# ADAPTIVE MACHINE LEARNING AND SIGNAL PROCESSING DETECTION SCHEMES FOR DDOS ATTACKS

*Maryam Ghanbari*

A Thesis submitted to the Faculty of Graduate Studies of

The University of Manitoba

in partial fulfilment of the requirements of the degree of

**Doctor of Philosophy**

Department of Electrical and Computer Engineering

University of Manitoba

Winnipeg

***In the Name of God***

"Wisdom is the supreme gift of God and God has opened the doors of art

to the righteous wise."

*(Abul-ghasim Ferdowsi)*

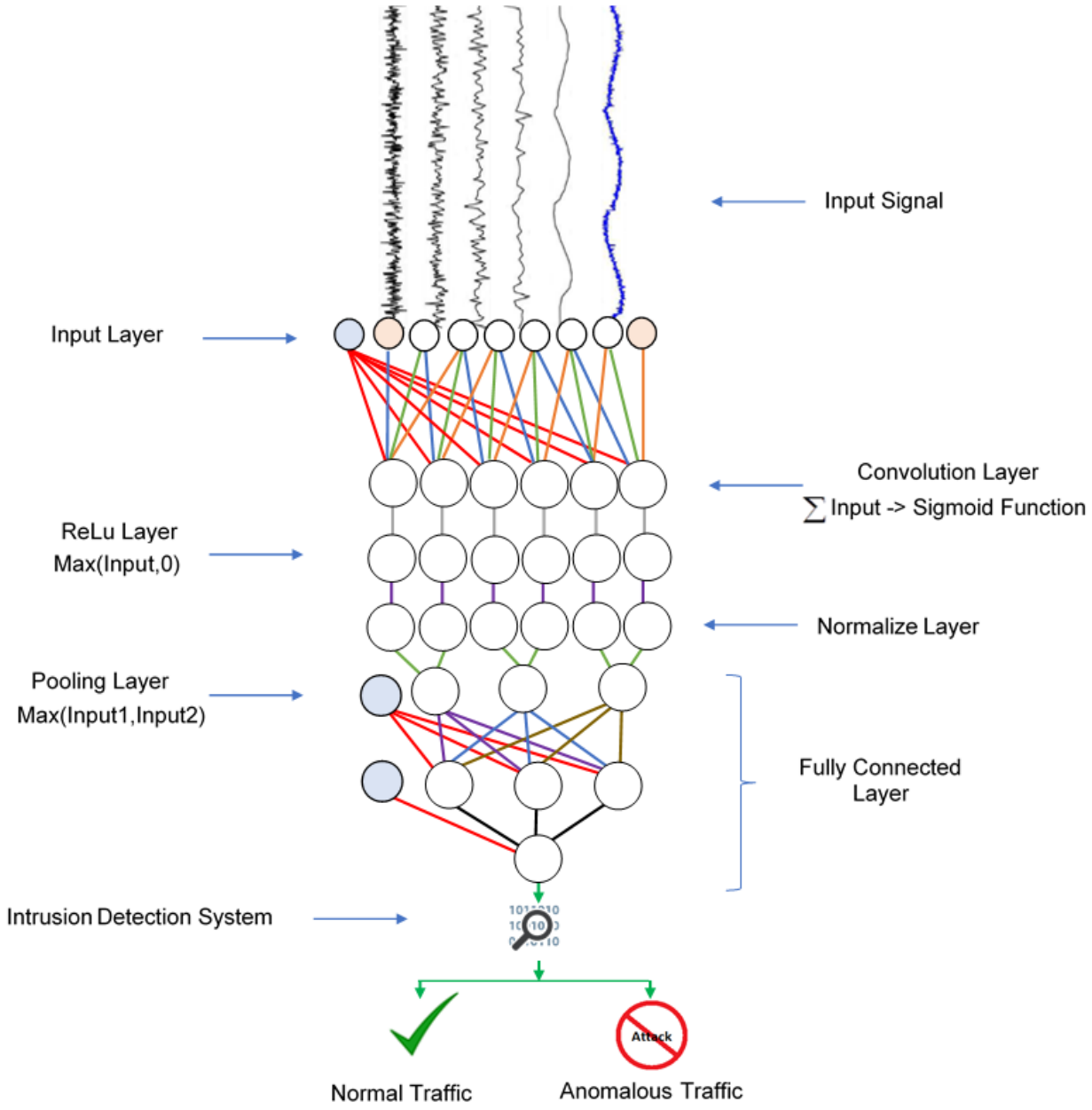
To my mother **Azam** for her love, understanding, sacrifice and awareness,

my father **Amir** for his love, gentility and training me how to present in an easy way

my sister **Shirin** for her support and kindness

and my advisor **Professor Kinsner** for his humanity and wisdom

# Visual Abstract



# Abstract

When cybersecurity flaws cause devices to be vulnerable to cyber-attacks, the functionality of these physical devices can be compromised which ultimately affects society. To prevent these cyber-physical-social security attacks cognitive informatics, and cognitive security systems can be used to design *intrusion detection system*s (IDSs). Using adaptive security systems, anomaly-based IDS methods are designed and implemented for reducing false positive and false negative detection and for increasing the detection speed of the proposed IDSs.

The proposed IDSs in the literature review are based on extracting patterns from static structures to improve the efficiency of anomaly detection in changing environments. In the first stage of this research, the feature extraction methods are designed as creating a mother wavelet and improving the VFD to enhance detection rate. The adaptive mother wavelet for a specific application, in this case DDoS attacks, is created to achieve the highest similarity and adaptability to the input data. Since an *Internet traffic data with distributed denial of service attacks* (DDoS ITD) is a long-range dependence signal, a multiscale analysis that measures a signal with various scales is created to extract the hidden characteristics of each scale of the DDoS ITD. As such, in this research, since IDSs that are based on fractal dimensions methods in the literature review are not online and are unable to properly select fractal dimensions methods to adapt to the input signal, this research uses and expands a method that works with online variance fractal dimension. This dimension is based on a polyscale analysis, which measures a signal with various scales and its outcome correlation requires all the scales to be used simultaneously. Therefore, the hidden features of the DDoS ITD are extracted in each scale, and the correlation between the hidden features are considered. As a result, the DDoS attacks can be detected with a higher detection rate

