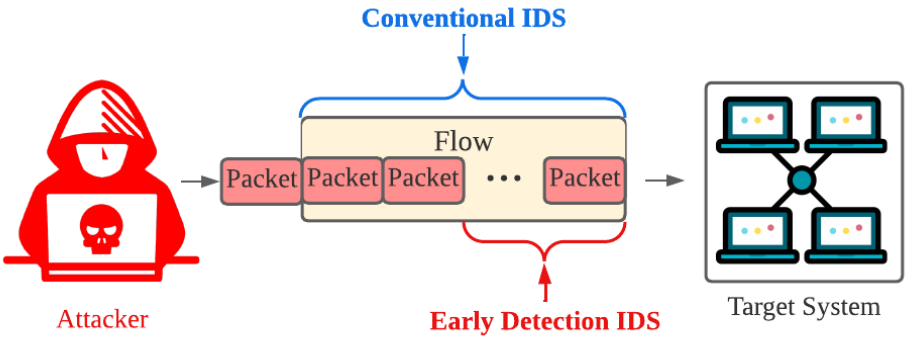


Figure 1.2: An Illustrative Comparison of a Conventional NIDS and an Early Detection IDS

1.5 Time Series Compression for NIDS Data

As we discussed in the last section, describing network flows/intrusions as time series can have various advantages; nevertheless, massive amounts of time series data are created, stored, and communicated as a result of the generation of IoT networks, and scientific research [235]. Transmitting such a large amount of time series can be costly in terms of network bandwidth and storage space [87]; consequently, many studies focus on compressing time series data with a high compression ratio [33]. Data compression can be roughly classified to two categories: lossless and lossy compression [47].

Lossless compression permits flawless recovery of the original time series such as bzip [192] and Sprintz [33]; nevertheless, the compression ratio of lossless compressors is relatively lower then lossy compressors [102]. Because of the intrinsic noise of time series, lossy time series compression can achieve a considerably higher compression ratio without compromising downstream tasks [65]. As lossy compression introduces errors to decompressed time series, it is important for lossy compressors to contain error-bound/distortion-constraint mechanisms to balance compression ratio and decompression errors [114].

AutoEncoder (AE) [27] is one of the important lossy time series compression techniques [196]. An AE encodes time series data as latent states with real-values and decodes them as prediction [222]. Compressed latent states are one of the major overheads of compressed data [217]. This work addresses the potential of encoding time series into Bernoulli distributed latent states rather than real-value latent states in order to drastically reduce the size of latent states and improve compression rate. In addition to AE, prediction-based compressors typically employ regression losses such as L1 and L2 [41]. This study identifies the issue of traditional regression losses and formulates a new loss to more accurately describe the problem based on the nature of entropy coders.

1.6 Motivations and Objectives of the Thesis

The overall objective is to establish defensive mechanisms under IoT and CPS environment to defend quickly and precisely against physical and cyber threats. With this in mind, the following demonstrates the motivations and objectives in details:

- Conventional access control techniques do not take the physical environment of users into account; for instance, unauthorized users can view credential information while authorized users leave the room without locking their devices. Thus, one of the

goals of this thesis is to use Wireless Sensor Network (WSN) to account for physical environment risks.

- Notwithstanding the advantages of IoT in smart surroundings, security concerns may arise. NSL-KDD is extensively used to assess IoT NIDS, however it is not an IoT-dedicated dataset. Thus, the purpose of this thesis is to validate and reproduce the NSL-KDD attacks in WSN (Contiki-NG) in order to assess IoT NIDS. In addition to conventional intrusions, another objective of this thesis is to establish an Advanced Persistent Threat NIDS dataset to investigate more sophisticated and complex attacks.

- Although various ML models have been developed to increase NIDS performance, network-based information can restrict their accuracy. Yet, host-based data can also be used to capture attack trails. Thus, we intend to validate the efficacy of host-based NIDS data.

- An ML-based NIDS can take a considerable amount of time to acquire distinguishing characteristics (CICIDS2017 using 2 minutes per sample). This thesis seeks to reduce response time without compromising too much precision.

- NIDS datasets can need a great deal of storage space and bandwidth; hence, this thesis also attempts a method to compress network-based data for more efficient data transmission.

1.7 Contributions of the Thesis

We summarize the contribution of this thesis as follows:

- Risk-Aware Smart Access Control (RASA) framework which is a context based access control system using unsupervised algorithm to infer the decision boundary. RASA is able to learn context information from widely spread IoT sensors and learn to define a context dependent and risk based access control scheme based on those contexts.

- Improving ML/DL-based NIDS from various perspectives including generating more datasets, boosting ML-based NIDS performance, combining multi-domain data, and reducing NIDS response time. 5 NIDS datasets used for ML/DL training: IoT-NSL-KDD, SCVIC-APT-2021, SCVIC-CIDS-2021, SCVIC-CIDS-2022, and TS- CICIDS2017. Attack Centric Method (ACM) combining advantages of multiple ML

methods to improve detection accuracy. CIDS-Net to fuse multi-domain (i.e., network and host) data. Multi-Domain Transformer (MDT) for reducing the detection time.

- A new compression framework named Deep Dict which achieves a higher compression ratio than state-of-the-art compressors. A novel Bernoulli Transformer-based AutoEncoder (BTAE) that can effectively reduce the size of latent states and reconstruct time series from the Bernoulli latent states. A new loss function namely Quantized Entropy Loss (QEL) which considers the characteristics of the problem and outperforms common regression losses in terms of compression ratio.

1.8 Structure of the Thesis

The rest of the thesis is organized as follows:

Chapter 2 introduce the background and literature study: Section 2.1 presents cyber-physical security continuous/implicit authentication/access control and risk-awareness. Section 2.2 further explains the current IoT network intrusions and APT detection, and illustrates the motivation of creating traditional and APT network intrusion datasets under IoT environment. Section 2.3 discuss NIDS, HIDS datasets and the datasets integrate network flows and host-based data. Section 2.4 lays the background for early detection on network intrusions. Section 2.5 summarizes the state-of-the-art of time series compressors.

Chapter 3 proposes an IoT assisted smart access control system. In this chapter, we present the system architecture of the proposed system, analyze the dataset we used in this chapter, and presents the numerical results of three different coupling mechanisms and clustering algorithms along with comprehensive discussions.

In Chapter 4, we reproduce the NSL-KDD attacks in IoT settings (i.e., Contiki-NG system), generate the intrusion dataset according to packets characteristics, and apply machine learning to detect and classify those intrusions. Within the section, we detail the settings and the topology of the IoT network, the attacks of NSL-KDD dataset that are threaten the Contiki-NG system, dataset generation strategy and ML-based detection approach. Furthermore, beside of traditional attack techniques from NSL-KDD, we further implement realistic APT attacks under IoT environment. A new ensemble algorithms named Attack Centric Method is proposed which builds classifier for each attack techniques.

Chapter 5 describes a novel Combined Intrusion Detection System (CIDS) which integrates NIDS and HIDS aiming at improving IDS performance. In order to combine

network and host-based data, we propose a CIDS dataset formation framework extracting features from network flows and various system logs. A transformer-based model is further proposed which can take network features and host-based data as input and predict attack types. Two CIDS datasets are created, named SCVIC-CIDS-2021 and SCVIC-CIDS-2022, out of the meta-data of well-known benchmark datasets CIC-IDS-2018 and NDSec-1. The results illustrate that host-based data can drastically improve the performance of NIDS by reducing the noise in the datasets.

Chapter 6 proposes a early detection system for network intrusions in order to reduce the responsive time of NIDS without sacrificing NIDS performance. With this in mind, we firstly propose a network flow-meter which describe the network flow as time series and propose a multi-domain transformer to extract useful features from the limited input. The results demonstrate that even with limited number of packets and duration of a flow, the proposed NIDS still can have a compelling performance.

Even though describing network flow as time series can boost NIDS response speed and detection performance, while storing time series data can cost huge amount of storage space and network bandwidth. Hence, Chapter 7 propose a lossy time series compressor, called Deep Dict, to reduce the size of time series data. Deep Dict consists Bernoulli Transformer AutoEncoder (BTAE) and Distortion Constraint. BTAE seeks to discover the Bernoulli latent states/representations of a time series and to recover the time series from the representations. Distortion Constraint limits the error of decompressed time series to a user defined range. We further propose a novel loss function, named Quantized Entropy Loss (QEL) to improve compression rate.

Finally, Chapter 8 concludes the thesis and summarizes the insights of numerical results.

Chapter 2

Background and Literature Review

This chapter discusses the background for access control and demonstrates a comprehensive survey regarding NIDS from various perspectives.

2.1 Access Control in IoT Environment

As we discussed in the previous chapter, NIDS can well protect IoT devices from network attacks and malicious traffic patterns [39]; whereas, since most of the IoT devices (e.g., tablet computers, camera, and WSN nodes) are easily captured or accessed by unauthorized users or legitimate users in insecure environment [129], access control plays indispensable role in IoT security solutions in order to provide a comprehensive protection for IoT devices [43]. In this section, we discuss and compare classical access control schemes such as MAC, DAC, and RBAC in detail; moreover state-of-the-art access control schemes are presented as well.

Some traditional authentication schemes, such as passwords or pincodes, can be inconvenient for some users. They also have well-known limitations to their effective security; for example, passwords set by mobile users can be determined by simple guessing [17], inferred from smudges left on users' screens [25], or stolen by malware [26]. These concerns have lead to plentiful prior work in cyberphysical context-dependencies for access control that has applied context dependency to multi-factor authentication, location-aware access control, and implicit or continuous authentication.

Multi-factor authentication establishes likely identity of users based on at least two pieces of evidences of identity, making identity assessment dependent upon multiple possibly-

contextual factors. Commonly, the factors relate to a users' unique knowledge (such as a private password), and physical possessions (something that is only in possession of the user such as a SecureID token). Common also is adding physical location to the authentication through definitions of geofences, such as a defined "home" location. Ramatsakane *et al.* [185] utilized location for improving accuracy of identity determination for authentication purposes.

So-called "implicit" authentication extends multi-factor one-time authentication could be more usable and secure [68, 122]. State of the art in context-aware access control calls for a system for continuous authentication that factors contextual risk into the continuous authentication in addition to other risks, such as the risk that the user is not who they appear to be, or that the resource being accessed is more sensitive than another [178]. Thanks to the rapid advancement of the Internet of Things (IoT), sensors can be utilized for easy-to-use additional access safeguards, such as through contextual, behavioral, and biometric signatures [183]. In particular, Internet of Things (IoT) devices and networks may be effective in recognizing certain context, as in the case of using sensors, communication and data (i.e. storage and analytics) as keys to IoT-based smart systems [89]. As Habibzadeh *et al.* [90] note, security is increasingly critical in smart access systems. Significant work on behavioural authentication has been published recently using a growing list of features collected through sensors including, but not limited to: smartphone touchscreens, wearables, and keystrokes [54, 207, 228].

Ashibani *et al.* [21] propose a framework which aims to reduce user intervention in access control by identifying the behavioral patterns about user-device interaction, such as service request and login duration. Rule-based behavior analysis aims for resiliency against password misuse, brute force attacks and unauthorized modification. Rauen *et al.* [187] proposed a primary-backup fashion to manage access control on smart mobile devices by using several machine learning algorithms that run on gestural information and spatiotemporal knowledge extracted from sessions on various applications. Thus, once the primary authentication module fails to authenticate a user (or fails to recognize a behavioral pattern), the backup module is activated. In case both primary and backup modules fails to authenticate a user implicitly, multi-factor authentication is triggered. Wu *et.al* [236] apply Support Vector Machines (SVMs) to classify genuine users as well as adversaries from keystroke signals. Lima *et al.* [146] introduced an architecture for behavioural data acquisition, and presented a behavior model with the building blocks of events, context, action and behavior. The architecture utilizes a belief analyzer and a recommendation filter to discover anomalous behaviour, and uncover hidden behavioural patterns. In addition to these, a probability analyzer is proposed to serve as a classifier that outputs three categories: normal, suspicious and abnormal. Hayashi *et al.* [94] proposed a framework

named "Context-Aware Scalable Authentication" (CASA) to make a trade-off between security and usability. In addition to multiple factors that contribute to a context, the authors report that location is the most influential factor on the context. Lee [138], a new continuous authentication system called "SmarterYou" is introduced to integrate the data from smart phone with wearable sensors. The proposed system builds on data analytics on these features in both time and frequency domains so to ensure fine-grained authentication. Flexibility is another important aspect to address in smart access control research as it can potentially compromise security. To address this challenge, Hulsebosch *et al.* [108] provide a secure and flexible access control scheme which builds on context sensitiveness and historical data analysis in the authorization of anonymous users.

While techniques to recognize behavioral patterns have improved, behavioral authentication still cannot guarantee the complete elimination of false positives [18]. In addition, even if false positives can be solved for authentication, implicit authentication alone cannot solve context-dependent risks that may result in undesired access.

Access control models such as Mandatory Access Control (MAC), Discretionary Access Control (DAC), Role-Based Access Control (RBAC), and Attribute-Based Access Control (ABAC) have been investigated by the researchers in this domain [66]. MAC offers operating system-constrained access control to manage process/thread operating files, network ports, memories, and devices. MAC is widely deployed on Linux, Windows, and databases. Rossi *et al.* [195] propose a framework that allows developers to establish fine-grained ad-hoc MAC strategies for applications, protecting the system from misbehavior (caused by bugs or compromised by attackers) of root privileged apps. Even though MAC provides high-level protection, it requires users to request permission for every resource. When compared to MAC, DAC is more flexible by allowing users to grant access to other users. Khan *et al.* [123] apply DAC combined with RBAC to access patient files dynamically. RBAC, unlike MAC and DAC, does not assign access to specific users but to pre-defined roles. By deploying RBAC, IT administration can operate more efficiently since RBAC can add or switch roles to cope with personnel changes. Cruz *et al.* [49] propose RBAC-SC, which utilizes RBAC and smart contracts to implement trans-organizational roles. Unlike static access control strategies, ABAC dynamically determines permissions according to a set of attributes such as user, resource, object, environment attributes. ABAC can achieve various levels of access control according to different requirements and environments in spite of its overhead. Ding *et al.* [59] propose combining ABAC and blockchain to cope with the massive connectivity of IoT so to avoid heavy role-engineering for various devices. As listed in Table 2.1, we compare the conventional access control schemes.

Besides the studies that aim to bridge access control with implicit authentication, identifying and assessing the risks of certain contexts is of paramount importance as well.

Table 2.1: Access Control Schemes Comparison

Access control scheme	Benefits	Limitation
Mandatory Access Control (MAC)	High security level; Fine-grained; Attackers cannot share or inherit access to other files	Difficult to maintain and configure; Difficult to scale to a large number of files and users; Users have to request permissions for new files
Discretionary Access Control (DAC)	Easy to maintain and configure; User friendly; Low implementation cost; Flexible	Less secure than MAC; Lack of centralized access management
Role-Based Access Control (RBAC)	Rules are transparent to users; Ease of use for the administrator; Flexible; Secure but no need to configure access for each user	Complex and laborious configuration of roles; Difficult to extend permission for individual users; Complexity is determined by the number of roles; Discards the principle of least privilege
Attribute-Based Access Control (ABAC)	Dynamically updates access permissions; Less effort after configuration; More control variables than RBAC	Low interpretability for the user; Scalability issues

With this in mind, the authors in [100] define a risk assessment scheme that associates the risk to the location. Cha *et al.* [38] introduce a data-driven risk assessment scheme that uses Data Flow Diagrams (DFDs) to model the flows of individual data and to recognize the components employed to process, store, and transmit data. Organizations can identify the potential/implicit incidences according to the associated components. Compared with asset-oriented and process-oriented approaches, the proposed approach can enhance the risk assessment accuracy by avoiding underestimating or neglecting the risks to sensitive individual data. In [120], Khambhammettu *et al.* propose a framework containing four threat assessment methods for subject-object accesses. Via assigning different weights to the sensitivity score of data/objects and the trustworthiness score of subjects, the proposed framework is flexible enough to satisfy the various preference of organizations. Wang *et al.* [231] present a new access control model that quantifies the risk of privacy violation in a statistical approach and further detects the physicians/users who over-access or misuse patients' private data. Atlam *et al.* [24] propose a risk estimation model that combines a new fuzzy logic algorithm and a set of rule-based policies to perform access control in IoT systems. To generate accurate and realistic risk values for each access request, the proposed fuzzy logic system, which involves twenty experts' effort, converts experts' qualitative expression into numeric values and offers a dynamic and context-aware access control by leveraging the contextual features such as resource sensitivity, action severity, and risk history.

This section indicates a gap remains between continuous/implicit authentication/access control and risk-awareness, particularly in cyberphysical environments where actions do not only appear in cyberspace but also stem from physical behaviors, interactions and roles.

2.2 IoT Network Intrusion Detection

As one of the adequate security mechanisms, Intrusion Detection System (IDS) is required for IoT environments to mitigate intrusion impacts. In general, IDSs fall into two categories: Network-based IDS (NIDS) and Host-based IDS (HIDS). Since IoT devices have limited resources (e.g., computation and memory), a centralized NIDS system is considered in this chapter. Another important characteristic of IoT is that IoT devices use tailored or particularly designed protocols such as IPv6 over Low Power Wireless Personal Area Networks (6LoWPAN), compressed UDP, RPL, and CoAP. Both feature extractor and machine learning algorithms should be able to deal with these protocols in different layers; for instance RPL and 6LoWPAN located in network layer provide extra features including length/status of compressed protocol, RPL types (ie.e, DIS, DIO, DAO), and RPL instance

id, while application layer protocol COAP can offer features like type of messages, and the length of tokens. Therefore, through this sub-chapter, we go through current IoT NIDS solutions and emphasis the need for IoT NIDS datasets.

Fu et al. [194] report the challenges in obtaining IoT intrusion detection datasets. In addition to the heterogeneous structures in IoT networks, the large scale deployment and distributed topology characteristics of IoT environments also challenge the existing centralized IDS techniques [172]. Due to the lack of public intrusion datasets or benchmarks such as the NSL-KDD dataset, researchers have to set up their unique network topology and generate attacks on it. Some researchers also insert attack records in regular traffic records. Pajaouh, et al. [173] apply the proposed IDS directly on the NSL-KDD dataset. Elike et al. [101] experiment with their IDS on their own IoT network, which contains five nodes, to simulate DDoS attacks. Fu et al. [194] employ Intel-Lab datasets and manually append some records for attacks. Mahmudul et al. [92] leverage Distributed Smart Space Orchestration System (DS2OS) traffic traces; DS2OS is a middleware for storage and brokerage of the state context. In that study, several attacks are also introduced to the environment manually. In [245], Zhang et al. employ Cooja, a Contiki simulator, to generate Distributed Denial of Service (DDoS) attacks and defend by checking consistency. To tackle the heterogeneous attribute of IoT networks, Nadun et al.[184] propose a framework allowing researchers to build their own intrusion dataset by inputting the network packet traffic as raw PACP files. Koroniotis et al. [130] simulate normal traffic using the MQTT protocol in Ubuntu virtual machines, and apply Kali (a operating system designed for penetration testing) to generate BotNet attacks. Argus and Bro-IDS are further used by the authors to extract features. Finally the dataset is fed into three ML algorithms: SVM, Recurrent Neural Network(RNN), and Long Short Term Memory network (LSTM). Gara et al. [79] utilize Cooja as a simulator and simulate the black-hole and gray-hole attacks targeting 6LoWPAN protocols. Instead of using ML algorithms, the authors propose threshold-based solutions to detect such attacks. Sagduyu et al. [197], implement DoS, spectrum poisoning attack, and priority violation attack through adversarial ML algorithms that build on Feed Forward Networks (FFN) to monitor the wireless channel.

Traditional threats like DoS and Probe used in NSL-KDD still threaten the IoT networks, and the current literature: 1) explores the feasibility of ML algorithms under network intrusions datasets that are tailored for non-IoT context, and/or 2) introduces attacks targeting routing protocols in IoT environments. Based on this observations, we integrate the conventional attacks in NSL-KDD dataset into an IoT environment, and validates the efficiency of various ML algorithms in this dataset.

The NSL-KDD dataset has been widely used in numerous studies to validate Network IDS (NIDS) systems and ML algorithms. Four different attack types are present in the

NSL-KDD dataset including Denial of Service (DoS), Remote to Local (R2L), User to Root (U2R), and Probe attacks. Further attack techniques are used under these four categories. Features in NSL-KDD offer rich information to identify the malicious traffic. For instance, essential features are extracted from the headers of packets revealing the necessary information of packets, content features hold the information of payloads, time-based features offer the analysis of the traffic input over two seconds, and host-based features analyze the behaviour over a series of connections established.

The most commonly used protocols in WSNs are IPv6, TCP, and UDP, whereas protocols such as FTP, mail, SNMP, ARP, and XTerm are not usually seen in WSN environments. Moreover, some of these attacks are designed specially to target Windows and Linux Operating Systems. Thus, these attacks may not be suitable when they are introduced to constrained systems like TinyOS or Contiki-NG. It is particularly worth noting that R2L and U2R are not in the scope of this study since we particularly focus on network layer protocols where user involvement such as login and elevating privileges is not present. More specifically, DoS and probe attacks are the most practical to introduce and test on resource limited devices. Based on these observations, we introduce these types of attack into the simulated environment while multiple ML algorithms are integrated with the IDS to explore their ability to detect these attacks.

Due to the significant risk of APT, an increasing number of studies have focused their efforts on building efficient NIDS against APT. Several of them propose graph analysis and hidden Markov models as candidates for intrusion detection systems [232, 81]. These approaches are responsible for predicting the attack trace within the internal network and allow administrators to take measures to avoid a deeper infiltration. However, the traditional approaches have several flaws since they are ineffective if the APT does not follow the predicted path indicated by the defensive techniques. Furthermore, wrong predictions by the defensive techniques is not of low probability due to the adaptable nature of APT attacks [237]. With this in mind, ML-based NIDSs are introduced to address the shortcomings of conventional methods [152]. A common way of implementation is as follows; ML-based monitoring is integrated with the firewall to provide a more comprehensive protection solution [46].

The training dataset for defensive machine learning models is one of the most critical requirements, and the quality of data affects the performance of machine learning models used in APT detection. However, there are few publicly accessible benchmarks for APT [251] due to the following reasons: 1) internal network data is private, 2) attack strategies vary according to the network structure. Since limited NIDS dataset contains complete APT stages as analyzed in Table 2.2, this thesis aims to construct a new APT dataset which contains actual APT attacks and complete APT stages, bridging the gaps of previous works

Table 2.2: Comparison of existing APT Datasets and the new SCVIC-APT-2021 dataset

Dataset	# of APT Stages	Network/Host	Synthetic/Real	Labels
Friedberg et al. [76]	2/6	Network	Synthetic	APT Stages
Siddiqui et al. [206]	-	Network	Real	Malware Names
Ghafir et al. [80]	6/6	Network	Syntheic	APT Stages
Laurenza [135]	-	Host	Real	Malware Names
Myneni et al. [167]	3/6	Network	Real	APT Stages
SCVIC-APT2021(Ours)	6/6	Network	Real	APT Stages

Table 2.3: Summary of Datasets that Contain Network and Host Data

Dataset	Network-based Features	Host-based Features	Network Meta-data	Host Meta-data	Environment	Labels
DARPA98/99	28	13	Tcpdump	Logs Audit Data File Systems	Simulated	Attack Traces Provided
SSHCure	5	Not Provided	Not Provided	Not Provided	Realistic	Labels Provided
Kent	3	5	Not Provided	Not Provided	Realistic	Labels Provided
NDSec-1	11	5	PCAPNG	Logs Audit Data	Realistic	Attack Traces Provided
Unified Host and Network Dataset	4	20	PCAP	Not Provided	Realistic	Not Provided
CIC-IDS-2018	76	Not Provided	PCAP	Logs	Realistic	Attack Traces Provided
SCVIC-CIDS-2021 (Ours)	**76**	**768**	**From CICIDS2018**	**From CICIDS2018**	**Realistic**	**Labels Provided**
SCVIC-CIDS-2022 (Ours)	**11**	**768**	**From NDSec-1**	**From NDSec-1**	**Realistic**	**Labels Provided**

in the literature.

This section demonstrates the lack of dedicated IoT environment datasets, especially those containing IoT protocols characteristics and attacks designed for IoT devices. Hence, network intrusion benchmarks are needed in IoT environments.

2.3 Combination of NIDS and HIDS

Flow-based NIDS, in general, is operated by sniffing packets from networks, producing network flows, extracting statistical features from network flows, and making judgments based on those features. Thus, the NIDS dataset's primary structure is tabular. To enhance detection performance, a range of machine or deep learning algorithms have been proposed in the recent years. Marteau et al. [159] propose DiFF-RF, a semi-supervised technique for detecting anomalies in network flows that overcomes the constraints of Isolation Forest. DiFF-RF is tested against a variety of network intrusion datasets, including the ISCX, UNSW-NB15, CIDDS, and Kitsune. Xu et al. [238] propose the FC-Net framework, a few-shot DNN framework for detecting malicious network flows utilizing a limited amount of training data. The proposed approach can be easily transferred to another dataset. Bitton et al. [32] construct their own network intrusion dataset by injecting the most

recent Remote Desktop Protocol (RDP) Common Vulnerabilities and Exposures (CSVs) and propose a fine-grained CBLOF model for detecting network traffic anomalies. Li et al. [143] utilize incremental learning in their proposed ensemble method and evaluate it on the NSL-KDD dataset and Key Infrastructure Protection Center datasets from Mississippi State University.

Recent studies on HIDS, particularly system log-based HIDS, have relied heavily on Natural Language Processing (NLP) approaches, due to the text-based nature of log entries [96]. Brown et al. [36] propose a Recurrent Neural Network (RNN) model based on an attention mechanism for classifying anomalies in log data. Tokenized log entities are fed into the proposed model. Not only the predictions are output, but also the relationships between the prediction and the features. Du et al. [64] offer DeepLog, which makes use of a Long Short-Term Memory (LSTM) network to represent log entries as time sequences and to discover anomalies in log files, which are defined as data that do not conform to normal log entry distributions. To mitigate the imbalance problem of intrusion detection dataset and boost model performance, Studiawan et al. [211] present Pylogsentiment, which classifies positive and negative sentiment in log files using the TomLink balancing dataset and a Gated Recurrent Unit (GRU) network. As text cannot be directly fed into GRU, an embedding layer is utilized as the input layer to convert it to hidden states. Nedelkoski et al. [169] incorporate transformer which is a new breakthrough in the field of Natural Language Processing (NLP) into their Logsy log anomaly detection framework. Logsy begins with an embedding layer and then a transformer encoder as a classifier.

After the release of DARPA98/99, multiple IDS datasets integrating network- and host-based data are generated to reflect more up to date intrusions, realistic network environments and normal traffic patterns. The following works are ordered chronologically as follows:

KDD98/99 [171, 215] contains both network packets and host-based data; nevertheless, the dataset is criticized because the synthetic network environment and outdated cyber intrusion methods. Moreover, only a small number of features are extracted from the host-based data. SSHCure dataset [98] is created to identify and prevent SSH incursions. Although Hellemons et al. gather network packets and system log files, their study focuses primarily on the features of network traffic. Kent dataset [119] contains realistic network flows and system log files from the Los Alamos National Laboratory's internal network. Network and host-based features are extracted; nevertheless, they are created separately and not aligned. The number of features is also restricted; for instance, only three network features and five host features are provided. Beer et al. [31] create the NDSec-1 dataset at the University of Applied Sciences Fulda and comprises a range of intrusion methods in addition to PCAP and text-based logs. The authors extract features from both network

traffic and system logs, while only five host-based features are obtained [30]. During a three-month period, the Unified Host and Network Dataset [223] is generated by gathering network flows and Windows logs from the corporate network of Los Alamos National Laboratory. Due to the absence of labels, the dataset is hard to analyse and assess. For instance, Beazley et al. [29] examine the dataset while the attacks cannot be identified precisely. Sharafaldin et al. [202] generate CIC-IDS-2018, including 50 perpetrators, 420 victim workstations, and 30 servers. Most of popular intrusions, according to research by McAfee, are introduced into target networks. The collected meta data compromises PCAP files, log files, and information on labelling. CICFlowMeter [134] is utilized to extract features from network packets whereas the host-based data is not being processed. Vinayakumar et al. [229] attempt to develop a framework for incorporating NIDS and HIDS while examining the individual performances of DNN on NIDS and HIDS datasets. Lu et al. [156] establish a virtual environment, implement four distinct types of cyber-attacks, and retrieve network traffic and multiple types of log files from IDS devices; however, most of the features extracted from logs are network-based features, including source IP, destination IP, and the number of bytes sent/received. Table 2.3 summarizes the characteristics of the datasets that include both network- and host-based data.

In attempt to incorporate network- and host-based features, previous work on combining NIDS and HIDS seems to have driven host-based data into a tabular format. Limited number of arbitrary features are extracted from host-based data (such as [215, 119, 31]), resulting in the loss of a substantial quantity of relevant information. Nonetheless, the recent success of HIDSs demonstrates the effectiveness of NLP techniques thanks to the text nature of host-based data [96]. With these in mind, forcing network-based and host-based data to the same format (e.g., tabular) limits the integration of state-of-the-art of HIDS methods and losses huge volume of information of host-based data.

In order to make optimal use of information from host-based data and intelligently extract features from host-based data, this thesis proposes a novel framework for CIDS dataset creation and a Transformer-based CIDS-Net model.

Instead of manually extracting features from host-based data, the CIDS Dataset Formation Framework maintains the maximum amount of original host-based information. In addition, by utilising NLP techniques, CIDS-Net accepts as inputs both the original host-based data and network-based features, automatically learns the feature representation, and makes the final predictions. As it is difficult to gather a large number of samples of specific types of attacks, the proposed CIDS system makes full use of the input data.

2.4 Early Detection on Network Intrusions

As this thesis combines early detection and NIDS to achieve quick response against attacks, this sections introduce related work regarding early detection and NIDS. Resende et al. [189] propose an anomaly-based intrusion detection system that is optimized using a genetic algorithm. They also integrate the other two approaches to achieve high performance. Marir et al. [158] propose a distributed intrusion detection system that combines Deep Belief Networks (DBN) and Support Vector Machines (SVM) to detect attacks in large volumes of traffic. The proposed strategy is able to significantly increase baseline performance. Min et al. [164] propose the SU-IDS AutoEncoder network, which learns from unlabeled examples using semi-supervised and unsupervised methods. Xu et al. [238] introduce few-shot learning into NIDS field. The authors extract image-like features from raw network packets and propose FC-Net, which allows NIDS to detect new types of intrusions with limited samples without sacrificing too much detection performance. Marteau et al. [159] propose DiFF-RF, a semi-supervised technique for detecting anomalies in network flows that overcomes the constraints of Isolation Forest. DiFF-RF is tested against a variety of network intrusion datasets. Chen et al. [46] propose an ensemble approach named All Predict Wisest Decides (APWD) that trains various machine learning models and relies on the decisions of specific ML models only if they outperform the rest of the classifiers for certain traffic types. Ding et al. [58] employ 1NN to multi-variate time series data; however, extracting useful features from time series remains a challenge; hence, the inference time is long. Ghalwash et al. [82] propose a method named Multivariate Shapelets Detection (MSD), which is a shapelet-based algorithm that incorporates HMM and SVM models. The proposed MSD is capable of identifying critical decision-making segments. Lin et al. [148] propose the REACT methodology for effectively classifying multivariate time series with a variety of features. The authors improve the overall performance of the proposed approach by extending it to GPUs. Huang et al. [107] propose the Multi-Domain DNN (MDDNN), which analyzes time series in both the time and frequency domains and incorporates CNN-LSTM to extract feature representations. Hsu et al. [103] propose an ETSCM based on MDDNN that utilizes an attention (weighted average) mechanism to make decisions based on segments with a high confidence score. ETSCM is capable of identifying critical components of time series, hence increasing the model's interpretability. We summarize the previous work in both domains (i.e., NIDS and early detection) in Table 2.4 Previous research in both domains has yielded promising results; however, they have not been integrated to improve the earliness of a NIDS. Thus, this thesis introduces early detection into NIDSs, enabling a NIDS to respond quickly before the entire flow is formed, so intercepting attacks before they reach target systems.

Table 2.4: Summary of Related Work of NIDS and Early Detection

Authors	Field	Data Format	Earliness	Deep/Machine Learning Method
Resende et al. [189]	NIDS	Tabular	No	Genetic Algorithm-based Adaptive Approach
Marir et al. [158]	NIDS	Tabular	No	DBN, SVM
Min et al. [164]	NIDS	Tabular	No	AutoEncoder
Xu et al. [238]	NIDS	Image Like	No	FC-Net
Marteau et al. [159]	NIDS	Tabular	No	DiFF-Random Forest
Chen et al. [46]	NIDS	Tabular	No	APWD
Ding et al. [59]	Medical	Time Series	Yes	KNN
Ghalwash et al. [82]	Medical	Time Series	Yes	MSD
Lin et al. [148]	Medical	Time Series	Yes	REACT
Huang et al. [107]	Medical	Time Series	Yes	MDDNN
Hsu et al. [103]	Medical	Time Series	Yes	ESTSCM
Ours	NIDS	Time Series	Yes	MDT

2.5 Lossy Time Series Compression

Describing network flow/attacks as time series can be costly in terms of storage space and network bandwidth; thus, this study develops a lossy compressor to reduce the size of data. This section summarizes the state-of-the-art of time series compressors.

Liang et al. [144] propose SZ2, a framework for adaptive prediction-based compression with Lorenzo or linear regression as the predictor. In addition to compressing data with a specified error constraint, SZ2 maximises Peak Single-to-Noise Ratio (PSNR) to ensure the integrity of recovered data with a high compression ratio. SZ2 has been evaluated on four scientific datasets including over a hundred domains. The results indicate a higher compression ratio and RSNR compared to state-of-the-art compressors such as SZ1, ZFP [149], and ISABELA [133]. The Sprintz framework proposed by Blalock et al. [33] is a lossless time series framework built for IoT networks. Sprintz aims to compress 8- or 16-bit integer data but lacks optimization for floating-point data. Vestergaard et al. [227] propose a lossless compression scheme which identifies the data chunks worth deduplicating and removes the repeated chunks. Their results illustrate that their proposed scheme can outperform GZip [10] and 7Z [136]. LFZip is a machine learning-based prediction-quantization-entropy coder lossy compressor proposed by Chandak et al. [41]. The auto-regressive predictor of LFZip has two types: Normalized Least Mean Square Predictor (NLMS) and Bidirectional GRU for learning nonlinear patterns in time series. L2 is used as the loss function and Maximum Absolute Error is used as the distortion function in LFZip. Results indicate that LFZip outperforms SZ2 on many time series datasets, including accelerometer/gyroscope data from smart watches, blood volume pulse data from PPG devices, and sensor data

Table 2.5: Summary of Lossy and Lossless Compressors

Authors	Compressor	Error Bounded	Lossy/Lossless	Model	Loss Function
Liang et al. [144]	SZ2	Yes	Lossy	Linear Regression, Lorenzo	L2
Blalock et al. [33]	SprintZ	-	Lossless	Delta Encoding, FIRE	L1
Vestergaard et al. [227]	GD	-	Lossless	Generalized Deduplication	Mutual Information
Chandak et al. [41]	LFZip	Yes	Lossy	NLMS, Bi-GRU	L2
Yu et al. [243]	AMMMO	-	Lossless	Reinforcement Learning	Compression Ratio
Zhao et al. [249]	SZ Auto	Yes	Lossy	Second-Order Lorenzo/Regression	L2
Zhao et al. [248]	SZ3	Yes	Lossy	Lorenzo, Spline Interpolation	L2
Xu et al. [239]	LSTM-AutoEncoder	No	Lossy	LSTM-based AutoEncoder	L2
Azar et al. [205]	1D Unet	Yes	Lossy	U-Net	Cross Entropy
Sun et al. [213]	LCR	Yes	Lossy	RNN	Cross Entropy
Ours	Deep Dict	Yes	Lossy	Bernoulli Transformer AutoEncoder	QEL

generated by Siemens instruments. Yu et al. [242] propose an Adaptive Multi-Model Middle-Out (AMMMO) approach for lossless reinforcement-based compression. AMMMO contains a two-level model which can decide compression techniques for each data point. As AMMMO is a byte-level compressor, it supports floating-point data well. Zhao et al. [249] enhance SZ2 by presenting SZ-Auto, which utilises second-order regression and Lorenzo and can efficiently find hyperparameters. The results indicate that SZ-Auto can enhance baseline compressors by 46%. Based on SZ2, Zhao et al. [248] offer SZ3, which replaces linear regression with a dynamic spline interpolation approach. SZ3 can identify nonlinear patterns, resulting in a high compression ratio and superior data quality (PSNR). It can be said that SZ3 can outperform SZ2 by a factor of four under certain datasets. Xu et al. [239] propose an AutoEncoder based on LSTM [243] to compress the Air-Quality dataset [52], and the proposed approach can fit non-linear time series, however it has considerable distortion owing to the lack of an error bound (5 to 12.57 Mean Absolute Error). Lossy Compression and Replay (LCR) is an RNN-based lossy compressor proposed by Sun et al. [213]. LCR represents communication traces as time series data and uses an RNN model to compress them. Their investigations demonstrate that LCR may drastically decrease the amount of the original data. Azar et al. propose a light SZ compressor which combines U-Net [205] to enhance data recovery. Their proposed lossy time series compressor is deployed in IoT devices (LoRa) and the results demonstrate that their method can outperform SZ1 [57] in terms of compression ratio and energy consumption.

Related works of compressors are summarized in Table 2.5. This research utilises a prediction-quantization-entropy coder paradigm and proposes Deep Dict to enhance prediction abilities. The proposed method can greatly reduce the size of latent states in comparison to conventional AutoEncoder-based compressors. Loss functions for conventional prediction-based compressors are L1/L2. This work demonstrates the drawbacks of conventional regression losses and proposes a novel loss function, QEL, which aimes to

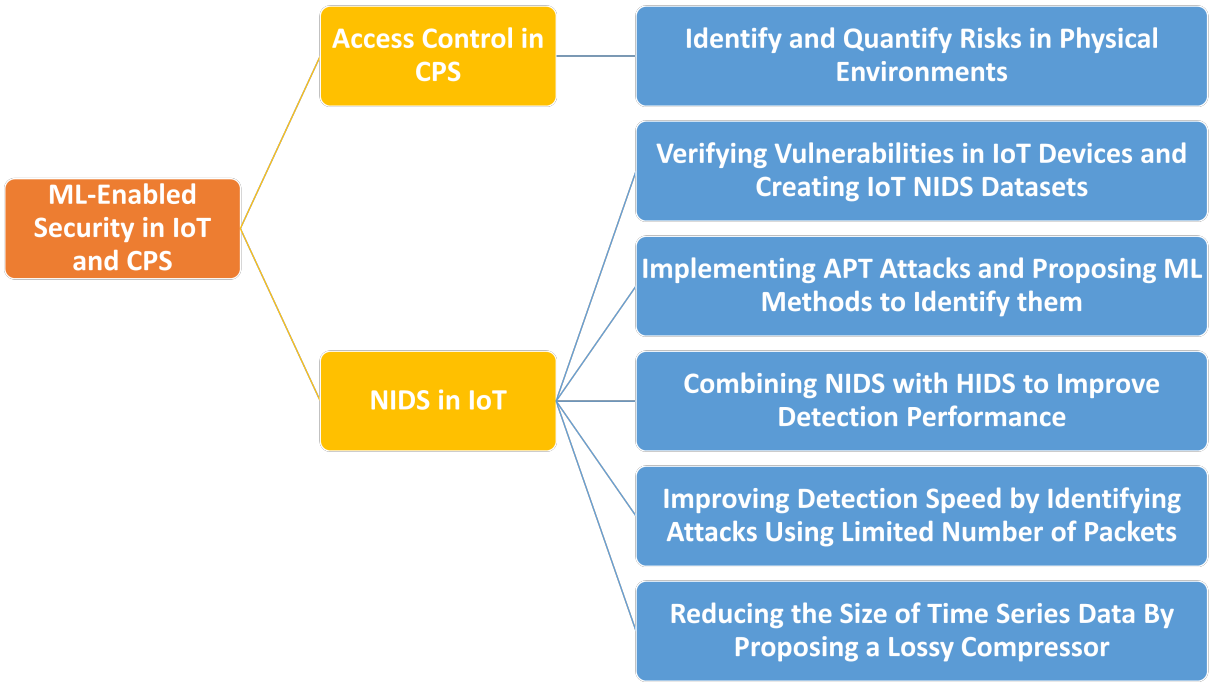


Figure 2.1: Summarize of Gaps and Improvements of This Thesis

improve the compression ratio. Since prediction-quantization-entropy coding scheme has been widely utilised for lossy signal, picture, and video compression, QEL is suitable for all compressors that use uniform quantization.

To conclude this chapter, we illustrate the gaps and our improvements in Fig. 2.1.

Chapter 3

Risk-Aware Fine-Grained Access Control in Cyber-Physical Contexts

3.1 Problem Statement

Access control aims to minimize unauthorized users accessing credential information or essential assets. Traditional access control schemes (e.g., MAC, RBAC, and ABAC) only verify users' identification without considering the operational context and interaction with the environment, which causes potential security issues when malicious users can physically access assets authorized by legitimate users. For instance, critical information (such as patients' Private Health Information) could be leaked if legitimate users leave the device unlocked with unauthorized users. With the development of IoT, especially Wireless Sensor Network (WSN), such problems can be mitigated by monitoring and collecting user and device actions from IoT sensors. With this in mind, IoT networks are assumed to be deployed at doors and devices to record users/devices entering/exiting a room, a document being accessed on a device. We propose the Risk-Aware Smart Access Control (RASA) framework which takes those actions as input and evaluates the risk of each action with the consideration of the environment. Take a healthcare environment as an example; an authorized physician attempts to access a patient's PHI. Without prior knowledge of users' identification, the RASA framework not only can identify whether the physician is authorized but assesses the actions' risk by checking whether the document is often read in the current room and how possible the surrounding people may endanger the patient's privacy. The following sections will further illustrate our methodology in detail.

3.2 RASA: Risk-Aware Smart Access Control

This section presents the system overview along with the cyber-physical action logs in a medical emergency room. We introduce the data pre-processing steps, feature extraction and risk inference methods in detail.

In our framework, actions are transformed into events which include contextual information that can be recognized through coupling mechanisms. Coupling defines interaction patterns between the objects, which can be recognized through clustering algorithms, that use couplings as the input features. Each coupling feature is associated to a risk level. Upon obtaining the clusters, an aggregate value of the associated risk levels of the features of all data points (i.e. actions) in a cluster is calculated as the risk value of that cluster, (i.e. label of cluster). Note that couplings can be formulated based on time, frequency or a combination of time and frequency.

The bold letters in the notation represent collections/sets of factors (***Ppl***, ***Dev***, ***Doc***, and ***Loc***), and lower case letters stand for the elements of the specific collection, e.g. $\boldsymbol{A} = \{a_1, ..., a_i, ..\}$. $\boldsymbol{A}$ and $\boldsymbol{B}$ denote any pair of the four collections/sets. C and R indicate coupling and risk respectively and can be denoted in terms of Frequency ($Freq$) and/or Duration (Dur). $F_n^{\boldsymbol{A},\boldsymbol{B}}$ denotes the n^{th} coupling feature extracted (from the n^{th} event (E_n)).

3.2.1 System Overview

The overview of the proposed RASA scheme is presented in Fig. 3.1. The use case in this study involves an emergency room of a healthcare organization where sensors record activities of people (i.e., patients and physicians) and devices, such as entering or exiting some locations and reading documents/records (as examples in Table 3.1). In order to protect patient privacy and physical security, protection systems should raise alarms or introduce access control levels against hazardous actions, such as a read access to a document by an unfamiliar patient or entering into unfamiliar rooms. The proposed scheme aims to quantify the risk of given actions and find potential threats without prior knowledge (e.g., document ownership, patient wards). Nevertheless, if only the action were considered, not only the context information would be overlooked but similar actions would be repeated, which might have led to many false alarms. Therefore, in the proposed framework, the first step is to reproduce the events where actions are performed. Then, these events are used for calculating the couplings which are then clustered and at the same time assigned some

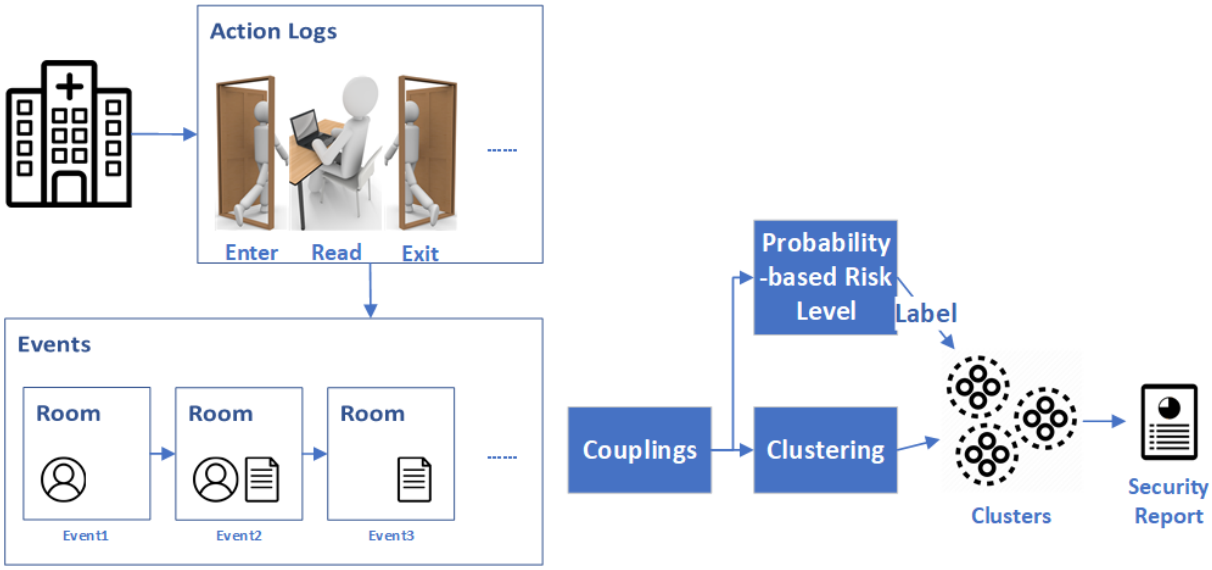


Figure 3.1: Overview of System Architecture

Table 3.1: Action Log Examples

time	act	agent	device	document	location	monitor
06-26 19:00	enter	actor: 35...9d			loc: fe...fb	dev: bd...77
06-26 20:02	exit	actor: 50...df			loc: fe...fb	dev: bd...77
06-26 19:31	read		dev: ae...2e	doc: 9a...de		

risk levels. The risk levels of couplings determine the risk level of a cluster of couplings. Finally, the risk levels of clusters are verified against a pre-defined policy.

An argument for using couplings as features described is as follows. Since the risk level is not only determined by actions but also their context, the actions are initially transformed into events to obtain the surrounding information, e.g. location, and co-presence. The coupling concept (see Section 3.2.3 for more details) is introduced to infer the relationship of objects in action logs. Coupling values are extracted from events and normalized. These values are stored in coupling matrices to explore their distributions and are marked with high, medium, and low-risk levels according to their mean value and standard deviation. Coupling features are fed into clustering algorithms. In this chapter, three different coupling features are used: frequency-based features, duration-based features, and a combination of them. A simple scenario that would trigger the RASA framework to compute coupling values is illustrated in Fig. 3.4. One physical action

of entering a room and one cyber action of accessing a document on a device is given as an example.

After the computation of couplings, their risk levels, and forming clusters of couplings, the risk of each cluster is calculated based on the number of high, medium and low labels of couplings in that cluster. Then, the cluster is labeled according to cluster risk value. Furthermore, the risk of the entire action log can be determined by the same calculation to provide the general risk information to the security experts to raise alarm.

In a privacy-sensitive environment such as a clinical setting, the inferred risk context can be utilized to define a rule-based access control policy managing access and defining risk-based system actions based on calculated risk. Given a risk level of a cluster of actions or events, a policy can be defined wherein access decisions are made per read attempt by calculating the overall risk within the current context. RASA estimates overall risk from the weighted sum of all per-feature risk values. Rather than assigning equal weights to the features, the policy author may adjust the weights of these coupling features based on direct inputs from subject matter experts. For instance, when patient documents are the assets considered, the couplings related to documents may be assigned greater weight factors compared to other couplings that do not consider documents. Furthermore, the policy may permit or deny read actions by tuning a threshold value for the risk factor. Although such policy definitions require a threshold to be tuned, the decisions are over clusters rather than individual actions, which is more scalable, stable, and explainable for policy analysts and users alike. In the conclusion part, we elaborate on AI outcomes versus policy decisions in order to analyze the efficiency of the AI results.

In the following sections, we explain the components of RASA in detail.

3.2.2 Action logs

In RASA's model, action logs consists on data of physical activity such as entering or exiting a room as well as cyber activity such as logging in a device or accessing a document. In a multi-user distributed system, the access control of documents is effective to all kinds of actions. Regardless of the devices used in a environment, wired or mobile, the access of a document can always be abstracted in association with user movements such as entering or exiting a room, device usage, and operating documents. Our simulated action logs are inspired by typical actors in clinical, in-person setttings.

An example of sequence of actions is given in Fig. 3.2. When users or devices enter or exit a specific location, the ID of the room and the ID of the user are logged with a timestamp. If a user attempts to access a document, the device first requests grant

Figure 3.2: Actions Examples

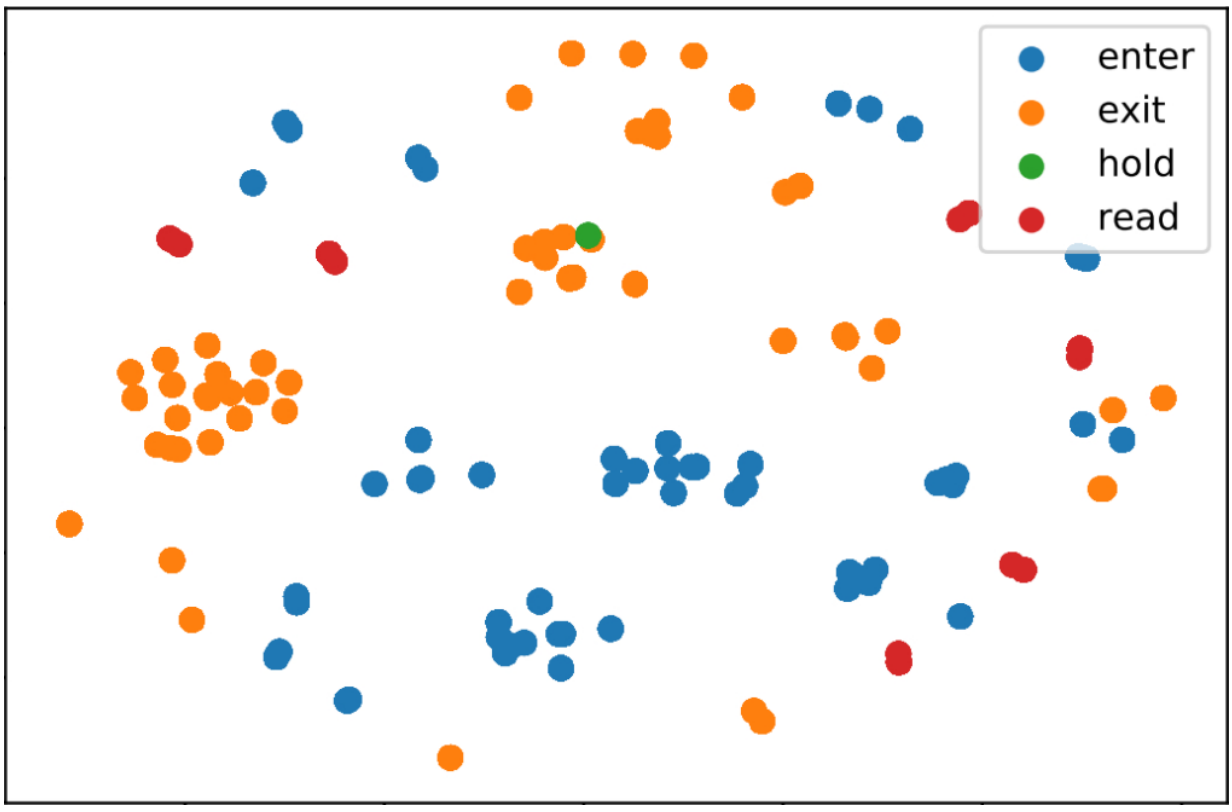


Figure 3.3: Visualization Result of Raw Action log using t-SNE

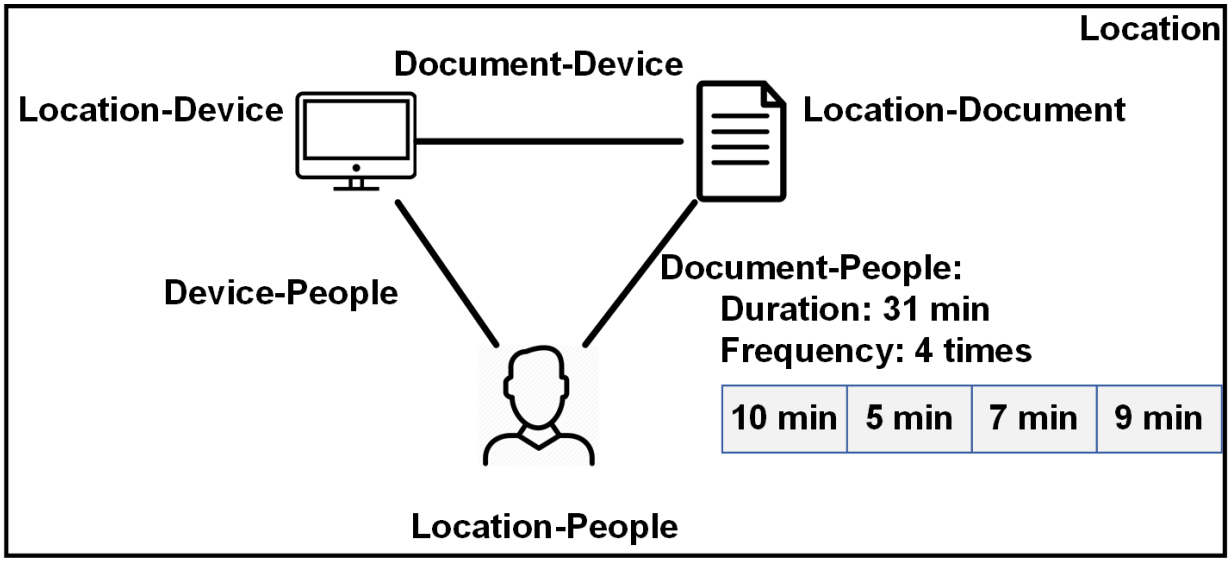


Figure 3.4: A simple scenario to illustrate the coupling concept

for access, and this attempt is also logged. The distribution of the simulated actions are visualized in Fig. 3.3 by using T-distributed Stochastic Neighbor Embedding (t-SNE). As a widely used and effective statistical visualization technique, mapping high dimensional data onto 2D or 3D, t-SNE calculates the probability distributions of each pair of data samples in high and low dimensional spaces, assigning high probabilities to similar ones and minimizes two distributions Kullback–Leibler divergence. t-SNE can exhibit the data distribution lying in various manifolds and clusters, while the distance between two samples in low dimensional space do not represent the distance in high dimensional space but the probabilities [157]. Hence, the axes in the plot do not represent physical units. By visualizing the dataset with t-SNE, we aim to understand the dataset distribution, discover potential clusters, and analyze the complexity of the problems. General characteristic of the actions can be formulated as moving these objects (i.e. people, documents and devices) across different locations over time. Thus, an event or a state machine is defined as a combination of time, location and a set of objects in this space.

3.2.3 Feature Extraction and Risk Inference

The coupling concept aims to reveal the interactions among different factors of action logs. As illustrated in Fig. 3.4, coupling can be defined between people and location, location and device as well as location and document. In the figure, the interaction between two factors:

document and people is presented in terms of duration and frequency. Duration denotes how long two objects spend time with each other; frequency denotes how many times two elements encounter each other. Using the logs, RASA first constructs the frequency and duration matrices for each object pair (i.e. person-person, person-device, device-location, person-location). The coupling values in the matrices are normalized to enable comparability between different coupling values.

The coupling factors are denoted as: $\boldsymbol{Ppl} = \{p_1, p_2, p_3, ..., p_n\}$, $\boldsymbol{Doc} = \{doc_1, doc_2, doc_3, ..., doc_n\}$, $\boldsymbol{Dev} = \{dev_1, dev_2, dev_3, ..., dev_n\}$, $\boldsymbol{Loc} = \{loc_1, loc_2, loc_3, ..., loc_N\}$. The computation of coupling values is given in Eq.3.1 and Eq.3.2 for frequency and duration, respectively. In the RASA framework, the first step is converting the action logs to events. Algorithm 1 presents the generation of event list dictionary which stores the status transitions of all factors. As discussed in Section 3.2.2, an action log contains a factor (e.g., a person, document, or device), an action (e.g., enter, read, or exit), and a location. For each action log, the algorithm determines the previous event, creates a new event according to whether the action is to move the factor in or out, and finally updates the statuses of all elements of the new event. It is worth to note that factors do not include locations as opposed to the element of an event. The event list dictionary is further processed by Algorithm 2 to generate $Freq_{a_i,b_j}$ which counts the number of times that a_i and b_j coexist in the same event and Dur_{a_i,b_j} which records the duration that two elements occur in the same event.

$$C^{Freq}_{a_i,b_j} = \frac{Freq_{a_i,b_j}}{\max_i Freq_{a_i,b_j}} \tag{3.1}$$

$$C^{Dur}_{a_i,b_j} = \frac{Dur_{a_i,b_j}}{\max_i Dur_{a_i,b_j}} \tag{3.2}$$

If $Freq_{a_i,b_j}$ and Dur_{a_i,b_j} are considered, they can be abstracted as G_{a_i,b_j}. Then, Eq.3.3 represents a coupling matrix (not yet normalized). The time window of the distribution is determined by the tuple (k_{start}, k_{end}). T_{E_k} indicates the duration of each event. Note that $\boldsymbol{A}$ and $\boldsymbol{B}$ should satisfy $Var(\boldsymbol{A}) > Var(\boldsymbol{B})$. The rationale behind this condition is as follows: according to Eq. 3.1 and Eq.3.2, these two formulas asymmetrically normalize the values. That said, in order to make coupling values comparable, the differences between the elements which have larger variance need to be reduced.

Algorithm 1: Event List Dictionary Generation

Input: *actionLogs*
Output: *eventListDict*
`// Create a list if visit missing keys`
eventListDict ← *EmptyDict*;
foreach *actionLog in actionLogs* **do**
    event ← *eventListDict*[*actionLog.loc*].*lastElement*;
    if *event not found* **then**
        event ← {*actionLog.loc*}
    end
    if *action is moving a factor in (e.g., enter)* **then**
        event ← *event* ∪ {*actionLog.factor*};
    else *action is moving a factor out (e.g., exit)*
        event ← *event* − {*actionLog.factor*}
    end
    foreach *elem in event* **do**
        eventListDict[*elem*].*add*(*event*);
    end
end
return *eventListDict*

Algorithm 2: $Freq_{a_i,b_j}$ and Dur_{a_i,b_j} Generation

Input: $eventListDict$, a_i, b_j
Output: $Freq_{a_i,b_j}$ and Dur_{a_i,b_j}

```
eventList ← eventListDict[a_i];
Freq_{a_i,b_j} ← 0;
Dur_{a_i,b_j} ← 0;
coupledFlag ← false;
startTime ← 0;
foreach event in eventList do
    if b_i ∈ event and coupledFlag = false then
        coupledFlag ← true;
        Freq_{a_i,b_j} ← Freq_{a_i,b_j} + 1;
        startTime ← event.time
    end
    if b_inotinevent and coupledFlag = true then
        coupledFlag ← false;
        Dur_{a_i,b_j} ← event.time − startTime;
    end
end
return Freq_{a_i,b_j}, Dur_{a_i,b_j}
```

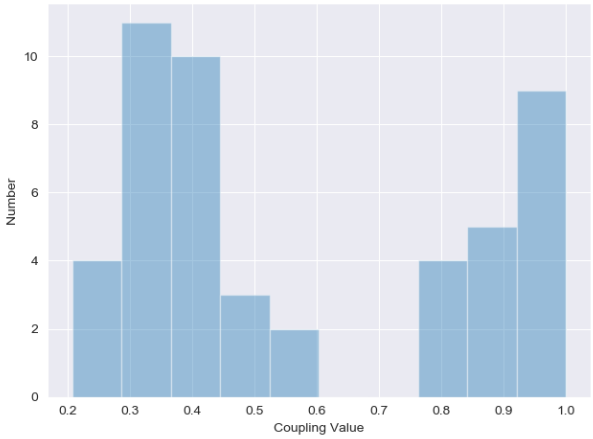


Figure 3.5: An Example of Distribution of Coupling Matrix

$$\begin{bmatrix} G_{a_1,b_1} & G_{a_1,b_2} & \dots & G_{a_1,b_{N_B}} \\ G_{a_2,b_1} & G_{a_2,b_2} & \dots & G_{a_2,b_{N_B}} \\ \dots & \dots & \dots & \dots \\ G_{a_{N_A},b_1} & G_{a_{N_A},b_2} & \dots & G_{a_{N_A},b_{N_B}} \end{bmatrix} \tag{3.3}$$

$$\begin{bmatrix} C_{a_1,b_1} & C_{a_1,b_2} & \dots & C_{a_1,b_{N_B}} \\ C_{a_2,b_1} & C_{a_2,b_2} & \dots & C_{a_2,b_{N_B}} \\ \dots & \dots & \dots & \dots \\ C_{a_{N_A},b_1} & C_{a_{N_A},b_2} & \dots & C_{a_{N_A},b_{N_B}} \end{bmatrix} \tag{3.4}$$

Since the action logs contain four factors–namely people, document, device and location–theoretically $\binom{4}{2} = 6$ different couplings can be obtained. These are: $C_{device,location}$, $C_{device,document}$, $C_{document,location}$, $C_{people,location}$, $C_{people,device}$, and $C_{people,document}$. As mentioned earlier, coupling values are asymmetric. For instance $C_{A,B}$ is applied instead of $C_{B,A}$ in case the variance of the elements over B is greater than those in A.

In case an object, such as a physician, interacts/couples with other objects or locations equally, its couplings will tend towards 1. If two objects less frequently interact, their coupling tends towards 0, which means the risk associated to their coexistence is high.

Fig. 3.5 illustrates how the coupling distribution in a matrix can be skewed.

When multiple coupling values are of the same type, lower coupling value is used to ensure that the impact of risky elements will not be alleviated by safe ones. This is

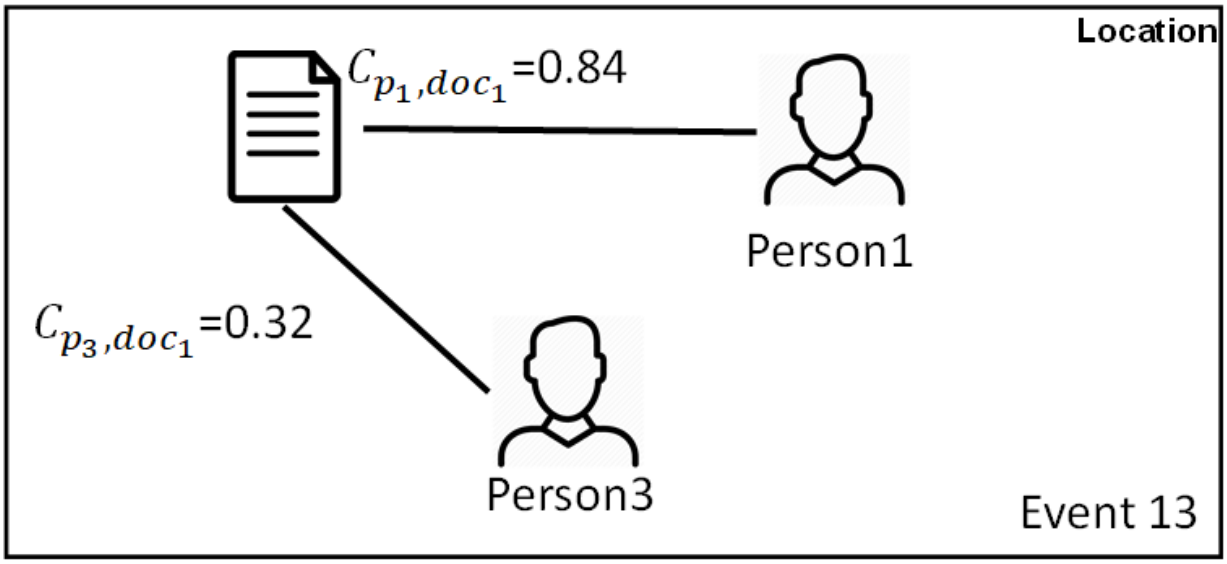


Figure 3.6: Coupling Feature Example

formulated in Eq. 3.5. For instance, given $C_{a_i,b_k} < C_{a_j,b_k}$, when a_i and a_j appear with b_k at the same time, $F_n^{\boldsymbol{A},\boldsymbol{B}} = C_{a_i,b_k}$. Specifically, if $E_i \subseteq E_j$, then $Risk(E_i) \leq Risk(E_j)$. For instance, as shown in Fig. 3.6, p_1 and p_3 co-exist with doc_1 in E_{13}. C_{p_1,doc_1} and C_{p_3,doc_1} are determined according to historical records and $C_{p_1,doc_1} > C_{p_3,doc_1}$. Therefore, for this event, $F_{13}^{\boldsymbol{Ppl},\boldsymbol{Doc}} = C_{p_3,doc_1}$

$$F_n^{\boldsymbol{A},\boldsymbol{B}} = \min_{i,j} C_{a_i,b_j}, \forall a_i, b_j \in E_n \tag{3.5}$$

Our final goal is to label each cluster of couplings with a risk level, which will be described in the next section. To be able to associate clusters with the risk, we need to first assign risk levels to each individual coupling. Therefore, for each coupling matrix, we map the coupling values to onto three risk levels as high, medium or low risk. To achieve this, we set a threshold as Eq. 3.6 and apply mean value binning over the coupling values.

$$Threshold_{High} = mean - stdev \tag{3.6}$$

To formulate the risk level of each event, several strategies are applied, one of which is to code high as three, medium as two and low as one. This is followed by calculating the average risk of each event. Then the average risk will be marked as high, medium, low-risk. The other one is to directly code the risk level of each feature, which can directly

Table 3.2: Map Cluster Risk Values to Risk Levels

CRV	1	1∼1.5	1.5∼2	2	2∼2.5	2.5∼3	3
RL	L	LM	ML	M	MH	HM	H

represent features' pattern. The latter one is used to evaluate the results. Even if the risk level of each coupling value is inferred, still, it can not be used as ground truth because we introduce human subjectivity and intuition into this inference.

Conventionally, researchers may adopt a risk matrix [77] to evaluate the risk level, e.g. $Risk \propto Probability * Impact$. However, this study infers risk by defining association between various objects and situations in the action logs. Thus, the use of conventional risk formulation is left to future work along with further investigations on the quantification of the impact.

3.2.4 Clustering of Couplings and Labeling of the Clusters

RASA framework clusters the couplings into groups and uses the cluster members' risk levels to determine the label of each cluster. Note that non-clustered couplings are less frequently observed, and so possess higher risk.

Most clustering algorithms put similar distance samples into one cluster. In this work, risk levels of couplings are used as the distance metric in cluster formation.

The cluster risk value (CRV) is determined based on the following formula:

$$CRV = \frac{N_H * C_H + N_M * C_M + N_L * C_L}{N_H + N_M + N_L} \tag{3.7}$$

where N_H, N_M, N_L denote the number of high, medium and low-risk features, respectively; and C_H, C_M, C_L are the codes that are assigned to high, medium and low-risk levels. C_H, C_M, C_L corresponds to 3, 2, 1, respectively, 3 denoting a higher risk. This is followed by mapping these values to Low (L), Low-Medium (LM), Medium-Low (ML), Medium (M), Medium-High (MH), High-Medium (HM), and High (H) risk levels according to Table 3.2. In case a cluster is labeled high-risk, all actions in this cluster would be denied. Otherwise, the clusters with medium or low-risk labels contain the actions that may be permitted. The decision for medium risk clusters depends on the coupling feature-based risk levels for each individual point.

3.3 Numerical Results

In this section we evaluate the initial promise of the RASA approach by comparing the decisions against an heuristic policy, and also validated with two supervised learning methods, namely decision tree and Support Vector Machine (SVM).

The action logs used in this work contain one device; hence the available features are Location-People, Location-Device, Location-Document, and Document-People. Because the action logs cover a single device, only $C_{document,location}$, $C_{people,location}$, $C_{people,people}$, and $C_{people,document}$ are used in the evaluations. We present the results under using duration-based features, frequency-based features and the combined (i.e. frequency and duration-based) features. In addition, three different clustering algorithms are applied, which are Gaussian mixture models (GMM) [190], Agglomerative Hierarchical [247], and DBSCAN [70]. The target is to cluster these features (couplings) and label the clusters with risk levels. The aim of the clustering algorithm is to form the least possible number of clusters where within each cluster co-existence of high and low risk patterns is avoided.

3.3.1 Numerical Results of Clustering and Labeling

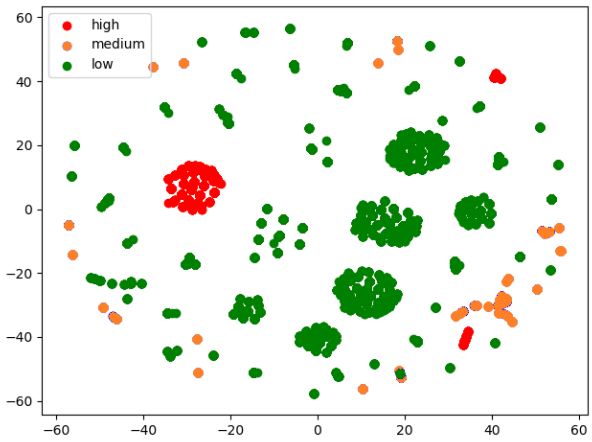
This subsection presents visualization of features followed by feature-by-feature risk of each cluster in the action logs. Then, results under different clustering algorithms are discussed, and finally cluster risk levels are presented

Visualization of features

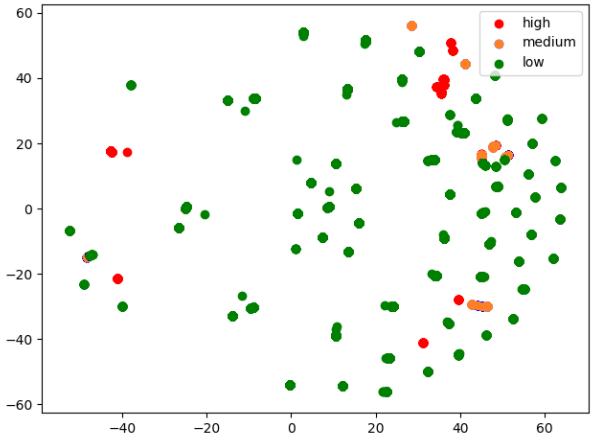
Fig. 3.7 illustrates the features visualized via using t-SNE [233][186]; features are presented by color-codes to show risk levels of samples (red=high-risk, green=low-risk and orange=medium-risk). In Fig. 3.7 (a–c), most of the data points are of low risk level (green) whereas the medium and high average-risk samples are less frequently observed. t-SNE maps every multi-dimensional input in a data set onto two or three dimensions. When t-sNE-based modeling is complete, similar data are mapped close to each other whereas dissimilar data are mapped onto distant points [157].

Feature-by-Feature risk of clusters

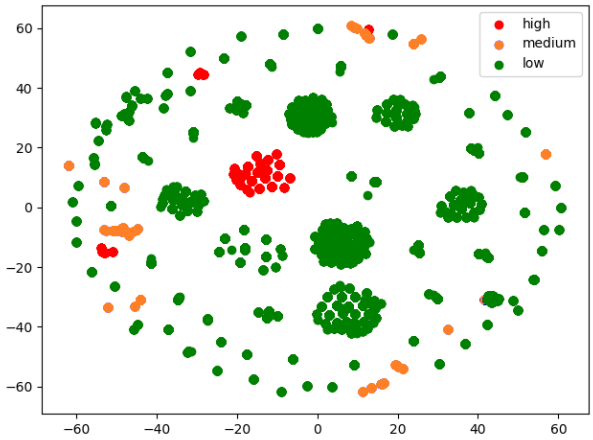
We calculate and present the percentage of feature-by-feature risk level (high/medium/low risk) found at each cluster when we use DBSCAN, Hierarchical and GMM clustering.



(a) Frequency-based Features

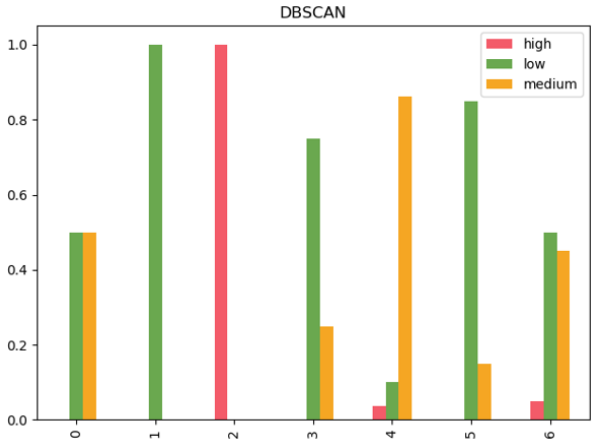


(b) Duration-based Features

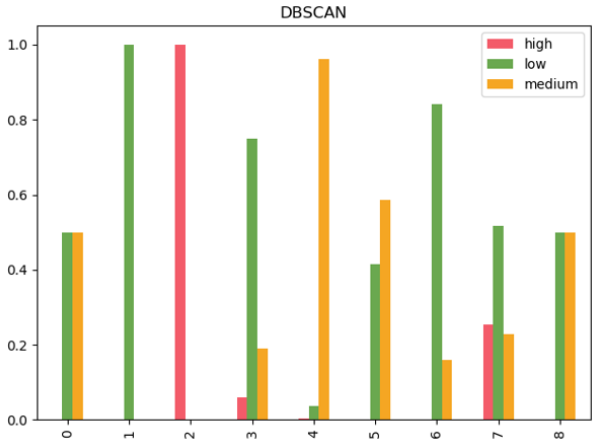


(c) Combined Features

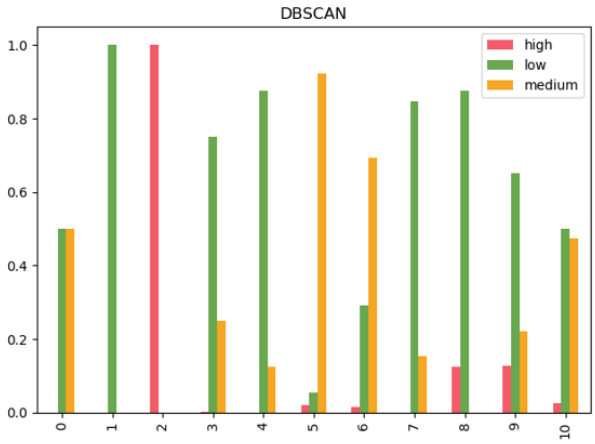
Figure 3.7: Features Visualization using t-SNE



(a) Using Frequency-based Features

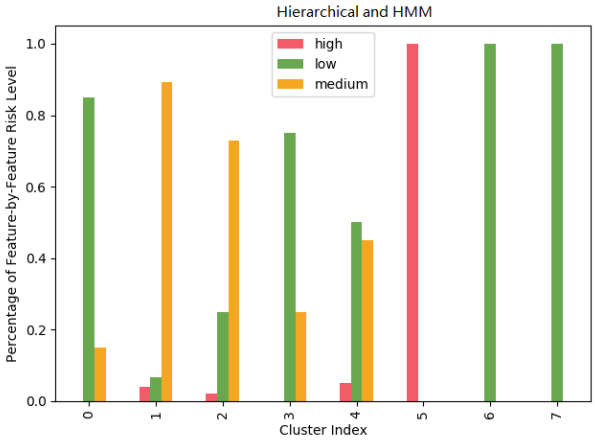


(b) Using Duration-based Features

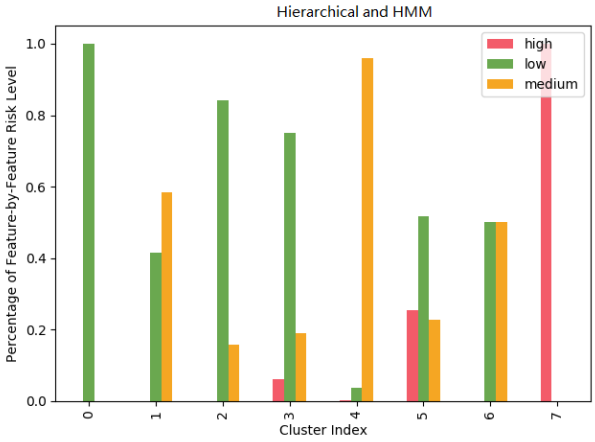


(c) Using Combined Features

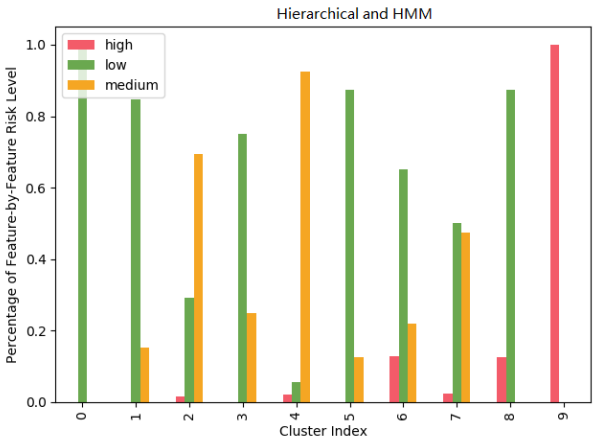
Figure 3.8: DBSCAN Result of Feature-by-Feature Risk Level using Different Features



(a) Using Frequency-based Features



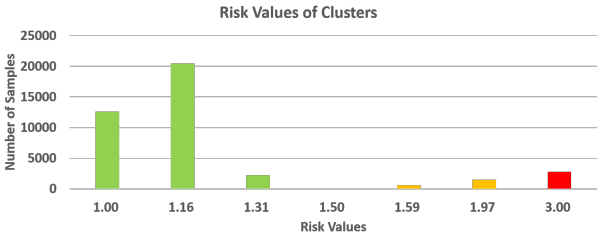
(b) Using Duration-based Features



(c) Using Combined Features

Figure 3.9: Hierarchical and GMM clustering results of feature-by-feature risk level using different features

(a) Using Frequency-based Features



(b) Using Duration-based Features

(c) Using Combined Features

Figure 3.10: DBSCAN Cluster Risk Values using Different Features

Note that hierarchical and GMM clustering schemes generate the same results, as shown in Fig. 3.9. It is not viable to apply the frequency or duration as the only metric to evaluate objects' relationship and their risk levels. In this study, we also consider the circumstances under which people may stay in a room for a long time while they enter and exit other rooms with high frequency. Therefore,the frequency-based features are also expected to affect the risk assessment. Since the duration-based coupling features and frequency-based features show distinctive perspective of the action logs, this study also employs the combined features defined as concatenation of two features. Clustering results, and risk value results are generated by using three different features. Fig. 3.8 and Fig. 3.9 present the percentage of feature-by-feature risk levels of clusters by using frequency-based, duration-based, and combined features. When Fig. 3.8 and Fig. 3.9 are compared when given the same number of clusters, DBSCAN, Hierarchical and HMM provide similar results, although DBSCAN results suggest it may increased noise.

Both Fig. 3.8 and Fig. 3.9 illustrate the diversity of three features; however the results of these three features have similar pattern. Low-risk events are dominant, other types of risk are minor, which is what one would hope is a realistic distribution of event risks in normal clinical practice. This pattern also results in the similarity of the three figures in Fig. 3.10. It is worth to note that the occurrence of this pattern is only due to the simulated action logs used in this study. When different action logs are employed, the result of three features may lead to a more significant variation.

Clustering algorithms: (HMM and DBSCAN)

The strength of DBSCAN is that we do not need to specify the number of clusters upfront while in hierarchical and GMM number of desired clusters are given. If there were more factors in the action logs, the number of features would increase, hence labeling of action clusters by using DBSCAN is generalizable. Indeed, outliers are expected, however, the outliers are not treated as high-risk samples directly, because the events maybe the sub-events of another cluster. For instance, in Tables 3.3,3.4 and 3.5, the sample in the cluster -1, does not have documents in this event; hence there is no possibility of leaking sensitive data in such an event. Nevertheless, in practice we expect that, as a matter of policy, outliers would be escalated to the security operations and policy experts.

Cluster Risk Levels

As clusters and their feature-by-feature risk levels are obtained, risk values of clusters are calculated according to Eq. 3.7 and binned into 7 groups: Low, Low-medium, medium-

low, medium, medium-high, high-medium, and high. Since the actions in medium and low-risk level clusters will be permitted, we intend to prevent high average-risk (defined in Section 3.2.4) samples occur in medium and low-risk level clusters. Although the average-risk level is not a kind of ground truth, if it shows high-risk, then most of the features of the sample raise alert or medium-risk. As explained in the feature extraction section, the coupling value of medium-risk is similar to that of high-risk. There is no obvious boundary between these two risk levels, and clustering algorithms provide some potential risk patterns in the action logs.

The risk values and levels of each cluster are shown on in Tables 3.3, 3.4, and 3.5. The cluster risk values are sorted and plotted as illustrated in Fig. 3.10. This chapter further compares the outcomes of three features in Fig. 3.11. Most of them are safe clusters whereas only one cluster raises a high-risk alert. Based on this, the risk value of the entire dataset (i.e. action log) can be calculated as well, which is 1.28 in this case.

Table 3.3: DBSCAN Result using Duration-based Features

Index	Risk Value	Risk Level	Number of samples
-1	1.5	LM	1
0	1	L	12612
1	3	H	2804
2	1.31	LM	2234
3	1.97	ML	1492
4	1.59	ML	608
5	1.16	LM	20489
6	1.74	ML	480
7	1.5	LM	246

3.3.2 Validation via Supervised Learning

In the final step, as shown in the Fig. 3.13, this work utilizes supervised learning methods, decision tree and SVM, to validate the results of unsupervised learning and to verify whether the coupling idea can be generalized.

As shown in Fig. 3.14, when the model is trained with dataset 1, and tested with dataset 2, 100% training accuracy and 99.86% test accuracy are achieved. As Fig. 3.3 and Fig. 3.12 present, despite the distinct distributions of dataset 1 and 2, the proposed

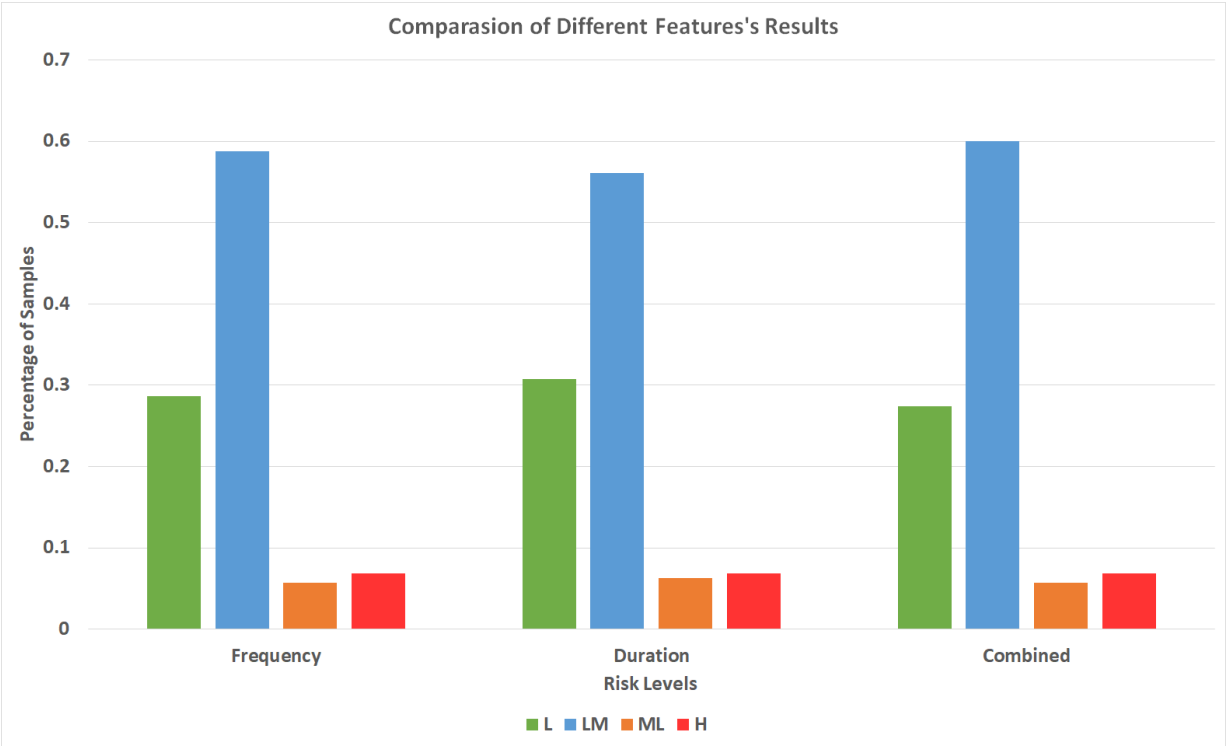


Figure 3.11: Comparison of Different Results

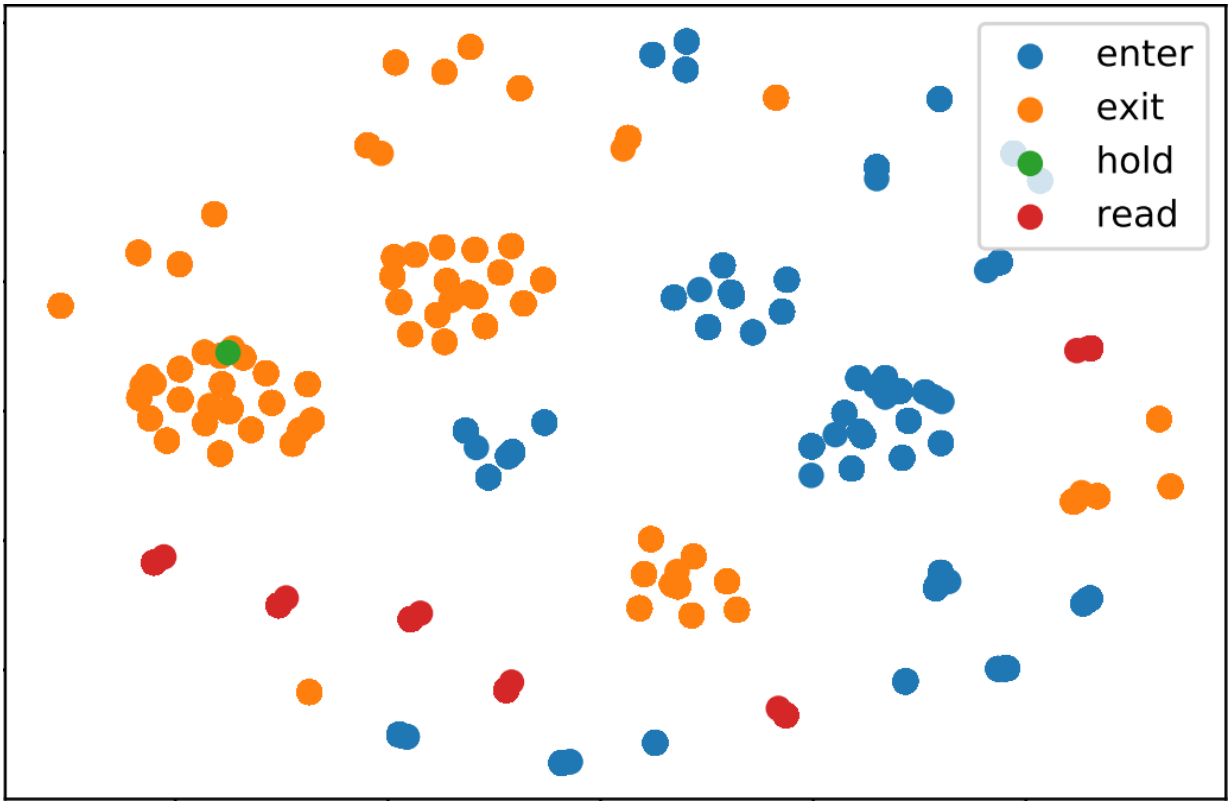


Figure 3.12: Visualization of Raw Actions in Dataset 2

Table 3.4: DBSCAN Result using Frequency-based Features

Index	Risk Value	Risk Level	Number of samples
-1	1.5	LM	1
0	1	L	11754
1	3	H	2804
2	1.25	LM	3082
3	1.94	ML	2100
4	1.15	LM	20979
5	1.55	ML	246

Table 3.5: DBSCAN Result using Combined-based Features

Index	Risk Value	Risk Level	Number of samples
-1	1.5	LM	1
0	1	L	11222
1	3	H	2804
2	1.25	LM	1702
3	1.13	LM	1390
4	1.97	ML	1492
5	1.72	ML	608
6	1.15	LM	20489
7	1.25	LM	532
8	1.48	LM	490
9	1.52	ML	246

methodology can still function well. Support Vector Machine (SVM) is also employed and it achieves 99.95% test accuracy.

3.3.3 Access Control Policy-based Decisions

The performance of RASA framework is validated against a policy-based access control scheme. Policy-based decisions can be considered as a Attribute-Based Access Control (ABAC) [104] approach, utilizing the four frequency-based coupling features introduced above in addition to the traffic, which indicates the number of people in the current event, and co-existence, which is triple-coupling, people-document-location coupling. Eventhough the traffic does not entail any coupling, it is possible to use Eq. 3.6, and the average value

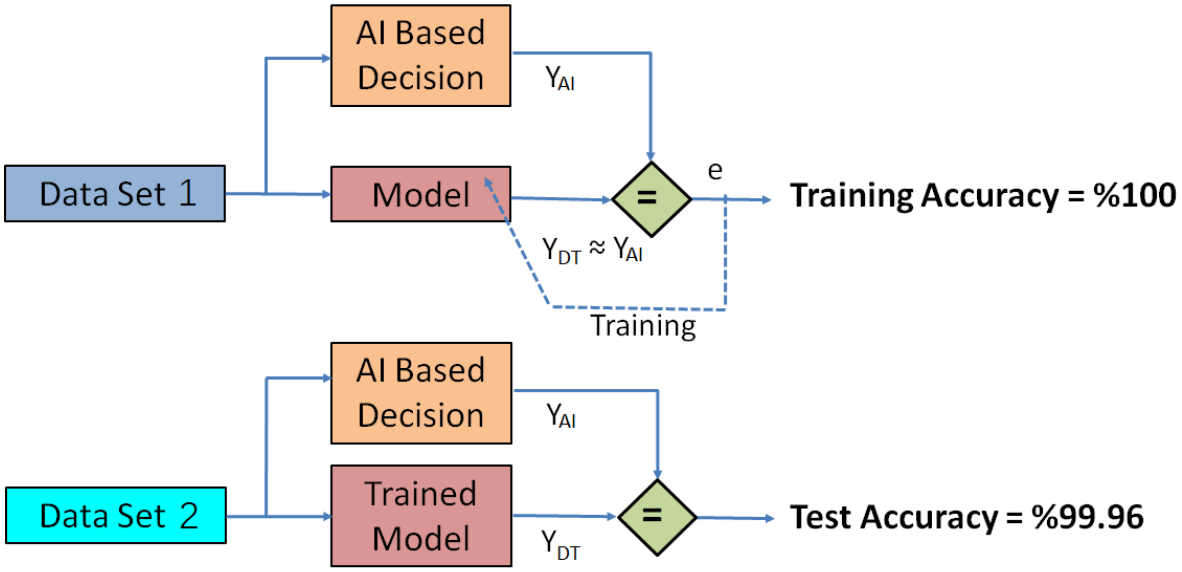


Figure 3.13: Supervised Learning Validation

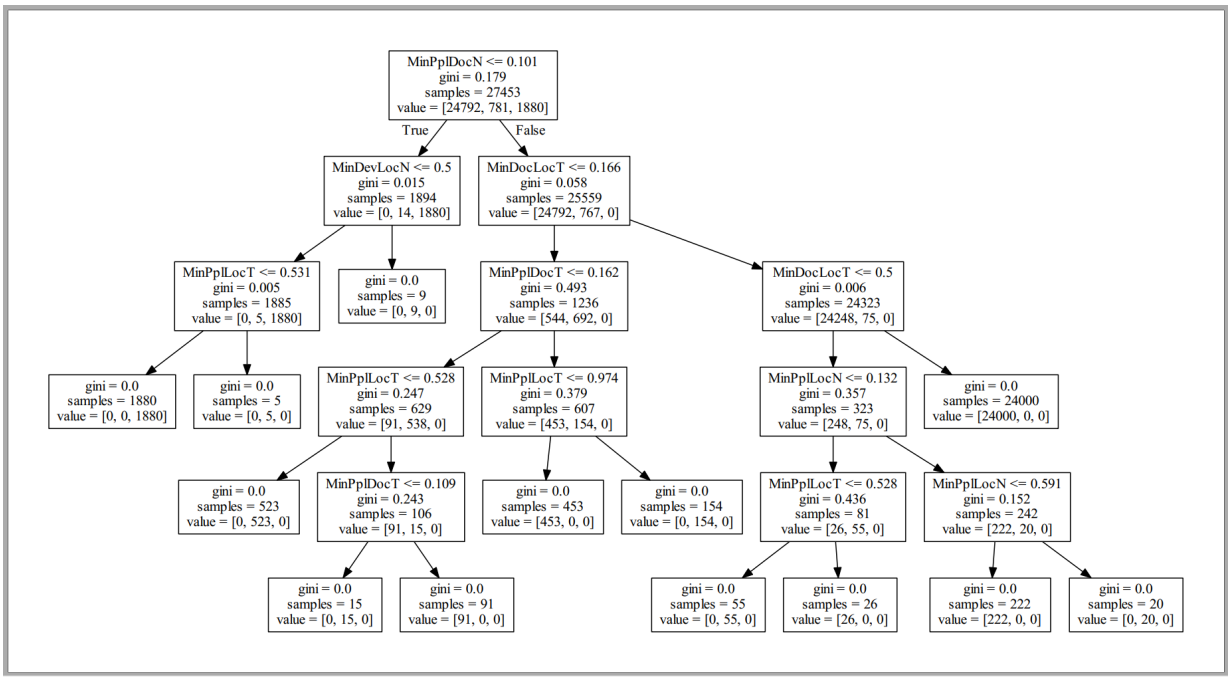


Figure 3.14: Trained Decision Tree

Table 3.6: Comparison between Policy and DBSCAN result using Frequency-based Features

Policy	Permit	Deny	
DBSCAN	Medium-Low	Medium-Low	High
Numer	586	23	2800
Consistency	100%	99.19%	
Overall Consistency	99.32%		

to determine the risk level of each coupling value.