with the polyscale analysis. In the second stage of this research, an IDS based on a *convolutional neural network* (CNN) is developed to enhance the sensitivity of DDoS attacks detection. As such, in this research, a weighted cost function is developed for evaluating the artificial neural network and the CNN structure. Moreover, an adaptive structure for the CNN is designed and created. For designing the IDS, the weighted cost function and the adaptive CNN structure are applied. The IDS produces 95% accuracy of detection rate. In the third stage of this research, a realistic IDS for real world is designed and implemented because the real-world data is unlabeled, so supervised learning methods are not realistic. To obtain a more realistic IDS, the proposed structure of the *polyscale convolutional neural network* (PCNN) regarding the *policy gradient based deep reinforcement learning* (PGDRL) is used to design and implement the IDS for unlabeled data. Since the DRL does not use labels to train its input, it is based on reward maximization, so the system is suitable for real-world applications. This IDS detects the anomalies with 93% accuracy.

# Acknowledgements

I would like to thank God for being my greatest hope and helper in every moment of my life and giving me the strength to achieve my goals.

I would like to declare my gratitude to Dr. Witold Kinsner for accepting me as his Ph.D. student and being my thesis advisor while providing me the freedom to select my path to do my PhD research. Furthermore, thank Dr. Witold Kinsner for challenging me during my PhD study and always patiently support me flourish in different aspects. Thank you very much that you believe in your students. In addition, I would like to thank him for helping me use my capabilities to accomplish this significant achievement. Words can't express how much I appreciate your support and encouragement in different activities to do more than just the thesis, such as volunteering in Verna Kirkness Program, volunteering in UMSATS, applying for Amateur Radio Operator Certificate, and conference and journal publications. Without exaggerations, I would like to state that I am very proud to have such a wonderful, wise, knowledgeable, kind, patient, humane, compassionate and supportive advisor. Your encouragement and support have been invaluable.

I am also very grateful to Dr. Ken Ferens for agreeing to be a committee member for my thesis, and for his excellent teaching, his comments and his support. Dr. Ken Ferens took significant steps in doing my PhD candidacy exam, vital courses for my research, and his valuable comments. I learned a lot from him, especially in Computer Network Security and Applied Computational Intelligence courses. Thank you for encouraging me to publish my first publication in my PhD study. It was not just a published paper, and it was a hope for a new horizon of my life. In addition, I would like to thank Dr. Nariman Sepehri for agreeing to be on the committee and for his valuable comments. Moreover, thank Dr. Nariman Sepehri for encouraging me to engage in IEEE activities

and his support. My thanks also go to Dr. Pourang Irani for agreeing to be a committee member for this thesis and for his valuable comments. In addition, I would like to thank Dr. George Baciu for agreeing to be an external committee member and for his valuable comments. Moreover, I would like to thank Dr. Ahmed Shalaby for agreeing to be the chair of my PhD oral examination. I am also very grateful to Dr. Miroslaw Pawlak, Dr. Gabriel Thomas, Ms. Antoanela Denchuk and Academic Learning Centre for their unconditional support at the University of Manitoba. The world would be a better place if there were more professors like you. Thank you for your encouragement and support.

I am also very grateful to the National Science Foundation and CAIDA Members for their assistance in acquiring CAIDA's Internet Traces provided by the US Department of Homeland Security.

In addition, I want to thank my colleagues, especially the members of the Delta Research Group, for creating a friendly and joyful environment in which to work and their useful discussions. I am grateful for the encouragement and the comments you gave me when I needed it.

These achievements could not have been possible without the love, constant support, help, and encouragement of my mother, who encourages me to be brave to fulfill my wishes and supports me to follow my dreams. Also, my father inspires me to make everything easy to learn, and my sister as my best friend helps me achieve my dreams. I thank my family for putting their faith in me over the last several years. I would not have been where I am today without your support and encouragement.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF SYMBOLS

| | |
|---|---|
| $\gamma$ | The discrete wavelet coefficient |
| $wt(a,b)$ | The discrete wavelet coefficient |
| * | The complex conjugate |
| $\psi^*$ | The complex conjugate of the scaled and shifted wavelet function. |
| $\psi(t)$ | Mother wavelet function (discrete basis function) |
| $\phi(t)$ | The scaling function |
| $\Psi(f)$ | The Fourier transform of the mother wavelet function |
| $M_k$ | The mother wavelet's vanishing moments |
| $h(n)$ | A low-pass filter |
| $g(n)$ | A high-pass filter |
| $\delta(k)$ | The delta function |
| $\|\bullet\|$ | The norm function |
| $H^*$ and $G^*$ | A pair of reconstruction filters |
| $D_\sigma$ | Variance fractal dimension |
| $E$ | Euclidean dimension |
| $H$ | The Hurst exponent |
| $N_T$ | The number of samples in each frame |
| $\delta_n$ | An interval between two successive sampled data |

| | |
|---|---|
| $\Delta t_{k_{\max}}$ | The maximum *volume element* (vel) size |
| $k_{low}$ | The lower bound (A parameter for the VFDv2 algorithm) |
| $k_{hi}$ | The upper bound (A parameter for the VFDv2 algorithm) |
| $N_K$ | The number of vels in the frame |
| $n_k$ | The size of vels in the frame |
| $\Delta A$ | The amplitude increment |
| $S$ | The slope of the log-log plot |
| $\beta$ | The power spectrum exponent |
| $D_\beta$ | The spectral fractal dimension |
| $w[]$ | A window containing a stationary segment of a signal |
| $tr_i$ | The turn occurring at a time interval |
| $TC$ | The turns count |
| $y_i$ | The desired output of the $i^{th}$ input |
| $\hat{y}_i$ | The actual output (the forward propagation output) of the neural network |
| $w$ | Vector of coefficients in SVM |
| $\theta_{ij}^{(k)}$ | Weights of ANN, CNN and PCNN |
| $w_j$ | Weights of SVM, CNN |
| $b$ | Bias of ANN and CNN |
| $g$ | sigmoid function |

| | |
|---|---|
| $C$ | Cost function of SVM |
| $J(\theta)$ | Cost function |
| ln | Logarithm with the base equals to the Euler's number |
| $\gamma$ | Discount factor in DRL |
| $\pi$ | Policy in DRL |
| $G_t$ | The discounted future return |
| $V_t$ | The value function |
| $T_E$ | A complete dataset is considered an epoch |
| $s$ | Scaling parameter |
| $\tau$ | Shifting parameter |
| $\mu$ | Location parameter in a levy distribution |
| $\sigma$ | Scale parameter in a levy distribution |
| $D_{\sigma i}$ | Local VFD |
| $\alpha$ | Learning rate in ANN and CNN |