Other features would further transform into risk as well. The policy maps the features' risk onto three risk categories: Device Risk, Environment Risk, and Action Risk. Furthermore it applies a weighted sum of these risks together to determine the overall risk.

$$R_{Dev} = 0.5 * R^{Freq}_{dev,loc}$$

$$R_{Env} = 0.5 * R_{traffic} + 0.5 * R^{Freq}_{co-existence}$$

$$R_{Act} = 0.5 * R^{Freq}_{doc,loc} + 0.5 * R_{doc,time}$$

$$R_{Overall} = 0.3 * R_{Dev} + 0.4 * R_{Env} + 0.3 * R_{Act}$$

Due to the lack of labels or other methods to prove the efficiency of these methods, to examine the performance of the proposed scheme, the unsupervised method and policy-based method are also compared with each other[137]. By tuning the parameters of policy, we also investigated the outcomes of RASA and policy and analyzed the action log. As shown in the Table. 3.6, Table. 3.7, and Table. 3.8, the frequency-based, duration-based, and combined coupling features show 99.32%, 99.27%, and 99.32% overall consistency, respectively. Since the policy uses the triple factor coupling, it is more likely to detect the rare cases but meanwhile, it is the only triple factor coupling since only one device exists in the action log. When the number of coupling factors increases, the types of couplings will reduce, and such a situation would occur where the frequency or duration is not enough to make statistical decision because the frequency or duration disperse on more coupling combinations.

3.4 Conclusion

In this chapter, we have proposed the Risk-Aware Smart Access Control (RASA) framework, which defines a simple overall structure to learn use contexts from observed re-

Table 3.7: Comparison between Policy and DBSCAN result using Duration-based Features

Policy	Permit	Deny	
DBSCAN	Medium-Low	Medium-Low	High
Numer	586	25	2798
Consistency	100%	99.19%	
Overall Consistency	99.27%		

Table 3.8: Comparison between Policy and DBSCAN result using Combined Features

Policy	Permit	Deny	
DBSCAN	Medium-Low	Medium-Low	High
Numer	586	23	2800
Consistency	100%	99.19%	
Overall Consistency	99.32%		

source access within cyberphysical contexts, and use simple parameters to define a context-dependent and risk-based authorization scheme based on those contexts. The basic framework can be utilized to extend other access control regimes that lack sufficient support for security policy analysts to craft and maintain context-dependent authorization policies. RASA utilizes combined cyber and physical activity logs, converts action sequence to events which contain contextual information, and identifies common relations through a coupling approach based on duration and frequency of interactions. Every coupling is assigned a risk value which results from the intuition that frequently occurring interactions can be considered as low-risk and rarely occurring actions as high-risk. RASA further feeds these couplings to clustering algorithms, after which the level of risk for a cluster is determined based on the risk levels of cluster members (i.e. couplings). Thereafter RASA denies actions in high-risk clusters and permits safe actions. One of the insights from our study in a simulated clinical environment is the importance of the selection of couplings on risk inference.

Three different features are explored in this study, namely, frequency-based, duration-based and the combination of frequency and duration. According to clustering and labelling results, these features can exhibit distinct perspectives of risk. We show that combined features can more comprehensively represent the risk of events and actions. It may be helpful in future work to define other risk factor compositions and understand better their interactions.

RASA is purposefully designed to require little in the way of security policy analyst

input–for example, no extensive labeling of actions into risk is required to develop a training set. Our evaluation established a basic validation of the promise of building on this approach.

In the tuning process, rule-based policy results are compared with the outcomes of RASA, and rule-based policy and RASA are updated in each iteration. After multiple tuning iterations, the decisions of two approaches achieved $> 99\%$ consistency. Furthermore, to validate the results of RASA, when two supervised methods (Decision tree and SVM) are trained and tested by using different action logs labeled by RASA, 99.95% test accuracy is achieved.

Chapter 4

ML-Driven Intrusion Detection under IoT Environment

4.1 ML-Driven Intrusion Detection for Contiki-NG-Based IoT Networks

4.1.1 Methodology

IoT Setting Under Study

The architecture of an IoT network used in this chapter consists of a group of sensors monitoring a physical environment and aggregating the sensed data for the sink node or gateway. The sink node further delivers the collected data to the cloud servers or to users via a LAN. Since sensors are distributed randomly in the field [200], they are prone to illegitimate access to view and modify the legitimate nodes in the network, or to introduce malicious nodes to the network. Hence, identifying the malicious nodes and classifying the attack types are important. Several researchers have studied the possible attacks targeting the RPL protocol, such as blackhole, gray hole, and warm hole, and proposed ML-based detection of these attacks [13]; however, the implementation of 6LoWPAN is still vulnerable to the traditional attacks such as those mentioned in KDD99 or NSL-KDD. With this in mind, we introduce possible network layer attacks in the NSL-KDD dataset to the Contiki-NG-based IoT network, and study the effectiveness of ML algorithms in terms of their classification performance under the NSL-KDD dataset. Below are the attacks considered here:

- **SynFlood**: Malicious nodes keep sending TCP packets with SYN flag creating multiple connections to drain the resources of other nodes [109].
- **Land**: Similar to SynFlood, but source and destination addresses are set as the target node leading the victims to establish TCP connections to themselves [212].
- **UDP Flood**: Malicious nodes randomly send UDP packets to victims, draining their bandwidth and having them send "ICMP port unreachable" messages [117].
- **Ping of Death (PoD)**: PoD is to deliver oversized payload ping requests to victim nodes to cause buffer overflow [241].
- **Smurf**: Malicious nodes broadcast ping requests, setting the source as the victim and leading other nodes to be involved in a flood of ping reply messages to the victims [244].
- **IP sweeping**: Malicious nodes discover active IP addresses by ping requests and enumerating addresses [121].
- **Port sweeping**: Malicious nodes scan the open TCP and/or UDP ports to attack victims [121].

ML-Integrated Intrusion Detection

Several researchers have studied various ML algorithms on network intrusion detection systems [201]. This study differs from the related work in many ways (as stated earlier), including the type of attacks and the variety of ML algorithms integrated with the IDS system. Eleven ML algorithms are employed, including both supervised and unsupervised ML algorithms. The detection mechanism used in this chapter takes the dataset as the input and splits the dataset according to a k-fold (k=10) cross-validation approach to keep the results stable and unbiased. Both training and test datasets are fed into various ML algorithms, and finally multiple evaluation metrics are measured for performance evaluation. The classification methods we test are briefly explained below:

- **Decision Tree (DT)** – A decision tree establishes a tree-like model of decisions in the form of if-else rules. [15]
- **XGBoost** – This is an ensemble learning method with fast training speed and high accuracy. XGBoost is based on DTs, and it leverages gradient boosting and tree-pruning [44].

- **Bagging Tree** – This refers to training multiple weak learners in parallel and aggregating the results in a certain way to avoid overfitting in order to obtain improved results [155].

- **Random Forest (RF)** – An RF combines multiple DTs and selects a subset of training samples and part of the features to prevent overfitting; it finally vote for the best result [155].

- **Bayes Net** – This is a probabilistic graphical model-based technique that builds on Bayesian inference [126].

- **Support Vector Machine (SVM)** – SVM can effectively handle high-dimensional datasets, including the situations where samples are outnumbered by the dimensions [15] .

- **Naïve Bayes** – This is a probabilistic approach that builds on Bayes theorem with the assumption of fully independent features [126].

- **AdaBoost** - This is an ensemble method that can perform oversampling to address the class imbalance problem [155].

In addition to the supervised techniques, clustering (unsupervised) algorithms are also investigated as summarized below:

- **Expectation Maximization (EM)** – The EM algorithm alternates between performing an expectation (E) step and a maximization (M) step to estimate the hidden variables [252].

- **DBSCAN** – This clusters the dataset according to density and is able to find outliers [61].

- **K-Means** – This aims to obtain k clusters out of n data points where each data point is placed in the cluster with the closest mean [61].

4.1.2 Experiments and Results

In the simulations, with the settings presented in Table 4.1, we first implement regular traffic by randomly distributing legitimate nodes. Three different kinds of legitimate nodes exist in the simulation: nodes with the UDP protocol; nodes with the TCP protocol; and

Table 4.1: IoT Network Node Types

Node Type	Num	Description
Sink Node	1	Aggregates network traffic and can receive both UDP and TCP taffics
Sensor Node (TCP)	10	Simulated as sensor node used for collecting environmental and transfer data via TCP protocol
Sensor Node (UDP)	10	Transfer data via UDP protocol
Malicious Node	1	Simulates the attacker that launches different attacks

a sink node which serves as a TCP and UDP server. To simulate realistic WSN traffic, each node randomly sends data with data size that varies according to the TCP and UDP protocols; these nodes are deployed in random locations. Network packets are collected at the sink node by filtering the IP and MAC address. Seven attacks in the NSL-KDD dataset (as also mentioned in Section 3) are implemented in Contiki-NG. Upon the collection of packets, PCAP files are fed into a feature extractor in order to form the intrusion dataset [1].

Attack Types

We implement seven types of attack techniques from the NSL-KDD dataset, i.e., SynFlood (Neptune), Land, UDP flood, Ping of Death (PoD), and Smurf. Other attacks in the NSL-KDD dataset target the mail protocol, Apache server and Linux systems, such as Mailbomb, Apache2, and Process table. These protocols or tools are not applicable to the Contiki-NG operating system since Contiki-NG is used in resource constrained devices. We further label the dataset according to the MAC address and timestamps. Injection of attacks into the Contiki-NG network is explained below:

- **Syn Flood** – Since Contiki-NG does not provide interfaces to send TCP packets with SYN flag without creating a connection, we directly create IP packets from scratch and send them to the MAC layer protocol. It is worth noting that when the checksum is calculated, the prefix of the IPv6 address is 0xFE80 instead of 0x0000; therefore, using Wireshark for packet analysis will result in several checksum errors. After

[1] Attack traces are available: http://nextconlab.academy/contikingdata.html

Syn Flood is introduced into the Contiki-NG network, the sink node keeps sending SynAck to respond to the messages, draining its resource.

- **Land** – As a special Syn Flood attack with identical source and destination IP address in the Contiki-NG system, a Land attack aims to have a node send packets to itself and directly forward these packets to the TCP/IP buffer instead of delivering them to the MAC layer and sending them through wireless channels. Thereby, theoretically, 6LOWPAN could prevent a Land attack by detecting such a situation where the MAC layer delivers a packet with the same source and destination IP address, leaving no chance for such attacks to succeed. This can be observed by monitoring the log of the sink node.

- **UDP Flood** – Unlike a TCP connection, UDP packets do not receive N/ACK from destination nodes. If the destination receives a UDP packet with a closed port, it responds with an unreachable port message (ICMPv6) to the source node. Therefore, in the experiment, UDP packets are sent with random source and destination ports to the sink node, and the sink node responds with ICMPv6 packets immediately. In Contiki-NG, if a source node keeps sending packets with random ports using their UDP interfaces, Contiki-NG refuses to send UDP packets after a few iterations. Hence, in the design of the attack, flooding is performed by generating IP packets and delivering them to the MAC layer. Meanwhile, by doing so, the speed of the flood can be controlled as well.

- **Ping of Death (PoD)** – Contiki-NG checks the size of every message received from a lower layer to prevent buffer overflow. Thus, PoD cannot paralyze or destabilize the sink node but since the sink node will still respond to the ping requests, PoD can be considered as a regular ping flood attack.

- **Smurf** – Smurf is another attack based on ICMPv6, which sends a Ping Request with a broadcast address as the destination and target address as the source IP address. In the simulator, all the nodes send a ping reply to the sink node, blocking other communication.

- **IP Sweeping** – Probes are utilized by adversaries to discover victim IP addresses and ports. In this study, we introduce IP sweeping, TCP port scanning, and UDP port scanning to perform probe attacks. IP sweeping is implemented by iterating on IP addresses while sending ping requests and recording replies.

- **Port Scan** – In TCP port scan, upon completion of the three-way TCP handshake, if the attacker node receives connected events from Contiki-NG, it records the port number.

In UDP port scan, if a node sends UDP packets to a closed port, an unreachable port message is replied back by the destination node. Through ICMPv6 events, open ports can be inferred.

Feature Extraction

The network intrusion dataset creator [184] is utilized to create our dataset. To reduce communication energy, Contiki-NG utilizes the 6LoWPAN protocol at an adaptive layer to compress the length of IP packets as the LibPCAP library does not support the 6LoWPAN protocol. On the other hand, although the Network Intrusion Dataset Creator is also designed for the IPv4 system, it can analyze network packets by using TShark as a backbone, which can also analyze 6LoWPAN protocols. To make it suitable for IPv6, ICMPv6, and 6LoWPAN protocols, we modify the feature extraction module and add more features related to the ICMPv6 protocol, IP address, and MAC address. Finally, twenty-eight features are extracted from the network flow, and each entry also has a label denoting the the type of the attack introduced through the corresponding packets. The extracted features can be categorized as follows:

- **Frame Info** – The total physical frame length of TCP, UDP, ARP, and ICMPv6 packets, and the number of packets.
- **IP Layer Info** – The total IP frame length of TCP, UDP, ARP, and ICMPv6 packets, the number of connection pairs, the number of ICMPv6 types, and the number of ARP and ICMPv6 packets.
- **IP Address Info** – Whether the source and the destination have the same IP address, and whether the IP address is consistent with MAC address.
- **Transport Layer Info** – The total frame length of TCP, UDP, and the number of ports used in both TCP and UDP packets.
- **Application Layer Info** – The number of SSL, HTTP, FTP, SSH, SMTP, DHCP, and DNS packets.

To better understand the generated dataset, we present a breakdown of the packets in the Contiki-NG network in Fig 4.1. Since most of the time (24 hours) the nodes in the simulator send regular traffic to the sink node, the normal packets form the majority class. Due to the limitation of the hardware and simulator, each DoS attack only runs for

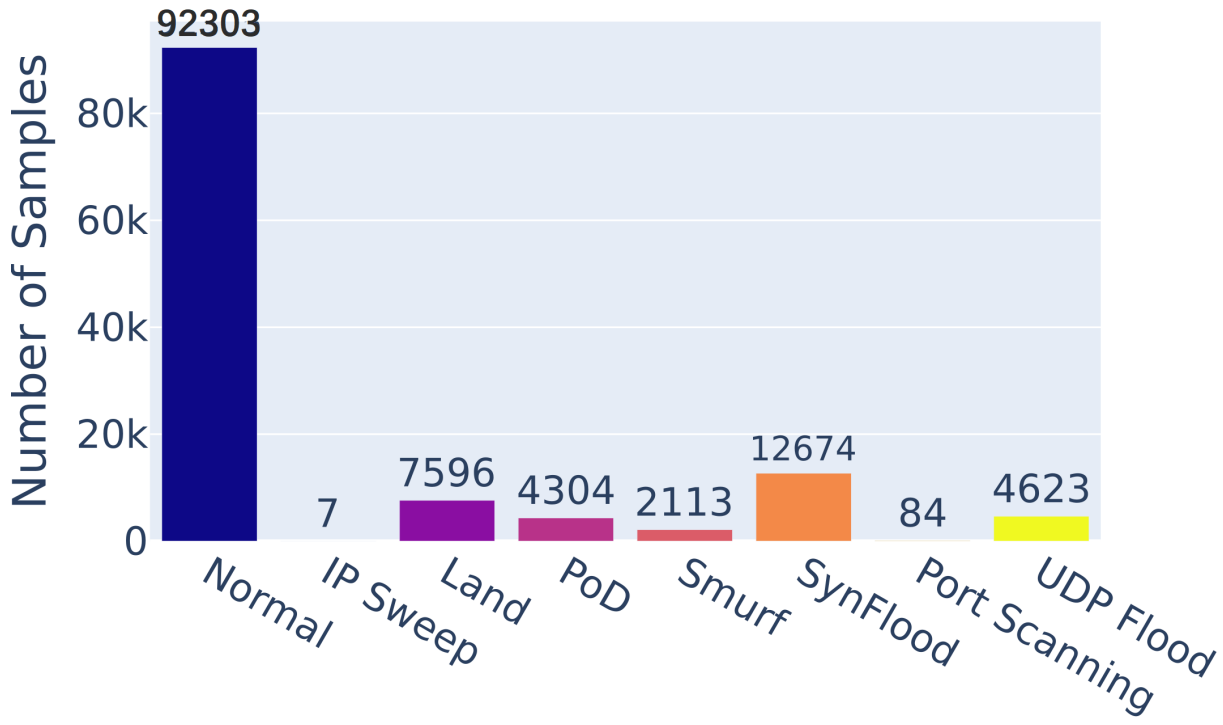


Figure 4.1: Dataset Distribution

Table 4.2: Accuracy, Precision, Recall and F-Scores

	Accuracy	Precision	Recall	F Score
RF	0.966	0.969	0.967	0.968
DT	0.966	0.969	0.967	0.968
Bagging	0.967	0.969	0.967	0.968
SVM	0.957	0.948	0.957	0.951
NB	0.452	0.904	0.452	0.545
BN	0.882	0.944	0.882	0.902
AdaBoost	0.740	0.663	0.740	0.646
XGBoost	**0.970**	**0.970**	**0.968**	**0.968**

Table 4.3: TP, RF, MCC, and AUC of ML Algorithms

	TP Rate	FP Rate	MCC	AUC
RF	0.966	0.046	0.903	0.995
DT	0.966	0.048	0.903	0.993
Bagging	0.967	0.048	0.904	0.995
SVM	0.957	0.072	0.865	0.942
NB	0.452	0.028	0.407	0.888
BN	0.882	**0.006**	0.789	0.985
AdaBoost	0.740	0.065	0.038	0.604
XGBoost	**0.970**	0.046	**0.905**	**0.996**

5 minutes. In the simulation, we also implement IP sweep and port scan, but the available IP address is obtained alongside the open port number, so such attacks will be blocked. Hence the sample size of probe attacks is negligible, especially for IP sweep. Since the generated dataset is significantly imbalanced, techniques and comparison metrics that can handle imbalanced datasets are used to train and evaluate the results.

ML Algorithms to Classify Intrusions

We employ the following ML algorithms to classify and cluster intrusions: AdaBoost, Random Forest, Decision Tree, Bagging, XGBoost, SVM with RBF kernel, Naïve Bayes, Bayes Network, KMeans, and DBSCAN. For supervised methods, accuracy, true positive rate (TP), false positive rate (FP), precision, recall, F measure, Matthews correlation coefficient (MCC), and Receiver Operating Characteristics area under the curve (AUC)

are used. MCC is calculated as $\frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP) \times (TP+FN) \times (TN+FP) \times (TN+FN)}}$, and is a useful ML algorithm metric especially for imbalanced datasets, which results in more meaningful results than F score [48] in such settings as it considers the proportion of the classes inside the confusion matrix.

As for the unsupervised methods, we simply use accuracy to present the result, since the results of clustering algorithms still need to be enhanced. As mentioned earlier, the generated dataset is imbalanced; hence accuracy itself is not enough to represent the performance, while F measure, MCC, and ROC area can better show the results of these ML algorithms. All the results of supervised methods discussed in this chapter use ten fold cross-validation to generate the final performance values.

Performance results of classification algorithms are listed in Table 4.2 and Table 4.3. Initially, Adaboost and RUSBoost were selected as strong ensemble methods to operate with imbalanced data. However, as shown in Table 4.2 and 4.3 the performance of these algorithms is not strong enough; in particular, the RUSBoost underperforms compared to AdaBoost. Therefore, only AdaBoost performance is shown in the performance results. Other ensemble methods are also tested such as Bagging, Random Forest (RF), and Extreme Gradient Boosting (XGBoost), among which XGBoost outperforms the others in terms of all performance metrics. RF and Bagging techniques almost have identical performance, which is slightly higher than that of a single decision tree. Considering the limitation of nodes in IoT or WSNs, (i.e., limited RAM, storage capability, and energy resource), decision trees stand out as the best fit for an IoT environment. Furthermore, DT is highly explainable so manual tuning of the trained model is also possible and suitable for specific use cases in real-life. Last but not least, the performance of decision trees is reasonably close to that of a more advanced method, XGBoost.

Different kernels of SVMs are tested and Radial Basis Function (RBF) outperforms the other functions. However, the performance of SVM with RBF is still outperformed by the tree-based methods. Furthermore, SVM requires an extensive amount of time to train and run. Naïve Bayes (NB) and Bayes Network (BN), due to their probabilistic nature, are not suitable for this problem, and perform poorly as seen in the table.

This chapter also utilizes unsupervised methods to validate their performance and reports them in Table 4.4. Regardless of tuning of the parameters, the performance of unsupervised methods is limited. For K-Means, when the number of clusters is set to six, the overall accuracy is 49.59%. Furthermore, when the parameters of DBSCAN are set to eps=0.4 and minPts=6, its accuracy is 51.1%. Expectation-Maximization (EM) stands out among the unsupervised techniques, offering the best results among unsupervised algorithms with 67.2% accuracy, which outperforms Naïve Bayes and approaches the

Table 4.4: Performance of Unsupervised Algorithms

	Accuracy	Clusters	Identified Classes
K-Means	0.496	6	Normal, Land, PoD, SynFlood, UdpFlood, Smurf
DBSCAN	0.511	4	Normal, Land, SynFlood
EM	0.672	4	Normal, Land, PoD, UdpFlood

performance of AdaBoost. Although the unsupervised methods underperform when compared to supervised ML techniques such as the tree-based methods, they do not require labels (which generally cannot be easily generated in real-life networks). In such a scenario, the EM algorithm can be considered as a potential solution.

4.1.3 Conclusion

In this chapter, we introduce possible DoS and probe attacks in the NSL-KDD dataset to an IoT network, specifically Routing Protocol for Low-Power and Lossy Networks (RPL) and 6LoWPAN networks, using the Contiki-NG operaing System. Moreover, the generated dataset is fed into eleven ML algorithms to explore their ability to classify different attacks. Tree-based methods and ensemble algorithms such as XGBoost, Decision Trees (DTs), Bagging Trees, and Random Forest perform well and achieve more than 96% accuracy, whereas the remaining methods, Bayes Network, Naive Bayes (NB), Adaboost, perform relatively poorly. Considering that the IoT devices consist of resource constrained devices, DTs, as a simple and highly explainable solution with high performance, is concluded to be utilized in the sink node to detect DoS and probe intrusions. Moreover, when unsupervised methods are utilized (EM, DBSCAN, and K-Means), DBCAN and K-Means are outperformed by EM while the EM algorithm outperforms NB by 22% in terms of accuracy. Therefore, EM is concluded to be more suitable for the NSL-KDD dataset and the Contiki-NG-based IoT scenario when it is not possible to acquire the labels from the network traffic.

Table 4.5: APT Attack techniques used in SCVIC-APT-2021

Attack Stage	Reconnaissance	Initial Compromise	Lateral Movement	Pivoting	Data Exfiltration
Attack Techniques	Active Scanning	VSFTPD	Pass the Hash/Ticket	AutoRoute	DNS Tunnelling
	Gathering Victim Host Information		Remote Desktop Protocol	Socks4a	C2 Tunnelling
	Gather Victim Network Information		WMI	Proxy Chain	Encode and Encrypt

Table 4.6: SCVIC-APT-2021 Dataset Class Distribution

	Train	Test	Total
DataExfiltration (DE)	527	74	601
InitialCompromise (IC)	73	77	150
LateralMovement (LM)	729	142	871
NormalTraffic (NT)	254836	55583	310419
Pivoting (P)	2122	360	2482
Reconnaissance (R)	833	251	1084

4.2 A New Realistic Benchmark for Advanced Persistent Threats in Network Traffic

4.2.1 SCVIC-APT-2021 Dataset

As illustrated in Figure 4.2, we create a multi-domain environment with two domains and four sub-networks to enable APT attacks such as lateral movement and pivoting. Kali is used to design attacks to machines and is capable of gaining access to the initial victims located within the first domain. Domain 1 consists of four machines, one of which is a Domain Controller (DC), and three of which are regular end devices. Two domains are connected by a virtual private network (VPN). Domain 2 has a domain controller (DC) and a regular PC, which is supposed to be the attacker's ultimate target for credential information.

Typically, an APT follows six stages: reconnaissance, initial compromise, lateral movement, pivoting, data exfiltration, and post-attack stage. MITRE Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK) is a global knowledge base of adversary tactics and approaches that serves as the foundation for our selection of common attack techniques. Table 4.5 summarizes the attack techniques used in SCVIC-APT-2021.

- **Reconnaissance**: Reconnaissance is used to gather information and credentials that will be used to support further APT stages.

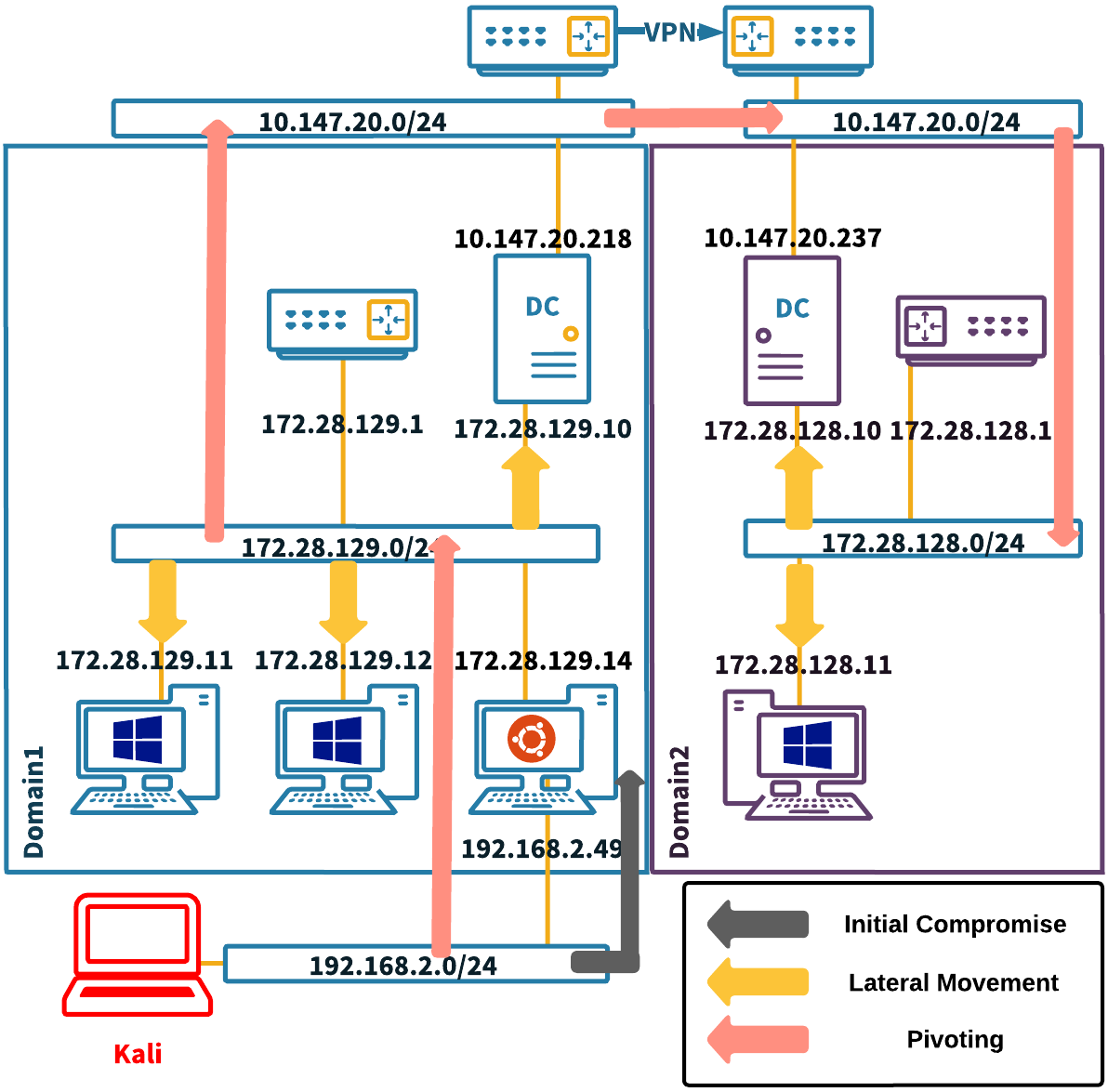


Figure 4.2: Network topology and detailed settings of the testbed

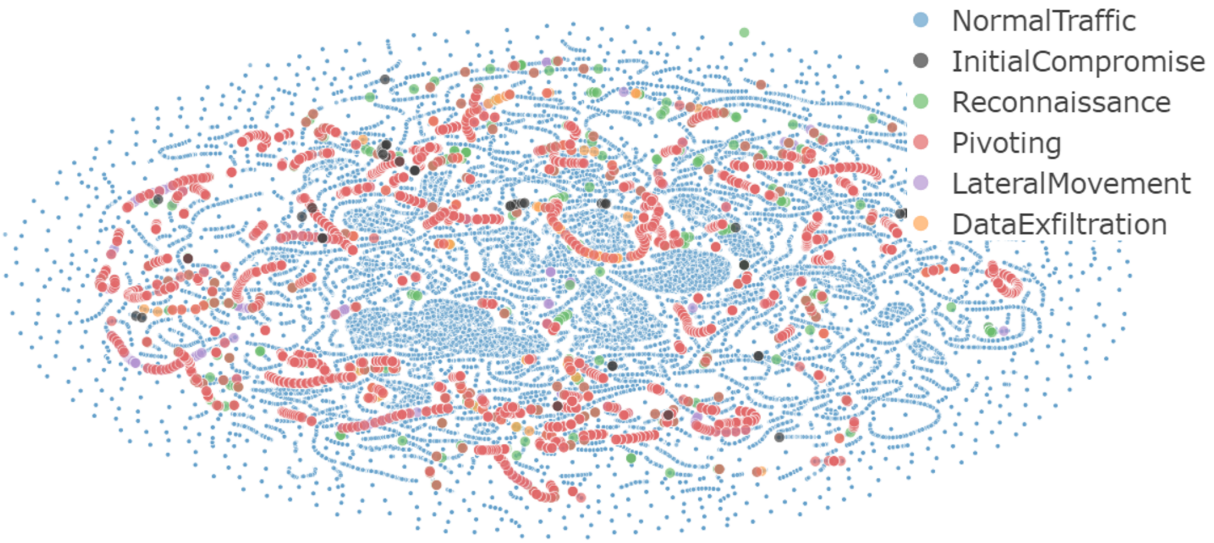


Figure 4.3: 2D distribution of the training set via t-SNE

- **Initial Compromise**: Numerous ways may be utilized as an initial compromise in order to establish a foothold within a network. Since this dataset focuses on attacks within the domain, only Metasploit framework's the VSFTPD attack is employed.
- **Lateral Movement**: Adversaries migrate laterally to access to other workstations to gain access to the primary target or pivot towards other networks.
- **Pivoting**: Companies and organizations may have several domains or subnetworks; as a result, attackers must establish tunnels or channels to traverse multiple networks.
- **Data Exfiltration**: After attackers compromise target networks and get access to valued targets, critical digital assets are stolen and transmitted to adversary-controlled servers.

A complete APT process is shown as follows. Through initial compromise, an attacker infiltrates the first machine and pivots to Domain 1. Additionally, the attacker moves laterally using a variety of approaches and collects all available information. The attacker pivots to domain 2 via the DC in domain 1 using VPN. After lateral movement within Domain 2, the attack compromises and exfiltrates data / files from the ultimate PC.

Unlike DOS/DDOS, which generates massive amounts of traffic, APT generates limited network flows leading to class imbalance. Therefore, to cope with the training set has four

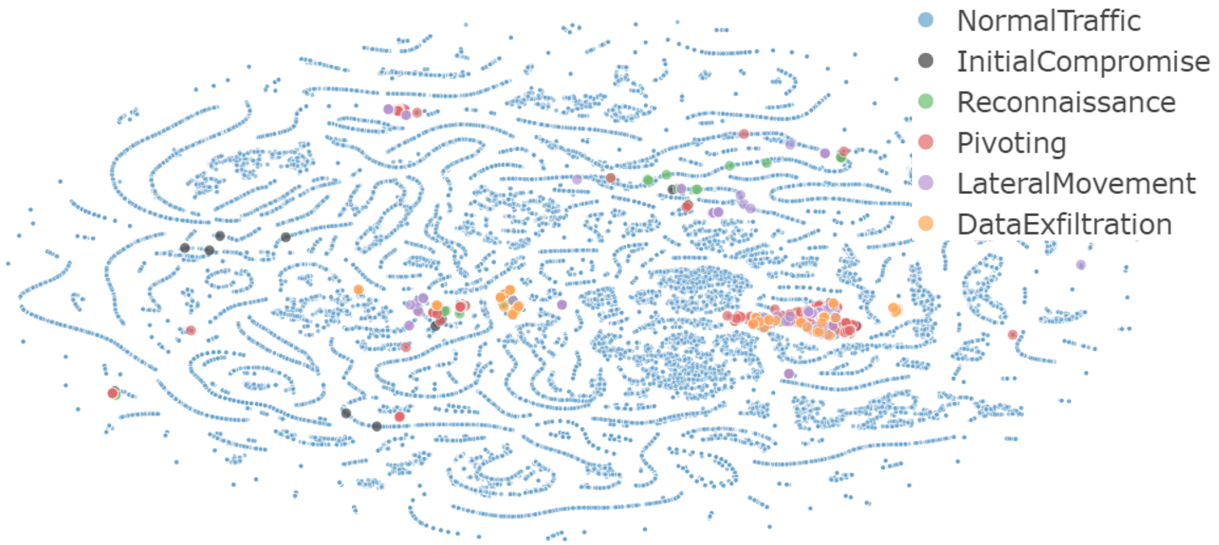


Figure 4.4: 2D distribution of the test set via t-SNE

rounds of the APT process, whereas the test set contains an isolated round from the training set. Each round of infiltration employs a different attack tactic randomly.

For normal traffic, besides the network flow collected from the setup network, we also merge the industrial IoT (IIoT) network flow (from 4SICS [2]) to incorporate IoT settings into our dataset. Network traces are saved in PCAP format, from which, network-based features are extracted using CICFlowMeter. A legitimate PC can generate malicious traffic due to lateral movement and pivoting; thus, labeling traffic requires not only IP and timestamp, but also precise port information. Table 4.6 summarizes the class distributions of training and test sets that are shown to express data structure in detail using 2D TSNE in Figures 4.3 and 4.4, respectively.

4.2.2 Detection Methodology

This section proposes a new ensemble method that builds on base (binary) machine learning models while training base models for a specific attack type and makes predictions using a machine learning aggregator. A minimalist overview of the test procedure for the proposed Attack Centric Method (ACM) is presented in Fig 4.5. A test sample is fed into a base model (binary classifier) for each class/attack, which outputs the posterior probability of

[2] 4SICS traffic can be found: https://www.netresec.com/?page=PCAP4SICS

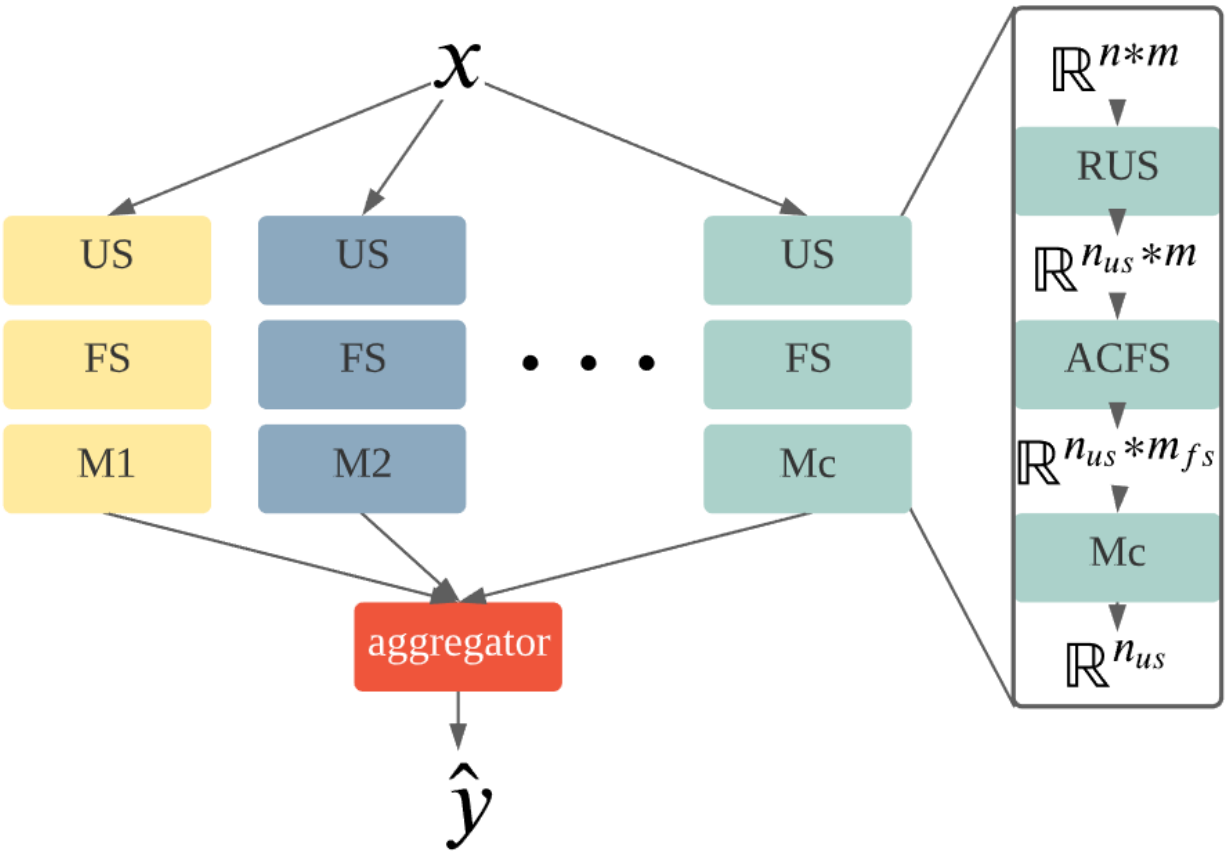


Figure 4.5: Attack Centric Method (ACM); **US**: Under-Sampling; **RUS**: Random Under-Sampling; **ACFS**: Attack Centric Feature Selection; M_i: Base Machine Learning Model. The right hand side shows the input/output dimensions of each block.

the test sample belonging to the attack type. The outputs of c classifiers are concatenated and fed into the aggregator that predicts the type of attack (c is the number of classes). Due to the unique characteristics of each attack, this chapter proposes Attack Centric Feature Selection (ACFS) which selects various feature sets prior to feeding machine learning algorithms according to attack types. For each attack, embedded feature selection method (e.g., random forest feature selection) is used to selection top k important features, where k is determined by validation set. ACFS improves the interpretability of ACM, explaining which features trigger the alert of an attack. During the training process, the labels in the entire training set are binarized into one-hot vectors, and each machine learning model is trained as a binary classifier. To avoid overfitting, the aggregator is trained using the base models' predictions on the validation dataset. Due to the rarity of minority instances, k-fold cross-validation is employed to make the most of data and improve the prediction with minority instances.

In general, a complex aggregator suffers from severe overfitting; hence, logistic/softmax regression is employed as the default aggregator in the majority of ensemble approaches, such as stacking. We suggest a novel aggregator, named Diagonal Softmax Regression (DSR), to further simplify the softmax regression. The hypothesis function for the original softmax regression is as follows:

$$h_{SR}(x) = \frac{1}{\sum_{j=1}^{c} exp(\theta_j^T x)} \begin{bmatrix} exp(\theta_1^T x) \\ exp(\theta_2^T x) \\ \vdots \\ exp(\theta_c^T x) \end{bmatrix}$$

, where $\theta \in \mathbb{R}^{m*c}$ is the parameter matrix, and m is the number of features. If SR is used as aggregator, then $m = c$. The proposed DSR aggregator further constrains the W as diagonal matrix; therefore the hypothesis function is simplified as follows:

$$h_{DSR}(x) = \frac{1}{\sum_{j=1}^{c} exp(w_j x_j + b_j)} \begin{bmatrix} exp(w_1 x_1 + b_1) \\ exp(w_2 x_2 + b_c) \\ \vdots \\ exp(w_c x_c + b_c) \end{bmatrix}$$

, where $w_j, b_j \in \mathbb{R}$ are the parameters of DSR. Since the input of DSR is the prediction of the base models, w_j and b_j denotes the reliability of the base models. By applying cross entropy, the objective function is as follows:

$$J(W, b) = -\sum_{i}^{n} \sum_{j}^{c} I(y^i = j) \log \frac{exp(w_j x_j^i + b_j)}{\sum_{k}^{c} exp(w_k x_k^i + b_k)}$$

Table 4.7: SCVIC-APT-2021 Dataset Baseline Results; AB: AdaBoost; GB: Gradient-Boost; DT: Dicision Tree; XGB: XGBoost; W Avg: Weighted Average F1; M Avg: Macro Average F1

	LR	SVM	AB	GB	DT	XGB	**RF**
DE	0.003	0.000	0.249	0.124	0.288	0.268	**0.301**
IC	0.000	0.000	0.001	0.612	0.800	0.861	**0.829**
LM	0.000	0.000	0.397	0.677	0.785	0.881	**0.782**
NT	0.000	0.992	0.685	0.999	0.999	1.000	**1.000**
P	0.000	0.000	0.644	0.826	0.733	0.757	**0.851**
R	0.000	0.000	0.691	0.648	0.403	0.490	**0.749**
W Avg	0.000	0.976	0.683	0.994	0.993	0.994	**0.996**
M Avg	0.000	0.165	0.444	0.648	0.668	0.709	**0.752**

where n is the number of samples,and $I(true) = 1$, $I(false) = 0$. Since Limited-memory Broyden Fletcher Goldfarb Shanno (L-BFGS) is more stable and able to handle large batches than gradient descent [151] , it is used in objective function for optimization. For each attack type, attack-specific feature sets are selected to improve the explainability of the model. Feature sets of attack types can explain what combination of features trigger alerts. Notably, besides the ACM approach, we also attempted to pursue clustering first and utilize clustering labels as prior knowledge as input to supervised machine learning (ML) algorithms; however, while this increases performance, the improvement was only approximately 3% in terms of the macro F1 score.

4.2.3 Numerical Results

This section discusses the baseline results and the performance of the proposed ACM on the SCVIC-APT-2021 dataset. Table 4.7 summarizes the performance of common machine learning models used as baselines. Due to the dataset's severe imbalance issue, the weighted average F1 score will be dominated by the performance of normal traffic; hence, the macro average F1 score is a more appropriate evaluation metric. For instance, logistic regression (LR) is incapable of classifying invasions as a linear model. Although SVM performs rather well in terms of weighted average F1 score, its macro average F1 score is limited because of

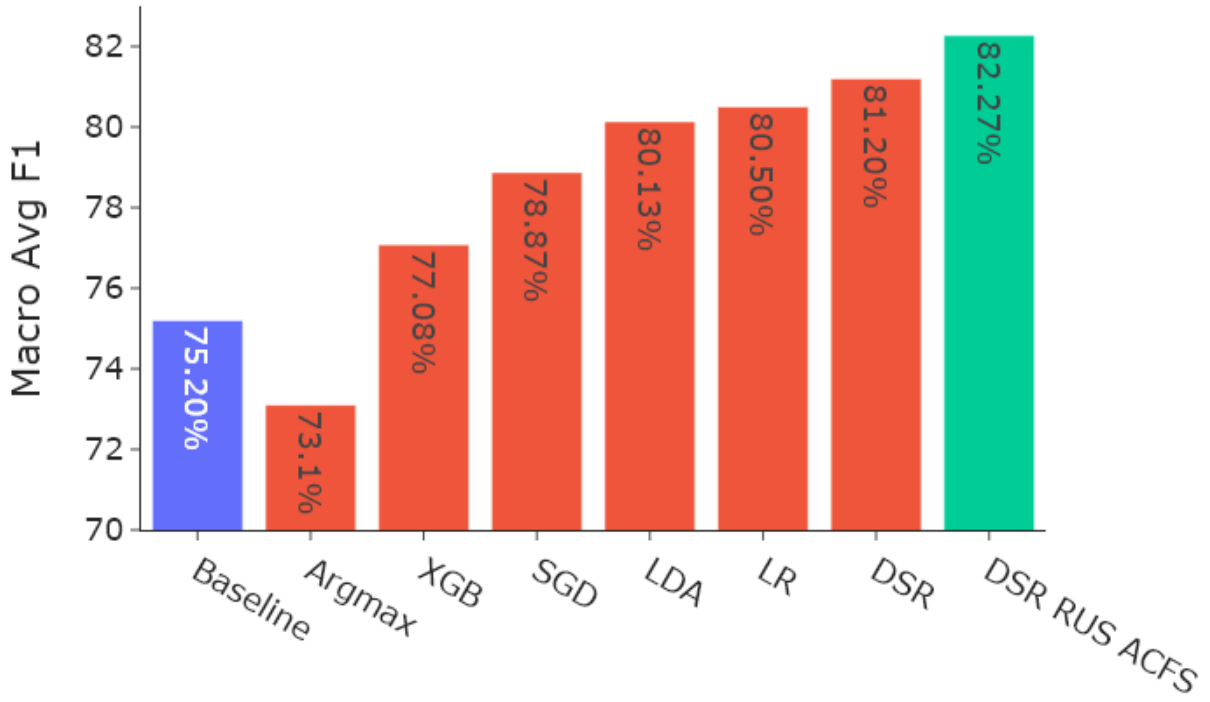


Figure 4.6: Attack Centric Method Results Using Different Aggregators as shown in 4.5

the poor performance of minority classes. Tree-based approaches yield better performance such as a decision tree (DT) can correctly identify the majority of samples with a macro average F1 score of 66.8%. XGBoost outperforms decision trees, with a macro average F1 score of 70.9%. Among the basic models, random forest (RF) achieves the highest macro average F1 score of 75.2%.

Due to the highest performance of RF, it is utilized as the base model in ACM and also served as a baseline model to demonstrate the efficiency of ACM. Fig. 4.6 presents the macro average F1 score of ACM calculated using various aggregators. Besides the machine learning-based aggregators, ACM's aggregator can be a simple argmax; however, an argmax aggregator does not outperform the baseline result whereas the aggregators powered by machine learning would outperform. As illustrated in the figure, complicated aggregators such as XGBoost can increase baseline performance, although simpler models such as LDA and LR perform better since they are able to mitigate overfitting. The proposed DSR aggregator is capable of improving the performance further up to 81.2%. In comparison to the highest baseline score, the proposed methodology enhances the macro average F1 score by 6%.

We further introduce random undersampling and ACFS into ACM. RF is used as an embedded feature selection method. Upon going through all possibilities, the best number of features is selected as 9. As illustrated in the result shows, the macro average F1 score improves up to 82.27% with random undersampling. As Figs. 4.7 and 4.8 show, feature importance of each attack type can be sorted and selected. For instance, the important features of Data Exfiltration (DE) and Initial Compromise (IC) are listed in Table 4.8. The meaning of the features can be found in CICFlowMeter [3]. As shown in the example, different attacks can be detected through different features.