# LIST OF ABBREVIATIONS

| | |
|---|---|
| adaptive-wavelet CNN | Adaptive-wavelet convolutional neural network |
| ANN | Artificial neural network |
| ATS | Packet arrival time-series signal |
| AMI | Advanced metering infrastructure |
| BDI attack | Bad data injection attack |
| BFD | Backward finite differences |
| CAIDA | Center for Applied Internet Data Analysis |
| CNN | Convolutional neural network |
| CFD | Central finite differences |
| CWT | Continuous wavelet transform |
| DARPA | Defense Advanced Research Projects Agency |
| DNS | Domain name system |
| db | Daubechies |
| db2 (D2) | Daubechies wavelet transform 2 |
| db4 (D4) | Daubechies wavelet transform 4 |
| DWT | Discrete wavelet transform |
| DoS attack | Denial of service attack |
| DDoS attack | Distributed denial of service attack |
| DDoS ITD | Internet traffic data with distributed denial of service attacks |
| DRL | Deep reinforcement learning |
| ECG | Electrocardiography |

| | |
|---|---|
| FFD | forward finite differences |
| FT | Fourier transform |
| GA | Genetic algorithm |
| GNN | Genetic neural network |
| H | The Hurst exponent |
| HAN | Home area networks |
| ITD | Internet traffic data |
| IDS | Intrusion detection system |
| IDSA | Intrusion detection system agent |
| DDoS ITD | Internet traffic data with distributed denial of service attacks |
| MC | Monte Carlo |
| MDP | Markov decision process |
| MRA | Multi-resolution analysis |
| NAN | Neighborhood area networks |
| KL-divergence | Kullback–Leibler divergence |
| MOEA | Multi-objective optimization using an evolutionary algorithm |
| MRA | Multi-resolution analysis |
| ms | Millisecond |
| NSGA II | Non-dominated Sorting Genetic Algorithm |
| PCA | Principal component analysis |
| PCNN | Polyscale convolutional neural network |
| PGDRL | Policy gradient based deep reinforcement learning |
| PMF | Probability mass function |
| PSD | Power spectrum density |

| | |
|---|---|
| ReLU layer | Rectified linear unit layer |
| SFD | The spectral fractal dimension |
| STFT | short term (windowed) Fourier transform |
| SVM | Support vector machine |
| TD | Temporal Difference |
| TC | Turn counting |
| vel | Volume element |
| VFD | Variance fractal dimension |
| VFDT | Variance fractal dimension trajectory |
| VFDTv2 | Improved version of the variance fractal dimension trajectory |
| WAN | Consists of wide area networks |
| WNN | Wavelet neural network |
| WT | Wavelet transform |
| ZC | Zero crossing |

# LIST OF DEFINITIONS

| Keyword | Definition | Def Page |
|---|---|---|
| **Adaptive System** | An *adaptive system* is a system that changes its behaviour in response to its environment in a way similar to evolutionary adaptation. The adaptive change that occurs is often relevant to achieving a goal. | 4 |
| **ANN** | An *artificial neural network* (ANN) is a learning model and efficient classifier, which has been influenced by human learning. | 48 |
| **CNN** | A c*onvolutional neural network* (CNN) is very similar to ordinary artificial neural networks. They are made up of neurons, non-learning pathways, and learning pathways which are called weights and biases. | 50 |
| **DRL** | The *Deep Reinforcement Learning* (DRL) is an area of machine learning. It is a computational approach in which machines learn from interacting with the environment using unlabeled data. The DRL purpose is to learn how to take an action to maximize the long-term rewards. | 55 |
| **Fractal Dimension** | A *fractal dimension* is calculated to analyze and measure a fractal signal complexity. A fractal dimension interprets degree of meandering of a fractal signal. Moreover, there are several fractal dimension methods. | 34 |
| **Fractal Signal** | A *fractal signal* is a signal whose scaled-down version is similar to a part of itself. Also, a fractal signal is self-affine when different scales | 34 |

| | | |
|---|---|---|
| | along different coordinates are using for a scale-down version of a signal. | |
| **GA** | A *genetic algorithm* (GA) is an optimizer algorithm, and it is a subset of evolutionary algorithms that is inspired by nature and biological processes such as parent selection, mutation, and crossover. | 71 |
| **IDS** | An *Intrusion Detection System* (IDS) is an early warning detection system capable of identifying and predicting departures from expected operations in networks to reduce anomalous behaviour. In general, there are two types of IDS: signature based and anomaly based. | 3 |
| **Machine Learning** | *Machine learning* is a field of study to teach computers without explicit programming, and this field of study deals with learning from tasks, gaining experience with data, and improving the performance. | 46 |
| **Monoscale** | A *monoscale* analysis measures a signal on one scale. This analysis is suitable for a simple signal with low complexity that has a Euclidean shape. | 46 |
| **Mother Wavelet** | A *mother wavelet* (wavelet) is used to approximate any given function. The mother wavelet is a small wave that has an oscillating wavelike characteristic. Mother wavelets are used as basis functions like sin and cos in Fourier analysis, to represent signals. A mother wavelet is used for a multi-scale analysis, and it represents a signal at different divisions in the time and scale domains using scaling and shifting functions. Therefore, it has different stages of resolution. | 26 |

| | | |
|---|---|---|
| **Multiscale** | A *multiscale* analysis measures a signal on distinct scales, so that the hidden characteristics of the signal are extracted in each scale. | 46 |
| **Polyscale** | A *polyscale* analysis measures a signal on distinct scales, and its outcome requires all the scales to be used simultaneously in order to extract the power-law relationship in the signal. | 46 |
| **WT** | The *wavelet transform* (WT) is a mathematical tool that represents a time series signal in time and scale domains. The WT is used to extract information or hidden features from a signal, so it reveals characteristics that are not observable in the time domain. The WT can decompose a signal into coefficients, and it can provide this information according to different scales. | 27 |

# Chapter 1

# INTRODUCTION

More than 2.4 billion individuals, as well as businesses and industries, use the Internet for services such as E-health, structural health, monitoring-of industrial control systems and communications. The amount of financial damages caused by cyber-crime increased from 17.8 billion U.S. dollars to 4200 billion U.S. dollars from 2001 to 2020 [Cybe21] and cybercrime incidents continue to threaten public safety. Since the Internet was intended for data communications, its security was not considered, which resulted in high threats and attacks from malicious software (malware) on these Internet services. It is essential to identify anomalies of network operation, because cyber-attacks cause a number of possible impacts, including unavailability of Internet services, data corruption, deceit, disruption of computer operations, computer security breaches (to gain private information from computer systems), or theft of sensitive information. Moreover, unavailability of the Internet services cause: shutdowns and a slowdown of industrial systems, such as brownouts and even blackouts of a smart grid. In addition, cyber space infections can have serious impacts on the real world, and unavailability of the Internet services cause threats to human life. For example, power shutdowns during a medical surgery can

place the patient's life at risk. Another example is of a power outage that could cause a nuclear system explosion, which could kill millions of people. Therefore, methods for detecting software and hardware vulnerabilities to overcome security flaws are necessary to overcome cybercrime. For example, in 2010, a new malware program, the Stuxnet worm, disrupted the *programmable logic controller*s (PLCs) of the Iranian industrial nuclear systems, and it took control of the systems [FaMC11][Karn11]. This sophisticated worm has demonstrated that not only can cyber-attacks affect computer networks and Internet services, but they can also affect critical industrial processes and threaten human life. The Stuxnet worm infected two groups of industrial control systems. The first group, consisted of uranium-enrichment controlling facilities in Iran [Proc20]. The second group included *supervisory control and data acquisition* (SCADA) systems, used in power generation, water management and oil pipelines. More information about SCADA is shown in Appendix A.

The Stuxnet attack consisted of two phases [Rose13]. In the first phase, the worm passed through the air gap layer and impacted the Windows based operating system of the nuclear enrichment facility. In the second phase, the worm manipulated the speed of centrifuge rotors in the purification process. This made the uranium in the plant impure, so the uranium was useless. Also, manipulating the speed cracked the rotors, so if the rotors had broken, an explosion may have happened. In addition, the Stuxnet worm hid itself and bypassed safety systems by sending false data reports. The reports stated that the SCADA system was working normally and was operating at normal speed. Eventually, the worm, functioning in cyber space, impacted on the nuclear system of the enrichment facility. This impact could have led to an explosion that would have killed millions of people. The Stuxnet worm showed that a cyber war can be a real war [Rose13].

Another cyber domain attack that affected the physical domain occurred at Ukraine's power grid in December 2015 [Fair16a]. Cybersecurity experts named this attack a "wake-up call" for North American industrial control systems that use a smart grid. Recognizing that defending against cyberattacks is an urgent necessity, the *Defense Advanced Research Projects Agency* (DARPA) in the USA has invested $77 million for the smart grid cybersecurity program from August 2016 [Fair16b]. More information about a smart grid is shown in Appendix B.

In 2017, another cyber domain attack, the WannaCry ransomware worm, disrupted computers running the Microsoft Windows operating system was using encrypting data and requesting payments in the Bitcoin cryptocurrency as ransom [Bhat18]. The WannaCry ransomware attack affected more than 300,000 computers in more than 150 countries and damaged billions of dollars. The attack demonstrated how vulnerable personal devices can be and forced cybersecurity experts to seriously consider the security flaws and threats [Bhat18].

Blockchain is a software protocol technology that provides the secure transfer of money using the Internet without any third party such as central bank or any financial organization [RaDe18]. A transaction is validated, executed, and recorded in a tamper-resistant database in order that event occurred. The database is always available on the Internet for on-demand lookup and verification. More information about blockchain is shown in Appendix C. A review of the literature shows that computer systems are at risk of security issues, and these systems are left unprotected and vulnerable to attacks. When cybersecurity flaws cause devices be vulnerable to cyber-attacks, the functionality of these physical devices can be compromised. To prevent these cyber-physical-social security attacks cognitive informatics, and cognitive security systems can be used to design an effective *intrusion detection system* (IDS) [Kins12]. An IDS is an early warning detection system that predicts problems in networks to reduce anomalous activities. In general,

there are two types of IDS: signature based and anomaly based. A signature-based IDS detects known attacks. In contrast, an anomaly-based IDS detects unknown (zero day) attacks and anomalous events such as variations of known attacks, so it has to detect not only a wide range of known vulnerabilities, but also future unknown vulnerabilities.

An effective IDS should be sensitive enough to notice warnings or at least clues of anomalies in network traffic and communications behavior with high accuracy. For designing a proper IDS, it is essential to find parameters (pattern size of data traffic, interval of transmission attempts, transmission power level, missing ACKs, request/reply pattern), features, and measurements to explore the normal and anomalous behavior patterns of the nodes in systems.

Cognitive informatics demonstrates the information acquisition, processing, interpretation and expression functions of the human learning [Wang02]. An IDS, which supports the prevention of cyber-physical-social attacks, could be improved by mimicking and simulating human learning activity using artificial intelligence and machine learning. In this research, adaptive IDSs are introduced based on a supervised learning algorithm and a *deep reinforcement learning* (DRL) for unlabeled data. An adaptive system is a system that changes its behavior in response to its environment in a way similar to evolutionary adaptation. Widrow and Lehr invented adaptive linear neurons (Adaline) [WiLe90] as the first commercial neural network to be used [Vela20]. In addition, they created the Madaline (Many Adeline) artificial neural networks for applications including speech and pattern generation, weather forecasting, adaptive controls, and adaptive signal processing [WiLe90]. The adaptive change that occurs is often relevant to achieving a goal. The goals of the designed adaptive IDSs in the current study are detection of anomalous behavior in real time, detection of anomalous behavior in various environments and improvement of detecting the *distributed denial of service* (DDoS) attacks. To further enhance the detection rate,

polyscale analysis, spectral analysis techniques in signal processing, artificial intelligence, machine learning algorithms and evolutionary algorithm are used.