Given that only DAPT2020 is publicly available among Table 2.2, we further evaluate ACM's performance under DAPT2020. Want et al. [167] evaluate anomaly detection methods on DAPT2020, but have not yet studied supervised algorithms; hence, we follow the similar procedure as described above, and the results are listed in Table 4.9. The results demonstrate that both the macro and weighted average F1 scores improved by 2% in comparison to the baseline (random forest).

4.2.4 Conclusions

This chapter has introduced a novel APT dataset, SCVIC-APT-2021, in which all stages of an APT are implemented. Furthermore, a novel ensemble APT detection scheme, called

[3]CICFlowMeter is available https://github.com/ahlashkari/CICFlowMeter

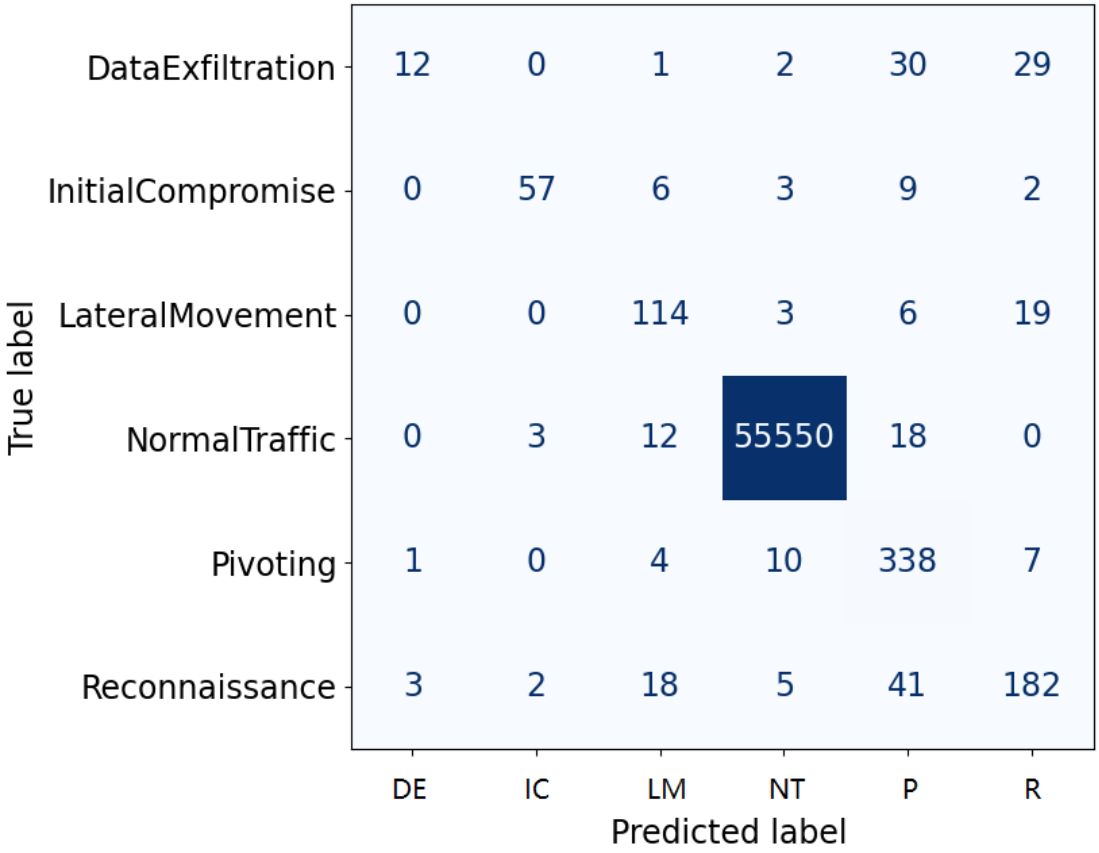


Figure 4.7: Confusion matrix of baseline

Table 4.8: Examples of Important Features of Attack Types

Attack Types	Top Important Features
DE	ACK Flag Count, Bwd Header Length, Bwd Init Win Bytes, Bwd Packet Length Max, Bwd Packet Length Std, Fwd Act Data Pkts, Idle Max, Idle Mean, Total Length of Fwd Packet, ...
IC	Bwd Init Win Bytes, Bwd Packet Length Max, Bwd Packet Length Mean, Bwd Packet Length Min, Fwd IAT Std, Fwd Seg Size Min, Packet Length Max, Total Length of Bwd Packet, ...

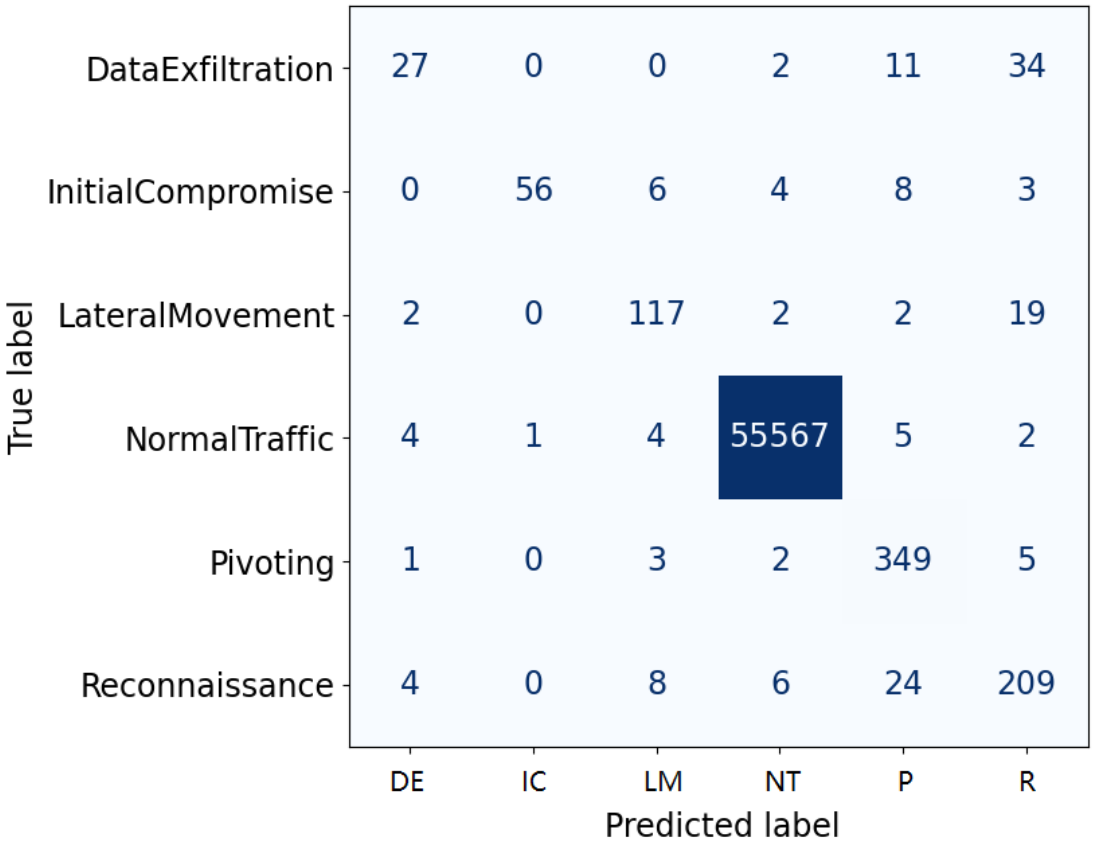


Figure 4.8: Confusion matrix of attack centric method using DSR aggregator with random under sampling and ACFS

Table 4.9: DAPT2020 Results

	RF (baseline)	ACM	Support
Benign	0.96	0.97	12742
Data Exfiltration	0.50	0.50	3
Establish Foothold	0.94	0.95	1720
Lateral Movement	0.55	0.56	490
Reconnaissance	0.87	0.90	2381
Macro Avg	0.76	0.78	17336
Weighted Avg	0.93	0.95	21930

Attack Centric Method (ACM) is proposed so to generate independent binary classifiers for each type of attack. With attack-centric feature selection (ACFS), ACM is capable of improving the performance and interpretability of the base model. To mitigate the potential overfitting issue, a Diagonal Softmax Regression (DSR) aggregator is also proposed. Out of various machine learning approaches, RF has shown the best performance with a macro F1 score of 75.2% under the SCVIC-APT-2021 dataset. The suggested ACM with DSR aggregator has been shown to increase the performance of the best baseline model by 6%. When combined with ACFS and random undersampling, ACM can increase the macro average F1 score to 82.27% corresponding to 9.4% improvement in the baseline performance.

Chapter 5

Bridging Networks and Hosts via Machine Learning-Based Intrusion Detection

5.1 Methodology

CIDSs gathers information from network packets and host data. In Section 5.1.2, the methods for creating and preparing the CIDS dataset are described in detail. We unveil the details of the proposed CIDS-Net to categorize network intrusions including imbalanced data via host-based data as described in Section 5.1.3.

5.1.1 Problem Definition

HIDSs analyze end device behaviors such as system call traces, application logs, and operating system audit trails. On the other hand, NIDSs sniff network packets that create flows, from which features are extracted and fed into machine/deep learning algorithms. To leverage the benefits of NIDS and HIDS, a CIDS incorporates network- and host-based features to improve the IDS performance.

Since CIDS dataset integrates features extracted from different domains (i.e., network and host), we propose a novel CIDS-Net to detect attacks using integrated features, and address the class imbalance issue.

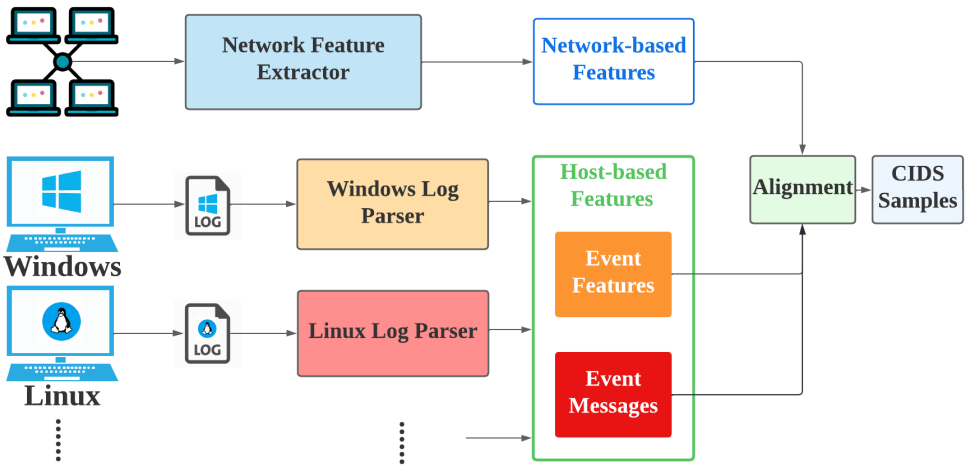


Figure 5.1: CIDS Dataset Formation Framework

5.1.2 CIDS Dataset Formation Framework

This section introduces a framework to produce CIDS samples which contain both network and host features.

Fig 5.1 depicts the detailed design of CIDS Dataset Formation Framework (CIDS-DFF) which retrieves features from network flows and system log files. This section explains an alignment approach along with forming CIDS instances as a result of varying timestamps and duration of host and network-based data.

The flow-based network feature extractors such as CICFlowMeter and NFStream are utilized in this research.

Each network flow is a collection of network packets sharing the same session key (i.e., IP addresses of source and destination hosts, TCP/UDP ports of source and destination, protocol type, and time window) from which network-based features are extracted.

Network-based instances are statistic, and they cover three aspects: flow directions, the feature type, and statistical summarization. For instance, 'Backward Packet Bytes Max" is defined as the maximum packet bytes in the backward flow.

Different Operating Systems (OSs) employ individual formats of logs to store host-based data. Security professionals can diagnose system issues and intrusions whereas ML-based HIDS are designed to mine patterns and learn outstanding features from vast amounts of data. In the Windows OSs, logs are saved in EVTX format, where each of the entry

Table 5.1: Example of Parsed Windows OS Logs

Message	Id	Version	Qualifiers	Level	Task	Opcode	Keywords	Provider Name	Time Created
The Windows Update service entered the stopped state.	7036	0	16384	4	0	0	-9.1E + 18	18	3/ 2/ 2018 14:50
An account was logged off. Subject: Security ID:S-*-*-** ...	0	0	0	1	0	0	0	7	4/ 5/ 2016 09:46

includes event information, warnings, and errors from OSs, services and applications. Several off-the-shelf log parsers (e.g., Get-WinEvent cmdlet and Log Parser) can be utilized to transform EVTX files to tabular and text data . The real-world examples parsed from Windows OS logs are listed in Table 5.1, with each entry containing both tabular/structured features and text/unstructured-based data. In the following content, event features are structured host-based features that can be employed by machine/deep learning models, such as level, task, and ID. Unstructured event messages are difficult for a log parser to interpret. Linux-based OS such as Ubuntu and CentOS have text-based logs including timestamp, entry providers and event messages. Consequently, to accommodate with Windows OS, Linux OS only have ProviderName as event features and all other features are NaN.

Each network-based instance correlates to several host-based instances due to the fact that each network flow is retrieved through a time-window, while each log entry is created with a specific timestamp. Hence, Algorithm 3 is employed to accommodate network- and host-based samples as though they were created in the same time frame. In brief, three

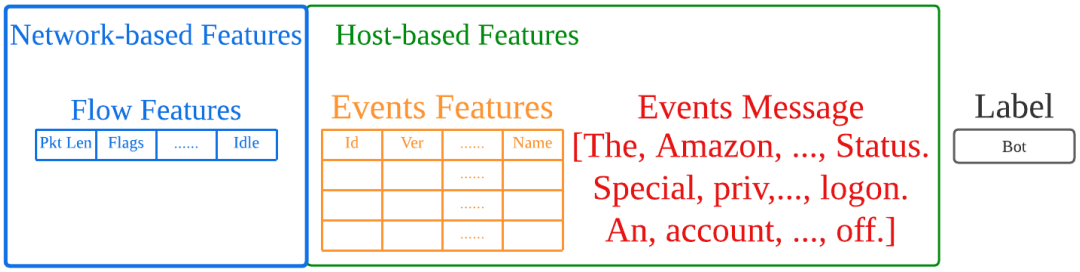


Figure 5.2: The structure of a CIDS sample which contains four components: network-based features, event features, event messages, and label. Event features and event messages are host-based features.

essential steps are followed. For each network-based instance, Algorithm 3 finds the host-based instances which share the same IP address and reside within the time window of the network-based sample. The event features of the host-based instances are concatenated into a matrix, and the event message are joined together to form a longer string. Finally, each sample inside the CIDS dataset is three-fold as follows: a network-based instance, the concatenated event features and the event messages.

The final shape and dimensionality of a CIDS instance is illustrated in Fig. 5.2. The network features are considered as a vector describing the statistical observation of a flow. The event features are represented by a matrix $\mathbb{R}^{n*m}$, where n denotes the number of log entries inside the time window and m represents the number of event features. Due to the text-based nature of event messages, Natural Language Processing (NLP) methods should be applied to process them. Further details are discussed in Section 5.1.3.

5.1.3 CIDS-Net

Given that the CIDS dataset comprises three different components in various forms, dimensions and kinds (i.e., network-based features, event features, and event messages), this chapter proposes CIDS-Net to bridge them and predict labels.

Figure 5.3 illustrates the architecture of CIDS-Net which comprises three encoders (i.e., Network Feature Encoder, Event Feature Encoder, and Event Message Encoder) tailored for the three distinct inputs and a Feed Forward Network (FFN) that receives the hidden states as input and predicts intrusions.

To encode network-based features into hidden states, a FFN is utilized since those features are essentially tabular data; however, a FFN can hardly outperform ML models

Algorithm 3: Network-based samples and host-based samples alignment. s_n, s_h, s_c represent network-based, host-based, and combined IDS sample; $\boldsymbol{S}$ is a set of samples; d is the time window of a network sample; e_f and e_m are event features and event message; $\boldsymbol{E}_f, \boldsymbol{E}_m$ are sets of e_f and e_m

Input: $\boldsymbol{S}_n$; $\boldsymbol{S}_h$
Output: CIDS Samples

$sort(samples_h)$;
$\boldsymbol{S}_c \leftarrow \emptyset$;
foreach $s_n \in \boldsymbol{S}_n$ **do**
    $ip \leftarrow get\_ip\_address(s_n)$;
    $d \leftarrow get\_timewindow(s_n)$;
    $\boldsymbol{E}_f \leftarrow \emptyset$;
    $\boldsymbol{E}_m \leftarrow \emptyset$;
    foreach $s_h \in \boldsymbol{S}_h$ **do**
        if $ip \in s_h$ *and* $s_h.timestamp \in d$ **then**
            $e_f, e_m \leftarrow s_h$;
            $\boldsymbol{E}_f \leftarrow \begin{bmatrix} \boldsymbol{E}_f \\ e_f \end{bmatrix}$;
            $\boldsymbol{E}_m \leftarrow \boldsymbol{E}_m \cup e_m$;
        end
    end
    $\boldsymbol{S}_c \leftarrow \boldsymbol{S}_c \cup (s_n, \boldsymbol{E}_f, \boldsymbol{E}_m)$;
end

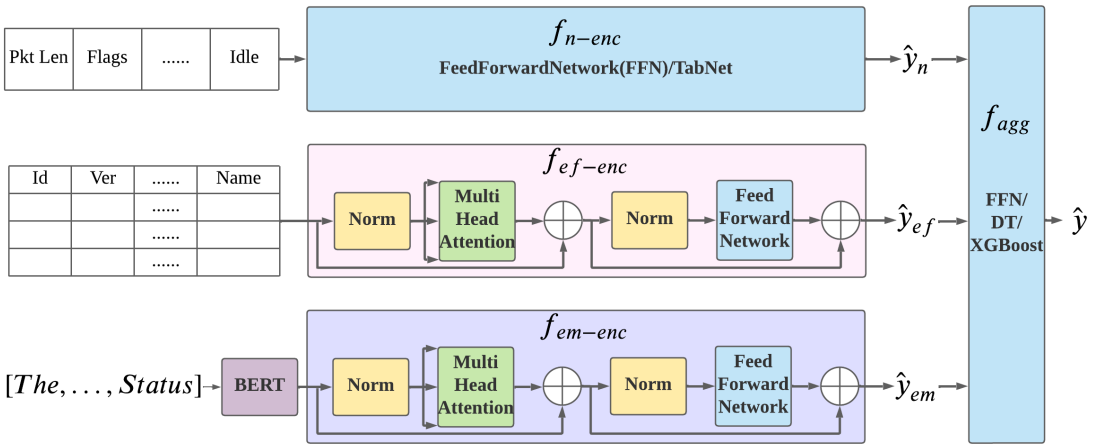


Figure 5.3: CIDS-Net; $\hat{y}_n$, $\hat{y}_{ef}$ and $\hat{y}_{em}$ stand for the outputs of network encoder, event feature encoder and event message encoder respectively; f_{n-enc}, f_{ef-enc} and f_{em-enc} denote the deep learning encoder for network features, event features and event messages.

such as XGBoost, SVM and Random Forest. Therefore, additional deep learning methods created for tabular data can also be considered as network feature encoders to enhance network feature representation. In this chapter, as TabNet is an explainable model with high performance, it is chosen as an example. Furthermore, the network feature encoder can be an identical layers, meaning the last FFN accepts network features as input.

Event features are the characteristics of a sequence of event entities that occur inside the time frame of a network flow; hence, they can be viewed as multivariate time series data. This research employs the state-of-the-art deep learning model for time series and NLP, and the Transformer encoder as the event feature encoder. In contrast, event messages are textual data that detail the reason or information of event entities. Using BERT word embedding, the event messages in our model are first turned into vectors. We use a pre-trained BERT network with fixed parameters to minimize the overfitting risk caused by the inclusion of such a large network into the CIDS-Net. Alternatively, other work embedding strategies such as Word2Vec and GloVe can be utilized. After word embedding, event messages are converted into a matrix form $\mathbb{R}^{i*j}$, where i is the number of words in the messages and j is the dimension of the embedding vectors. The event message encoder, like the event feature encoder, is a Transformer encoder.

To predict intrusions, CIDS-Net employs a FFN to aggregate the hidden states; nonetheless, FFN can hardly address the class imbalance issue which is commonly seen in IDS datasets. Hence, we replace the FFN with machine learning approaches to further enhance

the performance. The FFN aggregator is flexible and does not require the outputs of encoder having specific dimensionality. C Loss is formally defined as Eq. 5.1 where y is the actual label of an instance, $\hat{y}$ is the predicted label of the aggregator, $\hat{y}_n$ is the hidden states output by the network feature encoder, and α is a hyperparameter. C Loss requires that the number of hidden states of the network feature encoder's output to be same as the number of classes, despite the fact that the FFN aggregator has no restrictions.

$$C\_Loss = \alpha * loss(y, \hat{y}_n) + (1 - \alpha) * loss(y, \hat{y}) \tag{5.1}$$

Moreover, by using C Loss, network feature encoders can generate more complex outputs. For instance, if TabNet is used, the loss of the CIDS-Net can be expressed as follows:

$$loss = \alpha * (CE(\hat{y}_n, y) - \lambda * L_{sparse}) + (1 - \alpha) * CE(\hat{y}, y) \tag{5.2}$$

, where L_{sparse} is sparsity regularization, and λ is the hyperparameters of TabNet.

Due to the fact that machine learning methods such as Decision Tree, Random Forest, and XGBoost perform better on tabular data than deep learning models in general [204], this work proposes integrating traditional machine learning methods into the CIDS-Net, specifically replacing the aggregator. Because the majority of the traditional machine learning models do not support backpropagation, Fig. 5.4 depicts the procedure for incorporating machine learning algorithms into our CIDS-Net. During the training process, CIDS-Net with FFN aggregator is trained. The trained CIDS-Net encoders are fed with training set data and the latent states (i.e., $\hat{y}_n^{tr}$ $\hat{y}_{ef}^{tr}$ and $\hat{y}_{em}^{tr}$) are concatenated to combined latent states X_{ML}^{tr}. A machine learning model (e.g., decision tree, random forest and XGBoost) is trained with the combined latent states of training set X_{ML}^{tr} and training set labels y^{tr}.

For the testing process, the trained CIDS-Net encoders take test set data as input and predict three hidden states ($\hat{y}_n^{te}$ $\hat{y}_{ef}^{te}$ and $\hat{y}_{em}^{te}$). The three hidden states are concatenated to the combined hidden states X_{ML}^{te}. The trained machine learning model takes X_{ML}^{te} as input and makes the final predictions.

5.2 Experiments

This section introduces the two datasets generated by the CIDS Dataset Formation Framework, and presents abundant results and analysis of CIDS-Nets. Fig. 5.5 depicts the workflow of experiments and the relationship between the meta data and the generated datasets.

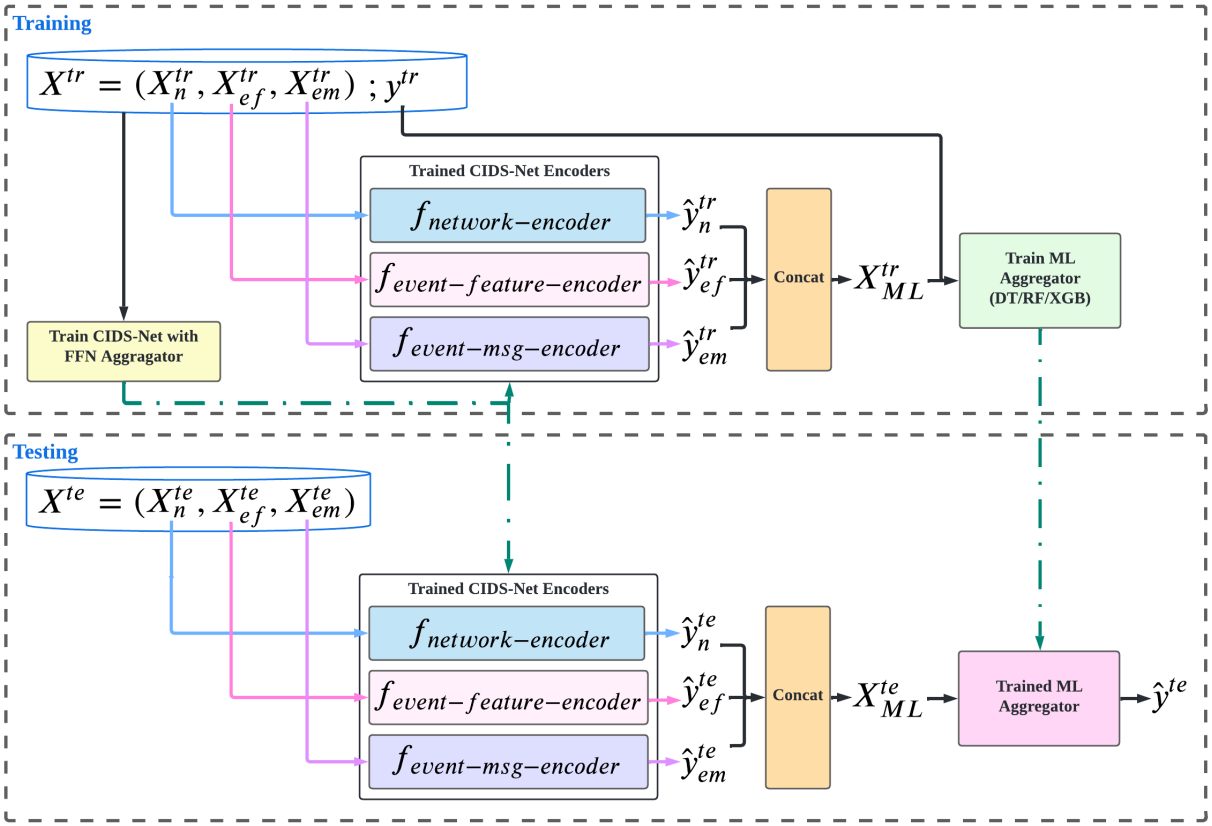


Figure 5.4: CIDS-Net using ML Aggregator. Solid lines stand for the data flow, and dashed lines represent an ML/DL model as the output of a process. X^{tr} is the training set data; y^{tr} is the training set label, and X^{te} is the test set data

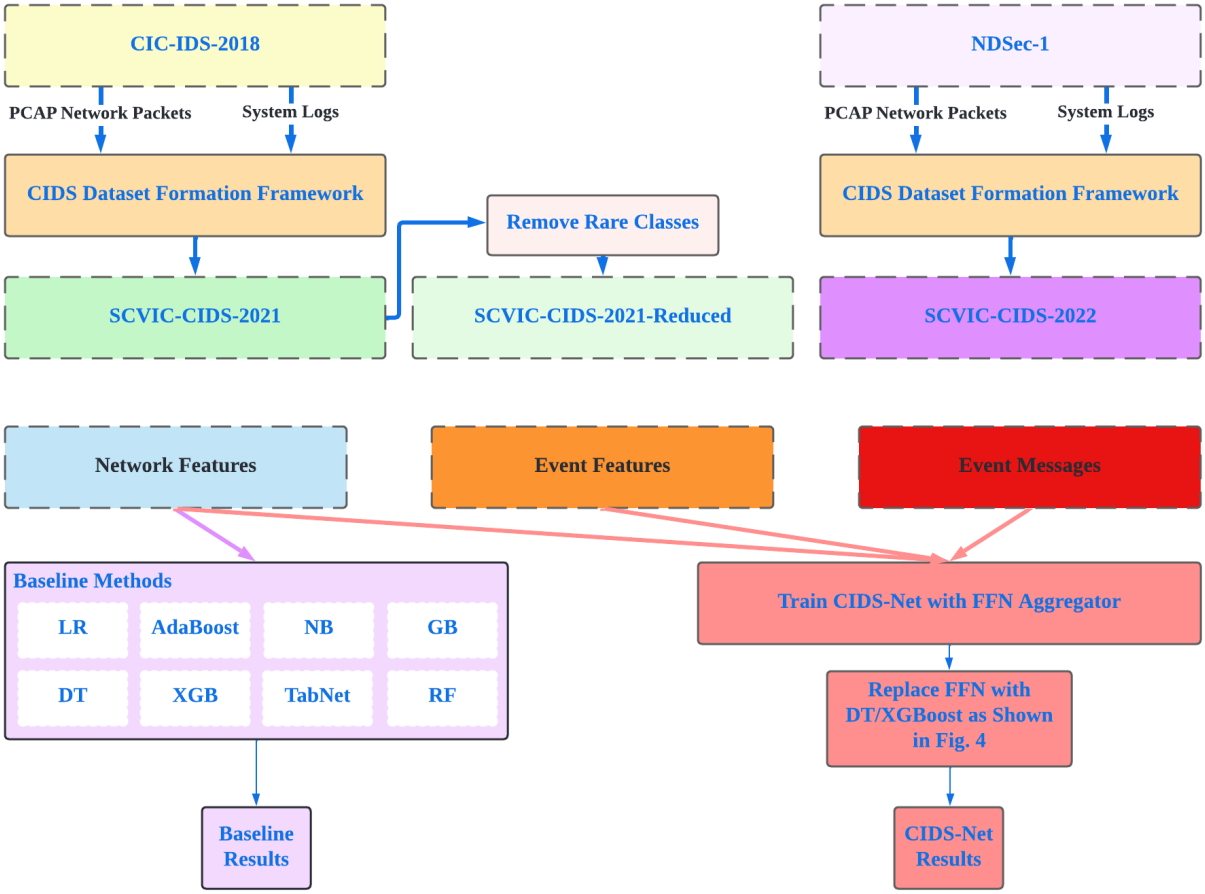


Figure 5.5: The flowchart for experiments and creating datasets.

5.2.1 Datasets

The CIDS-DFF aims to extract novel features and create CIDS dataset in a new format. As reported in [191], among all public network intrusion benchmark datasets, three of them contain complete metadata, network-based and host-based data, i.e., DARPA (former version dataset of NSL-KDD), NDSec-1, and CIC-IDS-2018. Since NSL-KDD and DARPA have been widely criticized for their out-of-date attacks and unrealistic network settings, this study uses the other two more recent datasets.

SCVIC-CIDS-2021 Dataset

The metadata of CIC-IDS-2018 is used to create SCVIC-CIDS-2021 due to the fact that CIC-IDS-2018 contains network traffic in PCAP format, host-based data such as system logs, and label information. Network-based features are extracted by CIC-IDS-2018, while host-related features are not yet retrieved. We use the raw CIC-IDS-2018 data and feed it into our CIDS-DFF to generate the SCVIC-CIDS-2021 dataset.

Since the tabular data offered in CIC-IDS2018 does not include IP addresses or timestamps, CICFlowMeter [134] is utilized with the same timewindow (i.e. two minutes) to reconstruct network-based features, and the instances are relabeled Based on the attack traces provided in [3].

In CIC-IDS-2018, the logs are obtained from Windows and Linux OS. CIDS-DFF transforms Windows OS logs to tabular format using Get-WinEvent and obtains ProviderName and event messages from Linux OS logs.

Algorithm 3 is then employed to align network- and host-based data.

Massive amount of data (more than 1 TB) is included in CIC-IDS-2018; thereby most of the research (e.g., [23, 28, 73]) utilize a subset of the dataset as reported in [140]. When host-based features are introduced, SCVIC-CIDS-2021 grows significantly. We randomly undersample the benign traffic to 50% since it often has an F1 score of higher than 99% [74, 28]. To avoid an excessive quantity of missing data from being included, SCVIC-CIDS-2021 removes the samples lacking host-based data. To illustrate the effectiveness of this technique, we also present evaluations when the CIDS-Net is not provided with host-based data. After excluding samples that lack host-based data, a number of minority classes include less than twenty instances (e.g., 'Brute Force - Web' (15), 'Brute Force - XSS' (6), 'Infiltration' (8), and 'SQL Injection' (8)). Even with over/under-sampling, minority classes will have a substantial impact on the performance of deep learning models; thus, we created SCVIC-CIDS-2021-Reduced, which removes these rare classes. Without these rare classes, oversampling is achievable without overfitting problems. Our previous work conducts the experiments only under the SCVIC-CIDS-2021-Reduced dataset. This study employs the full version of the SCVIC-CIDS-2021 dataset.

The network-based features are same as hose in CIC-IDS-2018. Event features have the dimension of $\mathbb{R}^{28*8}$ because the highest number of event entities is 28. Uncased pre-trained BERT Tokenizer tokenizes text-based event messages with the maximum length of 100, and the uncased pre-trained BERT encoder converts the tokenized messages to word embedding with the dimension of $\mathbb{R}^{100*768}$. Table 5.2 and 5.3 show the distribution of training and testing sets in SCVIC-CIDS-2021 and SCVIC-CIDS-2021 respectively.

Table 5.2: SCVIC-CIDS-2021 Class Distribution

	Training Set	Testing Set	Total
Benign	308823	151724	460547
Bot	60584	29876	90460
Brute Force -Web	27	15	42
Brute Force -XSS	10	6	16
DDOS-HOIC	136883	67713	204596
DDOS-LOIC-HTTP	39009	19176	58185
DDOS-LOIC-UDP	732	370	1102
DoS-GoldenEye	2301	1133	3434
DoS-Hulk	13382	6559	19941
DoS-SlowHTTPTest	10674	5256	15930
DoS-Slowloris	1416	680	2096
FTP-BruteForce	32234	15906	48140
Infiltration	21	8	29
SQL Injection	7	8	15
SSH-Bruteforce	11030	5531	16561
Sum	617133	303961	921094

Table 5.3: SCVIC-CIDS-2021-Reduced Class Distribution

	Training Set	Testing Set	Total
Benign	308375	152172	460547
Bot	60767	29693	90460
DDOS-HOIC	137147	67449	204596
DDOS-LOIC-HTTP	39019	19166	58185
DDOS-LOIC-UDP	760	342	1102
DoS-GoldenEye	2271	1163	3434
DoS-Hulk	13388	6553	19941
DoS-SlowHTTPTest	10579	5351	15930
DoS-Slowloris	1394	702	2096
FTP-BruteForce	32222	15918	48140
SSH-Bruteforce	11143	5418	16561
Sum	617065	303927	920992

Table 5.4: SCVIC-CIDS-2022 Class Distribution

	Training Set	Testing Set	Total
BOTNET	52	40	92
BRUTEFORCE	2150	1006	3156
DOS	12677	6389	19066
EXPLOIT	4	2	6
MALWARE	22	12	34
MISC	31	18	49
NORMAL	4716	2284	7000
PROBE	777	307	1084
WEBATTACK	18	12	30
Sum	20447	10070	30517

SCVIC-CIDS-2022 Dataset

SCVIC-CIDS-2021 is created using the raw data in CIC-IDS-2018, while SCVIC-CIDS-2022 is formed from NDSec-1 [31] meta-data by following the similar procedure in Section 5.2.1.

The network flows are preprocessed, and categorical features are one-hot encoded. Rather than EVTX files, NDSec-1 provides preprocessed tabular format logs containing structured and unstructured data. The available event features are 'Level' and 'Provider-Name'. The network and host data are aligned using Algorithm 3, and samples lacking host-specific data are eliminated. Because the spoofing class includes only one sample, it is also removed. NDSec-1 is a smaller dataset than CIC-IDS-2018; as such, we attempt to preserve as much information as possible. The shape of event features is $\mathbb{R}^{21*8}$, and the shape of event messages is $\mathbb{R}^{512*768}$, where 512 is the maximum length of BERT's inputs. The dataset is partitioned into training and test sets as listed in Table 5.4.

Table 5.5: Experimental Settings

Network Encoder	Identity, FFN, TabNet
Event Feature/Msg Encoder	Transformer
Aggregator	FFN, Decision Tree, XGBoost
Loss	Cross Entropy (default), C Loss
Optimizer	AdamW
GPU	GeForce RTX 3070
CPU	Intel Xeon

Table 5.6: Machine learning methods results of SCVIC-CIDS-2021-Reduced when only network-based features are used (Baseline Results)

	LogisticRegression	AdaBoost	NaiveBayes	TabNet	GradientBoost	DecisionTree	XGBoost	RandomForest
Benign	0.780	0.891	0.698	0.999	0.999	0.999	0.999	1.000
Bot	0.009	0.000	1.000	1.000	1.000	1.000	1.000	1.000
DDOS-HOIC	0.547	0.788	0.998	1.000	1.000	1.000	1.000	1.000
DDOS-LOIC-HTTP	0.000	0.000	0.998	1.000	1.000	1.000	1.000	1.000
DDOS-LOIC-UDP	0.000	0.000	0.423	0.715	0.746	0.759	0.761	0.761
DoS-GoldenEye	0.000	0.000	0.828	0.995	0.996	0.999	1.000	1.000
DoS-Hulk	0.763	0.372	1.000	0.999	1.000	1.000	1.000	1.000
DoS-SlowHTTPTest	0.000	0.000	0.500	0.440	0.544	0.543	0.543	0.542
DoS-Slowloris	0.000	0.000	0.020	0.907	0.905	0.993	0.994	0.996
FTP-BruteForce	0.498	0.853	0.498	0.866	0.875	0.875	0.875	0.875
SSH-Bruteforce	0.000	0.000	0.999	1.000	1.000	1.000	1.000	1.000
Macro Avg	0.236	0.264	0.724	0.902	0.915	0.924	**0.925**	**0.925**
Weighted Avg	0.556	0.674	0.810	0.982	0.984	0.985	**0.985**	**0.985**

5.2.2 CIDS-Net Results

This section details the results of CIDS-Nets using various network encoders and aggregators.

Section 5.2.2 shows the CIDS-Net results using SCVIC-CIDS-2021-Reduced which removes the classes with few samples. Section 5.2.2 presents the results of the complete version of SCVIC-CIDS-2021. Moreover, to demonstrate the generalization ability of the proposed methodology, the results of SCVIC-CIDS-2022 are shown in Section 5.2.2.

SCVIC-CIDS-2021-Reduced

This section describes the SCVIC-CIDS-2021-Reduced dataset results in depth. As the results have already been presented in our previous work [153], they are included in the

Table 5.7: CIDS-Net results on SCVIC-CIDS-2021-Reduced using FFN as aggregator and identity, FFN, and TabNet are utilized as network encoder

	Baseline(XGB)	CIDS-Net (Using FFN as Aggregator)			
		Identity	FFN	FFN (CLoss)	TabNet
Benign	0.9995	0.9995	0.9993	0.9963	0.9998
Bot	1.0000	0.9988	0.9978	0.9901	0.9994
DDOS-HOIC	1.0000	1.0000	0.9996	0.9998	1.0000
DDOS-LOIC-HTTP	0.9997	0.9997	0.9982	0.9989	0.9999
DDOS-LOIC-UDP	**0.7609**	**0.9771**	**0.9884**	**0.9899**	**0.9927**
DoS-GoldenEye	1.0000	0.9923	0.9914	0.9940	0.9996
DoS-Hulk	1.0000	0.9992	0.9998	0.9665	0.9988
DoS-SlowHTTPTest	**0.5425**	**0.9228**	**0.9964**	**0.9233**	**0.9990**
DoS-Slowloris	0.9936	0.9894	0.9929	0.9716	0.9993
FTP-BruteForce	**0.8749**	**0.9721**	**0.9991**	**0.9713**	**0.9996**
SSH-Bruteforce	0.9999	0.9998	0.9984	0.9994	0.9998
Macro Avg	**0.9246**	**0.9864**	**0.9965**	**0.9819**	**0.9989**
Weighted Avg	**0.9848**	**0.9967**	**0.9990**	**0.9934**	**0.9998**

appendices.

To provide an appropriate baseline, Table 5.6 summarizes the results of several machine/deep learning algorithms when only network-based features are used. Because network-based features dominate the final performance, different encoders for network-based features are tested, including an Identity Layer, FFN, and TabNet. To demonstrate the proposed CIDS-Net's ability to incorporate various models, this work runs experiments with two aggregators: an FFN aggregator and a traditional machine learning aggregator as shown in Fig. 5.4, which acts as a wrapper for all pre-trained models. Additionally, this section evaluates the effects of C Loss and cross-entropy loss. Table 5.6 illustrates the baseline results of several machine learning and deep learning methods utilizing just network-based features. Among these models, tree-based methods (such as Decision Tree (DT), XGBoost, and Random Forest(RF)) demonstrate the best performance. Although Decision Tree performs negligibly (i.e., 10^{-4}) worse than XGBoost and Random Forest on network-based features, as a basic but fast machine learning model, it is used for the following experiments and analysis. As XGBoost and Random Forest perform the best, their results are serve as reasonable benchmarks for comparing the CIDS-Net results to the results under the host-based data.

TabNet, a deep learning model designed for tabular datasets, is evaluated as well; nonetheless, it cannot outperform the DT, XGBoost, and RF. When the results of each

Table 5.8: CIDS-Net network encoder performance of SCVIC-CIDS-2021-Reduced and FFN, and TabNet are utilized as network encoder

	XGBoost	**FFN Aggregator**		
		FFN	**FFN (C Loss)**	**TabNet**
Benign	0.9995	0.0000	0.9668	0.9965
Bot	1.0000	0.0000	0.9885	0.9983
DDOS-HOIC	1.0000	0.0000	0.9897	0.9951
DDOS-LOIC-HTTP	0.9997	0.8202	0.9642	0.9997
DDOS-LOIC-UDP	0.7609	0.0000	0.3911	0.7365
DoS-GoldenEye	1.0000	0.0000	0.3649	0.9910
DoS-Hulk	1.0000	0.0000	0.9861	0.9991
DoS-SlowHTTPTest	0.5425	0.0000	0.4710	0.4978
DoS-Slowloris	0.9936	0.0000	0.6173	0.8830
FTP-BruteForce	0.8749	0.0000	0.7187	0.8493
SSH-Bruteforce	0.9999	0.0000	0.9628	0.9998
macro avg	0.9246	0.0746	0.7656	0.9042
weighted avg	0.9848	0.0517	0.9487	0.9796

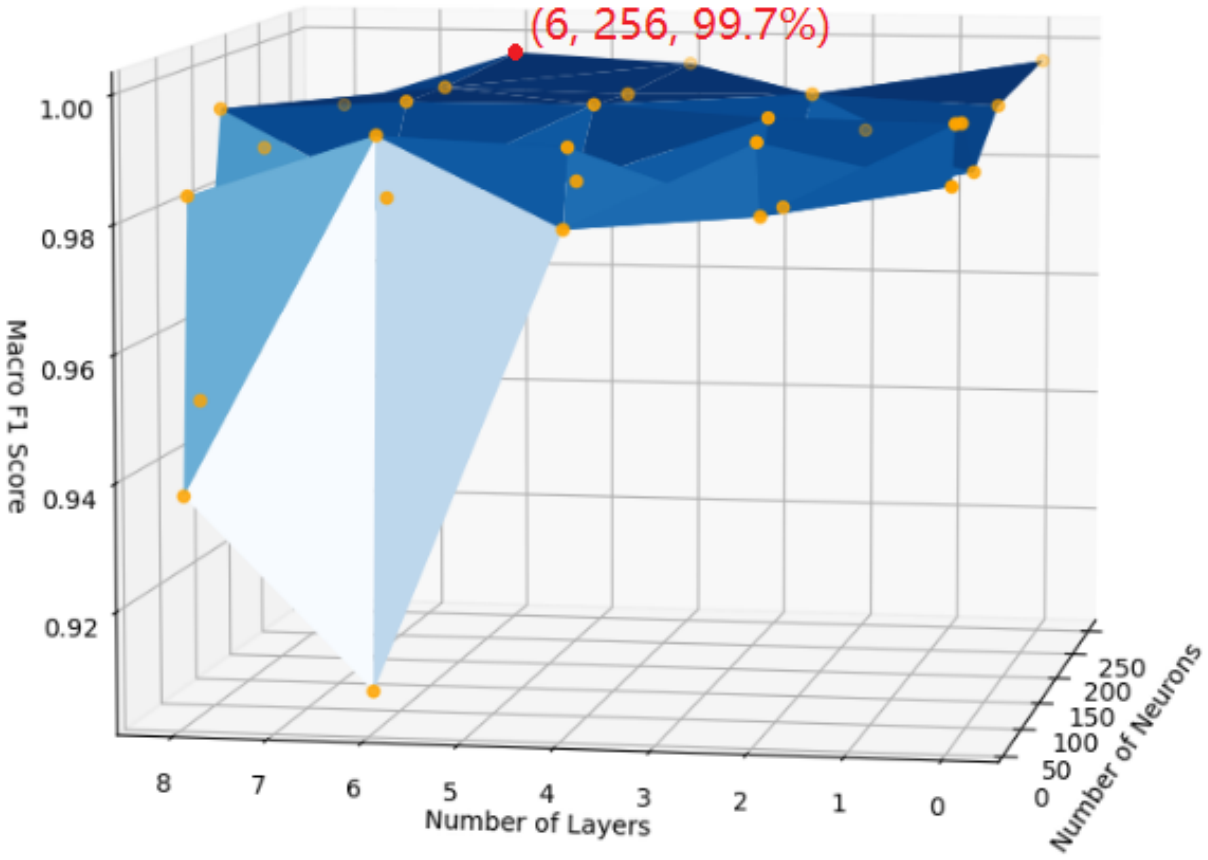


Figure 5.6: The change of CIDS (using FFN as network encoder and aggregator) macro F1 score along with the number of the network encoder's layers and neurons

Table 5.9: CIDS-Net results of SCVIC-CIDS-2021-Reduced using Decision Tree aggregator and identity, FFN, and TabNet are utilized as a network encoder

	Baseline(XGB)	Replacing Aggregator with Decision Tree			
		Identity	FFN	FFN (CLoss)	TabNet
Benign	0.9995	1.0000	0.9999	0.9999	1.0000
Bot	1.0000	1.0000	0.9999	0.9995	0.9999
DDOS-HOIC	1.0000	1.0000	1.0000	1.0000	1.0000
DDOS-LOIC-HTTP	0.9997	0.9999	0.9998	0.9999	0.9999
DDOS-LOIC-UDP	**0.7609**	**0.9942**	**0.9942**	**0.9927**	**0.9913**
DoS-GoldenEye	1.0000	0.9991	0.9987	0.9987	0.9996
DoS-Hulk	1.0000	1.0000	1.0000	0.9999	0.9998
DoS-SlowHTTPTest	**0.5425**	**0.9999**	**0.9992**	**0.9998**	**1.0000**
DoS-Slowloris	0.9936	1.0000	0.9979	0.9972	0.9993
FTP-BruteForce	**0.8749**	**1.0000**	**0.9999**	**0.9999**	**1.0000**
SSH-Bruteforce	0.9999	1.0000	1.0000	1.0000	1.0000
macro avg	**0.9246**	**0.9994**	**0.9990**	**0.9989**	**0.9991**
weighted avg	**0.9848**	**1.0000**	**0.9999**	**0.9999**	**1.0000**

class are compared, 'DDOS-LOIC-UDP,' 'DoS-SlowHTTPTest,' and 'FTP-BruteForce' are outperformed by the others (i.e., less than 90%). According to the attack traces in CIC-IDS-2018, the IP address shared by the victims of these three assaults. Specifically, both the slow HTTP test and FTP brute force use the TCP protocol, resulting in similar network characteristics. Given that the performance of other classes cannot be enhanced further, we anticipate that the performance of these three classes will be enhanced utilising host-based data without affecting the performance of other classes. Table 5.7 highlights the F1 score results of the CIDS-Net when different network encoders are combined with FFN as the aggregator. If network features are not transformed into hidden states before being feed into the FFN aggregator, the macro F1 score is 98.6%. Upon using the host-based features, the macro F1 score rose by more than 5% relative to one of the best baselines (XGBoost). The three attacks with the highest rate of improvement are 'DDOS-LOIC-UDP,' 'DoS-SlowHTTPPTest,' and 'FTP-BruteForce'. The F1 scores of these classes have increased by 21.62%, 38.03%, and 9.72%, respectively. Even though these three attacks have similar network patterns, they elicit distinct system reactions, such as an FTP login failure and an increase in the number of HTTP headers. When the FFN is utilized as a network encoder, the macro F1 score rises by more than 1%, indicating that the encoded hidden states can be more effectively with the FFN aggregator when extra parameters are given. To evaluate the effect of network size, we analyse the influence of varying numbers

of FFN layers and neurons on the macro average F1 score when the CIDS-Net encoder and aggregator is FFN. As seen in Figure 5.6, the number of layers has a little impact on performance, and when the number of neurons exceeds 32, performance stays consistent. This indicates that the planned CIDS-Net is resistant to hyperparameter adjustments (i.e., the number of layers and neurons).

This study also evaluates the C loss. The primary goal of C Loss includes avoiding the training of an extra NIDS and implementing a more complicated network encoder (e.g., TabNet). We explore the performance of a network encoder as a classifier later in this section. When C Loss is utilized, CIDS performance is marginally reduced; nevertheless, when TabNet (which also benefits from C Loss) is utilized as the network encoder, the macro average F1 score increases to 99.89%, which is 0.2% higher than when FFN is used as the network encoder. The performance of network encoders is assessed because of the potential that host-based data may not be accessible. When the FFN is used as an aggregator, the FFN network encoder (without C Loss) performs poorly as a classifier; however, when C Loss is applied, the performance of the FFN network encoder improves from 7.46% to 76.56%. When TabNet (which also benefits from C Loss) is employed as a network encoder, its performance is almost same to when TabNet is trained only with network-based features, proving the value of C Loss. Even with C Loss and TabNet, the performance of the network encoder does not surpass the XGBoost baseline. Consequently, NIDS based on machine learning remain as the dominant strategy until a deep learning model outperforms them under tabular datasets.

As shown in the baseline results, Machine Learning methods outperform deep learning models on tabular data, and previous results indicate that the ability to classify or encode network-based features affects the overall results. As a result, we replace the aggregator with an ML algorithm as illustrated in Fig. 5.4. Specifically, Decision Tree (DT) is used since its performance is comparable to that of XGBoost and Random Forest but is faster. The table 5.9 shows the outcomes of replacing the aggregator in all prior models with DT. In terms of macro average F1 score, the traditional ML aggregator outperforms all base CIDS-Nets.

SCVIC-CIDS-2021

In Section 5.2.2, we demonstrated the effectiveness of the proposed methodology by presenting the results of CIDS-Net on the SCVIC-CIDS-2021-Reduced dataset. We use all classes in the SCVIC-CIDS-2021 dataset in this section.

Table 5.10 lists the baseline performance of various machine learning models using

Table 5.10: Machine learning methods results of SCVIC-CIDS-2021 when network-based features are used (Baseline Results)

	GradientBoost	**DecisionTree**	**XGBoost**	**RandomForest**
Benign	0.999	0.999	0.999	0.999
Bot	1.000	1.000	1.000	1.000
Brute Force -Web	0.882	0.968	1.000	1.000
Brute Force -XSS	1.000	1.000	1.000	1.000
DDOS-HOIC	1.000	1.000	1.000	1.000
DDOS-LOIC-HTTP	1.000	1.000	1.000	1.000
DDOS-LOIC-UDP	0.772	0.776	0.777	0.775
DoS-GoldenEye	0.964	1.000	1.000	1.000
DoS-Hulk	0.998	1.000	1.000	1.000
DoS-SlowHTTPTest	0.550	0.550	0.550	0.550
DoS-Slowloris	0.958	0.991	0.989	0.993
FTP-BruteForce	0.877	0.877	0.877	0.877
Infiltration	0.203	0.800	1.000	1.000
SQL Injection	0.800	0.933	0.857	0.933
SSH-Bruteforce	1.000	1.000	1.000	1.000
Macro Average	0.867	0.926	0.937	0.942
Weighted Average	0.985	0.985	0.985	0.985

Table 5.11: CIDS-Net results of SCVIC-CIDS-2021 using the proposed machine learning aggregator

	Baseline	**CIDS-Net with Identity Network Encoder**			
		FFN	**Decision Tree**	**RandomForest**	**XGBoost**
Benign	0.9995	0.9988	0.9999	1.0000	1.0000
Bot	1.0000	0.9987	0.9999	1.0000	1.0000
Brute Force -Web	1.0000	0.7879	0.9677	0.9677	1.0000
Brute Force -XSS	1.0000	1.0000	1.0000	1.0000	1.0000
DDOS-HOIC	1.0000	1.0000	1.0000	1.0000	1.0000
DDOS-LOIC-HTTP	0.9998	0.9997	0.9999	0.9999	0.9999
DDOS-LOIC-UDP	**0.7748**	**0.9635**	**0.9933**	**0.9946**	**0.9933**
DoS-GoldenEye	1.0000	0.9917	0.9991	1.0000	1.0000
DoS-Hulk	1.0000	0.9999	1.0000	1.0000	1.0000
DoS-SlowHTTPTest	**0.5502**	**0.9117**	**1.0000**	**1.0000**	**1.0000**
DoS-Slowloris	0.9934	0.9920	0.9993	1.0000	1.0000
FTP-BruteForce	**0.8771**	**0.9669**	**1.0000**	**1.0000**	**1.0000**
Infiltration	1.0000	0.0800	0.6316	0.9333	1.0000
SQL Injection	0.9333	0.8889	0.9412	0.9333	1.0000
SSH-Bruteforce	1.0000	0.9992	1.0000	1.0000	0.9999
Macro Average	**0.9419**	**0.9052**	**0.9688**	**0.9886**	**0.9995**
Weighted Average	**0.9852**	**0.9958**	**0.9999**	**1.0000**	**1.0000**

Table 5.12: Machine learning methods results of SCVIC-CIDS-2022 when only network-based features are used (Baseline Results)

	RandomForest	**GradientBoost**	**XGBoost**	**DecisionTree**
BOTNET	0.889	0.783	0.919	0.861
BRUTEFORCE	0.999	0.998	0.999	0.997
DOS	0.999	0.998	0.999	0.999
EXPLOIT	0.667	0.667	0.500	0.667
MALWARE	0.909	0.909	0.909	0.800
MISC	0.261	0.519	0.571	0.563
NORMAL	0.989	0.986	0.990	0.989
PROBE	0.997	0.987	0.992	0.997
WEBATTACK	0.818	0.769	0.857	0.889
Macro Avg	0.836	0.846	0.859	0.862
Weighted Avg	0.994	0.993	0.995	0.994

network-based features. The poor performed models are not included. Tree-based machine learning algorithms include DT, XGBoost, and RF obtain satisfactory results, among which RF achieving the highest macro F1 score of 94.2%. Thus, the RF is selected as baseline for comparison.

Given the previous section's examination and comparison of various network encoders and aggregators, this section will focus on the one with the best performance with fewest parameters: the CIDS-Net, which uses the Identity layer as the network encoder and traditional ML as the aggregator. As shown in Table 5.11, without replacing the aggregator, the CIDS-Net performs even worse than the baseline, due to the low F1 score of minority classes. When the FFN is substituted with traditional machine learning algorithms (e.g., DT, XGBoost, and RF), performance is improved and outperforms the baseline. The XGBoost aggregator outperforms the baseline by 5.76% in terms of macro F1 score and works well in minority classes.

SCVIC-CIDS-2022

SCVIC-CIDS-2022 is tested using CIDS-Net to demonstrate that the CIDS dataset formation framework and the CIDS-Net can be generalized. Table 5.12 summarizes the baseline performance of various machine learning and deep learning methods. DT achieves the highest macro average F1 score of 86.2%, while XGBoost achieves the highest weighted

Table 5.13: CIDS-Net results of SCVIC-CIDS-2022 using the proposed machine learning aggregator

	Baseline	**CIDS-Net with Identity Network Encoder**			
		FFN	**GB**	**XGBoost**	**DecisionTree**
BOTNET	0.861	0.333	0.716	0.841	0.759
BRUTEFORCE	0.997	0.976	0.999	0.999	0.999
DOS	0.999	0.928	1.000	1.000	0.999
EXPLOIT	0.667	0.053	0.400	0.500	1.000
MALWARE	0.800	0.019	0.818	0.909	0.909
MISC	0.563	0.250	0.621	0.774	0.750
NORMAL	0.989	0.763	0.991	0.993	0.991
PROBE	0.997	0.974	1.000	1.000	0.997
WEBATTACK	0.889	0.468	0.818	0.909	0.815
Macro Avg	0.862	0.529	0.818	0.880	0.913
Weighted Avg	0.994	0.891	0.995	0.997	0.995

average F1 score of 99.95%. When the identity layer is used as the network encoder and FFN is used as the aggregator, CIDS-Net outperforms the best baseline performance; however, when the aggregator is replaced with XGBoost and DT, the predictions are boosted. When DT is used as an aggregator, the CIDS-Net achieves the highest macro average F1 score of 91.3%, 5.1% higher than the best baseline, and the XGBoost aggregator achieves the highest weighted average F1 score of 99.7%.

5.2.3 Interpretation of the model

This section introduces theoretical support for combining network-based and host-based data. Furthermore, a noise estimation method is presented in this section as well.

Machine learning and deep learning errors can be decomposed into three types: noise, which is the inherent mistake in the data, bias, and variation. Bias and variance are intensively explored, whereas this study introduces additional features (i.e., host-based data) that genuinely boost performance when compared to using solely network-based features. Thus, we focus on the Noise term to determine how host-based features function efficiently.

Eq 5.3 formulates the decomposition of classification errors, where L is a loss function,

y is the true label of testing samples, $\hat{y}$ is the predicted label from the trained model, $\hat{y}_*$ is the optimal theoretical predictions of the Bayes Classifier, and $\hat{y}_m = \arg\min_{\hat{y}'} E_D[L(\hat{y}, \hat{y}')]$ is the main prediction for a loss function [60].

$$\begin{aligned} E[L(\hat{y} - y)] &= Noise + Bias + Variance \\ &= c_1 E_y[L(y, \hat{y}_*)] + \\ & L(\hat{y}_*, \hat{y}_m) + \\ & c_2 E_D[L(\hat{y}_m, \hat{y})] \end{aligned} \tag{5.3}$$

Since we aim at classification, 0-1 Loss $I(y \neq \hat{y})$ is used, and the noise is formulated as in Eq. 5.4.

$$E_y[L(y, \hat{y}_*)] = E_y[I(y \neq \hat{y}_*)] \tag{5.4}$$

where $\hat{y}_*$ is predicted by the theoretical Bayes optimal model which is difficult to obtain. Thus, we approximate it by two approaches to measure the upper and lower limit of actual data noise. For the lower limit, the optimal model is assumed to have zero bias and variance, as well as an infinite capacity for generalization. Keeping this in mind, the noise is formulated as follows:

$$E_y[L(y, \hat{y}_*)] = Pr(y \neq \hat{y}_*) \tag{5.5}$$

As we assume the optimal model has infinite generalization ability to ensure its bias and variance are 0, $\hat{y}_* = y$ if $x \notin D_{tr}$. In other words the classifier makes wrong predictions only on the noise samples that have the same features as some samples in the training set but with different labels; for instance, a test sample is (x_{te}, c_1), and three training points $\{(x_1, c_1), (x_2, c_2), (x_2, c_2)\}$ have the same feature as the test sample $x_{te} = x_1 = x_2 = x_3$. Without other knowledge, any Bayes model will tend to predict the test sample as $\hat{y}_{te} = mode(c_1, c_2, c_2) = c_2$; however the predicted label is different from the actual label $\hat{y}_{te} \neq c_i$, resulting in errors in prediction process. Additionally, we can conclude that adding information improves the overall performance as long as it is capable of reducing the amount of noise samples without adding new ones. On the other hand, more information does not add much benefit to non-noisy samples.

It is worth noting that although Eq. 5.5 reveals why and when additional features will work, it is a theoretical formulation that cannot be accurately calculated due to $E_y[\cdot] = \int_y \cdot Pr(y|X)dy$ which requires infinity samples. ; hence, we propose Algorithm 4 which uses bootstrap techniques to estimate the noise in the dataset.

Although assumed optimal model (i.e., 0 bias and variance and infinity generalization ability) can eliminate the influence (i.e., bias and variance) of a machine learning model and locate actual noise samples, it is only sensitive to the samples in the training set. However,

Algorithm 4: Estimation of Data Noise

Input: D_{tr}, D_{te}

Output: noise `// Data noise in the machine learning dataset`

$noise \leftarrow 0$;
foreach $D_{tr}^{'} \in Bootstrap(D_{tr})$ **do**
  $noise^{'} \leftarrow 0$;
  foreach $x_{te}, y_{te} \in D_{te}$ **do**
    $P \leftarrow wrong\_predictions(D_{tr}^{'}, x_{te}, y_{te})$;
    $noise^{'} \leftarrow noise^{'} + P$;
  end
  $noise \leftarrow noise + \frac{1}{|D_{te}|} noise^{'}$
end
$noise \leftarrow \frac{1}{|Bootstrap(D_{tr})|} noise$

even if additional information is random perturbation, the noise computed from the optimal model $noise_{opt}$ can be decreased as well; consequently, it is used for detecting the absolute noise in the data and providing a lower limit of actual noise $noise_{opt} < noise_{true}$. In order to estimate the upper limit of data noise, we use an actual machine learning model and assume its $bias = 0$, which is equivalent to $\hat{y}_* = \hat{y}_m$. With this in mind, the noise can be calculated as follows

$$E_y[L(y, \hat{y}_*)] = E_y[I(y \neq \hat{y}_m)] \tag{5.6}$$

, where $\hat{y}_m$ is predicted by majority voting of a set of classifiers trained by bootstrapping datasets. In practice, models with lower bias will lead $\hat{y}_m$ to be more similar to $\hat{y}_*$; that is, if a model's bias approaches to be 0, the approximated noise will be actual data noise $\lim_{bias \to 0} noise_{ML} = noise_{true}$. As a result, models (such as Decision Tree and Boosting) which are well-know for low bias are recommended for calculating $noise_{ML}$.

The question of whether more features may be useful can be approached from the distance standpoint. For instance, 2 original samples $x_1 = x_2$, while $y_1 \neq y_2$ have zero distance in between in terms of their features $d(x_1, x_2) = 0$. As long as the new features are capable of ensuring $d(x_1, x_2) \neq 0$, the noise will be reduced. To generalize, if the additional features can increase the distances among noise samples, it will benefit the predictions. This reasoning is based on the assumption of the optimal model, while a similar rule holds for other non-optimal models. For generic machine learning models, useful features should be able to be more distinguishable for different classes $d_{inter} = \frac{1}{|C|} \sum_{c \neq k} d(\mu_k, \mu_c)$, and maintain or reduce the distance among the samples of the same class $d_{intra} = \frac{1}{|x \in k|} \sum_{x \in k} d(x, \mu_k)$,

Table 5.14: The Effectiveness of Adding Host-based Features

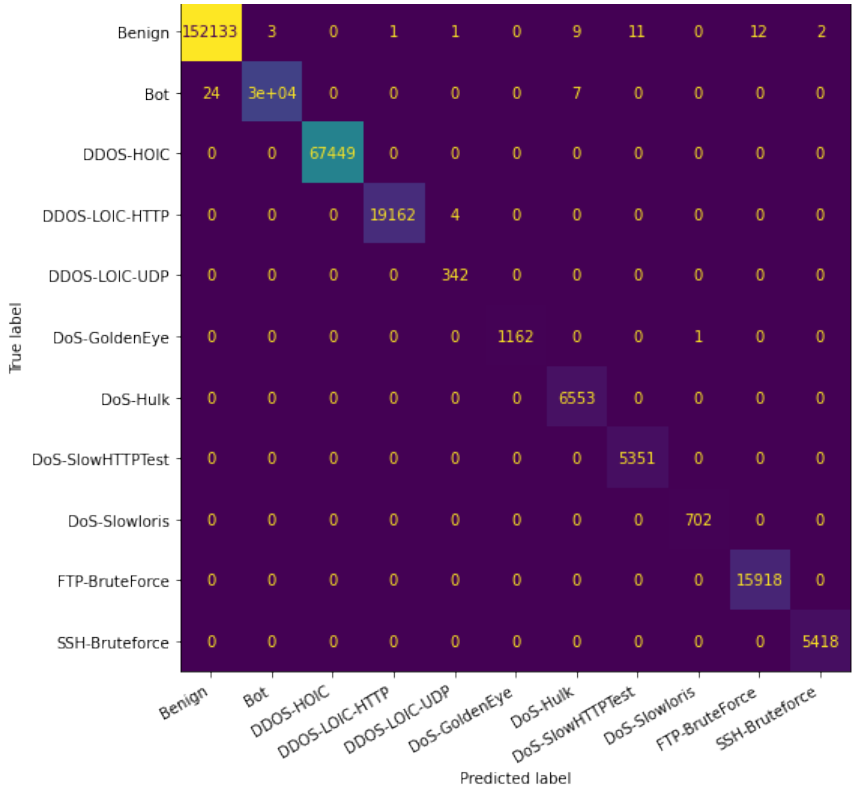
Features	Noise Sample Classes	$noise_{opt}$	$noise_{DT}$	Avg Class Centroids Distance	
				L1 Norm	L2 Norm
Network Features	(DoS-SlowHTTPTest, FTP-BruteForce): 21247	$1.42 * 10^{-2}$	$1.43 * 10^{-2}$	$8.2 * 10^{-2}$	$1.8 * 10^{-2}$
	(Benign, DDOS-LOIC-HTTP, DDOS-LOIC-UDP): 369				
Network+Host Features	(DDOS-LOIC-HTTP, DDOS-LOIC-UDP): 136	$1.32 * 10^{-5}$	$2.63 * 10^{-5}$	0.45	0.18
Network+Perturbation	-	-	$1.46 * 10^{-2}$	0.137	$2.40 * 10^{-2}$

where C is the set of classes, k is the true class of x, and μ_k is the centroid of class k.

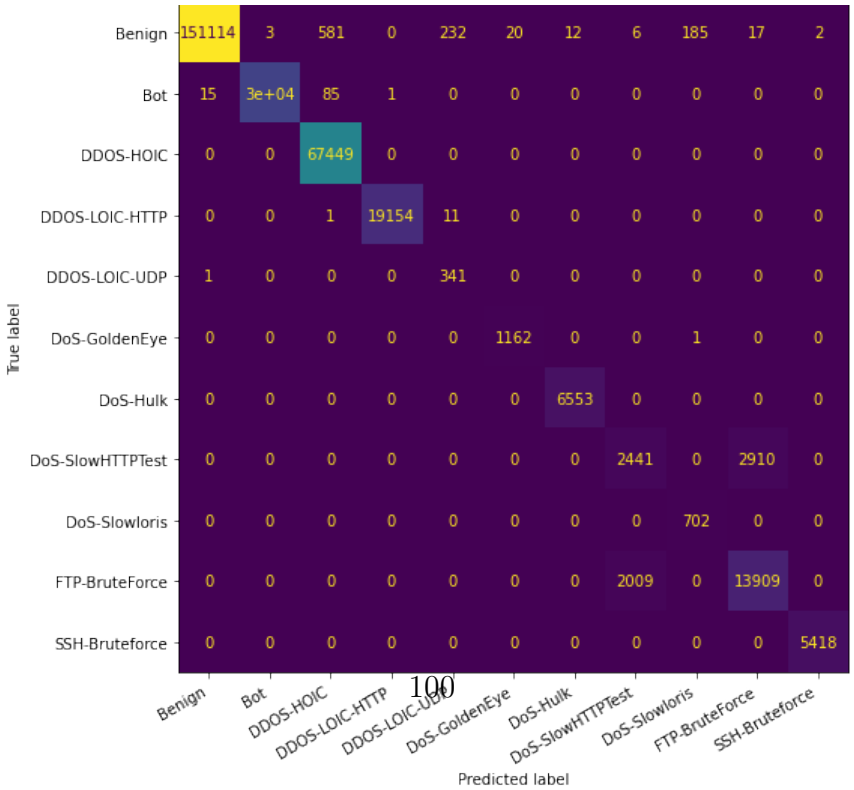
To support the hypothesis presented above, noise analysis under the SCVIC-CIDS-2021 dataset is listed in Table 5.14.

Since Decision Tree (DT) estimates the noise upper limit, $noise_{ML}$ is equivalent to $noise_{DT}$. Data noise varies between 0.0142 and 0.0143 when network-based features are used, indicating that more than 1% of samples contain noise. Within those noise sample, we further reveal which classes conflict with one another. "DoS-SlowHTTPTest" and "FTP-BruteForce" have the highest number of noise samples that no machine learning model can accurately predict, and these noise instances directly contribute to the errors between these two classes. The CIDS confusion matrix with only network based features is plotted in Fig. 5.7b where the major loss occurs in "DoS-SlowHTTPTest" and "FTP-BruteForce". The numbers following the noise sample classes indicate the number of samples with multiple labels in the training set; however, they do not mean that classifier predicts these samples incorrectly; for instance, a x_{te} may correspond to multiple classes in training set like $[C_1, C_1, C_2]$, if the y_{te} is C_1 which is the majority label of the training set, the x_{te} will be classified correctly even if it belongs to multiple classes in the training set; nonetheless the high number of samples with multiple labels in the training set indicates high possibility of making wrong predictions. After the host-based features are added into the data, the data noise is reduced significantly from $1.32 * 10^{-5}$ to $2.63 * 10^{-5}$, and the noise sample classes exist only in "DDOS-LOIC-HTTP" and "DDOS-LOIC-UDP" leading to 4 wrong predictions between these classes in Fig 5.7a, which depicts the CIDS confusion matrix with host-based features. When perturbation is added as an additional feature, the noise is around 0.01456. As previously explained, the lower limit is neglected since $noise_{opt}$ is insensitive to noise. Moreover, we further estimate the variance using DT as it is well-known for its low bias and high variance. Without host-based features, the variance is already at $2.11 * 10^{-4}$, and, with the host-based features, the variance is $4.28 * 10^{-5}$. Regardless of the impact of the DT, the datasets (with/without host-based features) do not raise any variance concern, which also explains why Random Forest and XGBoost do not improve the Decision Tree performance significantly.

The distance of average class centroids is also measured in SCVIC-CIDS-2021. As



(a) CIDS Confusion Matrix Using Host-based Feature



(b) CIDS Confusion Matrix without Using Host-based Feature

Figure 5.7: CIDS Confusion Matrix when Host-based features are given and not given

[11] points out, distance may lose its utility for higher dimensional data. Thus, L1 is a more useful distant measure than L2. To avoid inflating the dimensions, we use the dataset obtained from Fig. 5.4 which has less dimensions than the original. The inter-class distance (L2) is 0.018 when network-based features are solely used, but increases to 0.18 when host-based features are appended, which is 10 times larger than original distance, making classification easier. Furthermore, L1 distances are calculated, demonstrating the huge improvement as well. To further exclude the influence of additional dimensions, we substitute random noise for host-based features; the L1 and L2 distances are 0.137 and 0.024 respectively. Especially, when host-based features are included, the distance between the two classes that improved the most ('DoS-SlowHTTPPTest' and 'FTP-BruteForce') increased from 0.007 to 0.571.

5.3 Conclusion

In this chapter, first we have proposed a novel framework for constructing CIDS datasets that combine and align network flow and host-based data. Additionally, a transformer-based CIDS-Net has been proposed that utilizes the combined feature to classify network intrusions. Three datasets (SCVIC-CIDS-2021-Reduced, SCVIC-CIDS-2021, and SCVIC-CIDS-2022) have been constructed from well-known benchmark datasets (CIC-IDS-2018 and NDSec-1), and have been evaluated in this chapter utilizing network-based features as baselines.

For SCVIC-CIDS-2021-Reduced, the highest performing baseline models are XGBoost and Random Forest, which achieve 92.5% in terms of macro F1 score. We have examined several combinations of network encoders (i.e., Identity, FFN, and TabNet), aggregators (i.e., FFN, additive, and machine learning), and loss functions (i.e., cross entropy loss and C Loss) for the CIDS-Net . When FFN is utilized as the aggregator, the CIDS-Net performs optimally when TabNet is used as the network encoder, achieving a macro F1 score of 99.89%. If an additive aggregator is utilized as an aggregator, the FFN network encoder can achieve 99.75% macro F1 score. By substituting DT for FFN and additive aggregators, the results are enhanced and more stable, reaching 99.94% macro F1 score. The CIDS results demonstrate that host-based features can significantly improve IDS performance. For SCVIC-CIDS-2021, the best baseline model is RF, which achieves a macro average F1 score of 94.2%. When the identity layer is utilized as the network encoder and XGBoost is used as the aggregator, the CIDS-Net achieves the highest macro average F1 score of 99.95%. SCVIC-CIDS-2022 has been evaluated to demonstrate generalizability of the CIDS-Net. The CIDS-Net, which uses the identification layer as the network encoder and

DT as the network aggregator, performs the best, achieving a macro F1 score of 91.3%, surpassing the baseline of 5.1%. In comparison to our previous work, the proposed CIDS-Net handles imbalanced datasets better while leading to 38.4% improvement under the SCVIC-CIDS-2022 dataset.

To interpret the improvements, we have proposed a method for decomposing errors and approximating the upper and lower limits of data noise. The results prove that host-based features can significantly reduce the data noise of network-based features (from [1.42e-2, 1.43e-2] to [1.32e-5, 2.63e-5]) and increase the average distance among different classes. The results demonstrate that host-based features can lead to significant noise reduction in IDS datasets.

Chapter 6

Multidomain transformer-based deep learning for early detection of network intrusion

6.1 IoT Intrusion Early Detection System

This section discusses the detailed methodology for the early detection of IoT network intrusions. To begin, a new network feature extractor is presented to characterize a network flow in a time-series format in order to identify them prior to their complete arrival. Additionally, inspired by MDDNN, which demonstrates that frequency domain can improve the performance of deep learning models, we propose a Multi-Domain Transformer (MDT) for detecting network intrusions using the beginning packets of flows. MDT integrates Fourier transformation with a Transformer encoder that learns additional information from the frequency domain.

6.1.1 Problem Definition

Traditionally, a machine learning-based Network Intrusion Detection System (NIDS) detects network intrusions by collecting network packets to form flows, extracting features from the flows to generate a tabular dataset, feeding the flow-based tabular dataset to machine learning algorithms, and evaluating performance,while this chapter is concerned with detecting intrusions as early as possible. Two important issues should be addressed

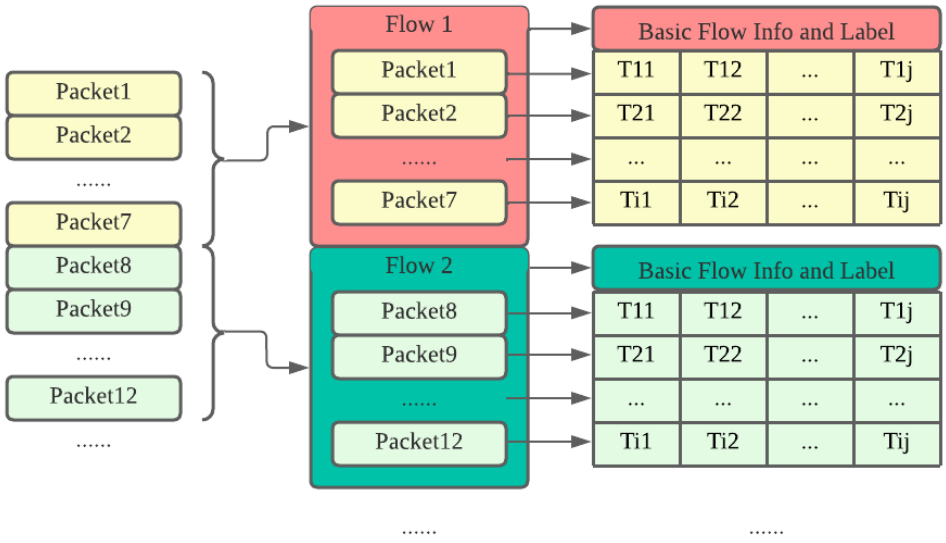


Figure 6.1: Time Series Representation of Network Flow Meter Features

to accomplish this goal: representing flows as time series and detecting intrusions using only the beginning subsequence of time series.

Conventional flow-based feature extractors can successfully obtain features from flows but they always have earliness limitations as features are observed using a completed time window. Hence, we propose extracting packet-based features from the network flows forming time-series datasets. Each flow f is defined as a set of packets $P = \{p_1, p_2, ..., p_L\}$ with the same session key: source IP, source port, destination IP, destination port, and time window. From each flow, TSNFM extracts a Multivariate Time Series (MTS) $\mathbb{R}^{L*d}$ which consists of features from packets, where L is the total length of MTS and d is the number of features extracted from a packet. To classify intrusions in early stages, only the Early Subsequence (ES) $\mathbb{R}^{l*d}$ is used for classification, where $l < L$.

Besides classification performance, another important metric of early detection is Earliness $E = \frac{l}{L}$. As shown in Table 1.1, some payloads or attacks may have less number of packets while being executed for longer duration than others, such as Reverse TCP and Reverse HTTP in Meterpreter. Therefore, besides earliness, a duration-based earliness is considered to prevent intrusions with low number of packets but long duration of launching period. Duration-based Earliness (DE) can be defined as $DE = \frac{Dur(ES)}{Dur(MTS)}$. Since E and DE should be considered together, we further propose Flow Earliness ($FE = (E, DE)$) as a comprehensive metric for network intrusion early detection.

Table 6.1: A General Description of Traditional NIDS Feature Extractors

Direction	Feature Type	Statistic Characteristics
Forward	# of Packets	Min
Backward	# of Packets Bytes	Max
Flow	Inter-Arriaval Time	Sum
	TCP Flags	Mean
	Active/Idle Time	Standard Deviation
		Speed

6.1.2 Time Series Network Flow Meter

A Conventional NIDS feature extractors (NIDSFE), such as CICFlowMeter [88] or NF-Stream [19], extracts features as a combination of the three aspects (i.e., direction, feature kinds) described in Table 6.1; for example, Forward + Packet Bytes + Min specifies the flow's minimum packet bytes. The primary constraint on classic NIDSFE's earliness is not only their process (i.e., extracting features after the flow is generated) but also the fact that those basic statistical characteristics require a certain amount of packets to adequately describe a flow. Thus, rather than manually summarizing flow properties using statistical characteristics, we propose extracting features directly from packets constituting an MTS and allowing machine learning models to learn to define and classify flows.

To extract MTS from a network flow, we propose a time series-based feature extractor named Time Series Network Flow Meter (TS-NFM). As shown in Fig. 6.1, packets with the same session key (i.e., source IP, source port, destination IP, destination port, timewindow) are grouped together to form a flow. Instead of extracting statistical features from a flow, TS-NFM extracts features from each packet inside a flow. The packet-based features include direction, Inter-Arrival Time (IAT), number of bytes, and TCP flags. In order to label network flows, the flow session information (basic info) is stored in the tabular format, while the features are stored in MTS format $\mathbb{R}^{L*d}$, where L is the number of packets inside a flow, and d is the number of features extracted from each packet.

6.1.3 Multi-Domain Transformer

This section offers the Multi-Domain Transformer (MDT), a deep learning model for early classification that incorporates the frequency domain into the original Transformer model.

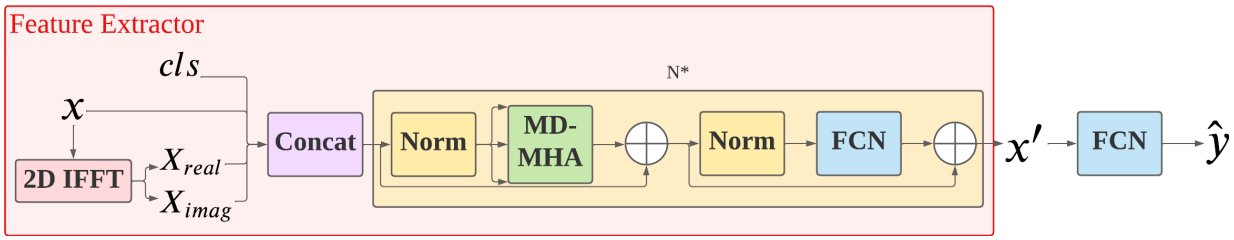


Figure 6.2: MDTransformer Architecture

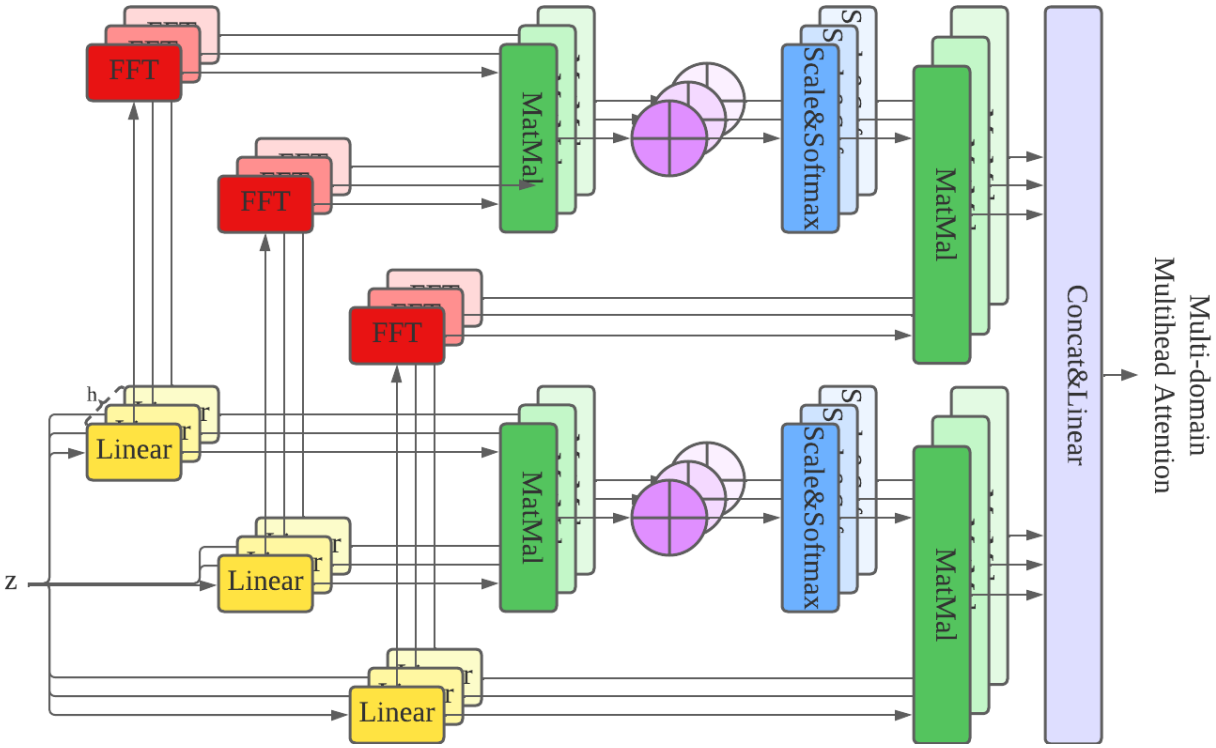


Figure 6.3: Multi-Domain Multi-Head Attention (MD-MHA)

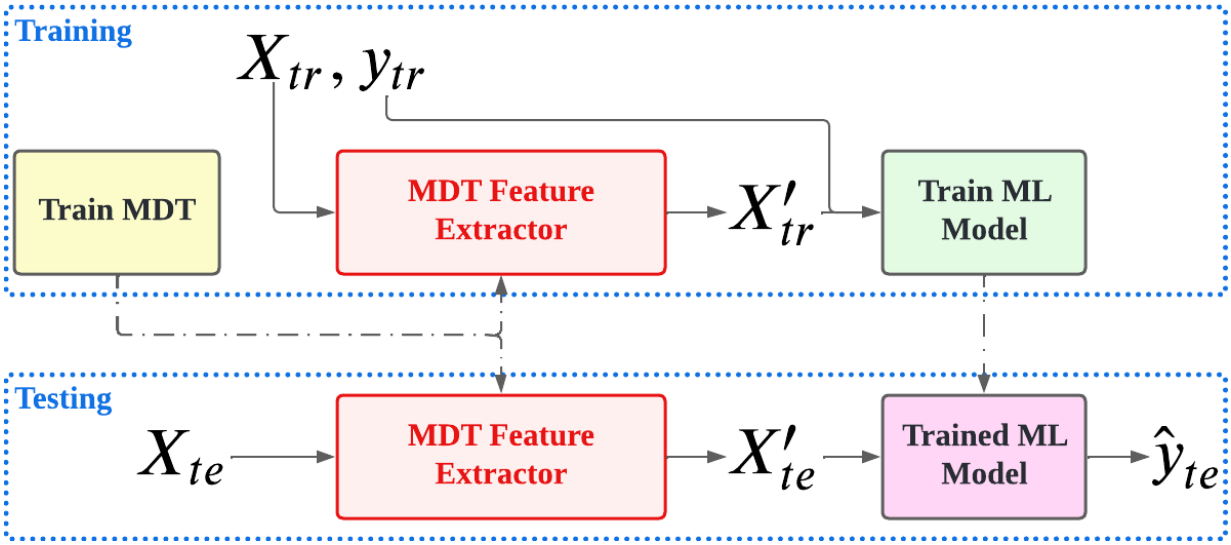


Figure 6.4: MDT Using ML Classifier

Each input of early classification problem contains limited information; hence, it is essential to extract useful features from MTS. The frequency domain can give extra information in addition to the time domain of time series. The architecture of Multi-Domain Transformer is depicted in Figure 6.2 (MDT). An MTS is initially be converted using two-dimensional IFFT. The outputs of the IFFT (i.e., the real and imaginary sections) are concatenated with MTS and fed into a Transformer encoder [55]. IFFT is required since, for MTS with a small number of dimensions/features d, the layer normalization of the Transformer encoder makes time series difficult to differentiate. For example, the ECG dataset contains only two dimensions. After layer normalization, all time series are be changed to [0, 0], [1, -1], and [-1, 1], which means that all amplitude information in MTS is lost. As a result, the dimensions can be increased using IFFT. MDDNN makes use of the one-dimensional IFFT; however, the imaginary part of the one-dimensional IFFT may be zero. Thus, to increase the number of dimensions even further, a 2D IFFT is used, with the real and imaginary components concatenated with the original input. Additionally, 2D IFFT may extract a greater amount of local information from MTS than 1D IFFT.

This work proposes a Multi-Domain Multi-Head Attention (MD-MHA) approach for further enhancing the Transformer encoder. Fig. 6.3 depicts the architecture of MD-MHA. Consider z as input of MD-MHA. Three linear layers (W_Q, W_K, W_V) are fed z to get Q, K, V. Q, K, V are further converted to frequency domain $Q' = FFT(Q)$, $K' = FFT(K)$,

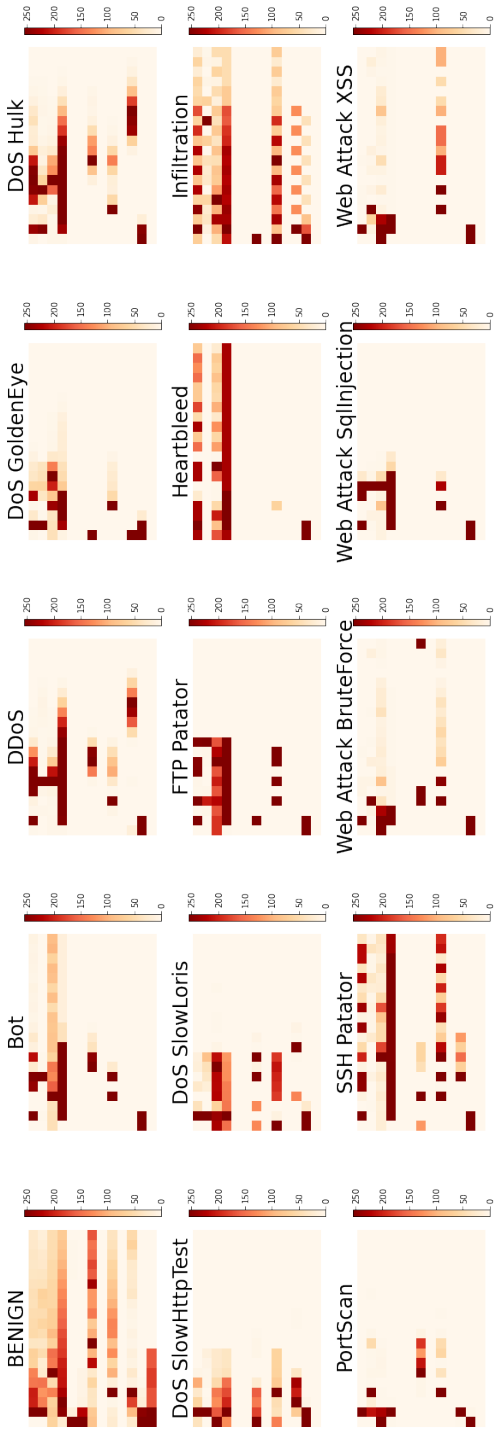


Figure 6.5: Visualization of the TS-CICIDS2017 Dataset

$V' = FFT(V)$. MD-MHA is formulated as Eq. 6.1.

$$MD - MHA(z) = [h_1.h_2, ..., h_h, h'_1, h'_2, ...h'_h]W_o \tag{6.1}$$

, where

$$h_i = softmax(\frac{QK^T}{\sqrt{d_{model}}})V,\ h'_i = softmax(\frac{Q'K'^T}{\sqrt{d_{model}}})V'$$

As FFT transforms the time domain to the frequency domain, it can explicitly provide MDT a variety of views. MD-MHA increases the number of heads without increasing the number of parameters, hence decreasing the possibility of overfitting.

Network intrusion detection datasets contains imbalanced classes; hence, machine learning models (such as XGBoost) perform better than deep learning based classifiers including MLP, and TabNet on tabular datasets [153]. Therefore, this work incorporates machine learning models into MDT. As Fig. 6.4 shows, MDT is firstly trained with FCN classifier. The inputs (MTS) are fed to MDT feature extractors to obtain latent states. The latent state is further classified by a ML model.

6.2 Experiments

This section provides an overview of datasets and their numerical results. The datasets utilized in this section include TS-CICIDS2017 created by TSNFM and two other widely-used MTS datasets (i.e., ECG and Wafer) in order to compare MDT to other approaches and demonstrate its generalizability.

6.2.1 Datasets

TS-CICIDS2017

To apply TSNFM, a network intrusion dataset must have original raw network packets rather than extracted features and complete labeling information; consequently, this chapter uses the CIC-IDS-2017 dataset. The raw network packets (PCAP format) are fed into the proposed TSNFM with a time window of two minutes (same to the extracted features from CIC-IDS-2017), resulting in the TS-CICIDS2017 dataset. MTS's maximum length L is 511681 due to the network's fast speed. The number of features/dimensions d is thirteen, which includes the direction of a packet, IAT, the size in bytes, and ten TCP flags.

To aid in comprehension of the MTS in TS-CICIDS2017, the MTS of various intrusions are visualized in Fig. 6.5. MTS for each class are averaged and scaled to [0, 255]. Due to space constraints, the first twenty timestamps are shown. Each subplot has an x-axis for timestamps and a y-axis for features. By visual comparison, MTS of various intrusion types exhibit distinct patterns, establishing a basis for classification. Additionally, the remarkable features are focused on the left side of the picture, indicating that the critical information is concentrated at the start of the sequence, laying the foundations for early detection of network intrusions.

ECG

In order to show the generalizability of the proposed method, we apply the MDT to two more datasets that have been extensively studied in early detection research. The ECG database is made of multivariate time series, each of which represents the sequence of measurements taken by a single electrode during a single heartbeat. Each heartbeat is classified as normal or pathological. All aberrant heartbeats are indicative of a heart condition called a supraventricular premature beat. The ECG dataset contains 200 samples, 133 of which are normal and 67 of which are abnormal. ECG MTS consist of two dimensions/features, with a maximum length of 152.

Wafer

The wafer dataset is a collection of MTS including the sequence of measurements taken by a single vacuum-chamber sensor during the etch process used to create semiconductor microelectronics. Each wafer is classified as normal or abnormal. The irregular wafers are indicative of a variety of issues that frequently occur during semiconductor fabrication. There are 1194 instances in the Wafer dataset, 1067 of which are normal and 127 of which are abnormal. Wafer MTS have a maximum length of 198 and six dimensions/features.

6.2.2 Experimental Results

This section discusses the detailed results of MDT on the TS-CICIDS2017 dataset; additionally, to demonstrate MDT's generalizability, this research performs MDT tests on the EEG and Wafer datasets and compares them to other state-of-the-art works. To compare with others' work, accuracy, detection rate, and macro (non-weighted) F1 score are used.