## 1.1 Thesis Statement

The focus of this study is to study a new anomaly-based IDS for *domain name system* (DNS) servers that is under the DDoS attacks. To develop an IDS algorithm in this study, an adaptive mother wavelet that is specifically adapted to the input signal of the IDS is designed and created. In addition, an adaptive IDS based on a *convolutional neural network* (CNN) structure with a new cost function is designed for supervised learning to obtain a high detection rate. Moreover, an IDS based on policy gradient DRL is developed to obtain an IDS that detects DDoS attacks using unlabeled data. This system is a realistic IDS, which is more suitable for real world Internet traffic.

## 1.2 Aim

The goal of this proposed research is to detect anomalous behavior in DNS servers under the *denial of service* (DoS) and the *distributed denial of service* (DDoS) attacks. More information about DNS is shown in Appendix D and Appendix E, and more information about DDoS attacks is shown in Ch. 5 and Appendix F.

## 1.3 Approach

This research is based on extracting features from the raw data and then passing the extracted features to feed a non-parametric learning model (a machine learning algorithm). A critical task in developing such a system is to use suitable signal analysis techniques to extract and examine the underlying features to carry out the identification and classification of normal and anomalous data.

## 1.4 Purpose

The main purpose of this research is to optimize the detection of anomalous behaviors in computer networks through the introduction of an adaptive IDS that reduces the false positive and false negative detection, increases the detection speed, and uses less memory for the IDS algorithm.

## 1.5 Scope

The scope of this proposed research is to consider the Internet traffic as the cyber domain that contains DDoS attacks. Moreover, there are two classes for classification of the signals: normal and anomalous. In addition, the network availability is considered. Additionally, time-frequency features are extracted by the *wavelet transform* (WT), the *variance fractal dimension* (VFD) and an *improved version of the variance fractal dimension* (VFDTv2). Furthermore, the features of nonstationary signals with fractal characteristics are extracted using extraction methods.

## 1.6 Methodology

Five anomaly-based *intrusion detection systems* (IDSs) have been developed and tested. The methodologies extract distinguishing features to enhance the sensitivity of detection regarding anomalous behaviors. Then, based on learning models, anomalous behaviors are detected. The main procedure of developing the IDSs are consists of the following steps:

S.1. The dataset that is used for training and testing in this research is downloaded from the Center for Applied Internet Data Analysis (CAIDA) [CAIDA15]. Also, simulating normal and anomalous patterns of the smart grid signal (the data series) is used.

S.2. The signals are pre-processed by:

S.2.1 Analyzing signals by digital signal processing tools such as the *wavelet transform* (WT) to reveals important underlying features and dimension reduction. Therefore, a mother wavelet that is specifically adapted to the input of the IDS is created for this research.

S.2.2. Applying a fractal dimension to extract more features from the signals. Therefore, an extended version of the VFD is created to extract more features in this research.

S.3. Processing the signal by applying a non-parametric learning model such as *artificial neural network* (ANN), the deep learning CNN, and DRL to learn and classify normal and anomalous behaviors based on the extracted features.

## 1.7 Research Questions

In order to design the IDS algorithm, security, statistics, machine learning, and signal processing issues are considered. The following research questions guide this study.

a) What are the most common vulnerability issues in computer network systems and how can they be detected?

b) How to design a real-time learning algorithm similar to a cybersecurity expert to classify the normal and anomalous behaviors of a computer network system under attack?

c) How do normal bursty traffic behaviors change when a computer network system is under DDoS attacks? Do DDoS attacks change the statistical characteristics of arrival packets?

d) What methodologies are suitable for adaptability in this research?

e) Why are mother wavelets good?

f) How to design an adaptive mother wavelet that adapts with an arbitrary input signal to extract the highest hidden features?

## 1.8 Thesis Organization

This thesis includes seven chapters. The introduction of the thesis has introduced the problem, the research objectives, and the research questions. The background and literature review are described in Ch. 2. In Ch. 3, the techniques for designing IDS, including all the components of the solution, the eight different feature extraction and the supervised and unsupervised learning used in this thesis are explained. Chapter 4 focuses on the proposed intrusion detection systems. In this chapter, five anomaly detection algorithms are proposed, a cost function and an adaptive mother wavelet that increases the detection rate of DDoS attacks with more accuracy are designed. In Ch.5, two complete time series datasets, algorithms, software and hardware implementations are described. In Ch. 6, the design of the experiments and their results are included and the results of the simulation of the proposed anomaly detection algorithms are explained. Chapter 7 provides conclusion, as well as the contributions of this research and future research suggestions.

# Chapter 2

# BACKGROUND AND LITERATURE REVIEW

This chapter introduces an attack taxonomy for computer network systems, and a literature review regarding IDSs in three categories. The literature review shows the weakness of each IDS in each category. Therefore, this research proposed an IDS with an expanded version of the online fractal calculation method for extracting hidden features for the input data, and an adaptive mother wavelet for detecting DDoS attacks in input data. Moreover, a fit IDS matched for detecting DDoS ITD is designed using a weighted cost function and an adaptive structure for convolutional neural networks. This research proposes a computer field-based IDS for detecting anomalous DDoS attacks.

## 2.1 Intrusion Detection Systems

Attack taxonomy is a useful tool for analyzing complex systems and helps experts to understand different classes of security algorithms. There are several attack classifications, but in this thesis, attack taxonomy is based on security properties. An attack taxonomy is shown in

Table 2.1. In general, security has five properties [ZhJS11]: data integrity, availability, authentication, confidentiality, and freshness or timeline of data. Therefore, there are five attack classes. First is data integrity attack. In this kind of attack, hackers try to manipulate data or signals by insert, delete, and change of control commends between consumers and controllers. The purpose of this attack is to deceive controller by making wrong decision, control device or software, and lunch furthermore attacks. one example of data integrity attack is changing information of an account. Second is availability attack. In this kind of attack, hackers try to overwhelm the server or control system by sending several fake requests. The purpose of this attack is preventing service to clients, delaying result to the control system's action which have deadline constrain in control system, and failure of data communications. one example of data integrity attack is DoS attack. A DoS attack is a serious internet attack that can damage the national economy by causing a delay of a few seconds. Third is authentication attack. In this kind of attack, hackers try to bypassing normal authentication. The purpose of this attack is getting service without permission. One example of authentication attack is man in the middle attack. forth is confidentiality attack. In this kind of attack, hackers try to reveal private information. The purpose of this attack is to get private information illegally such as users' password and encryption keys. One example of confidentiality attack is usage information. Fifth is timeline attack. In this kind of attack, hackers try to retransmission of legal packet or signal again. The purpose of this attack is controlling power system that cause controllers make wrong decisions. one example of timeline attack is replay attack. Authentication attack, confidentiality attack, and timeline attack are out of scope in this thesis.