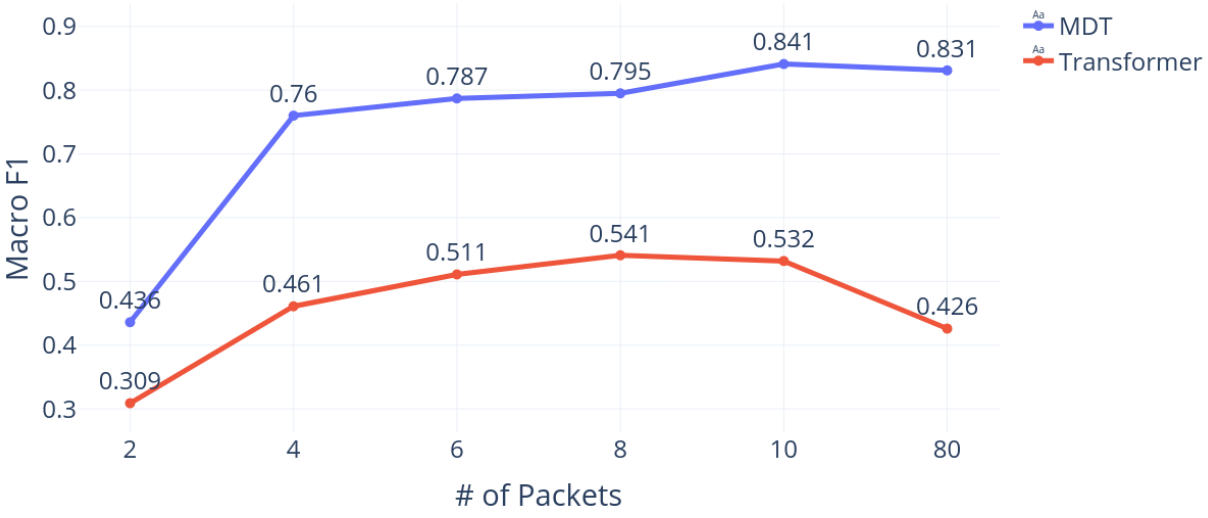


Figure 6.6: F-score Under Varying Number of Packets

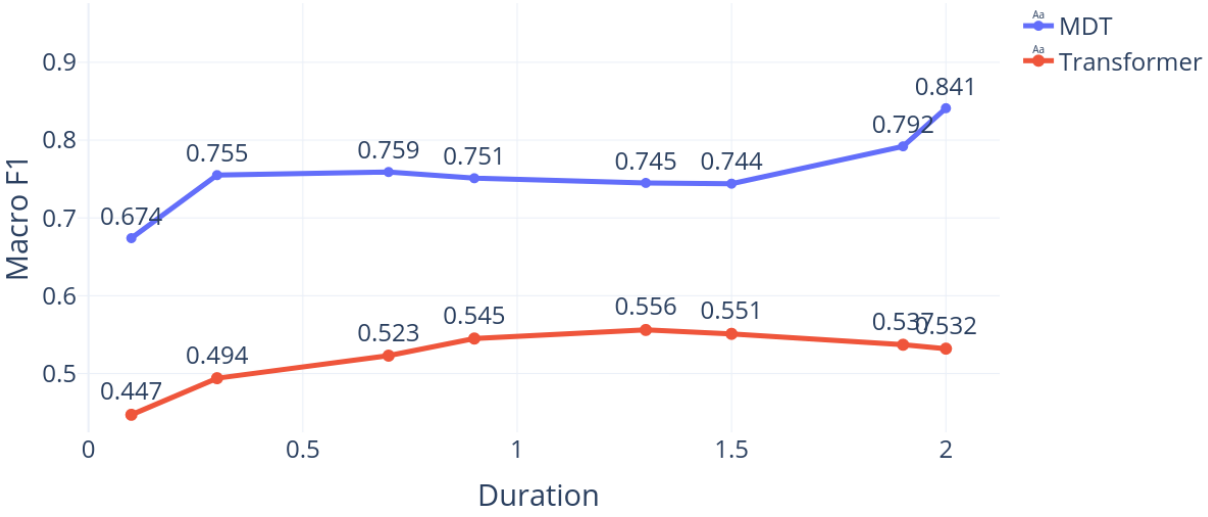


Figure 6.7: F-score Under Varying Durations

Table 6.2: Comparison of Detection Performance, Earliness (E) and Duration-based Earliness (DE) among Proposed Method and Related Studies; ACC: Accuracy; DR: Detection Rate

Method	Feature Type	E	DE	ACC (%)	DR (%)
GA-based Adaptive Method [189]	Tabular	1	1	-	92.85
Ensemble SVM [158]	Tabular	1	1	-	94.94
IFSE-AD [230]	Tabular	1	1	97.3	-
SU-IDS [164]	Tabular	1	1	71.02	-
FC-Net [238]	Image-like	1	1	94.64	99.62
MDT (Ours)	**MTS**	**2e-5**	**1.6e-2**	**99.7**	**99.7**

To better understand the influence of earliness and the efficacy of FFT, this study conducts an experiment examining the relationship between performance and earliness/duration-based earliness. Given that the L of TS-CICIDS2017 is 511681, the earliness is exceedingly small, ranging from 4e-6 to 2.3e-4; thus, for better depiction, the number of packets is used as the x-axis rather than the earliness. As illustrated in Fig. 6.6, the F-score is less than 70% when the first two packets are employed. The F-score increases as the number of packets increases. When the number of packets exceeds 10, the F-score can exceed 80%. Additionally, the MDT performance is compared with Transformer. Transformer performs 12% to 40.8% worse than MDT in terms of macro F-score because it is unable to extract usable features from a small number of packets, FCN struggles to handle unbalanced classes, and oversampling produces significant overfitting. MDT performs significantly better than Transformer as the number of packets increases. Additionally, Fig. 6.7 illustrates the change in F-score associated with duration-based earliness. When the first 0.1s of packets are used, MDT can get a 67.4% F-score. MDT's performance is enhanced when a longer duration is employed. Both Fig. 6.6 and Fig. 6.7 demonstrate that MDT can consistently outperform Transformer without FFT. At 0.016 duration-based earliness, the maximum improvement of 30.9% is achieved. As duration-based earliness impacts classification results, it is important to apply it as another evaluation metric. During the experiments, we also notice that FFT can accelerate the convergence of MDT.

Although the focus of this chapter differs from existing efforts, TS-CICIDS2017 uses the same network packets as the original CICIDS2017 dataset and has the same flow timeout (2 minutes). Thus, their samples/flows are comparable. Table 6.2 compares MDT to other studies in terms of earliness, duration-based earliness, and performance (i.e., accuracy and detection rate). Tabular features denote the original CICIDS2017's statistical features in CSV format. The authors' use of network packet headers as inputs results in image-like

Table 6.3: Earliness and F-score for each method and dataset; E: earliness; DE: Duration-based Earliness

Method \ Datasets	TS-CICIDS2017			ECG		Wafer	
	E	DE	F-score	E	F-score	E	F-score
MSD [82]	-	-	-	0.08	0.59	-	-
REACT [148]	-	-	-	**0.06**	0.77	**0.23**	0.92
1NN-full [58]	-	-	-	1	0.79	1	0.87
MDDNN [107]	-	-	-	**0.06**	0.81	**0.23**	0.91
ETSCM [103]	-	-	-	0.13	0.89	0.5	0.93
MDT (Ours)	**2e-5**	**0.016**	**0.84**	**0.06**	0.84	**0.23**	**0.98**
MDT (Best F-score)	**2e-5**	**0.016**	**0.84**	0.4	**0.94**	**0.23**	**0.98**
Transformer	2e-5	0.016	0.53	**0.06**	0.59	**0.23**	0.94

features. In comparison to other approaches, our proposed method increases earliness by 50000 times and duration-based earliness by 60 times; additionally, our method improves the baseline performance. MDT achieves a greater accuracy of 99.7% than other baselines and a (weighted-average) detection rate of 99.7%. Since TS-CICIDS2017 does not provide any additional feature types compared with CICIDS2017 (as listed in Table 6.1), the results demonstrate that MDT is capable of extracting more valuable information using a very short duration and a limited amount of network packets.

To demonstrate the proposed MDT's generalizability, it is evaluated on the ECG and Wafer datasets, which are extensively used in research of MTS early detection. Table 6.3 compares our proposed MDT to five other state-of-the-art approaches in terms of earliness and F-score. ETSCM is likewise inspired by the MDDNN; however, rather than using the earliness of earlier work, ETSCM chooses earliness based on its best F-score. Thus, it is compared to MDT's best F-score. REACT, and MDDNN have the best earliness for ECG datasets. MDT achieves an 84% F-score, which is 3% higher than MDDNN when the same earliness (0.06) is used. In comparison to ETSCM, it achieves the greatest F-score of 89% at 0.13 earliness, while MDT obtains the highest F-score of 94% (5% better than ETSCM) at 0.4 earliness. Due to the fact that the ECG dataset contains only two features, it is difficult for Transformer to analyze, providing 59% F-score; as a result, MDT improves Transformer by 25% on the ECG dataset. Previous work achieved 92% F-scores within 0.23 earliness for the Wafer dataset, whereas the proposed MDT achieves 98% F-scores under the same earliness, an improvement of 6%. Even when compared to the best

performance of ETSCM, we are able to get a 5% increase in F-score with better earliness. In comparison to Transformer, the results indicate that MDT with FFT can improve the Transformer's F-score by 4% on the Wafer dataset.

6.3 Conclusion

NIDSs have been extensively investigated but traditional approaches take a relatively long time to form flows and accumulate distinguishable features (e.g., 2 minutes for CICIDS2017). With this in mind, this work introduces early detection into the NIDS field, allowing for the detection of malicious traffic before it reaches target systems in its whole. This chapter introduces a novel feature extractor, TSNFM, that describes network flow as MTS with explainable features, and generates the TS-CICIDS2017 dataset; additionally, a new early detection model, MDT, is proposed. In comparison to the conventional methodology, our proposed method improves earliness by 50000 times and duration-based earliness by 60 times. This study reveals that when the first ten packets in 2s are used for detection, the proposed method is capable of achieving satisfactory detection performance (higher than 84% F-score). To demonstrate MDT's generalizability further, two additional datasets are analyzed. MDT is capable of increasing the F-score of state-of-the-art algorithms by 5% and 6%, respectively, on the ECG and Wafer datasets. Our ongoing study involves evaluation of the results on multiple network intrusion datasets and empowering unsupervised learning to extract abundant features from different number of packets and with various duration.

Chapter 7

Deep Dict: Deep Learning-based Lossy Time Series Compression

The preceding chapter described the benefits of representing NIDS flows as time series; nevertheless, time series might need a substantial amount of storage and bandwidth. This chapter presents a lossy time series compressor, called Deep Dict, to conserve storage space and network bandwidth. Section 7.1 describes the architecture of the proposed Deep Dict. The 7.2 illustrates the numerical output of several datasets. We employ CICIDS2017 as the NIDS dataset and 10 other datasets from diverse fields to validate the proposed Deep Dict. Section 7.3 concludes the chapter and highlights the important results.

7.1 Methodology

This section lays out the technical details of the proposed Deep Dict framework for lossy time series compression. Section 7.1.1 sets the scope of the problem and presents the evaluation metrics. In Section 7.1.2, the design of Deep Dict, network architecture, and the novel loss function are described in detail. Table 7.1, presents the notation (symbols and their descriptions) used in the presentation of the proposed lossy compression technique in the rest of this section.

7.1.1 Problem Definition

Time series are described as a collection of time-dependent data that can be categorized broadly into Univariate Time Series (UTS) and Multivariate Time Series (MTS). Univariate Time Series $UTS \in \mathbb{R}^l$ consist of a single variable that varies over time, whereas Multivariate Time Series $MTS \in \mathbb{R}^{l \times d}$ contain multiple variables that are not only related to timestamp but also dependent on one another, where l and d represent the time series length and the number of variables, respectively. AutoEncoder (AE)-based compressors encode time series into a latent representation consisting of floating-point numbers; thus, the size of the latent representation has a direct effect on the compression ratio. This study proposes Deep Dict, a new compressor which compresses time series (UTS/MTS) into Bernoulli distributed latent states, significantly reducing the size of latent states, and limits the distortion of decompressed/reconstructed time series. For evaluating lossy time series compression, two key metrics are used: data distortion and compression ratio. *Data distortion* measures the error between the original and reconstructed time series, including the mean square error, the mean absolute percentage error, and the maximum absolute error. Maximum Absolute Error (MaAE) is a widely accepted measure of distortion in the absence of downstream tasks or domain knowledge [41]. *Compression ratio* is defined as the ratio of original data size to compressed data size.

7.1.2 Deep Dict

Overview

The two major components of Deep Dict are the Bernoulli Transformer AutoEncoder (BTAE) and distortion constraint. Figure 7.1 illustrates an overview of the proposed Deep Dict compressor. Initially, Deep Dict chunks the original long time series into smaller time series (x) by a time window. BTAE encodes x into Bernoulli latent states (c) and decodes c to predict time series ($\hat{x}$). In order to limit the error of the reconstructed time series to a desired range, the residual ($r = x - \hat{x}$) is quantized uniformly to r_q and an entropy coder is used to compress $r_{encoded}$ in a lossless manner. For transmission or storage after compression, c, the decoder, and $r_{encoded}$ are used. During decompression, c is fed into the decoder to recover $\hat{x}$, and the entropy coder decodes $r_{encoded}$ to r_q.

The time series reconstruction is formulated as $x_{recon} = \hat{x} + r_q$. Thus, the quantization error determines the distortion of the reconstructed time series. Since c functions as indicated during decompression whereas BTAE's decoder serves as a dictionary. Therefore, we name our framework Deep Dict.

Table 7.1: Notation Table

Notation	Description
ϵ	User specified Maximum Absolute Error
c	The Bernoulli distributed latent states
C	The coefficient matrix used for generating synthetic dataset
D	A distance matrix where $d_{\cdot,j} = r_{\cdot} - s_j$
$f_{binarization}$	Binarization funtion used for transforming real-value latent state to Bernoulli latent states
g_ϵ, g'_ϵ	A rectangular function and its approximation
H	The objective function which calculates r's quantized entropy
l, d	The length and the number of variables of time series
$n(s_j)$	The number of s_j occurs in r_q
P	A differentiable function used to replace p
$p(s_j)$	The possibility of s_j appears in r_q
r	The residual between original time series and the predicted time series
$r_{encoded}$	The encoded r_q compressed by entropy coder
r_q	The uniformly quantized r
S	A collection of unique values of r_q
t, t_p	Timestamp, and polynomial timestamp
x	Uni/multi-variate time series
$\hat{x}$	The time series predicted by BTAE model
x_{recon}	The reconstructed time series
y	Real-value latent states generated by the BTAE's encoder

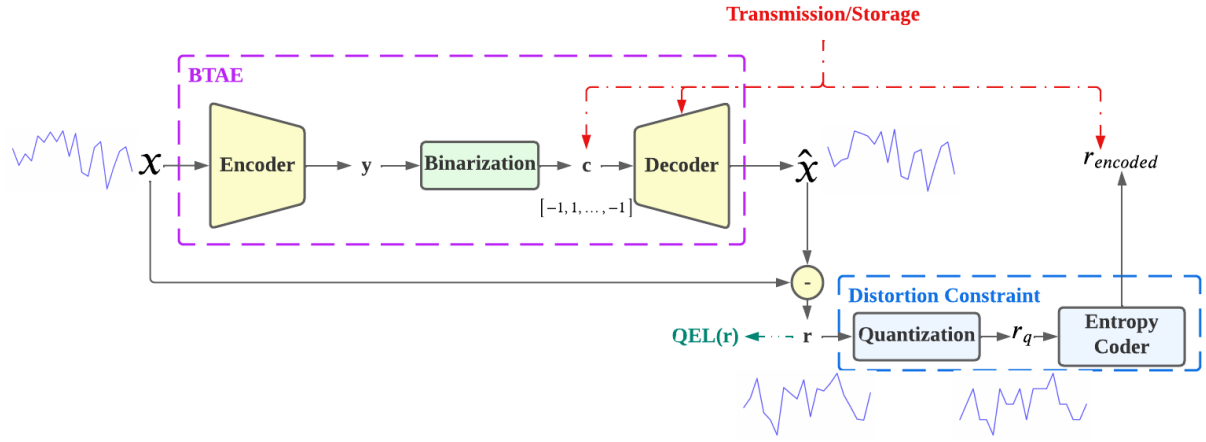


Figure 7.1: Overview of Deep Dict

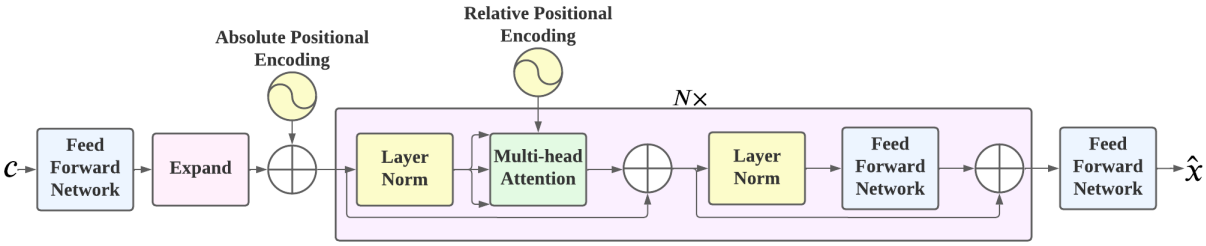


Figure 7.2: Detailed Architecture of the Decoder of Deep Dict

BTAE

BTAE seeks to discover the Bernoulli latent states/representations of a time series and to recover the time series from the representations. The encoder takes the flattened time series x as input and transforms it into real-value latent states y, which are then binarized to Bernoulli latent states c as shown in Eq. 7.1 where $tanh$ approximation is used to make the binarization function differentiable.

$$f_{binarization}(y_i) = \begin{cases} 1 & \text{if } y_i \geq 0 \\ -1 & \text{else} \end{cases} \tag{7.1}$$

Consequently, the derivative of the binarization function is computed as in Eq. 7.2

$$\frac{df_{binarization}}{dy_i} = 1 - tanh^2(y_i) \tag{7.2}$$

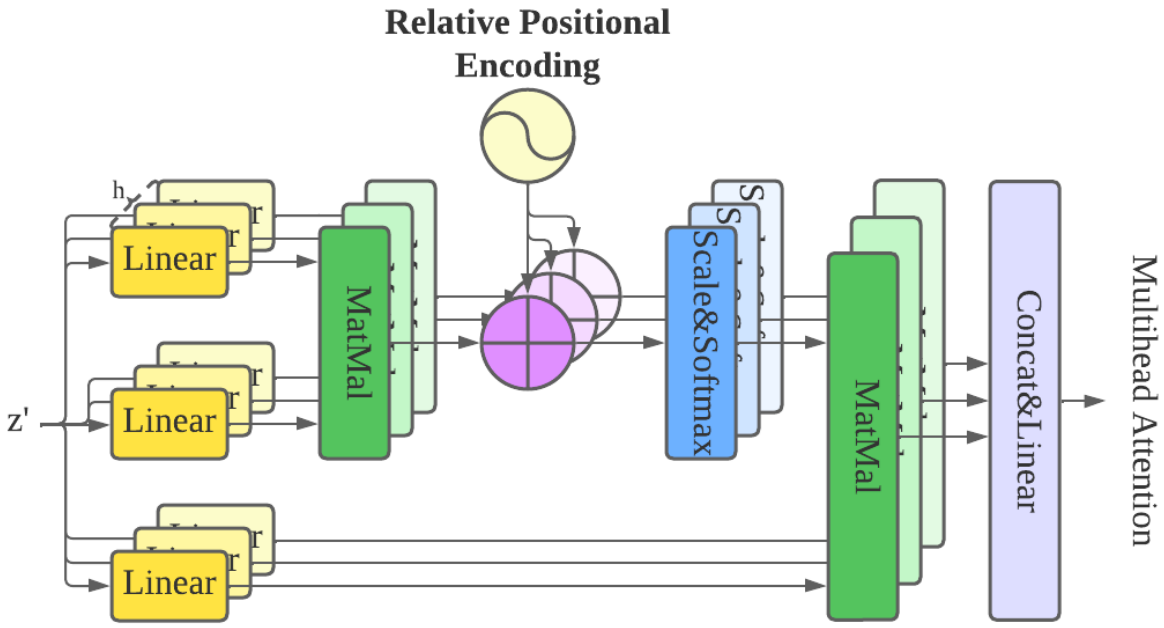


Figure 7.3: Detailed Architecture of Multihead Attention with Relative Positional Encoding

A transformer-based decoder is then fed the Bernoulli latent states (c) to predict the corresponding time series. Since c contains limited information, a feed-forward network (FFN) is used as the encoder in this study. Figure 7.2 demonstrates the architecture of the decoder in detail. A FFN augments $c \in \{-1, 1\}^{|c|}$ to $c' \in \mathbb{R}^{d_{model}}$ which is then replicated to $c'' \in \mathbb{R}^{l \times d_{model}}$, where l is the length of time series, and d_{model} is considered as one of important hyperparameters of Deep Dict. To avoid additional overhead of parameters in absolute positional encoding ($APE \in \mathbb{R}^{l \times d_{model}}$), sine and cosine functions are used as seen in Eqs. 7.3-7.4, and $z = c'' + APE$ is fed into the Transformer [225] encoder blocks followed by a FFN to obtain the prediction of time series.

$$APE(pos, 2i) = sin(pos/10,000^{2i/d_{model}}) \tag{7.3}$$

$$APE(pos, 2i+1) = cos(pos/10,000^{2i/d_{model}}) \tag{7.4}$$

Due to the fact that each input time series x is truncated from a long sequence, relative position can be more meaningful than the absolute position. Therefore, we include relative positional encoding in the proposed decoder ($RPE \in \mathbb{R}^{l \times l}$). Layer normalization transforms z into z'. Figure 7.3 illustrates the details of multihead attention (MHA) with RPE. Three linear layers (W_q, W_k, and W_v) are fed z' to get Q, K, and V. Dot production is employed to obtain attentions. The output of MHA can be formalized as Eq. 7.5.

$$MHA(z') = [head_1, head_2, ...head_h]W_o \tag{7.5}$$

where

$$head_i = softmax(\frac{QK^T + RPE}{\sqrt{d_{model}}})V$$

RPE is introduced to the attention matrix to allow the decoder to learn relative position-related information. Skewing technique (introduced by Huang et al. [106]) is used by RPE to reduce the dependency on extra parameters. The RPE can be formulated as $RPE_{i,j} = j - i$ and an example is demonstrated in Eq. 7.6.

$$\begin{bmatrix} 0 & 1 & ... & l-1 \\ -1 & 0 & ... & l-2 \\ \vdots & \vdots & ... & \vdots \\ -l+1 & -l+2 & ... & 0 \end{bmatrix} \tag{7.6}$$

BTAE employs 32-bit float for the training phase, and 16-bit float for validation and testing to decrease the size of the decoder. Time series contain recurring or similar patterns;

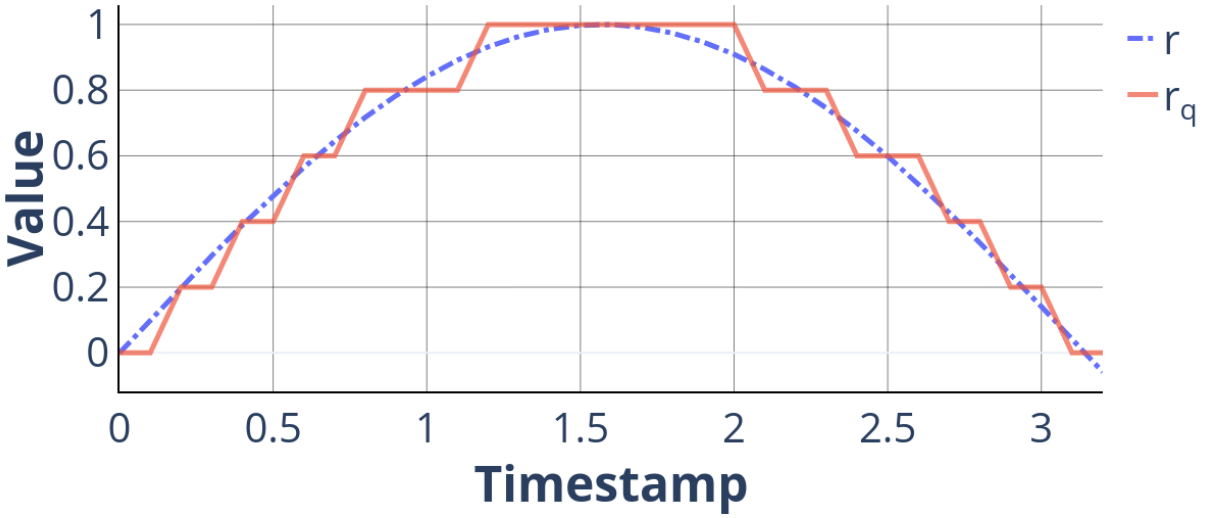


Figure 7.4: Intuitive Example of Uniformed Quantization

hence, not all Bernoulli hidden states are unique. Encoding Bernoulli latent states with an entropy encoder can further improve the compression ratio.

In comparison to the traditional AE, BTAE reduces the size of the latent state by a factor of the bitcount of a floating point $|FP|$. For example, $|FP| = 32$ bits by default in Pytorch and Tensorflow. With its non-auto-regressive architecture, the proposed method leads to faster training and performs better on long sequences than RNN models such as LSTM and GRU in terms of compression ratio.

It consumes long time and computational resources to train a new model from scratch each time a new time series is compressed; therefore, transfer learning can be used to accelerate the compression process. For univariate time series, a trained model can be directly applied to another, whereas for a multivariate time series, the encoder and the last FFN of the decoder have to be retrained from scratch due to the varying number of multivariate variables.

Distortion Constraint

The predicted time series $\hat{x}$ typically has more than 10% Mean Absolute Percentage Error (MAPE) loss. By utilizing BTAE, distortion can be constrained to a small range at the

expense of more parameters. Thus, with limited number of parameters in BATE, the distortion constraint is utilised to reduce the distortion to a desired range. As depicted in Figure 7.4, r is quantized uniformly to r_q as follows: $r_q = 2\epsilon \times round(r/2\epsilon)$, where ϵ is the desired MaAE. To avoid float-point overflow, r_q is stored in 64-bit format. Adaptive Quantized Local Frequency Coding (powered by libbsc, a well-known library for lossless compression [85]) is used as the entropy coder to encode r_q as $r_{encoded}$ in this work.

Quanized Entropy Loss (QEL)

In general, many ML-based compression schemes contain quantization and entropy coder and apply L1/L2 as loss function. Nevertheless, because of the entropy coder, the size of $r_{encoded}$ is constrained to the total entropy of r_q. Since common regression losses such as L1/L2 do not consider quantization nor minimizing the entropy of r_q, they cannot describe the problem precisely.

This chapter introduces a new loss function, Quantized Entropy Loss (QEL), so as to minimize the size of $r_{encoded}$. In addition, QEL is not specific to Deep Dict; rather, it is a loss function applicable to all compression methods that employ uniform quantization followed by an entropy coder.

To formulate QEL, we first clarify the necessary symbols. Given ϵ as the desired MaAE, $r_q = 2\epsilon \times round(r/2\epsilon)$. $S = \{s_1, s_2, ..., s_{|S|}\}$ is the set of unique values of r_q whereas $n(s_j)$ is a counter to keep track of the number of times s_j appears in r_q, and $p(s_j)$ specifies the probability of s_j appearing in r_q, where $|\cdot|$ counts the number of items in $\cdot$. A formal expression of $p(s_j)$ can be given as $\frac{n(s_j)}{|r_q|}$.

As stated in Eq. 7.7, the size of $r_{encoded}$ is always greater than the total entropy of r_q due to the nature of the entropy coder. Good-performing entropy coders with high compression ratio tend to approach the limits.

$$|r_{encoded}| \geq -\sum_{j=0}^{|S|} n(s_j) \log p(s_j) \tag{7.7}$$

In light of these, the objective function can be formulated as in Eq. 7.8.

$$\min_{r} H(r) = -\sum_{j=0}^{|S|} p(s_j) \log p(s_j) \tag{7.8}$$

However, since neither the probability (p) nor the counter (n) is differentiable, we define P to replace p as shown in Eq. 7.9.

$$P(x) = \frac{1}{|x|} \sum_{k=0}^{|x|} g_\epsilon(x_k) \tag{7.9}$$

, where

$$g_\epsilon(x) = \begin{cases} 1 & \text{if } -\epsilon \leq x < \epsilon \\ 0 & \text{else} \end{cases} \tag{7.10}$$

To make g_ϵ differentiable, g_ϵ is approximated as g'_ϵ in Eq. 7.11 where $g_\epsilon = \lim_{b\to\infty} g'_\epsilon$..

$$g_\epsilon \approx g'_\epsilon = \frac{1}{(\frac{x}{\epsilon})^b + 1} \tag{7.11}$$

Under the circumstances, the objective function can be reformulated as in Eq. 7.12.

$$H(r) \approx -\sum_{j=0}^{|S|} P(r - s_j) \log P(r - s_j) \tag{7.12}$$

With these in mind, once the first order derivative of H is taken, it will appear as in Eq. 7.13 , where $d_{ij} = r_i - s_j$.

$$\begin{aligned} \frac{\partial H}{\partial r_i} &\approx \frac{\partial H}{\partial P} \times \frac{\partial P}{\partial g'_\epsilon} \times \frac{\partial g'_\epsilon}{\partial r_i} \\ &\approx -\sum_{j=0}^{|S|} [1 + \ln P(r - s_j)] \times \frac{1}{|r|} \times \frac{-\frac{b}{\epsilon^b}(r_i - s_j)^{b-1}}{[\frac{1}{\epsilon^b}(r_i - s_j)^b + 1]^2} \\ &\approx \frac{b}{|r|\epsilon^b} \sum_{j=0}^{|S|} [1 + \ln P(r - s_j)] \frac{(r_i - s_j)^{b-1}}{[\frac{1}{\epsilon^b}(r_i - s_j)^b + 1]^2} \\ &\approx \frac{b}{|r|\epsilon^b} \sum_{j=0}^{|S|} [1 + \ln p(s_j)] \frac{d_{ij}^{b-1}}{(\frac{1}{\epsilon^b} d_{ij}^{b-1} d_{ij} + 1)^2} \end{aligned} \tag{7.13}$$

Furthermore, QEL can be easily generalized to multidimensional matrix as presented in Eq. 7.14 where $\cdot$ is the index of r's elements, and $d_{\cdot j} = r_\cdot - s_j$.

$$\frac{\partial H}{\partial r_\cdot} \approx \frac{b}{|r|\epsilon^b} \sum_{j=0}^{|S|} [1 + \ln p(s_j)] \frac{d_{\cdot j}^{b-1}}{(\frac{1}{\epsilon^b} d_{\cdot j}^{b-1} d_{\cdot j} + 1)^2} \tag{7.14}$$

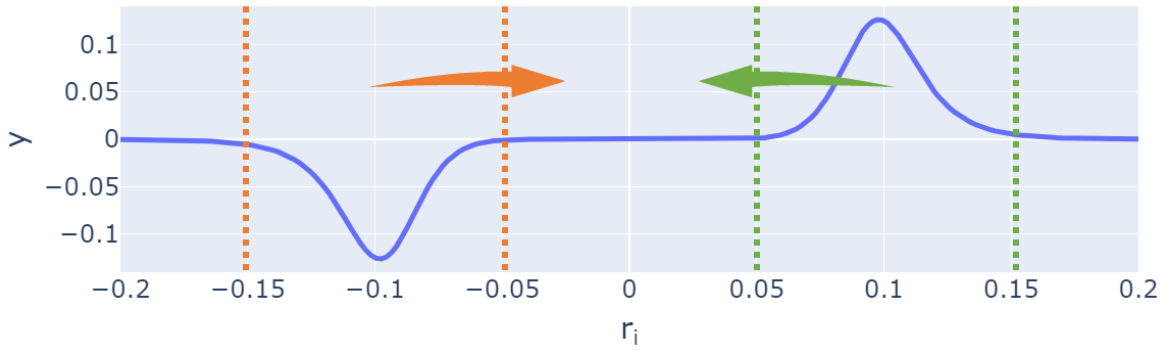


Figure 7.5: The example function of $R(r_i - s_j)$, where $s_j = 0$, $|x| = 200$, $b = 10$ and $\epsilon = 0.1$

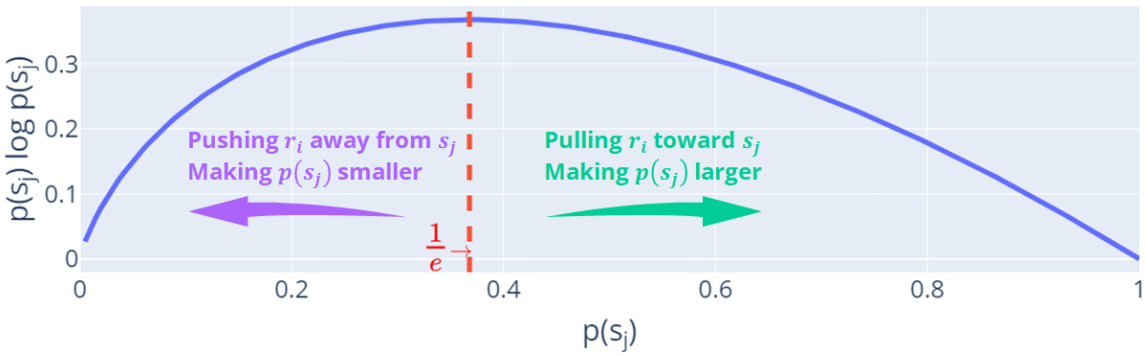


Figure 7.6: Illustration of QEL behavior

To accelerate the process, the forward procedure leverages the objective function formulation in Eq. 7.8, utilizes the occurrences of unique values of r_q, and saves S and $p(s_j)$ for backward procedure, while the backward process constructs the distance matrix $D = [d_{\cdot j}]_{\cdot \times j}$ and uses Eq. 7.14 to calculate derivatives.

Interpretation of QEL

The forward process of QEL calculates the entropy of time series. In order to explain the behavior of QEL during the backpropagation, we recall the gradient of H as Eq. 7.15.

$$\frac{\partial H}{\partial r_i} = \lim_{b\to\infty} \sum_{j=0}^{|S|} [1 + \ln p(s_j)] \times R(r_i - s_j) \tag{7.15}$$

, where

$$R(r_i - s_j) = \frac{b}{|r|\epsilon^b} \frac{(r_i - s_j)^{b-1}}{[\frac{(r_i - s_j)^b}{\epsilon^b} + 1]^2} \tag{7.16}$$

. Figure 7.5 illustrates a special case of $R(r_i - s_j)$, where $s_j = 0$. The effect of $R(r_i - 0)$ in QEL tends to move any r_i that is around ± 0.1 ($\pm\epsilon$) towards to 0 (s_j), and $R(r_i - 0)$ has no effects on any r_i that is close to or far from 0 (s_j). To be generalized, $R(r_i - s_j)$ pulls any r_i around $s_j \pm \epsilon$ to s_j and $-R(r_i - s_j)$ pushes any r_i around $s_j \pm \epsilon$ away from s_j. With this in mind, (and because of the property of $1 + \ln p(s_j)$), QEL has following key behaviors:

- If $p(s_j) < \frac{1}{e}$, QEL pushes r_j that is close to s_j away from s_j, making $p(s_j)$ samller;
- If $p(s_j) > \frac{1}{e}$, QEL pulls r_j that is close to s_j toward to s_j, with the bigger $p(s_j)$ possessing the greater pulling force, making $p(s_j)$ even larger;
- Adjacent s compete with each other, and the one with higher $p(s)$ pulls more r_i towards to it;
- For any r_i that is far from s_j, QEL ignores (does not move) it and consider it as outliers;
- For any r_i that is very close to s_j, QEL ignores it as well, as it would not affect the entropy of r_q after quantization.

Figure 7.6 illustrates the behaviors of QEL. For each s_j, both pulling or pushing r_i can reduce the entropy of r_q.

Due to the imperfection of sensors, randomness of human activities and network jitters, vast majority of time series, particularly real-world time series, are noisy [111]. Forcing a deep/machine learning model to remember these noise will need a significantly high number of parameters. Other regression losses, such as L1 or L2, have a tendency to drive all r_i

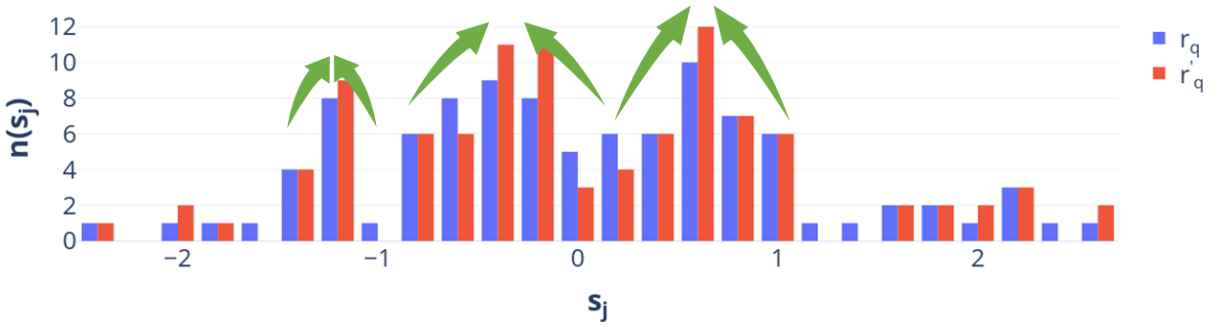


Figure 7.7: An example of QEL effect

to zero, holding a strong assumption: after optimization, the unique value of r_q should only be zero. However, owing to the abundance of noise, this objective is unattainable and models would waste efforts (adjusting weights) to achieve this objective without directly reducing the entropy. In addition, while L1/L2 can assist a model in learning of patterns and reduce the entropy of the learnable portion of data, they cannot guarantee to reduce the entropy of the non-learnable noise.

Compared with other regression losses, the advantages of QEL are listed as follows:

- QEL directly minimizes the entropy of r_q and recognize the presence of noise allowing models to make mistakes.
- Instead of only pulling all r towards 0, QEL pulls r towards multiple s providing multiple path to minimize entropy.
- For each iteration, QEL minimizes entropy via moving r_i to the most adjacent s_j (where $p(s_j) > \frac{1}{e}$), significantly reducing the runtime complexity of the optimization
- QEL does not request r_i to be exact s_j further reducing optimization difficulty.
- As shown in Figure 7.5, each s_j cannot affect remote r_i. Consequently, QEL is remarkably robust to outliers.

In addition to the mathematical explanations, we further prepare an example to understand the behavior of QEL intuitively. Consider r as noise that cannot be fitted by deep/machine learning models. The noise, $r \in \mathbb{R}^{100}$ is randomly generated, and $r_q =$

Table 7.2: Datasets Description

Name	Length	# of Variables	Description
dna	1,167,877	1	Nanopore DNA sequencing raw current data
pow	2,049,280	1	Household electric power consumption [7]
watch_gyr	3,205,431	3	Heterogeneity Activity Recognition (HAR): smartwatch accelerometer [209]
watch_acc	3,540,962	3	HAR: smartwatch gyroscope [209]
cicids2017	11,323,100	78	CICIDS2017: Network Intrusion Detection Dataset [202]
phones_acc	13,062,475	3	HAR: phone accelerometer [209]
phones_gyr	13,932,632	3	HAR: phone gyroscope [209]
bar_crawl	14,057,567	3	Accelerometer data collected from a college bar crawl [127]
soybeans	22,824,499	1	Enhanced Vegetation Index (EVI) collected from thematic maps produced by the GLAD [6]
synthtic	43,000,000	5	The synthetic dataset used by default
ppg_ecg	90,642,300	1	ECG signals (provided by PPG-DaLiA dataset) collected from chest-worn devices [188]

$2\epsilon \times round(\frac{r}{2\epsilon})$. By utilizing Eq. 7.14, $r' = r - \frac{\partial H(r)}{\partial r}$ (one iteration) and $r'_q = 2\epsilon \times round(\frac{r'}{2\epsilon})$. Figure 7.7 depicts the distribution of r_q and r'_q. As shown in the figure, r_q itself contains multiple peaks (where the green arrows point). Instead of pulling all values in r toward 0 as other regression losses would, QEL automatically identifies the peaks and moves the values around the peaks towards the peaks. Notably, since QEL expects the existence of noise, it can lead to better compression ratio under large, complicated datasets that include more noise.

7.2 Experiments

This section starts with an introduction of the time series datasets used for evaluation, and presents a comprehensive evaluation of Deep Dict.

7.2.1 Datasets

Table 7.2 provides a summary of the datasets used for evaluation, ordered by the length of time series. We apply CICIDS2017 as NIDS dataset. To further evaluate Deep Dict, we also test it on other datasets which cover various domains containing sensor data from mobile devices and smartwatches, agriculture data, DNA data, and ECG data. The time series datasets are accessible to the general public via the UCI ML Repository and Kaggle datasets. To investigate the effects of the length of the time series and the number of variables on the proposed method, we generate additional polynomial synthetic datasets as follows.

Timestamp $t = [\ t_{low}, \ldots, t_{high}\ ]^T \in \mathbb{R}^l$ determines the length of a time series. Polynomial timestamp is $t_p = [t^0, t^1, \ldots, t^{d_p}]^T \in \mathbb{R}^{d_p \times l}$, where d_p is the degree of polynomial. To

Table 7.3: Compression ratio comparison on UTS among CA, SZ, LFZip, and Deep Dict, under 0.1 MaAE. Red Indicates L1 is superior than QEL, green indicates that QEL is better than L1, and orange indicates the boost from RPE (using QEL as loss). Imp.: Improvement compared to the best baseline methods.

Dataset	length	CA	SZ	LFZip	SZ3	DeepDict(L1)	Imp.	DeepDict(QEL)	Imp.	DeepDict(RPE)	Imp.
dna	1,167,877	4.86	**8.62**	8.40	7.78	8.58	-0.46%	8.09	-6.15%	8.36	-3.02%
pow	2,049,280	12.47	23.99	17.98	**24.21**	23.92	-1.20%	23.98	-0.95%	24.00	-0.87%
watch_gyr	3,205,431	10.75	24.79	**28.77**	26.85	27.10	-5.80%	24.68	-14.22%	25.63	-10.91%
watch_acc	3,540,962	5.19	11.00	12.71	10.78	**13.24**	**4.17%**	**12.74**	**0.24%**	**12.87**	**1.26%**
cicids2017	11,323,100	6.11	6.35	3.72	**6.38**	**6.73**	**5.49%**	6.71	5.18%	**6.78**	**6.28%**
phones_acc	13,062,475	7.13	12.63	14.12	12.64	**15.95**	**12.96%**	**15.84**	**12.18%**	**15.89**	**12.54%**
phones_gyr	13,932,632	27.32	52.00	46.28	55.11	**56.44**	**2.41%**	**55.46**	**0.64%**	52.66	-4.45%
bar_crawl	14,057,567	4.99	18.09	19.14	18.39	**27.57**	**44.04%**	**29.41**	**53.66%**	**29.78**	**55.59%**
soybeans	22,824,499	3.43	14.35	15.61	13.50	**17.04**	**9.16%**	**18.32**	**17.36%**	**18.45**	**18.19%**
synthetic	43,000,000	113.07	119.00	58.35	127.23	**125.16**	**-1.63%**	**154.65**	**21.55%**	**149.54**	**17.54%**
ppg_ecg	90,642,300	46.01	69.20	45.47	71.42	**90.31**	**26.45%**	**95.31**	**33.45%**	**97.28**	**36.21%**

Table 7.4: Compression ratio comparison between univariate and multivariate mode under 0.1 MaAE. Uni.: Univariate mode; Mul.: Multivariate mode; Imp.: The improvement of multivariate mode compared to univariate mode

Dataset	Shape	L1			QEL			RPE		
		Uni.	Mul.	Imp.	Uni.	Mul.	Imp.	Uni.	Mul.	Imp.
watch_gyr	3,205,431 × 3	**31.14**	28.83	-7.42%	**31.64**	28.52	-9.86%	**32.93**	28.72	-12.78%
watch_acc	3,540,962 × 3	**13.06**	11.28	-13.63%	**13.53**	11.48	-15.15%	**13.57**	10.79	-20.49%
phone_acc	13,062,475 × 3	11.34	**13.15**	15.96%	11.77	**13.55**	15.12%	11.8	**13.89**	17.71%
phone_gyr	13,932,632 × 3	37.28	**47.09**	26.31%	42.85	**47.81**	11.58%	42.1	**46.9**	11.40%
bar_crawl	14,057,567 × 3	28.08	**29.1**	3.63%	22.56 (b=3)	**28.57** (b=3)	26.64% (b=3)	23.65 (b=3)	**28.85** (b=3)	21.99% (b=3)
synthetic	43,000,000 × 5	24.42	**42.68**	74.77%	15.9	**43.5**	173.58%	28.17	**44.31**	57.29%
cicids2017	11,323,100 × 78	8.43	9.04	7.24%	8.71	9.47	8.73%	8.72	9.43	8.14%

introduce randomness into synthetic data, the coefficient $C \in \mathbb{R}^{d \times d_p}$ is randomly generated.

A time series is defined as $x = (C \times t_p)^T \in \mathbb{R}^{l \times d}$ and the concatenation of multiple x forms a long sequence.

The listed datasets include both multivariate and univariate time series. The first variable is utilised as a univariate dataset for these multivariate time series; for instance, the univariate dataset of watch_gyr indicates that only the x-axis is utilised; and univariate mode indicates that the multivariate time series is flattened prior to being fed into compressors.

7.2.2 Experimental Settings

This section lists the details of experiment settings. Furthermore, the impact and discussions regarding the hyperparameters are presented in Section 7.2.3, and the hyperparam-

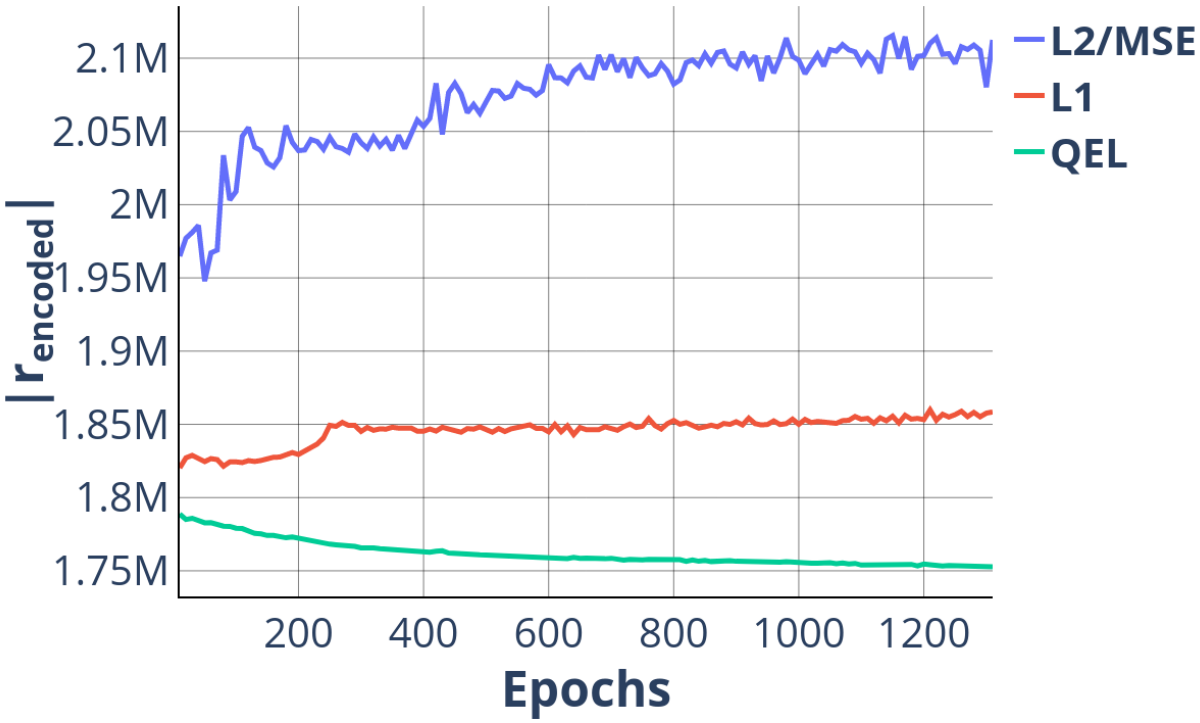


Figure 7.8: Comparison of L1, L2/MSE, and QEL under bar_crawl univariate dataset

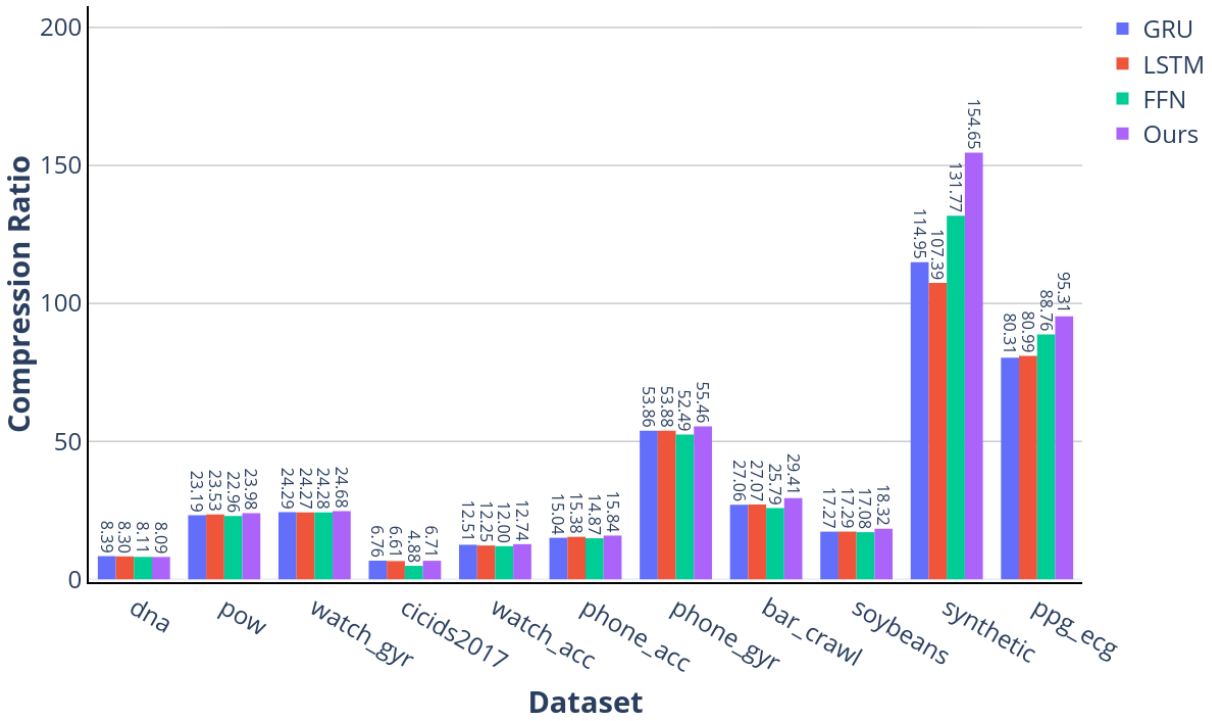


Figure 7.9: Comparison among different architectures of decoder on univariate datasets

Table 7.5: Transferability; NTL: Non-Transfer Learning; TL: Transfer Learning; Imp.: Improvement between TL and NTL in terms of compression ratio

(a) Compression ratio comparison between non-transfer learning and transfer learning under univariate datasets

Dataset	Length	NTL	TL	Imp.	TL+RPE	Imp.
dna	1,167,877	8.09	8.01	-0.99%	8.02	-0.87%
pow	2,049,280	23.98	21.84	-8.92%	22.01	-8.22%
watch_gyr	3,205,431	24.68	24.01	-2.71%	24.19	-1.99%
watch_acc	3,540,962	12.74	12.78	0.31%	12.73	-0.08%
cicids2017	11,323,100	6.71	6.13	-8.64%	6.53	-2.68%
phones_acc	13,062,475	15.84	14.87	-6.12%	14.83	-6.38%
phones_gyr	13,932,632	55.46	53.43	-3.66%	53.76	-3.07%
bar_crawl	14,057,567	29.41	28.54	-2.96%	28.71	-2.38%
soybeans	22,824,499	18.32	17.59	-3.98%	17.85	-2.57%
synthetic	43,000,000	154.65	119.97	-22.42%	120	-22.41%
ppg_ecg	90,642,300	95.31	94.15	-1.22%	96.67	1.43%

(b) Compression ratio comparison between non-transfer learning and transfer learning on multivariate datasets

Dataset	Shape	NTL	TL	Imp.	TL+RPE	Imp.
watch_gyr	3,205,431 × 3	28.52	21.79	-23.60%	27.19	-4.66%
watch_acc	3,540,962 × 3	11.48	10.92	-4.88%	10.71	-6.71%
phone_acc	13,062,475 × 3	13.55	12.31	-9.15%	12.52	-7.60%
phone_gyr	13,932,632 × 3	47.81	43.4	-9.22%	46.31	-3.14%
bar_crawl	14,057,567 × 3	28.57	28.36	-0.74%	30.58	7.04%
synthetic	43,000,000 × 5	43.5	44.84	3.08%	44.13	1.45%
cicids2017	11,323,100 × 78	9.47	8.97	-5.28%	9.06	-4.33%

eters are selected according to the observations in Section 7.2.3. The long sequence in the datasets are truncated with a time window size of 128, and $|c|$ is set at 32 by default. BTAE's encoder has 3 layers with 64 hidden states. The FFN that is used for augmenting c has 1 layer with 64 hidden states. The decoder of BTAE has two layers and all feed forward layers inside the decoder has 64 hidden states. The hyperparameter d_{model} of transformer encoder is set at 32, and the number of heads of MHA is 8. The FFN used for projecting output time series has 1 layer with 64 hidden states. We apply the Gaussian Error Linear Unit (GeLU) as the activation function. The following three loss functions are used: L1, L2 and QEL. The other hyperparameter b (for QEL) is set to 10 as default. Batch size is set to 64. AdamW optimizer is used with a learning rate of 0.0001, weight decay of 0.01, β_1 of 0.9 and β_2 of 0.999. The model is trained by using PyTorch 1.11 on NVIDIA GeForece RTX 3070 and Intel Xeon W-2295.