**TABLE 2.1:** An attack taxonomy.

| Attack Category | Definition | Purpose |
|---|---|---|
| Data Integrity Attacks (changing bill, changing information) | Manipulating data or signals by insert, delete, or change control commends. | Controller makes wrong decision, so hackers can control devices or software. Therefore, the hackers can launch further attacks. |
| Availability Attacks (DoS) (damage national economy by delay few seconds) | Making server or control system too busy by sending lots of fake request to overwhelm them. | Preventing to service to clients, delaying result to the control system's action that have deadline constrain in control system, or failure of data communications. |
| Authentication Attacks (man in the middle) | Bypassing normal authentication. | Getting service without permission. |
| Confidentiality Attacks (usage information) | Reveal private information. | Getting private information such as passwords, encryption keys by unauthorized person. |
| Timeline Attacks (replay attack) | Retransmission of legal packet or signals. | Controlling power systems, or controller makes wrong decision. |

The literature review regarding the proposed IDS is reported in four categories as follows:

## 2.2 Intrusion Detection Systems in the Cyber Domain

### 2.2.1 Intrusion Detection Based on Extracting Patterns of Attacks

Mohammadi et al. [MMMK12] studied a framework for detecting attacks in. The NAN-IDS consists of a distributed and hierarchical IDS which combines anomaly-based and signature-based methods. It extracts patterns and parameters of a NAN network in a smart grid such as data traffic, transmission power level, interval of transmission queries and request/reply patterns. The proposed NAN-IDS has three different kinds of IDS nodes: field IDS, WAN IDS and central IDS. Field IDS monitors and collects trace data communications of the neighboring

smart meters, and it reports detected attacks to the central IDS. WAN IDS is responsible for incoming and outgoing traffic from/to collectors. It reports the malicious nodes to the central IDS. The central IDS is responsible for making global decisions based on alarms and notifications coming from the WAN IDSs and field IDSs. The proposed IDS checks anomalous behavior in their communications. Finally, the proposed IDS makes final decisions about the anomalous behavior, whether it is a malicious attack or just a transient failure. Then the IDS keeps the history of the monitored nodes to make accurate decisions in the future.

This paper [FFSN11] tries to overcome one of the most important attacks in the network, which is a DoS or a DDoS attack. It is based on the estimated characteristics of the regular traffic. The objective of this paper was to use well known patterns of traffic and request to detect malicious activities. For example, a group of nodes trying to perform a DDoS attack, but they would not use the typical traffic pattern. With this information, the system could detect an anomaly that was present in the network, and the nodes that were responsible for it. This detection scheme was based on the principal that legitimate traffic has well defined characteristics. However, if an attacker is able to develop a pattern that is statistically close to typical patterns, the detection method cannot properly identify the attack.

Intrusion detection based on extracting patterns use static structures for designing IDSs. In addition, like neural network IDSs, most of the machine learning based IDSs use a common technique for evaluating the machine learning output. In this research, a weighted cost function is designed to evaluate the artificial neural network and the convolutional neural network output. Also, an adaptive structure for the convolutional neural network as an IDS is created. As a result, a fit IDS that is matched for detecting DDoS ITD is designed.

### 2.2.2 Intrusion Detection Based on Fractal Dimensions

This paper [ShJa12] tries to detect anomaly behavior in networks by using a fractal dimension method. The proposed method uses WT to decompose network traffic to coefficients parameters. Then, it uses function *G*, to compare variations between two successive windows. This function use Katz's algorithm to calculate the fractal dimension which is

$$FD = \frac{\log(L/a)}{\log(d/a)} \tag{2.1}$$

Where *L* is length of time series, and *d* is diameter estimation of the distance between first data point and the data with the highest distance. Also, *a* is the average distance between two successive data points. s

$$G_k = | FD_{k+1} - FD_k | \qquad k = 1,2,3,...,N \tag{2.2}$$

where *N* is the number of samples in the *G* function. Also, it finds local maxima of *G* and defines a threshold to detect anomaly in network traffic. Moreover, it found proper window length based on energy of *G* by Eq. (1.3)

$$E_G = \frac{\sum_k | G_k |^2}{N} \tag{2.3}$$

The threshold value is defined by mean (*G*) + 2 (standard deviation of *G*): mean(*G*) + $2\sigma_G$. However, Katz's algorithm is not an online method because it needs to traverse whole input data to find the point with the highest distance with the first data.

This paper [MaDy14] introduces a network anomaly detection method based on a statistical self-similarity factor that affects the autocorrelation function. Therefore, this method can detect an anomaly based on deviation, which was the mean value of the Hurst (*H*) parameter. The *H* parameter is explained with details in Eq. (3.42). In general, this paper has three steps. First, this

method tries to find features of traffic, so this method collects network traffic based on wireshark software. Second, this method uses statistical analysis, where statistical change affects the autocorrelation function. Third, this method tries to find *power spectrum density* (PSD) of features and linearize PSD of them. Then, based on linearized PSD it finds the slope, which is $\beta$. The purpose of this method is to find the Hurst parameter, which is $H = 1 - \beta/2$, to compare with normal traffic models. Any deviation shows anomaly in network traffic. The Hurst parameter is defined with detail in Sec. 3. 3.1.3 using Eq. (3.25). However, if an attacker is able to develop a pattern that is statistically close to typical patterns, the detection method cannot properly identify the attack.

Most intrusion detection software based on fractal dimensions methods are not online (the calculation of the fractal dimension is not in real time). Moreover, the selection of proper fractal dimensions methods that adapt to the input signal is important. This research uses and expands a VFD method that works in real-time and adapts to the Internet traffic that contains DDoS attacks.

### 2.2.3 Intrusion Detection Based on Signal Processing

This paper [BKPR02] introduces a network anomaly detection method based on deviation score algorithm. In the first step, from decomposition, three output signals are derived: high frequency, mid frequency, and low frequency. In the second step, it normalizes the high frequency and mid frequency parts within a one-week window (to have variance one). In the third step, it computes the local variability of the (normalized) high frequency and mid frequency parts, assessing the variance of the data falling within a moving window of specified size. In the fourth step, it combines the local variability (weighted sum): weighted (H-part) + weighted (M-part) -> the V-part. In the fifth step, it detects anomaly behaviors by thresholding to the V-part as follows. 2 or higher: “high-confidence” of detection anomaly behavior, between 1.25 and 2: “grey band”,

and below 1.25: “low-confidence” of detection anomaly behavior. Long lived events such as flash crowd events exposed by low and mid frequency. Short lived events such as network failure, DDoS attack, Measurement failures exposed by deviation score of mid frequency and high frequency such as flash crowd events. The time frequency detection technique can efficiently increase detection rate of anomalous behavior. Due to their anomaly detection method, the 5 minutes’ interval for gathering data is too slow, and the length of window and weighted sum for detection attack in deviation score algorithm is static.

Intrusion detection based on signal processing methods need selection proper signal analysis tools that are adapt with the input signal. This research creates a mother wavelet for a specific application such as detecting the DDoS attacks.

### 2.2.4 Intrusion Detection Based on Deep Reinforcement Learning

Arturo proposed an anomaly detection algorithm based on reinforcement learning to detect flooding-based DDoS attacks [Serv07]. The algorithm is based on the Q-learning algorithm to estimate a value function; however, the algorithm does not have any exploration technique. Servin *et al.* offered a distributed Q-learning reinforcement learning approach in a hierarchical architecture of network sensor agents, and Boltzmann exploration provided an action selection strategy during learning [SeKu08]. Each network sensor agent learned to interpret local state observations and communicates them to a central agent higher up in the agent hierarchy. These central agents, in turn, learned to send signals up the hierarchy, based on the signals received. Finally, the agent at the top of the hierarchy learned when to signal an intrusion alarm. Due to their architecture, there may be instances where the attackers attack the top node and take advantage of the bottleneck problem. Xu *et al.* offered a reinforcement learning technique for host-based IDS using sequences of system calls [XuXi05]. A Markov reward process model was introduced for

modeling the behaviors of system call sequences. Also, the IDS problem was converted to predicting the value functions of the Markov reward process. In addition, a temporal difference learning algorithm was used for value function prediction, so anomalous behaviors of the host could be predicted. However, the Markov process model is the theoretical foundation of DRL, and a proper practical model should be used to implement it.

## 2.3 Intrusion Detection in Power Systems

In this section, the IDSs that are considered detect anomalous behaviors based on voltage, current and phase angle that are introduced as follows.

Liu et al. [LGWG13] showed how to detect one of dangerous attacks, which is the *bad data injection* (BDI) attack. The aims of this attack are energy stealing on the consumer side, manipulating of energy costs, modifying smart meters, taking control of power system, and breaking down power generation. Liu et al's detection algorithm has four steps. In the first step, a large smart grid system is divided into several subsystems by a partitioning graph method. The aim of partitioning is to enhance the sensitivity of the detection method. In the second step, each subsystem of the power system is modeled with a mathematic equation. In the third step, a BDI detection algorithm is launched to detect the attack in each subsystem. A chi-square test was used to test and compare it with a threshold in each subsystem. If the chi-square test's result is less than or equal to the threshold, then the detection attack algorithm finds the subsystem is normal. on the other hand, if the chi-square test result is greater than the threshold, then the detection attack algorithm suspects a BDI attack in the subsystem. In the fourth step, the algorithm can detect the exact location of the BDI by zooming in the subsystem's graph. Therefore, an iterative algorithm to perform steps 2, 3 and 4 in a loop is needed to detect and locate the BDI. Wrinch et al. [WrFW12] proposed anomaly detection by comparing the test building's results with an ideal case (threshold).

This method finds an anomaly behavior by extracting periodic energy and demand parameters of the power consumption of a building. In the first step, this method extracts information based on frequency domain (discrete Fourier transform) instead of time domain, and it finds energy from discrete Fourier transform. Then, it extracts periodic energy demand parameters of a building in the frequency domain. Next, the proposed method extracts periodic schedules. High energy periodical points present both periodic computer control and occupant activities. If the energy of each harmonic is outside of a threshold, then the method can detect an anomalous behavior (by simple observing a trend of increasing periodic demand energy ratios from the original baseline).

Hu et al. offers a taxonomy of attack based on cyber and physical schemes [HuPG14]. According to this paper, for detecting anomalous behaviors in control systems, security experts should consider both sides of cyber space domain and physical space domain. In some attack cases, although there are not attacks' impact on cyber space domain, there is some impact on the physical space such as changing the rotation of a machine (nuclear system). Therefore, any change in cyber and physical domain features can be detected by IDS. According to this paper, power systems represent three phase current and voltage in a frame. A device in the smart grid is modeled in terms of a differential equation of device dynamics. Voltage, current and phase angle have to satisfy algebraic constraint. Using this phase angle, this paper tries to detect attacks. For example, the proposed method can detect an integrity attack when the system is in an unstable mode. The paper proposed a comprehensive scheme for physical and cyber-attack detection. Due to their architecture, there may be instances where the attackers attack the root node, and takes advantages of the bottleneck problem.

Intrusion detection based on power system methods require high power electricity. If the anomalies present in low power electricity, power system methods are not sensitive enough to

detect them. This research proposes methods that detect anomalous behaviour in low power electricity currents. A further problem with power system methods is that only electrical power experts can conduct the design and evaluation of such anomaly detection systems since this is out of the range of expertise of IT professionals. Therefore, a computer field-based IDS is preferable.