7.2.3 Numerical Results

This section discusses Deep Dict's performance under a variety of time series datasets.

Results on Univariate Datasets

To evaluate the performance of Deep Dict, five lossy time series compressors are used as baselines. These compressors include Critical Aperture (CA)[1], SZ2[2], LFZip[3], and SZ3[4]. CA is an industrially well-received compressor that is computationally simple and efficient. We utilize the most recent versions of SZ2 (version 2.1.12.2) and SZ3 (version 3.1.5.3), which are the state-of-the-art prediction-based compressors. LFZip is a cutting-edge framework for prediction-quantization-entropy that uses bidirectional GRU to capture nonlinear structure.

Table 7.3 compares the proposed method (RPE is not used by default) to the baselines under the datasets that are ordered with respect to the length of their time series. Under seven out of eleven datasets, the proposed method outperforms the state-of-the-art algorithms. Due to the overhead of BTAE and codes, Deep Dict performs similarly to the baseline on small datasets; however, under large datasets, Deep Dict outperforms the baselines by at most 53.66%. Since the majority of time series datasets are noisy, L1 loss

[1] https://github.com/shubhamchandak94/LFZip/blob/master/src/ca_compress.py
[2] https://github.com/szcompressor/SZ
[3] https://github.com/shubhamchandak94/LFZip
[4] https://github.com/szcompressor/SZ3

is utilised. We compare the QEL loss and L1 loss for further analysis as presented in Table 7.3. As the size of the datasets increases, QEL outperforms L1 under four datasets (marked as green).

It is worth noting that QEL reduces $|r_encoded|$ better than L1 on nearly all datasets; nevertheless, QEL also increases the unique value of the size of the Bernoulli latent states ($|c|$), resulting in an increase in $|c|$ for small datasets.

As depicted in Figure 7.8, when the bar_crawl dataset is considered as a representative example, because L1 and L2 are not particularly designed to reduce the size of $r_{encoded}$, L1 and L2 losses can result in an increase of $|r_{encoded}|$ during the training process; however, QEL can handle such situations and increase the compression ratio. We further enable the RPL (using QEL loss) in Deep Dict. The results indicate that RPL can improve the performance of Deep Dict on eight out of eleven datasets. The relative location of time series can provide the model with additional information.

As demonstrated in Figure 7.1, the decoder of BTAE can be other network architecture intended for time series data, such as FFN, LSTM, or GRU. In order to illustrate the effectiveness of the proposed decoder, we compare the various decoder designs in Figure 7.9. Similar to the typical decoder of an RNN-based autoencoder, the auto-regressive approach is employed for LSTM and GRU, with c serving as the initial input timestamp. All architectures have the same number of hidden states and layers. Under six out of eleven datasets, the results indicate that LSTM performs better than FFN and GRU. Our proposed decoder outperforms the other network architectures under nine out of eleven datasets. In addition, due to the auto-regressive behavior of LSTM and GRU, they require remarkably longer training time when compared to the proposed decoder which is trained in a non-auto-regressive manner.

Results on Multivariate Datasets

Six out of the eleven datasets listed in Table 7.2 contain multivariate time series. Table 7.4 compares the performance of the univariate mode (i.e., flattening the MTS prior to feeding into Deep Dict) and multivariate mode. Due to the fact that flattening MTS increases the size of the Bernoulli latent states (c) by the dimensions, the compression ratio of multivariate mode improves as the length and dimension increase. The default value of b (one of the hyperparameters of QEL defining the accuracy of approximation) is set at 10; however, since the bar_crawl dataset contains extremely large values (greater than 108), a large b will lead to float-point overflow in the derivative, as shown in Eq. 7.14, resulting in NaN in BTAE's outputs. Therefore, we set b to 3 for this dataset.RPE is not employed

by default. Compared with the performance of L1 and QEL, QEL outperforms L1 on all datasets except for the bar_crawl dataset. When RPL is leveraged (QEL is used as loss function), it is able to improve the performance of the univariate and multivariate modes.

Transferability

Training a new model for each time series from scratch is inefficient and time-consuming. Transfer learning is used to accelerate the compression process. The model is initially pre-trained using the largest dataset (i.e., ppg_ecg for univariate datasets and synthetic for multivariate datasets), and then fine-tuned for just 10 epochs (so to align the speed of Deep Dict with that of LFZip) with the remaining datasets. By default, only APE is utilized. As shown in Table 7.5a, the compression ratio of Deep Dict with transfer learning (Deep Dict + TL) reduces by less than 5% under 7 out of 10 univariate datasets when compared to training a model from scratch. Under five out of seven univariate datasets (where Deep Dict outperforms the best baseline), Deep Dict + TL continue to outperform the best baseline. Table 7.5b shows the comparative results between NTL and TL for multivariate datasets. Under five out of six multivariate datasets, Deep Dict + TL decreases the compression ratio by up to 9.22%. Experimental results indicate that Deep Dict can achieve considerably higher speed without sacrificing significant compression ratio. The impact of RPE is examined further. On univariate datasets, the performance of Deep Dict with and without PRE is comparable, demonstrating that RPE cannot significantly enhance Deep Dict's transferability on univariate datasets. Nonetheless, RPE can increase Deep Dict's transferability under four out of six multivariate datasets. Since RPE has no negative impact, it is safe to employ RPE in Deep Dict.

Empirical Study

In the last step, we conduct a series of empirical studies concerning the impact of hyper-parameters and data size. As defined by Eq. 7.11, a large b improves the precision of the approximation, but it may also cause the derivative float-point overflow. As shown in Fig. 7.10, compression ratio increases with b. When $b > 6$, QEL performs better than L1 loss. $b = 10$ is regarded as the default value, since, based on our experiments, it does not result in float-point overflow and b values that are more than 10 do not provide significant advantages.

Since QEL can only minimize the entropy of each batch for each backpropagation, batch size is one of the critical hyperparameters for QEL. Fig. 7.11 indicates that Deep

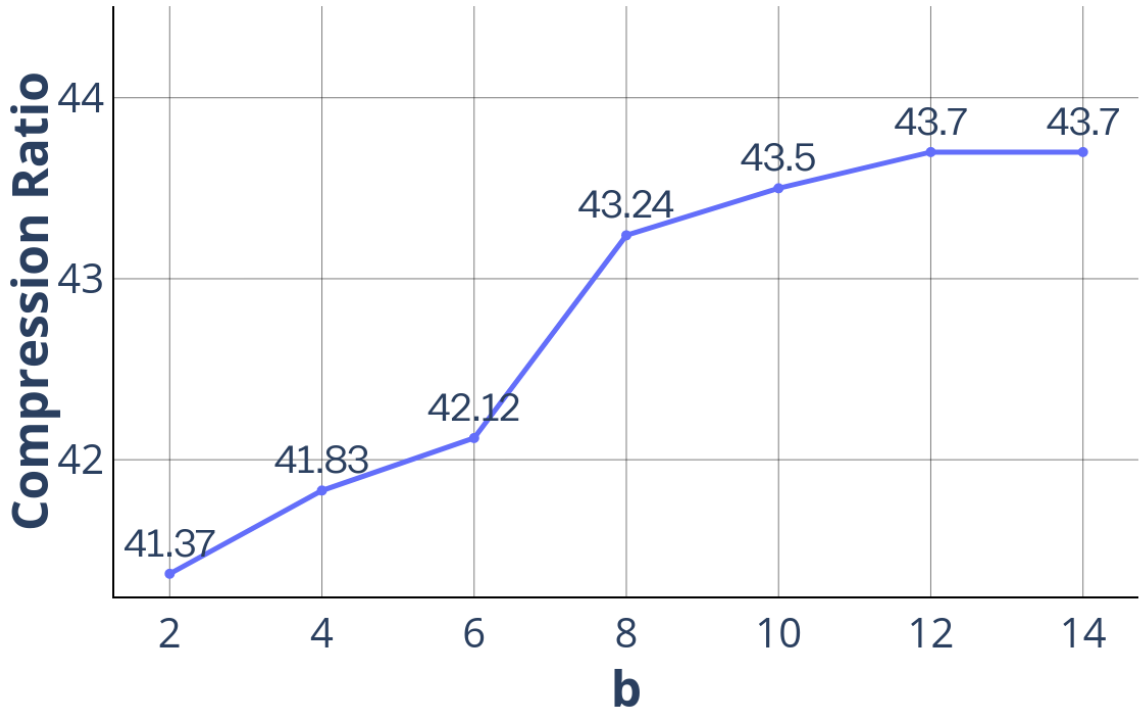


Figure 7.10: The effect of b on compression ratio; synthetic dataset (43,000,000 $\times$ 5) is used.

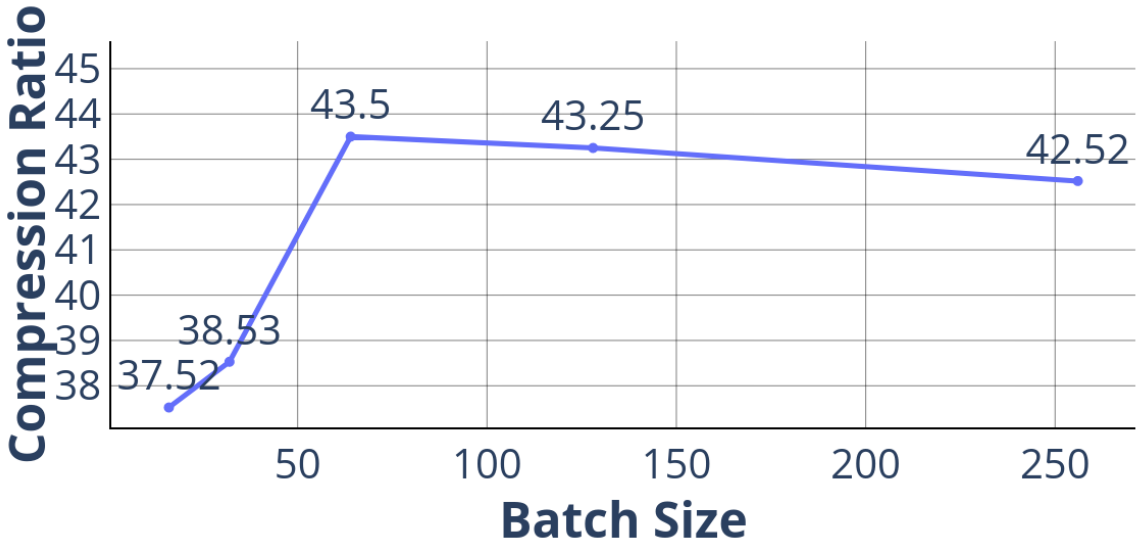


Figure 7.11: The effect of batch size on compression ratio; synthetic dataset (43,000,000*5) is used.

Dict achieves the highest compression ratio when the batch size is 64, and that compression ratio is steady for batch sizes greater than 64. Moreover, a big batch size can significantly increase training speed. Size of the window influences the length of the time series x. Similar to the batch size, a larger window size allows QEL to more accurately estimate data distribution and enables Deep Dict to train more quickly; however, window size also impacts the complexity of x. A long time series makes it difficult for Deep Dict to estimate. As seen in Fig. 7.12, Deep Dict performs effectively with a small window, however its compression ratio decreases rapidly under a large window.

Previous results indicate that Deep Dict outperforms the baselines under large time series datasets. Figure 7.13 illustrates the effect of the dimensionality on compression ratio (with the same hyperparameters). As network size cannot be increased, Deep Dict's compression ratio is limited by the number of parameters. There are two ways to increase the number of parameters: stacking more layers and expanding the network. As shown in Figure 7.14, stacking more transformer encoders does not result in a significant improvement; rather, as the number of layers increases, the compression ratio decreases because of the increase in the decoder size. On the other hand, Figure 7.15 demonstrates that compression ratio can be improved with a large d_{model} (one of the Transformer hyperparameters).

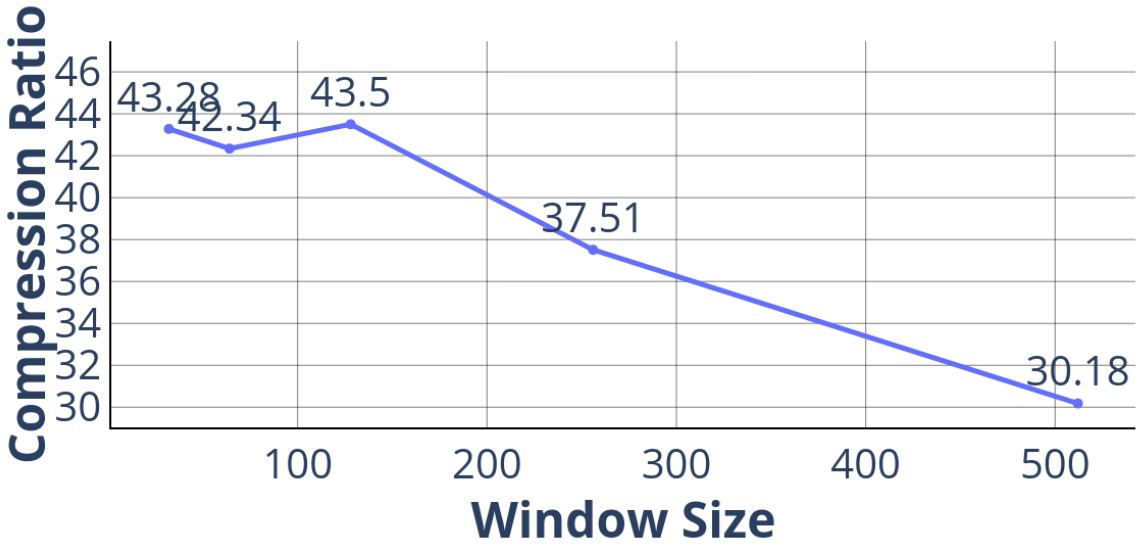


Figure 7.12: The effect of window size on compression ratio; synthetic dataset (43,000,000 $\times$ 5) is used.

It is worth noting that large d_{model} is not suitable for small datasets since large d_{model} will notably increase the number of parameters in BTAE. Figure 7.16 depicts the variation of compression ratio under varying Bernoulli latent states ($|c|$). Increasing $|c|$ is possible to considerably enhance Deep Dict performance for big datasets, although, similar to d_{model}, a large $|c|$ can also increase the number of parameters. In summary, increasing b, d_{model}, and $|c|$ can further enhance the performance of long time series.

7.3 Conclusion

In this chapter , we have proposed a Bernoulli Transformer Autoencoder-based lossy time series compressor, namely Deep Dict to improve the compression ratio by learning the Bernoulli representation of time series. Upon pointing out the limitations of conventional regression losses such as L1 and L2, to provide a more accurate definition of the problem, we have substituted the conventional regression loss with a novel loss function, Quantized Entropy Loss (QEL), which further improves the compression ratio and reduces the difficulty of optimization. eleven time series datasets (spanning a variety of domains) are used to

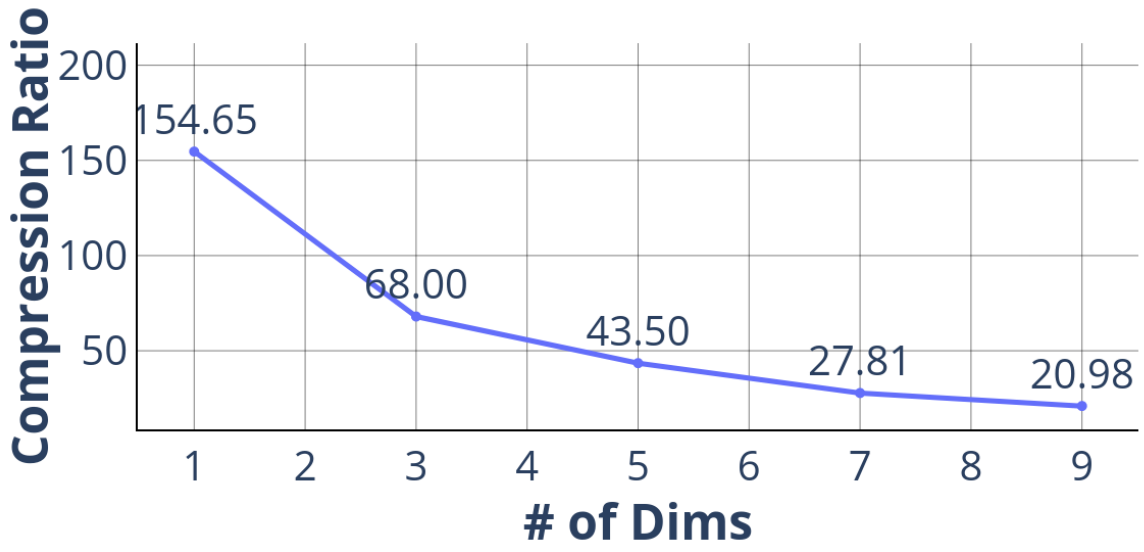


Figure 7.13: The change of compression ratio with the number of dimension of data; synthetic datasets are used which has the same length (43,000,000).

test Deep Dict. Deep Dict outperforms the state-of-the-art time series compression under 7 out of 11 datasets, particularly under the lengthy time series datasets. We have shown that Deep Dict can outperform the best baseline by a maximum of 53.66%. The proposed loss function, QEL can boost compression ratios more than the traditional regression losses such as L1 and L2. The experiment on the transferability demonstrates that Deep Dict can be accelerated by transfer learning without significantly sacrificing much compression ratio (less than 5%). Moreover, RPE can improve Deep Dict's transferability on multivariate datasets. When multivariate and univariate modes are compared, the results indicate that Deep Dict's multivariate mode performs better under larger multivariate time series. Through a series of empirical tests, b, d_{model}, and $|c|$ have been identified to be effective hyperparameters.

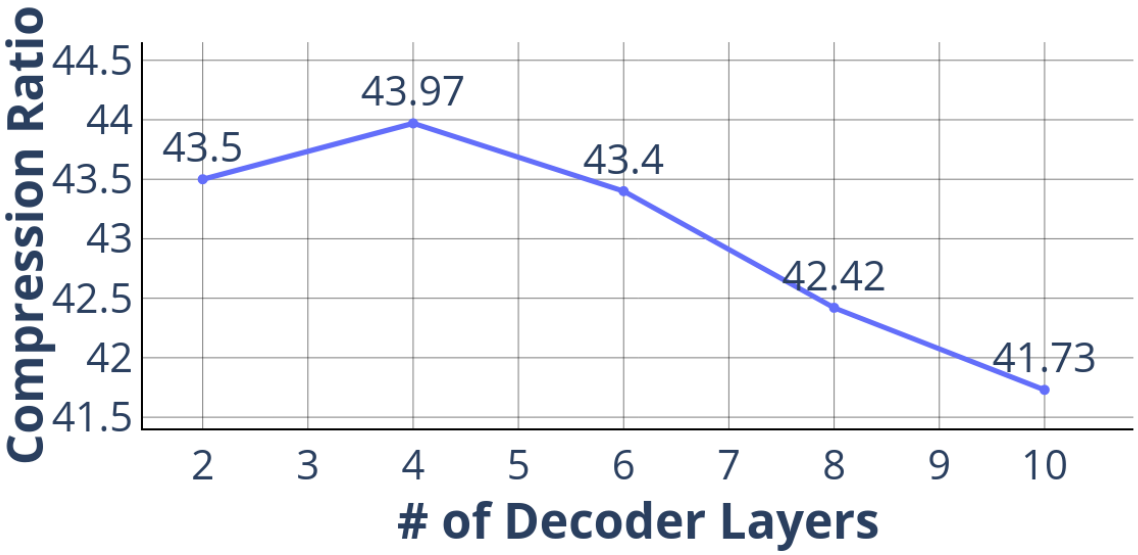


Figure 7.14: The effect of the number of layers on compression ratio; synthetic dataset (43,000,000 $\times$ 5) is used.

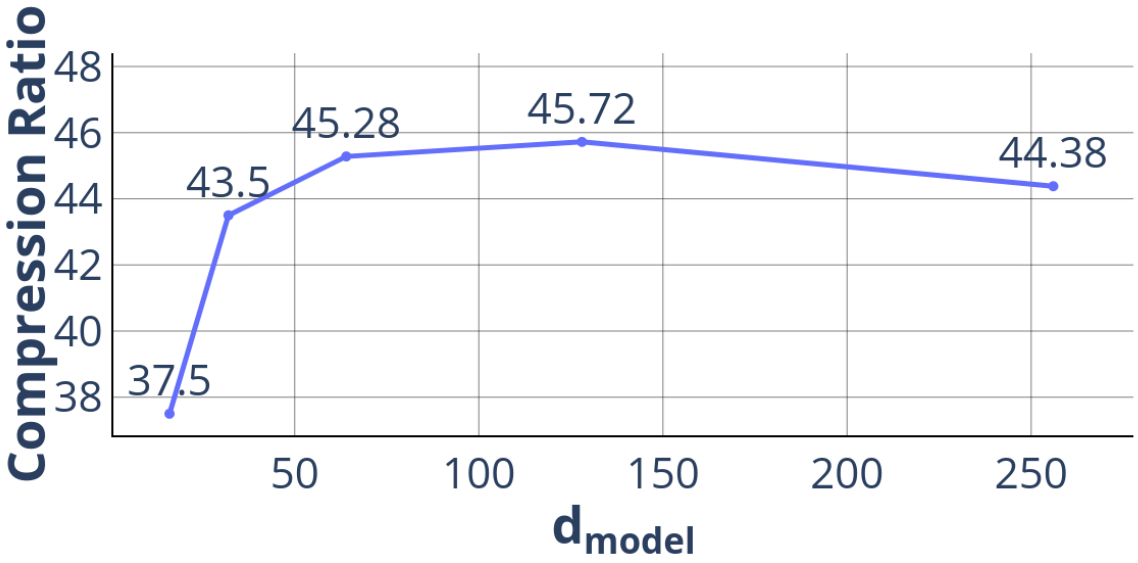


Figure 7.15: The effect of d_{model} on compression ratio; synthetic dataset (43,000,000 $\times$ 5) is used.

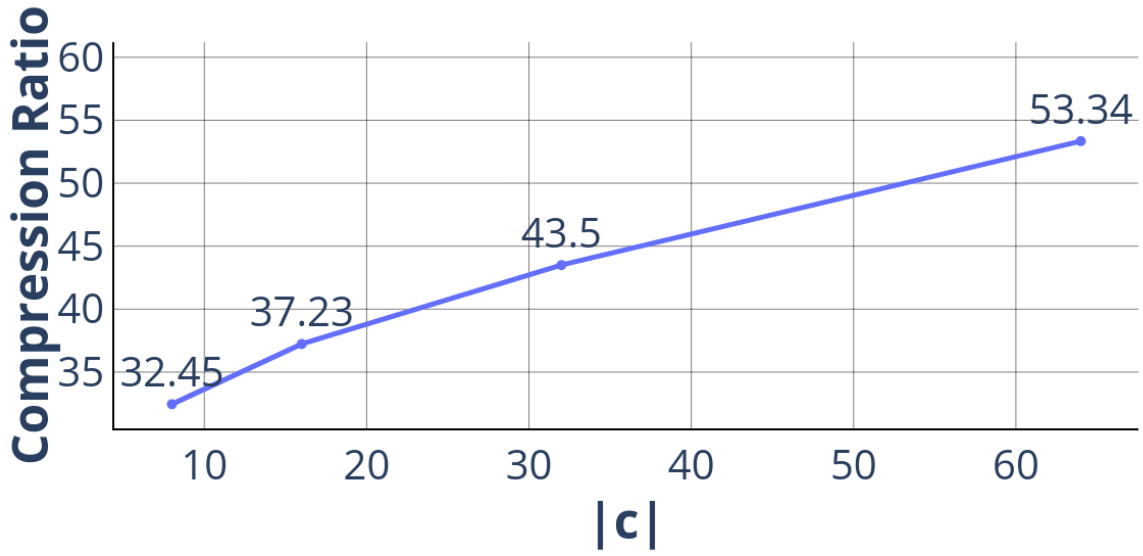


Figure 7.16: The effect of $|c|$ on compression ratio; synthetic dataset (43,000,000 $\times$ 5) is used.

Chapter 8

Conclusions and Future Directions

8.1 Conclusions

It is anticipated that the Internet of Things (IoT) would see rapid expansion due to the proliferation of communication technologies, the availability of devices, and the advancement of computer systems. Therefore, IoT security is a concern in order to safeguard the hardware and networks of IoT systems. This thesis has mitigated security problems originating from a variety of perspectives and areas. First, we have presented Risk-Aware Smart Access Control (RASA), an IoT-assisted intelligent access control system, utilizing unsupervised ML techniques such as KMeans and DBSCAN. In order to evaluate the framework, the outcomes of rule-based policy have been compared to those of RASA, and both have been adjusted after each iteration. Following several cycles of fine-tuning, the decisions of the two techniques have obtained $> 99.9\%$ consistency. In addition, to validate the outcomes of RASA, when two supervised techniques (Decision Tree and SVM) have been trained and evaluated using distinct action logs labelled by RASA, a test accuracy of 99.95% has been attained.

This thesis has significantly enhanced NIDS in several ways. We have launched standard incursions in an IoT environment and generated a NIDS dataset. On the dataset, several ML approaches (including supervised and unsupervised algorithms) have been evaluated. The results demonstrate that contemporary IoT devices are still susceptible to classical DoS and probing assaults and that ML-based NIDS can efficiently recognize these threats. Among supervised approaches, tree-based methods have an F1 score of 96%. For the unsupervised technique, EM earns an F1 score of 67.2%. This thesis has examined the effects and defence methods of APT assaults, which have emerged in recent decades. We

have set up the network using virtual machines, where a multi-domain environment has been created. Kali and the Metasploit framework have been used to exploit and penetrate the whole network. We have formed the SCVIC-APT-2021 dataset by collecting network packages from the target environment and features are extracted from the packets. The SCVIC-APT-2021 dataset comprises all APT phases.

ACM is a novel ensemble approach, named Attack Centric Method (ACM), for developing classifiers for each assault type. We have proposed Attack Centric Feature Selection (ACFS) to boost ACM which selects different feature sets for each class. In order to reduce the risk of overfitting issues, Diagonal Softmax Regression has been derived to serve as the aggregator of ACM. To offer a fair baseline, various machine learning methods have been tested on the SCVIC-APT-2021 dataset, among which, Random Forest has performed the best with a macro F1 score of 75.2%. Results indicate that ACM can reach a macro F1 score of 82.3%, 9.4% higher than the best baseline performance.

Despite the fact that several ML/DL models have been suggested over the years to enhance performance, data noise still is one of the most important impediments limiting prediction outcomes. This thesis, therefore, has proposed a novel CIDS that combines the benefits of NIDS and HIDS. A methodology for the formation of CIDS datasets has been suggested, called CIDS Dataset Formation Framework (CDFF). CDFF collects network packets and system logs from various Operating Systems (OS) including Windows and Linux OSs. CDFF also contains an alignment algorithm to align the network flow and system log events. Three datasets titled SCVIC-CIDS-2021-Reduced, SCVIC-CIDS-2021, and SCVIC-CIDS-2022 have been constructed via CDFF. SCVIC-CIDS-2021-Reduced and SCVIC-CIDS-2021 use the meta-data of CIC-IDS-2018 and SCVIC-CIDS-2022 applies the meta-data from NDSec-1.

Further, we have presented the CIDS-Net deep learning model, which can mix inputs from network and host domains. CIDS-Net has achieved a macro F1 score of 99.5% under SCVIC-CIDS-2021, 5.76% higher than the best baseline result. On the SCVIC-CIDS-2022 dataset, CIDS-Net has achieved a macro F1 score of 91.3%, which is 5.1% more than the best baseline result. In addition, we have given an explanation of the CIDS enhancements. The results indicate that incorporating host-based data greatly reduces data noise and increases the distance between classes.

In addition to enhancing NIDS's detection performance, we have enhanced its reaction speed. This thesis has combined the concepts of early detection and network intrusion detection systems to make choices before harmful traffic reaches target computers. In light of this, we have proposed a framework capable of describing each network flow as a time series and created a new dataset called TS-CICIDS2017. This thesis has proposed

MDT to extract features from inputs with limited information. Compared to current strategies, our proposed approach can enhance earliness by 500,000 times and duration-based earliness by 60 times. When the first 10 packets in two seconds have been employed for detection, the proposed approach has achieved a satisfactory detection performance (F-score greater than 84%). The formation of network traffic as time series can significantly enhance detection speed, but storing and transferring incur bandwidth and storage costs. We, therefore, have proposed Deep Dict to increase the compression ratio by learning the Bernoulli representation of time series. In order to define the problem more precisely, we have replaced the trivial regression loss with a novel loss function, Quantized Entropy Loss (QEL), which further increases the compression ratio and minimizes the complexity of optimization. Deep Dict has been tested on ten time series datasets (covering a number of areas). Deep Dict has surpassed the state-of-the-art time series compression in seven out of ten datasets, especially with respect to large time series datasets. We have demonstrated that Deep Dict can outperform the best baseline by up to 53.66%. The suggested loss function, QEL, can increase compression ratios more than the conventional regression losses L1 and L2.

8.2 Future Directions

This thesis enhances access control and intrusion detection systems (IDS) from a variety of angles, yet further work is required to strengthen the security of IoT networks.

Future work of RASA framework could seek evaluation against data derived from authentic use cases. Similar approaches to automatic inference of authorization contexts and associated risks may be particularly applicable in rich cyber-physical environments where sensors are deployed to generate activity logs by monitoring physical actions, in addition to common network and endpoint cyber action logs. Such sensor-rich environments are becoming increasingly common in extremely diverse cyber-physical contexts, including clinical, industrial, home, automotive, smart-building, and smart-city contexts. As these cyber-physical systems grow in size and complexity, human-written context-dependent authorization access rules will not scale, nor will expert risk analysis be cost-effective and reliable without evidence-based risk assessment of such accesses. The RASA framework proposed in this thesis may be a stepping stone to a rich exploration of similar approaches to address such growing problems in cyber-physical security.

Even though this thesis has incorporated classical and APT attacks and constructed a dataset, further modern and sophisticated attacks can be added in the future: for instance, IoT nodes may also suffer from firmware update attacks; hence, such attacks will also be

integrated with the simulation environment in the future. Additionally, the datasets are generated in virtual or simulated environments (such as virtual machines and Contiki-simulator). The future datasets can be generated under physical PCs and IoT nodes. The created data can be also tested under unsupervised methods (such as Hidden Markov Model, Gaussian Mixture Model, and other EM-based algorithms) to explore anomalies.

In chapter 5, we have suggested a novel CIDS where only system logs are studied. Nonetheless, as host-based data can be various, such as system calls, audit records, and file systems, more research can incorporate other forms of meta-data into CIDS datasets. As we have tested on the generated CIDS datasets, the performance reaches 99.98%. Future CIDS datasets can have more sophisticated intrusions to evaluate the efficacy of the combination of network- and host-based features.

The proposed early detection system has demonstrated several advantages for NIDS. Currently, it is tested on CIC-IDS-2017 while more NIDS datasets can be tested to evaluate the proposed method. MDT is trained in a supervised manner, however many real-life intrusion datasets may not contain enough labels. Consequently, it may be enhanced by unsupervised learning to extract abundant features from packets of varying amounts and lengths. The storage or bandwidth overhead of Deep Dict contains three components: the binary distributed codes, the decoder and the encoded residuals. The proposed Deep Dict minimizes the size of encoded residuals. Nonetheless, for small datasets, the other two components are still the major overhead. Future work on Deep Dict may include neural network quantization to further minimize the size of the decoder and explore a method which can also minimize the size of the binary codes.

Moreover, since diverse data sizes and types have distinct hyperparameters, selecting hyperparameters automatically based on NIDS or other time series data is a promising direction.

References

[1] Ant Colony Induced Decision Trees for Intrusion Detection.

[2] DS2OS traffic traces.

[3] IDS 2018 Datasets Research Canadian Institute for Cybersecurity UNB.

[4] Internet of Things (IoT) Operating Systems Market – Global Industry Trends and Forecast to 2026 | Data Bridge Market Research.

[5] Machine Learning-Based Adaptive Anomaly Detection in Smart Spaces.

[6] Soybean South America Time Series.

[7] UCI Machine Learning Repository: Individual household electric power consumption Data Set.

[8] Data Mining with Decision Trees: Theory and Applications. *Online Information Review*, 39(3):437–438, January 2015. Emerald Group Publishing Ltd.

[9] contiki-ng/contiki-ng, May 2020. original-date: 2017-05-13T17:37:59Z.

[10] Mohamed S. Abdelfattah, Andrei Hagiescu, and Deshanand Singh. Gzip on a chip: high performance lossless data compression on FPGAs using OpenCL. In *Proceedings of the International Workshop on OpenCL 2013 & 2014*, IWOCL '14, pages 1–9, New York, NY, USA, 2014. Association for Computing Machinery.

[11] Charu C. Aggarwal, Alexander Hinneburg, and Daniel A. Keim. On the Surprising Behavior of Distance Metrics in High Dimensional Space. In Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, Jan Van den Bussche, and Victor Vianu, editors, *Database Theory — ICDT 2001*, volume 1973, pages 420–434. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

[12] Alaa Alhowaide, Izzat Alsmadi, and Jian Tang. Towards the design of real-time autonomous IoT NIDS. *Cluster Computing*, January 2021.

[13] Iman Almomani, Bassam Al-Kasasbeh, and Mousa AL-Akhras. WSN-DS: A Dataset for Intrusion Detection Systems in Wireless Sensor Networks, 2016.

[14] Hazim Almuhimedi et al. Your Location has been Shared 5,398 Times! A Field Study on Mobile App Privacy Nudging. In *ACM Conf. Human Fact. in Comp. Sys.*, pages 787–796, Seoul, Korea, April 2015.

[15] Ethem Alpaydin. *Introduction to Machine Learning*. The MIT Press, Cambridge, MA, fourth edition edition edition, February 2020.

[16] Adel Alshamrani, Sowmya Myneni, Ankur Chowdhary, and Dijiang Huang. A Survey on Advanced Persistent Threats: Techniques, Solutions, Challenges, and Research Opportunities. *IEEE Communications Surveys Tutorials*, 21(2):1851–1877, 2019. Conference Name: IEEE Communications Surveys Tutorials.

[17] Panagiotis Andriotis, Theo Tryfonas, George Oikonomou, and Can Yildiz. A pilot study on the security of pattern screen-lock methods and soft side channel attacks. In *Proceedings of the Sixth ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec '13, page 1–6, New York, NY, USA, 2013. Association for Computing Machinery.

[18] Fazel Anjomshoa, Moayad Aloqaily, Burak Kantarci, Melike Erol-Kantarci, and Stephanie Schuckers. Social Behaviometrics for Personalized Devices in the Internet of Things Era. *IEEE Access*, 5:12199–12213, 2017.

[19] Zied Aouini and Adrian Pekar. NFStream: A flexible network data analysis framework. *Computer Networks*, 204:108719, February 2022.

[20] Preeti Arora, Deepali, and Shipra Varshney. Analysis of K-Means and K-Medoids Algorithm For Big Data. *Procedia Comp. Sci.*, 78:507–512, January 2016.

[21] Yosef Ashibani, Dylan Kauling, and Qusay H. Mahmoud. A context-aware authentication framework for smart homes. In *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 1–5, April 2017. ISSN: null.

[22] Koorosh Aslansefat, Ioannis Sorokos, Declan Whiting, Ramin Tavakoli Kolagari, and Yiannis Papadopoulos. SafeML: Safety Monitoring of Machine Learning Classifiers Through Statistical Difference Measures. In Marc Zeller and Kai Höfig, editors,

Model-Based Safety and Assessment, Lecture Notes in Computer Science, pages 197–211, Cham, 2020. Springer International Publishing.

[23] Ramin Atefinia and Mahmood Ahmadi. Network Intrusion Detection using Multi-Architectural Modular Deep Neural Network. *The Journal of Supercomputing*, 77, April 2021.

[24] Hany F. Atlam and Gary B. Wills. An efficient security risk estimation technique for risk-based access control model for iot. *Internet of Things*, 6:100052, 2019.

[25] Adam J. Aviv, Katherine Gibson, Evan Mossop, Matt Blaze, and Jonathan M. Smith. Smudge attacks on smartphone touch screens. In *Proceedings of the 4th USENIX Conference on Offensive Technologies*, WOOT'10, page 1–7, USA, 2010. USENIX Association.

[26] Adam J. Aviv, Benjamin Sapp, Matt Blaze, and Jonathan M. Smith. Practicality of accelerometer side channels on smartphones. In *Proceedings of the 28th Annual Computer Security Applications Conference*, ACSAC '12, page 41–50, New York, NY, USA, 2012. Association for Computing Machinery.

[27] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders, April 2021. arXiv:2003.05991 [cs, stat].

[28] Ram Basnet, Riad Shash, Clayton Johnson, Lucas Walgren, and Tenzin Doleck. Towards Detecting and Classifying Network Intrusion Traffic Using Deep Learning Frameworks. December 2019.

[29] Catherine Beazley, Karan Gadiya, Ravi K U Rakesh, David Roden, Boda Ye, Brendan Abraham, Donald E. Brown, and Malathi Veeraraghavan. Exploratory Data Analysis of a Unified Host and Network Dataset. In *2019 Systems and Information Engineering Design Symposium (SIEDS)*, pages 1–5, April 2019.

[30] Frank Beer and Ulrich Bühler. Feature selection for flow-based intrusion detection using Rough Set Theory. In *2017 IEEE 14th International Conference on Networking, Sensing and Control (ICNSC)*, pages 617–624, May 2017.

[31] Frank Beer, Tim Hofer, David Karimi, and Ulrich Bühler. A new attack composition for network security. In Paul Müller, Bernhard Neumair, Helmut Raiser, and Gabi Dreo Rodosek, editors, *10. DFN-Forum Kommunikationstechnologien*, pages 11–20, Bonn, 2017. Gesellschaft für Informatik e.V.

[32] Ron Bitton and Asaf Shabtai. A Machine Learning-Based Intrusion Detection System for Securing Remote Desktop Connections to Electronic Flight Bag Servers. *IEEE Transactions on Dependable and Secure Computing*, 18(3):1164–1181, May 2021. Conference Name: IEEE Transactions on Dependable and Secure Computing.

[33] Davis Blalock, Samuel Madden, and John Guttag. Sprintz: Time Series Compression for the Internet of Things. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2(3):93:1–93:23, 2018.

[34] Remco R Bouckaert. Bayesian Network Classifiers in Weka for Version 3-5-7. page 47.

[35] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, August 1996.

[36] Andy Brown, Aaron Tuor, Brian Hutchinson, and Nicole Nichols. Recurrent Neural Network Attention Mechanisms for Interpretable System Log Anomaly Detection. In *Proceedings of the First Workshop on Machine Learning for Computing Systems*, MLCS'18, pages 1–8, New York, NY, USA, June 2018. Association for Computing Machinery.

[37] Aniello Castiglione, Kim-Kwang Raymond Choo, Michele Nappi, and Stefano Ricciardi. Context Aware Ubiquitous Biometrics in Edge of Military Things. *IEEE Cloud Computing*, 4(6):16–20, November 2017.

[38] S. Cha and K. Yeh. A data-driven security risk assessment scheme for personal data protection. *IEEE Access*, 6:50510–50517, 2018.

[39] Nadia Chaabouni, Mohamed Mosbah, Akka Zemmari, Cyrille Sauvignac, and Parvez Faruki. Network Intrusion Detection for IoT Security Based on Learning Techniques. *IEEE Communications Surveys & Tutorials*, 21(3):2671–2701, 2019.

[40] Timothy Chadza, Konstantinos G. Kyriakopoulos, and Sangarapillai Lambotharan. Contemporary Sequential Network Attacks Prediction using Hidden Markov Model. In *2019 17th International Conference on Privacy, Security and Trust (PST)*, pages 1–3, August 2019. ISSN: 2643-4202.

[41] Shubham Chandak, Kedar Tatwawadi, Chengtao Wen, Lingyun Wang, Juan Aparicio Ojea, and Tsachy Weissman. LFZip: Lossy Compression of Multivariate Floating-Point Time Series Data via Improved Prediction. In *2020 Data Compression Conference (DCC)*, pages 342–351, March 2020. ISSN: 2375-0359.

[42] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Trans. on Intel. Sys. and Technology*, 2(3):1–27, April 2011.

[43] Shehzad Ashraf Chaudhry, Khalid Yahya, Fadi Al-Turjman, and Ming-Hour Yang. A Secure and Reliable Device Access Control Scheme for IoT Based Sensor Cloud Systems. *IEEE Access*, 8:139244–139254, 2020. Conference Name: IEEE Access.

[44] Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. In *ACM SIGKDD Int Conf. on Knowledge Discovery and Data Mining*, pages 785–794, San Francisco, CA, USA, August 2016.

[45] Zhiyan Chen, Jinxin Liu, Yu Shen, Murat Simsek, Burak Kantarci, Hussein T. Mouftah, and Petar Djukic. Machine learning-enabled iot security: Open issues and challenges under advanced persistent threats. *ACM Comput. Surv.*, apr 2022. Just Accepted.

[46] Zhiyan Chen, Murat Simsek, Burak Kantarci, and Petar Djukic. All predict wisest decides: A novel ensemble method to detect intrusive traffic in iot networks. In *2021 IEEE Global Communications Conference (GLOBECOM)*, pages 01–06, 2021.

[47] Giacomo Chiarot and Claudio Silvestri. Time series compression: a survey, January 2021. Number: arXiv:2101.08784 arXiv:2101.08784 [cs].

[48] Davide Chicco and Giuseppe Jurman. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics*, 21, January 2020.

[49] Jason Paul Cruz, Yuichi Kaji, and Naoto Yanai. RBAC-SC: Role-Based Access Control Using Smart Contract. *IEEE Access*, 6:12240–12251, 2018. Conference Name: IEEE Access.

[50] Antitza Dantcheva, Petros Elia, and Arun Ross. What Else Does Your Biometric Data Reveal? A Survey on Soft Biometrics. *IEEE Transactions on Information Forensics and Security*, 11(3):441–467, March 2016.

[51] Venkat Surya Dasari, Maryam Pouryazdan, and Burak Kantarci. Selective versus Non-Selective Acquisition of Crowd-Solicited IoT Data and Its Dependability. In *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6, May 2018. ISSN: 2474-9133.

[52] S. De Vito, E. Massera, M. Piga, L. Martinotto, and G. Di Francia. On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario. *Sensors and Actuators B: Chemical*, 129(2):750–757, February 2008.

[53] Jyoti Deogirikar and Amarsinh Vidhate. Security attacks in IoT: A survey. In *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, pages 32–37, February 2017. ISSN: null.

[54] I. Deutschmann, P. Nordström, and L. Nilsson. Continuous authentication using behavioral biometrics. *IT Professional*, 15(4):12–15, July 2013.

[55] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs]*, May 2019. arXiv: 1810.04805.

[56] L. Dhanabal and S. P. Shantharajah. A Study on NSL-KDD Dataset for Intrusion Detection System Based on Classification Algorithms. 2015.

[57] Sheng Di and Franck Cappello. Fast Error-Bounded Lossy HPC Data Compression with SZ. In *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 730–739, 2016. ISSN: 1530-2075.

[58] Hui Ding, Goce Trajcevski, Peter Scheuermann, Xiaoyue Wang, and Eamonn Keogh. Querying and mining of time series data: experimental comparison of representations and distance measures. *Proceedings of the VLDB Endowment*, 1(2):1542–1552, 2008.

[59] Sheng Ding, Jin Cao, Chen Li, Kai Fan, and Hui Li. A Novel Attribute-Based Access Control Scheme Using Blockchain for IoT. *IEEE Access*, 7:38431–38441, 2019. Conference Name: IEEE Access.

[60] Pedro Domingos. A unified bias-variance decomposition for zero-one and squared loss. *AAAI/IAAI*, 2000:564–569, 2000.

[61] G. Dong, Y. Jin, S. Wang, W. Li, Z. Tao, and S. Guo. Db-kmeans:an intrusion detection algorithm based on dbscan and k-means. In *20th APNOMS Symposium*, pages 1–4, 2019.

[62] Martin Drahansky, Štěpán Mráček, Marina L. Gavrilova, Svetlana N. Yanushkevich, Jan Vana, Vlad P. Shmerko, Radim Dvora, and Ahmad Poursaberi. Facial biometrics for situational awareness systems. *IET Biometrics*, 2(2):35–47, June 2013.

[63] Gerard Draper-Gil, Arash Habibi Lashkari, Mohammad Saiful Islam Mamun, and Ali A. Ghorbani. Characterization of Encrypted and VPN Traffic using Time-related Features. pages 407–414, December 2021.

[64] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 1285–1298, New York, NY, USA, October 2017. Association for Computing Machinery.

[65] Yann Dubois, Benjamin Bloem-Reddy, Karen Ullrich, and Chris J Maddison. Lossy Compression for Lossless Prediction. In *Advances in Neural Information Processing Systems*, volume 34, pages 14014–14028. Curran Associates, Inc., 2021.

[66] Besik Dundua and Mikheil Rukhaia. Towards Integrating Attribute-Based Access Control into Ontologies. In *2019 IEEE 2nd Ukraine Conference on Electrical and Computer Engineering (UKRCON)*, pages 1052–1056, July 2019.

[67] A. Dunkels, B. Gronvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *29th Annual IEEE International Conference on Local Computer Networks*, pages 455–462, November 2004. ISSN: 0742-1303.

[68] Serge Egelman, Sakshi Jain, Rebecca S. Portnoff, Kerwell Liao, Sunny Consolvo, and David Wagner. Are You Ready to Lock? In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 750–761, Scottsdale, Arizona, USA, November 2014. Association for Computing Machinery.

[69] Felix Erlacher and Falko Dressler. On High-Speed Flow-Based Intrusion Detection Using Snort-Compatible Signatures. *IEEE Transactions on Dependable and Secure Computing*, 19(1):495–506, 2022. Conference Name: IEEE Transactions on Dependable and Secure Computing.

[70] Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. page 6, 1996.

[71] Dave Evans. How the Next Evolution of the Internet Is Changing Everything, 2011. Library Catalog: www.semanticscholar.org.

[72] Huan Feng, Kassem Fawaz, and Kang G. Shin. Continuous Authentication for Voice Assistants. In *Proceedings of the 23rd Annual International Conference on Mobile*

Computing and Networking, MobiCom '17, pages 343–355, New York, NY, USA, 2017. ACM. event-place: Snowbird, Utah, USA.

[73] Mohamed Amine Ferrag, Leandros Maglaras, Sotiris Moschoyiannis, and Helge Janicke. Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. *Journal of Information Security and Applications*, 50:102419, February 2020.

[74] Qusyairi Ridho Saeful Fitni and Kalamullah Ramli. Implementation of Ensemble Learning and Feature Selection for Performance Improvements in Anomaly-Based Intrusion Detection Systems. In *2020 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT)*, pages 118–124, July 2020.

[75] Travis Floyd, Matthew Grieco, and Edna F. Reid. Mining hospital data breach records: Cyber threats to U.S. hospitals. In *2016 IEEE Conference on Intelligence and Security Informatics (ISI)*, pages 43–48, September 2016. ISSN: null.

[76] Ivo Friedberg, Florian Skopik, Giuseppe Settanni, and Roman Fiedler. Combating advanced persistent threats: From network event correlation to incident detection. *Computers & Security*, 48:35–57, 2015.

[77] Kevin Fu, Tadayoshi Kohno, Daniel Lopresti, Elizabeth Mynatt, Klara Nahrstedt, Shwetak Patel, Debra Richardson, and Ben Zorn. Safety, security, and privacy threats posed by accelerating trends in the internet of things, 2020.

[78] Sunanda Gamage and Jagath Samarabandu. Deep learning methods in network intrusion detection: A survey and an objective comparison. *Journal of Network and Computer Applications*, 169:102767, November 2020.

[79] Fatma Gara, Leila Ben Saad, and Rahma Ben Ayed. An intrusion detection system for selective forwarding attack in IPv6-based mobile WSNs. In *Int. Wireless Communications and Mobile Computing Conf.*, pages 276–281, June 2017.

[80] Ibrahim Ghafir, Mohammad Hammoudeh, Vaclav Prenosil, Liangxiu Han, Robert Hegarty, Khaled Rabie, and Francisco J. Aparicio-Navarro. Detection of advanced persistent threat using machine-learning correlation analysis. *Future Generation Computer Systems*, 89:349–359, 2018.

[81] Ibrahim Ghafir, Konstantinos G. Kyriakopoulos, Sangarapillai Lambotharan, Francisco J. Aparicio-Navarro, Basil Assadhan, Hamad Binsalleeh, and Diab M. Diab.

Hidden markov models and alert correlations for the prediction of advanced persistent threats. *IEEE Access*, 7:99508–99520, 2019.

[82] Mohamed F. Ghalwash, Dušan Ramljak, and Zoran Obradović. Early classification of multivariate time series using a hybrid HMM/SVM model. In *2012 IEEE International Conference on Bioinformatics and Biomedicine*, pages 1–6, 2012.

[83] Julius Francis Gomes, Marika Iivari, Petri Ahokangas, Lauri Isotalo, and Niemel Riikka. Cybersecurity Business Models for IoT-Mobile Device Management Services in Futures Digital Hospitals. *Journal of ICT Standardization*, 5(1):107–128, July 2017.

[84] Mohit Goyal, Kedar Tatwawadi, Shubham Chandak, and Idoia Ochoa. DZip: improved general-purpose loss less compression based on novel neural network modeling. In *2021 Data Compression Conference (DCC)*, pages 153–162, March 2021. ISSN: 2375-0359.

[85] Ilya Grebnov. IlyaGrebnov/libbsc, May 2022. original-date: 2011-05-11T06:39:49Z.

[86] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645 – 1660, 2013.

[87] Adrián Gómez-Brandón, José R. Paramá, Kevin Villalobos, Arantza Illarramendi, and Nieves R. Brisaboa. Lossless compression of industrial time series with direct access. *Computers in Industry*, 132:103503, November 2021.

[88] Arash Habibi Lashkari, Gerard Draper Gil, Mohammad Saiful Islam Mamun, and Ali A. Ghorbani. Characterization of Tor Traffic using Time based Features:. In *Proceedings of the 3rd International Conference on Information Systems Security and Privacy*, pages 253–262, Porto, Portugal, 2017. SCITEPRESS - Science and Technology Publications.

[89] Hadi Habibzadeh, Cem Kaptan, Tolga Soyata, Burak Kantarci, and Azzedine Boukerche. Smart city system design: A comprehensive study of the application and data planes. *ACM Comput. Surv.*, 52(2), May 2019.

[90] Hadi Habibzadeh, Tolga Soyata, Burak Kantarci, Azzedine Boukerche, and Cem Kaptan. Sensing, communication and security planes: A new challenge for a smart city system design. *Computer Networks*, 144:163 – 200, 2018.

[91] Shangbin Han, Qianhong Wu, Han Zhang, Bo Qin, Jiankun Hu, Xingang Shi, Linfeng Liu, and Xia Yin. Log-Based Anomaly Detection With Robust Feature Extraction and Online Learning. *IEEE Transactions on Information Forensics and Security*, 16:2300–2311, 2021. Conference Name: IEEE Transactions on Information Forensics and Security.

[92] Mahmudul Hasan, Md. Milon Islam, Md Ishrak Islam Zarif, and M. M. A. Hashem. Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches. *Internet of Things*, 7:100059, September 2019.

[93] Trevor Hastie, Saharon Rosset, Ji Zhu, and Hui Zou. Multi-class AdaBoost. *Statistics and Its Interface*, 2(3):349–360, 2009.

[94] Eiji Hayashi, Sauvik Das, Shahriyar Amini, Jason Hong, and Ian Oakley. CASA: Context-aware Scalable Authentication. In *Proceedings of the Ninth Symposium on Usable Privacy and Security*, SOUPS '13, pages 3:1–3:10, New York, NY, USA, 2013. ACM. event-place: Newcastle, United Kingdom.

[95] Debiao He and Sherali Zeadally. Authentication protocol for an ambient assisted living system. *IEEE Communications Magazine*, 53(1):71–77, January 2015.

[96] Shilin He, Pinjia He, Zhuangbin Chen, Tianyi Yang, Yuxin Su, and Michael R. Lyu. A Survey on Automated Log Analysis for Reliability Engineering. *ACM Computing Surveys*, 54(6):130:1–130:37, July 2021.

[97] Hussin J Hejase, Hasan F Fayyad-Kazan, and Imad Moukadem. Advanced persistent threats (apt): An awareness review. *Journal of Economics and Economic Education Research*, 21(6):1–8, 2020.

[98] Laurens Hellemons, Luuk Hendriks, Rick Hofstede, Anna Sperotto, Ramin Sadre, and Aiko Pras. SSHCure: A Flow-Based SSH Intrusion Detection System. In Ramin Sadre, Jiří Novotný, Pavel Čeleda, Martin Waldburger, and Burkhard Stiller, editors, *Dependable Networks and Services*, Lecture Notes in Computer Science, pages 86–97, Berlin, Heidelberg, 2012. Springer.

[99] Hanan Hindy, David Brosset, Ethan Bayne, Amar Seeam, Christos Tachtatzis, Robert Atkinson, and Xavier Bellekens. A taxonomy and survey of intrusion detection system design techniques, network threats and datasets, June 2018. Num Pages: 35 Place: Ithaca, N.Y. Publisher: arXiv.org.

[100] Daniel Hintze, Eckhard Koch, Sebastian Scholz, and René Mayrhofer. Location-based Risk Assessment for Mobile Authentication. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*, UbiComp '16, pages 85–88, New York, NY, USA, 2016. ACM. event-place: Heidelberg, Germany.

[101] Elike Hodo, Xavier Bellekens, Andrew Hamilton, Pierre-Louis Dubouilh, Ephraim Iorkyase, Christos Tachtatzis, and Robert Atkinson. Threat analysis of IoT networks using artificial neural network intrusion detection system. In *Int. Symp. on Networks, Comp. and Comm.*, pages 1–6, Yasmine Hammamet, TUN, May 2016.

[102] Gregor Hollmig, Matthias Horne, Simon Leimkühler, Frederik Schöll, Carsten Strunk, Adrian Englhardt, Pavel Efros, Erik Buchmann, and Klemens Böhm. An evaluation of combinations of lossy compression and change-detection approaches for time-series data. *Information Systems*, 65:65–77, April 2017.

[103] En-Yu Hsu, Chien-Liang Liu, and Vincent S. Tseng. Multivariate Time Series Early Classification with Interpretability Using Deep Learning and Attention Mechanism. In Qiang Yang, Zhi-Hua Zhou, Zhiguo Gong, Min-Ling Zhang, and Sheng-Jun Huang, editors, *Advances in Knowledge Discovery and Data Mining*, Lecture Notes in Computer Science, pages 541–553, Cham, 2019. Springer International Publishing.

[104] Vincent C. Hu, D. Richard Kuhn, David F. Ferraiolo, and Jeffrey Voas. Attribute-Based Access Control. *Computer*, 48(2):85–88, February 2015. Conference Name: Computer.

[105] Yanpei Hua. An Efficient Traffic Classification Scheme Using Embedded Feature Selection and LightGBM. In *2020 Information Communication Technologies Conference (ICTC)*, pages 125–130, May 2020.

[106] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Ian Simon, Curtis Hawthorne, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck. Music Transformer, December 2018. arXiv:1809.04281 [cs, eess, stat].

[107] Huai-Shuo Huang, Chien-Liang Liu, and Vincent S. Tseng. Multivariate Time Series Early Classification Using Multi-Domain Deep Neural Network. In *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 90–98, October 2018.

[108] R. J. Hulsebosch, A. H. Salden, M. S. Bargh, P. W. G. Ebben, and J. Reitsma. Context sensitive access control. In *Proceedings of the tenth ACM symposium on Access control models and technologies*, SACMAT '05, pages 111–119, Stockholm, Sweden, June 2005. Association for Computing Machinery.

[109] Khalid Hussain, Syed Jawad Hussain, NZ Jhanjhi, and Mamoona Humayun. SYN Flood Attack Detection based on Bayes Estimator (SFADBE) For MANET. In *Int. Conf. on Computer and Information Sci. (ICCIS)*, pages 1–4, April 2019.

[110] P Illavarason and B Kamachi Sundaram. A Study of Intrusion Detection System using Machine Learning Classification Algorithm based on different feature selection approach. In *2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, pages 295–299, December 2019.

[111] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33(4):917–963, July 2019.

[112] Mohammad S. Jalali, Sabina Razak, William Gordon, Eric Perakslis, and Stuart Madnick. Health Care and Cybersecurity: Bibliometric Analysis of the Literature. *Journal of Medical Internet Research*, 21(2):e12644, 2019.

[113] Gareth James and Trevor Hastie. Generalizations of the bias/variance decomposition for prediction error. *Dept. Statistics, Stanford Univ., Stanford, CA, Tech. Rep*, 1997.

[114] Sian Jin, Sheng Di, Xin Liang, Jiannan Tian, Dingwen Tao, and Franck Cappello. DeepSZ: A Novel Framework to Compress Deep Neural Networks by Using Error-Bounded Lossy Compression. In *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing*, HPDC '19, pages 159–170, New York, NY, USA, 2019. Association for Computing Machinery.

[115] Roxanne Joncas. MQTT and CoAP, IoT Protocols | The Eclipse Foundation.

[116] Shijoe Jose, D. Malathi, Bharath Reddy, and Dorathi Jayaseeli. A Survey on Anomaly Based Host Intrusion Detection System. *Journal of Physics: Conference Series*, 1000:012049, April 2018. Publisher: IOP Publishing.

[117] Kamaldeep, M. Malik, and M. Dutta. Contiki-based mitigation of UDP flooding attacks in the Internet of things. In *Int. Conf. on Comp., Comm. and Automation*, pages 1296–1300, May 2017.

[118] Gozde Karatas, Onder Demir, and Ozgur Koray Sahingoz. Increasing the Performance of Machine Learning-Based IDSs on an Imbalanced and Up-to-Date Dataset. *IEEE Access*, 8:32150–32162, 2020. Conference Name: IEEE Access.

[119] Alexander D. Kent. Cyber security data sources for dynamic network research. In *Dynamic Networks and Cyber-Security*, volume Volume 1 of *Security Science and Technology*, pages 37–65. WORLD SCIENTIFIC (EUROPE), December 2015.

[120] Hemanth Khambhammettu, Sofiene Boulares, Kamel Adi, and Luigi Logrippo. A framework for risk assessment in access control systems. *Computers & Security*, 39:86 – 103, 2013. 27th IFIP International Information Security Conference.

[121] Nattawat Khamphakdee, Nunnapus Benjamas, and Saiyan Saiyod. Improving Intrusion Detection System based on Snort rules for network probe attack detection. In *Int. Conf. on Information and Communication Technology*, pages 69–74, May 2014.

[122] Hassan Khan, Urs Hengartner, and Daniel Vogel. Usability and security perceptions of implicit authentication: convenient, secure, sometimes annoying. In *Proceedings of the Eleventh USENIX Conference on Usable Privacy and Security*, SOUPS '15, pages 225–239, Ottawa, Canada, July 2015. USENIX Association.

[123] M. Fahim Ferdous Khan and Ken Sakamura. Fine-grained access control to medical records in digital healthcare enterprises. In *2015 International Symposium on Networks, Computers and Communications (ISNCC)*, pages 1–6, May 2015.

[124] M. Fahim Ferdous Khan and Ken Sakamura. Fine-grained access control to medical records in digital healthcare enterprises. In *2015 International Symposium on Networks, Computers and Communications (ISNCC)*, pages 1–6, May 2015.

[125] Minhaj Ahmad Khan and Khaled Salah. IoT security: Review, blockchain solutions, and open challenges. *Future Gen. Comp. Sys.*, 82:395–411, May 2018.

[126] K. C. Khor, C. Y. Ting, and S. Phon-Amnuaisuk. Comparing single and multiple bayesian classifiers approaches for network intrusion detection. In *Int. Conf. on Computer Engineering and Applications*, volume 2, pages 325–329, 2010.

[127] Jackson A Killian, Kevin M Passino, Arnab Nandi, Danielle R Madden, and John Clapp. Learning to Detect Heavy Drinking Episodes Using Smartphone Accelerometer Data. page 8.

[128] Huy Kang Kim. IoT network intrusion dataset. September 2019. type: dataset.

[129] SuHyun Kim and ImYeong Lee. IoT device security based on proxy re-encryption. *Journal of Ambient Intelligence and Humanized Computing*, 9(4):1267–1273, August 2018.

[130] Nickolaos Koroniotis, Nour Moustafa, Elena Sitnikova, and Benjamin Turnbull. Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset. *Future Gen. Comp. Sys.*, 100:779–796, November 2019.

[131] Anna Magdalena Kosek. Contextual anomaly detection for cyber-physical security in Smart Grids based on an artificial neural network model. In *2016 Joint Workshop on Cyber- Physical Security and Resilience in Smart Grids (CPSR-SG)*, pages 1–6, April 2016. ISSN: null.

[132] Devdatta Kulkarni and Anand Tripathi. Context-aware role-based access control in pervasive computing systems. In *Proceedings of the 13th ACM symposium on Access control models and technologies*, SACMAT '08, pages 113–122, Estes Park, CO, USA, June 2008. Association for Computing Machinery.

[133] Sriram Lakshminarasimhan, Neil Shah, Stephane Ethier, Seung-Hoe Ku, C. S. Chang, Scott Klasky, Rob Latham, Rob Ross, and Nagiza F. Samatova. ISABELA for effective in situ compression of scientific data: ISABELA FOR EFFECTIVE *IN-SITU* REDUCTION OF SPATIO-TEMPORAL DATA. *Concurrency and Computation: Practice and Experience*, 25(4):524–540, February 2013.

[134] Arash Habibi Lashkari, Gerard Draper Gil, Mohammad Saiful Islam Mamun, and Ali A. Ghorbani. Characterization of Tor Traffic using Time based Features. pages 253–262, December 2021.

[135] Giuseppe Laurenza, Riccardo Lazzeretti, and Luca Mazzotti. Malware triage for early identification of Advanced Persistent Threat activities. *arXiv:1810.07321 [cs]*, October 2018. arXiv: 1810.07321.

[136] E Jebamalar Leavline and DAAG Singh. Hardware implementation of lzma data compression algorithm. *International Journal of Applied Information Systems (IJAIS)*, 5(4):51–56, 2013.

[137] Chung-Hong Lee. Unsupervised and supervised learning to evaluate event relatedness based on content mining from social-media streams. *Expert Systems with Applications*, 39(18):13338–13356, December 2012.

[138] Wei-Han Lee and Ruby B. Lee. Implicit Smartphone User Authentication with Sensors and Contextual Machine Learning. In *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 297–308, June 2017. ISSN: 2158-3927.

[139] Wei-Han Lee and Ruby B. Lee. Implicit Smartphone User Authentication with Sensors and Contextual Machine Learning. In *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 297–308, June 2017. ISSN: 2158-3927.

[140] Joffrey L. Leevy and Taghi M. Khoshgoftaar. A survey and analysis of intrusion detection models based on CSE-CIC-IDS2018 Big Data. *Journal of Big Data*, 7(1):104, November 2020.

[141] Fagen Li, Yanan Han, and Chunhua Jin. Practical access control for sensor networks in the context of the Internet of Things. *Computer Communications*, 89-90:154–164, September 2016.

[142] Meicong Li, Wei Huang, Yongbin Wang, Wenqing Fan, and Jianfang Li. The study of apt attack stage model. In *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, pages 1–5, 2016.

[143] Xinghua Li, Mengyao Zhu, Laurence T. Yang, Mengfan Xu, Zhuo Ma, Cheng Zhong, Hui Li, and Yang Xiang. Sustainable Ensemble Learning Driving Intrusion Detection Model. *IEEE Transactions on Dependable and Secure Computing*, 18(4):1591–1604, July 2021. Conference Name: IEEE Transactions on Dependable and Secure Computing.

[144] Xin Liang, Sheng Di, Dingwen Tao, Sihuan Li, Shaomeng Li, Hanqi Guo, Zizhong Chen, and Franck Cappello. Error-Controlled Lossy Compression Optimized for High Compression Ratios of Scientific Datasets. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 438–447, 2018.

[145] Andy Liaw and Matthew Wiener. Classification and Regression by randomForest. 2:6, 2002.

[146] Joao Carlos D. Lima, Cristiano C. Rocha, Iara Augustin, and M´rio A.R. Dantas. A Context-Aware Recommendation System to Behavioral Based Authentication in Mobile and Pervasive Environments. In *2011 IFIP 9th International Conference on Embedded and Ubiquitous Computing*, pages 312–319, October 2011. ISSN: null.

[147] Francisco Sales de Lima Filho, Frederico A. F. Silveira, Agostinho de Medeiros Brito Junior, Genoveva Vargas-Solar, and Luiz F. Silveira. Smart Detection: An Online Approach for DoS/DDoS Attack Detection Using Machine Learning. *Security and Communication Networks*, 2019:e1574749, October 2019. Publisher: Hindawi.

[148] Yu-Feng Lin, Hsuan-Hsu Chen, Vincent S. Tseng, and Jian Pei. Reliable Early Classification on Multivariate Time Series with Numerical and Categorical Attributes. In Tru Cao, Ee-Peng Lim, Zhi-Hua Zhou, Tu-Bao Ho, David Cheung, and Hiroshi Motoda, editors, *Advances in Knowledge Discovery and Data Mining*, Lecture Notes in Computer Science, pages 199–211, Cham, 2015. Springer International Publishing.

[149] Peter Lindstrom. Fixed-Rate Compressed Floating-Point Arrays. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2674–2683, 2014. Conference Name: IEEE Transactions on Visualization and Computer Graphics.

[150] R.P. Lippmann, D.J. Fried, I. Graf, J.W. Haines, K.R. Kendall, D. McClung, D. Weber, S.E. Webster, D. Wyschogrod, R.K. Cunningham, and M.A. Zissman. Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation. In *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00*, volume 2, pages 12–26, Hilton Head, SC, USA, 1999. IEEE Comput. Soc.

[151] Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1):503–528, 1989.

[152] Jinxin Liu, Michele Nogueira, Johan Fernandes, and Burak Kantarci. Adversarial machine learning: A multilayer review of the state-of-the-art and challenges for wireless and mobile systems. *IEEE Communications Surveys Tutorials*, 24(1):123–159, 2022.

[153] Jinxin Liu, Murat Simsek, Burak Kantarci, Mehran Bagheri, and Petar Djukic. Collaborative Feature Maps of Networks and Hosts for AI-driven Intrusion Detection, August 2022. arXiv:2208.05085 [cs].

[154] Judith Liu-Jimenez, Raul Sanchez-Reillo, and Belen Fernandez-Saavedra. Iris Biometrics for Embedded Systems. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 19(2):274–282, February 2011.

[155] N. Lower and F. Zhan. A study of ensemble methods for cyber security. In *Computing and Communication Workshop and Conf.*, pages 1001–1009, 2020.

[156] Jiazhong Lu, Fengmao Lv, Zhongliu Zhuo, Xiaosong Zhang, Xiaolei Liu, Teng Hu, and Wei Deng. Integrating Traffics with Network Device Logs for Anomaly Detection. *Security and Communication Networks*, 2019:e5695021, June 2019. Publisher: Hindawi.

[157] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[158] Naila Marir, Huiqiang Wang, Guangsheng Feng, Bingyang Li, and Meijuan Jia. Distributed Abnormal Behavior Detection Approach Based on Deep Belief Network and Ensemble SVM Using Spark. *IEEE Access*, 6:59657–59671, 2018. Conference Name: IEEE Access.

[159] Pierre-Francois Marteau. Random Partitioning Forest for Point-Wise and Collective Anomaly Detection—Application to Network Intrusion Detection. *IEEE Transactions on Information Forensics and Security*, 16:2157–2172, 2021. Conference Name: IEEE Transactions on Information Forensics and Security.

[160] Guy Martin, Paul Martin, Chris Hankin, Ara Darzi, and James Kinross. Cybersecurity and healthcare: how safe are we? *BMJ*, 358, July 2017.

[161] Francesca Meneghello, Matteo Calore, Daniel Zucchetto, Michele Polese, and Andrea Zanella. IoT: Internet of Threats? A Survey of Practical Security Vulnerabilities in Real IoT Devices. *IEEE Internet of Things Journal*, 6(5):8182–8201, October 2019.

[162] Weizhi Meng, Elmar Wolfgang Tischhauser, Qingju Wang, Yu Wang, and Jinguang Han. When Intrusion Detection Meets Blockchain Technology: A Review. *IEEE Access*, 6:10179–10188, 2018. Conference Name: IEEE Access.

[163] Weizhi Meng, Duncan S. Wong, Steven Furnell, and Jianying Zhou. Surveying the Development of Biometric User Authentication on Mobile Phones. *IEEE Communications Surveys Tutorials*, 17(3):1268–1293, 2015.

[164] Erxue Min, Jun Long, Qiang Liu, Jianjing Cui, Zhiping Cai, and Junbo Ma. SU-IDS: A Semi-supervised and Unsupervised Framework for Network Intrusion Detection. In Xingming Sun, Zhaoqing Pan, and Elisa Bertino, editors, *Cloud Computing and Security*, Lecture Notes in Computer Science, pages 322–334, Cham, 2018. Springer International Publishing.

[165] Syed Misbahuddin, Junaid Ahmed Zubairi, Abdulrahman Saggaf, Jihad Basuni, Sulaiman A-Wadany, and Ahmed Al-Sofi. IoT based dynamic road traffic management

for smart cities. In *2015 12th International Conference on High-capacity Optical Networks and Enabling/Emerging Technologies (HONET)*, pages 1–5, December 2015.

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

[167] Sowmya Myneni, Ankur Chowdhary, Abdulhakim Sabur, Sailik Sengupta, Garima Agrawal, Dijiang Huang, and Myong Kang. DAPT 2020 - Constructing a Benchmark Dataset for Advanced Persistent Threats. In Gang Wang, Arridhana Ciptadi, and Ali Ahmadzadeh, editors, *Deployable Machine Learning for Security Defense*, volume 1271, pages 138–163. Springer International Publishing, Cham, 2020. Series Title: Communications in Computer and Information Science.

[168] Ali Nauman, Yazdan Ahmad Qadri, Muhammad Amjad, Yousaf Bin Zikria, Muhammad Khalil Afzal, and Sung Won Kim. Multimedia Internet of Things: A Comprehensive Survey. *IEEE Access*, 8:8202–8250, 2020. Conference Name: IEEE Access.

[169] Sasho Nedelkoski, Jasmin Bogatinovski, Alexander Acker, Jorge Cardoso, and Odej Kao. Self-Attentive Classification-Based Anomaly Detection in Unstructured Logs. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 1196–1201, November 2020. ISSN: 2374-8486.

[170] T. D. Nguyen, J. Y. Khan, and D. T. Ngo. A distributed energy-harvesting-aware routing algorithm for heterogeneous iot networks. *IEEE Transactions on Green Communications and Networking*, 2(4):1115–1127, 2018.

[171] Adetunmbi A Olusola, Adeola S Oladele, and Daramola O Abosede. Analysis of kdd'99 intrusion detection dataset for selection of relevance features. In *Proceedings of the world congress on engineering and computer science*, volume 1, pages 20–22. WCECS, 2010.

[172] Marc-Oliver Pahl and François-Xavier Aubet. All Eyes on You: Distributed Multi-Dimensional IoT Microservice Anomaly Detection. In *14th International Conf. on Network and Service Management (CNSM)*, pages 72–80, November 2018. ISSN: 2165-9605.

[173] Hamed Haddad Pajouh, Reza Javidan, Raouf Khayami, Ali Dehghantanha, and Kim-Kwang Raymond Choo. A Two-Layer Dimension Reduction and Two-Tier Classification Model for Anomaly-Based Intrusion Detection in IoT Backbone Networks. *IEEE Trans. on Emerging Topics in Computing*, 7:314–323, April 2019.

[174] Sung Bum Pan, Daesung Moon, Younhee Gil, Dosung Ahn, and Yongwha Chung. An ultra-low memory fingerprint matching algorithm and its implementation on a 32-bit smart card. *IEEE Transactions on Consumer Electronics*, 49(2):453–459, May 2003.

[175] Vishal M. Patel, Rama Chellappa, Deepak Chandra, and Brandon Barbello. Continuous User Authentication on Mobile Devices: Recent progress and remaining challenges. *IEEE Signal Processing Magazine*, 33(4):49–61, July 2016.

[176] Al-Sakib Khan Pathan. *Security of Self-Organizing Networks: MANET, WSN, WMN, VANET*. CRC Press, April 2016. Google-Books-ID: ZtBnZoijaDcC.

[177] Abdurrahman Pektaş and Tankut Acarman. A deep learning method to detect network intrusion through flow-based features. *International Journal of Network Management*, 29(3):e2050, 2019. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nem.2050.

[178] Giuseppe Petracca, Frank Capobianco, Christian Skalka, and Trent Jaeger. On risk in access control enforcement. In *Proceedings of the 22nd ACM on Symposium on Access Control Models and Technologies*, SACMAT '17 Abstracts, page 31–42, New York, NY, USA, 2017. Association for Computing Machinery.

[179] Leonid Portnoy, Eleazar Eskin, and Sal Stolfo. Intrusion Detection with Unlabeled Data Using Clustering. In *In Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001*, pages 5–8, 2001.

[180] Supriyanto Praptodiyono, Moh. Jauhari, Rian Fahrizal, Iznan H. Hasbullah, Azlan Osman, and Shafiq Ul Rehman. Integration of firewall and ids on securing mobile ipv6. In *2020 2nd International Conference on Industrial Electrical and Electronics (ICIEE)*, pages 163–168, 2020.

[181] Abena Primo, Vir V. Phoha, Rajesh Kumar, and Abdul Serwadda. Context-Aware Active Authentication Using Smartphone Accelerometer Measurements. pages 98–105, 2014.

[182] Kun Qian, Maik Schott, Wenju Zheng, and Jana Dittmann. Context-based approach of separating contactless captured high-resolution overlapped latent fingerprints. *IET Biometrics*, 3(2):101–112, June 2014.

[183] Kyle Quintal, Burak Kantarci, Melike Erol-Kantarci, Andrew Malton, and Andrew Walenstein. Contextual, Behavioral and Biometric Signatures for Continuous Authentication. *IEEE Internet Computing*, pages 1–1, 2019.

[184] Nadun Rajasinghe, Jagath Samarabandu, and Xianbin Wang. INSECS-DCS: A Highly Customizable Network Intrusion Dataset Creation Framework. In *IEEE CCECE,*, pages 1–4, May 2018.

[185] K. I. Ramatsakane and W. S. Leung. Pick location security: Seamless integrated multi-factor authentication. In *2IST-Africa Week Conference (IST-Africa)*, pages 1–10, May 2017.

[186] Paulo E Rauber, Alexandre X Falcão, and Alexandru C Telea. Visualizing Time-Dependent Data Using Dynamic t-SNE. page 5, 2016.

[187] Zachary I. Rauen, Fazel Anjomshoa, and Burak Kantarci. Gesture and Sociability-based Continuous Authentication on Smart Mobile Devices. In *Proceedings of the 16th ACM International Symposium on Mobility Management and Wireless Access*, MobiWac'18, pages 51–58, New York, NY, USA, 2018. ACM. event-place: Montreal, QC, Canada.

[188] Attila Reiss, Ina Indlekofer, Philip Schmidt, and Kristof Van Laerhoven. Deep PPG: Large-Scale Heart Rate Estimation with Convolutional Neural Networks. *Sensors*, 19(14):3079, January 2019. Number: 14 Publisher: Multidisciplinary Digital Publishing Institute.

[189] Paulo Angelo Alves Resende and André Costa Drummond. Adaptive anomaly-based intrusion detection system using genetic algorithm and profiling. *SECURITY AND PRIVACY*, 1(4):e36, 2018.

[190] D. A. Reynolds. A Gaussian mixture modeling approach to text-independent speaker identification. page 1, 1993.

[191] Markus Ring, Sarah Wunderlich, Deniz Scheuring, Dieter Landes, and Andreas Hotho. A Survey of Network-based Intrusion Detection Data Sets. *Computers & Security*, 86:147–167, September 2019. arXiv: 1903.02460.

[192] Martin Ringwelski, Christian Renner, Andreas Reinhardt, Andreas Weigel, and Volker Turau. The hitchhiker's guide to choosing the compression algorithm for your smart meter data. In *2012 IEEE International Energy Conference and Exhibition (ENERGYCON)*, pages 935–940. IEEE, 2012.

[193] Eyal Ronen, Adi Shamir, Achi-Or Weingarten, and Colin O'Flynn. IoT Goes Nuclear: Creating a ZigBee Chain Reaction. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 195–212, May 2017. ISSN: 2375-1207.

[194] Rongrong Fu, Kangfeng Zheng, Dongmei Zhang, and Yixian Yang. An intrusion detection scheme based on anomaly mining in Internet of Things. In *IET Int. Conf. on Wireless, Mobile & Multimedia Networks*, pages 315–320, 2011.

[195] Matthew Rossi, Dario Facchinetti, Enrico Bacis, Marco Rosa, and Stefano Paraboschi. Seapp: Bringing mandatory access control to android apps. August 2021.

[196] Olivér Rákos, Szilárd Aradi, Tamás Bécsi, and Zsolt Szalay. Compression of Vehicle Trajectories with a Variational Autoencoder. *Applied Sciences*, 10(19):6739, January 2020. Number: 19 Publisher: Multidisciplinary Digital Publishing Institute.

[197] Yalin E. Sagduyu, Yi Shi, and Tugba Erpek. IoT Network Security from the Perspective of Adversarial Deep Learning. In *IEEE Int. Conf. on Sensing, Commu., and Networking*, pages 1–9, June 2019.

[198] Gerry Saporito. A Deeper Dive into the NSL-KDD Data Set, November 2019. Library Catalog: towardsdatascience.com.

[199] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN. *ACM Transactions on Database Systems*, 42(3):19:1–19:21, July 2017.

[200] Mustapha Réda Senouci, Abdelhamid Mellouk, and Amar Aissani. Random deployment of wireless sensor networks: a survey and approach. *Int. J. Ad Hoc Ubiquitous Comp.*, 2014.

[201] Muhammad Shafiq, Zhihong Tian, Yanbin Sun, Xiaojiang Du, and Mohsen Guizani. Selection of effective machine learning algorithm and Bot-IoT attacks traffic identification for internet of things in smart city. *Future Generation Computer Systems*, 107:433–442, June 2020.

[202] Iman Sharafaldin., Arash Habibi Lashkari., and Ali A. Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy - ICISSP,*, pages 108–116. INSTICC, SciTePress, 2018.

[203] Nathan Shone, Tran Nguyen Ngoc, Vu Dinh Phai, and Qi Shi. A Deep Learning Approach to Network Intrusion Detection. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2(1):41–50, February 2018. Conference Name: IEEE Transactions on Emerging Topics in Computational Intelligence.

[204] Ravid Shwartz-Ziv and Amitai Armon. Tabular Data: Deep Learning is Not All You Need. *arXiv:2106.03253 [cs]*, November 2021. arXiv: 2106.03253.

[205] Nahian Siddique, Sidike Paheding, Colin P. Elkin, and Vijay Devabhaktuni. U-Net and Its Variants for Medical Image Segmentation: A Review of Theory and Applications. *IEEE Access*, 9:82031–82057, 2021. Conference Name: IEEE Access.

[206] Sana Siddiqui, Muhammad Salman Khan, Ken Ferens, and Witold Kinsner. Detecting Advanced Persistent Threats using Fractal Dimension based Machine Learning Classification. In *Proceedings of the 2016 ACM on International Workshop on Security And Privacy Analytics*, IWSPA '16, pages 64–69, New York, NY, USA, 2016. Association for Computing Machinery.

[207] Z. Sitová, J. Šeděnka, Q. Yang, G. Peng, G. Zhou, P. Gasti, and K. S. Balagani. Hmog: New behavioral biometric features for continuous authentication of smartphone users. *IEEE Transactions on Information Forensics and Security*, 11(5):877–892, May 2016.

[208] Jan Spooren, Davy Preuveneers, and Wouter Joosen. Mobile Device Fingerprinting Considered Harmful for Risk-based Authentication. In *Proceedings of the Eighth European Workshop on System Security*, EuroSec '15, pages 6:1–6:6, New York, NY, USA, 2015. ACM. event-place: Bordeaux, France.

[209] Allan Stisen, Henrik Blunck, Sourav Bhattacharya, Thor Siiger Prentow, Mikkel Baun Kjærgaard, Anind Dey, Tobias Sonne, and Mads Møller Jensen. Smart Devices are Different: Assessing and MitigatingMobile Sensing Heterogeneities for Activity Recognition. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, SenSys '15, pages 127–140, New York, NY, USA, 2015. Association for Computing Machinery.

[210] Maria Stoyanova, Yannis Nikoloudakis, Spyridon Panagiotakis, Evangelos Pallis, and Evangelos K. Markakis. A Survey on the Internet of Things (IoT) Forensics: Challenges, Approaches, and Open Issues. *IEEE Communications Surveys & Tutorials*, 22(2):1191–1221, 2020. Conference Name: IEEE Communications Surveys & Tutorials.

[211] Hudan Studiawan, Ferdous Sohel, and Christian Payne. Anomaly Detection in Operating System Logs with Deep Learning-Based Sentiment Analysis. *IEEE Transactions on Dependable and Secure Computing*, 18(5):2136–2148, September 2021. Conference Name: IEEE Transactions on Dependable and Secure Computing.

[212] Subrina Sultana, Sumaiya Nasrin, Farhana Kabir Lipi, Md Afzal Hossain, Zinia Sultana, and Fatima Jannat. Detecting and Preventing IP Spoofing and Local Area Network Denial (LAND) Attack for Cloud Computing with the Modification of Hop Count Filtering (HCF) Mechanism. In *Int. Conf. on Comp., Comm., Chemical, Materials and Electronic Eng. (IC4ME2)*, pages 1–6, July 2019.

[213] Jingwei Sun, Tao Yan, Hao Sun, Huancheng Lin, and Guangzhong Sun. Lossy Compression of Communication Traces Using Recurrent Neural Networks. *IEEE Transactions on Parallel and Distributed Systems*, 33(11):3106–3116, 2022. Conference Name: IEEE Transactions on Parallel and Distributed Systems.

[214] Syeda Manjia Tahsien, Hadis Karimipour, and Petros Spachos. Machine learning based solutions for security of internet of things (iot): A survey. *Journal of Network and Computer Applications*, 161:102630, 2020.

[215] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali Ghorbani. A detailed analysis of the KDD CUP 99 data set. *IEEE Symposium. Computational Intelligence for Security and Defense Applications, CISDA*, 2, July 2009.

[216] Shejin Thavalengal, Petronel Bigioi, and Peter Corcoran. Iris authentication in handheld devices - considerations for constraint-free acquisition. *IEEE Transactions on Consumer Electronics*, 61(2):245–253, May 2015.

[217] Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. Lossy Image Compression with Compressive Autoencoders, March 2017. arXiv:1703.00395 [cs, stat].

[218] Matt Tigner, Hayden Wimmer, and Carl M. Rebman. Analysis of Kali Linux Penetration Tools: A Survey of Hacking Tools. In *2021 International Conference on Electrical, Computer and Energy Technologies (ICECET)*, pages 1–6, 2021.

[219] Igor Tomičić, Petra Grd, and Miroslav Bača. A review of soft biometrics for IoT. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1115–1120, May 2018. ISSN: null.

[220] Michal Trnka and Tomas Cerny. On security level usage in context-aware role-based access control. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, SAC '16, pages 1192–1195, Pisa, Italy, April 2016. Association for Computing Machinery.

[221] Hien Thi Thu Truong, Xiang Gao, Babins Shrestha, Nitesh Saxena, N. Asokan, and Petteri Nurmi. Using contextual co-presence to strengthen Zero-Interaction Authentication: Design, integration and usability. *Pervasive and Mobile Computing*, 16:187–204, January 2015.

[222] Michael Tschannen, Olivier Bachem, and Mario Lucic. Recent Advances in Autoencoder-Based Representation Learning. Technical Report arXiv:1812.05069, arXiv, December 2018. arXiv:1812.05069 [cs, stat] type: article.

[223] Melissa J. M. Turcotte, Alexander D. Kent, and Curtis Hash. Unified Host and Network Data Set. *arXiv:1708.07518 [cs]*, August 2017. arXiv: 1708.07518.

[224] Melissa J. M. Turcotte, Alexander D. Kent, and Curtis Hash. Unified Host and Network Data Set. In *Data Science for Cyber-Security*, volume Volume 3 of *Security Science and Technology*, pages 1–22. WORLD SCIENTIFIC (EUROPE), March 2018.

[225] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc.

[226] A Velinov and A Mileva. Running and Testing Applications for Contiki OS Using Cooja Simulator. page 7, 2016.

[227] Rasmus Vestergaard, Qi Zhang, and Daniel E. Lucani. Lossless Compression of Time Series Data with Generalized Deduplication. In *2019 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, 2019. ISSN: 2576-6813.

[228] S. Vhaduri and C. Poellabauer. Multi-modal biometric-based implicit authentication of wearable device users. *IEEE Transactions on Information Forensics and Security*, 14(12):3116–3125, Dec 2019.

[229] R. Vinayakumar, Mamoun Alazab, K. P. Soman, Prabaharan Poornachandran, Ameer Al-Nemrat, and Sitalakshmi Venkatraman. Deep Learning Approach for Intelligent Intrusion Detection System. *IEEE Access*, 7:41525–41550, 2019. Conference Name: IEEE Access.

[230] Jinfa Wang, Hai Zhao, Jiuqiang Xu, Hequn Li, Hongsong Zhu, Shuai Chao, and Chunyang Zheng. Using Intuitionistic Fuzzy Set for Anomaly Detection of Network Traffic From Flow Interaction. *IEEE Access*, 6:64801–64816, 2018. Conference Name: IEEE Access.

[231] Qihua Wang and Hongxia Jin. Quantified risk-adaptive access control for patient privacy protection in health information systems. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '11, page 406–410, New York, NY, USA, 2011. Association for Computing Machinery.

[232] Wenhui Wang, Liandong Chen, Longxi Han, Zhihong Zhou, Zhengmin Xia, and Xiuzhen Chen. Vulnerability assessment for ics system based on zero-day attack graph. In *2020 International Conference on Intelligent Computing, Automation and Systems (ICICAS)*, pages 1–5. IEEE, 2020.

[233] Martin Wattenberg, Fernanda Viégas, and Ian Johnson. How to Use t-SNE Effectively. *Distill*, 1(10):e2, October 2016.

[234] George Edward Williams. Critical aperture convergence filtering and systems and methods thereof, July 2006.

[235] Timothy Wong and Zhiyuan Luo. Recurrent Auto-Encoder Model for Large-Scale Industrial Sensor Signal Analysis. In Elias Pimenidis and Chrisina Jayne, editors, *Engineering Applications of Neural Networks*, Communications in Computer and Information Science, pages 203–216, Cham, 2018. Springer International Publishing.

[236] Changsheng Wu, Wenbo Ding, Ruiyuan Liu, Jiyu Wang, Aurelia C. Wang, Jie Wang, Shengming Li, Yunlong Zi, and Zhong Lin Wang. Keystroke dynamics enabled authentication and identification using triboelectric nanogenerator array. *Materials Today*, 21(3):216–222, April 2018.

[237] Yang Xin, Lingshuang Kong, Zhi Liu, Yuling Chen, Yanmiao Li, Hongliang Zhu, Mingcheng Gao, Haixia Hou, and Chunhua Wang. Machine learning and deep learning methods for cybersecurity. *Ieee access*, 6:35365–35381, 2018.

[238] Congyuan Xu, Jizhong Shen, and Xin Du. A Method of Few-Shot Network Intrusion Detection Based on Meta-Learning Framework. *IEEE Transactions on Information Forensics and Security*, 15:3540–3552, 2020. Conference Name: IEEE Transactions on Information Forensics and Security.

[239] Xinghan Xu and Minoru Yoneda. Multitask Air-Quality Prediction Based on LSTM-Autoencoder Model. *IEEE Transactions on Cybernetics*, 51(5):2577–2586, 2021. Conference Name: IEEE Transactions on Cybernetics.

[240] Kai Yang, Jie Ren, Yanqiao Zhu, and Weiyi Zhang. Active Learning for Wireless IoT Intrusion Detection. *IEEE Wireless Communications*, 25(6):19–25, December 2018.

[241] Fekadu Yihunie, Eman Abdelfattah, and Ammar Odeh. Analysis of ping of death DoS and DDoS attacks. In *IEEE Long Island Sys., Applications and Technology Conf.*, pages 1–4, May 2018.

[242] Xinyang Yu, Yanqing Peng, Feifei Li, Sheng Wang, Xiaowei Shen, Huijun Mai, and Yue Xie. Two-Level Data Compression using Machine Learning in Time Series Database. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 1333–1344, April 2020. ISSN: 2375-026X.

[243] Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures. *Neural Computation*, 31(7):1235–1270, July 2019.

[244] Gholam Reza Zargar and Peyman Kabiri. Identification of effective network features to detect Smurf attacks. In *IEEE Student Conf. on Research and Development*, pages 49–52, November 2009.

[245] Congyingzi Zhang and Robert Green. Communication security in internet of thing: preventive measure and avoid DDoS attack over IoT network. In *18th Symposium on Communications & Networking*, pages 8–15, Alexandria, VA, April 2015.

[246] Harry Zhang. The Optimality of Naive Bayes. page 6.

[247] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. BIRCH: an efficient data clustering method for very large databases. *ACM SIGMOD Record*, 25(2):103–114, June 1996.

[248] Kai Zhao, Sheng Di, Maxim Dmitriev, Thierry-Laurent D. Tonellot, Zizhong Chen, and Franck Cappello. Optimizing Error-Bounded Lossy Compression for Scientific Data by Dynamic Spline Interpolation. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 1643–1654, April 2021. ISSN: 2375-026X.

[249] Kai Zhao, Sheng Di, Xin Liang, Sihuan Li, Dingwen Tao, Zizhong Chen, and Franck Cappello. Significantly Improving Lossy Compression for HPC Datasets with Second-Order Prediction and Parameter Optimization. In *Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing*, HPDC '20, pages 89–100, New York, NY, USA, 2020. Association for Computing Machinery.

[250] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. 2021.

[251] Aaron Zimba, Hongsong Chen, Zhaoshun Wang, and Mumbi Chishimba. Modeling and detection of the multi-stages of advanced persistent threats attacks based on semi-supervised learning and complex networks characteristics. *Future Generation Computer Systems*, 106:501–517, 2020.

[252] Yüksel Öner and Hasan Bulut. A robust EM clustering approach: ROBEM. *Communications in Statistics - Theory and Methods*, 0(0):1–19, February 2020. Publisher: Taylor & Francis.