## 2.4 Summary of Chapter 2

This chapter summarizes an attack taxonomy for computer network systems and presents a literature review of IDS to detect attacks. Due to the similarity between regular DDoS attacks on DNS servers and DDoS attacks on smart grid infrastructure, the literature review of the power system has been used for this thesis.

Comparing the IDSs shows the strength and weaknesses of each algorithm. The goals of the IDS algorithms are to reduce false positive and false negative rates, detect attacks in real-time, reduce the required memory for IDSs, and detect anomalous behaviors in unlabeled data for real-world applications. Therefore, this research is conducted to design and create an IDS that meets the above conditions.

# Chapter 3

# TECHNIQUES FOR DESIGNING IDS

This chapter summarizes the techniques used in the proposed development of the Intrusion Detection System (IDS). It consists of two subsections: feature extraction and machine learning. The techniques required by the proposed IDS and associated with the feature extraction research include: (i) signal analysis, (ii) spectral analysis, (iii) multiscale analysis, and (iv) polyscale analysis. The techniques required by the proposed IDS and associated with the machine learning research include: (i) supervised learning, and (ii) deep reinforcement learning.

The DDoS ITD is a complex signal, so the monoscale analysis is not enough for it. A multiscale analysis measures a signal with various scales, so the hidden characteristics of the DDoS ITD are extracted in each scale. Using DWT, which is a multiscale analysis, an adaptive mother wavelet is created to extract distinguishing features from the DDoS ITD. Because the DDoS ITD is a fractal signal, so the multiscale analysis is not enough for it. Therefore, a polyscale analysis measures a signal with various scales, so the hidden characteristics of the DDoS ITD are extracted in each scale. A polyscale analysis measures a signal with various scales, and its outcome correlation requires all the scales to be used simultaneously. Therefore, the hidden features of the

DDoS ITD are extracted in each scale, and the correlation between the hidden features are considered. As a result, the DDoS attacks can be detected with a higher detection rate with the polyscale analysis. Therefore, an improved version of the variance fractal dimension trajectory (VFDTv2) is proposed and created. The VFDTv2 is adapted to the DDoS ITD to extract distinguishing features.

Identifying normal burst-data behaviors of a network and the abnormal burst-data behaviors caused by DDoS attacks is challenging. Both classes of network traffic have similar intrinsic characteristics. They are both stochastic time-series signals, self-affine and multi-fractal, non-periodic and broadband. Therefore, to differentiate the two, distinguishing features must be extracted. Next, new attack patterns and behaviors must be detected in the Internet, which is a frequently changing environment. Therefore, a learning method that can detect new attack patterns and behaviors in frequently changing environments must be used [Beqi09]. A deep learning algorithm is a good candidate to learn and classify normal behaviors from anomalous behaviors in such an environment.

## 3.1 Feature Extraction Methods

This research focused on three categories of features based on: mono-scale technique, multi-scale technique and poly-scale (power-law related) technique. Mono-scale techniques are statistical features, *turns count* (TC), *zero crossing* (ZC), distance and *principal component analysis* (PCA). A multi-scale technique is *discrete wavelet transform* (DWT). A poly-scale technique is *variance fractal dimension* (VFD). In this section, eight different methods for extracting features from input data are presented.

### 3.1.1 Statistical Features

Mean: the mean is the arithmetic average of data. The mean is the average value of a distribution [MoMC09].

$$\bar{x} = \frac{1}{n}\sum x_i \tag{3.1}$$

Variance: the variance is a common measure about the mean as center. It measures spread by looking at how far the data is from its mean [MoMC09].

$$s^2 = \frac{1}{n-1}\sum (x_i - \bar{x})^2 \tag{3.2}$$

The standard deviation is the square root of the variance.

$$s = \sqrt{\frac{1}{n-1}\sum (x_i - \bar{x})^2} \tag{3.3}$$

Skewness: the skewness is a numeric metric to measure the degree of asymmetry of a distribution [ShCh15]. The skewness for a normal distribution that is a symmetric distribution is zero. A negative skewness value indicates the data distribution has skewed to the right, and a positive skewness value indicates the data distribution has skewed to the right. The skewness indicates the data distribution has skewed to the right or to the left [ShCh15].

With this convention, positive skewness (or skewed to the right) indicates a heavy and long-extending tail on the right side of the location alignment. Similarly, a negative skewness (or skewed to the left) indicates a heavy and long-extending left tail. The skewness is calculated using Eq. (3.4) [ShCh15].

$$\gamma = E[(x_i - \bar{x})^3] / s^3 = \frac{\sum_{i=1}^{N} (x_i - \bar{x})^3 / N}{s^3} \tag{3.4}$$

Kurtosis: the kurtosis is a measure of peakedness of the distribution, and it measures the tail weights [ShCh15]. Kurtosis is a measure of both the peakedness of the distribution in and around the location measure (center of mass) and a measure of the tail weights that jointly characterize the accumulation of probability mass toward the center. The kurtosis indicates the impact of the heavy tail of the distribution. Positive values of kurtosis indicate heavy tail distributions. The kurtosis is calculated using Eq. (3.5) [ShCh15].

$$k = E[(x_i - \bar{x})^4] / s^4 = \frac{\sum_{i=1}^{N} (x_i - \bar{x})^4 / N}{s^4} \tag{3.5}$$

## 3.1.2 Signal Processing

*3.1.2.1 The Signal Processing Purposes*

The purposes of signal processing concept are dimensionality reduction and feature extraction. These issues are shown with more detail in the following subsections.

3.1.2.1.1 Dimensionality Reduction

The signal processing tools such as *Fourier transform* (FT) and *Wavelet transform* (WT) can be used as dimension reduction. When the dimension is reduced, the dataset classification (an input signal as data) is easier and more accurate. All transformations in signal processing cause dimension reduction. By sampling the signal, it is feasible to reconstruct it, so time series data can be obtained. One problem with time series domain is that there are about one million data points, so it is difficult and time consuming to analysis the data. Therefore, a transform from the time domain to the frequency domain is essential to reduce this large amount of data points to two or three points. This is the concept of dimension reduction. The figures below demonstrate the concept of dimension reduction. Figure 3.1.a illustrates a complex signal in its time domain. Figure 3.1.b shows sampling of the complex signal in Fig. 3.1.a.

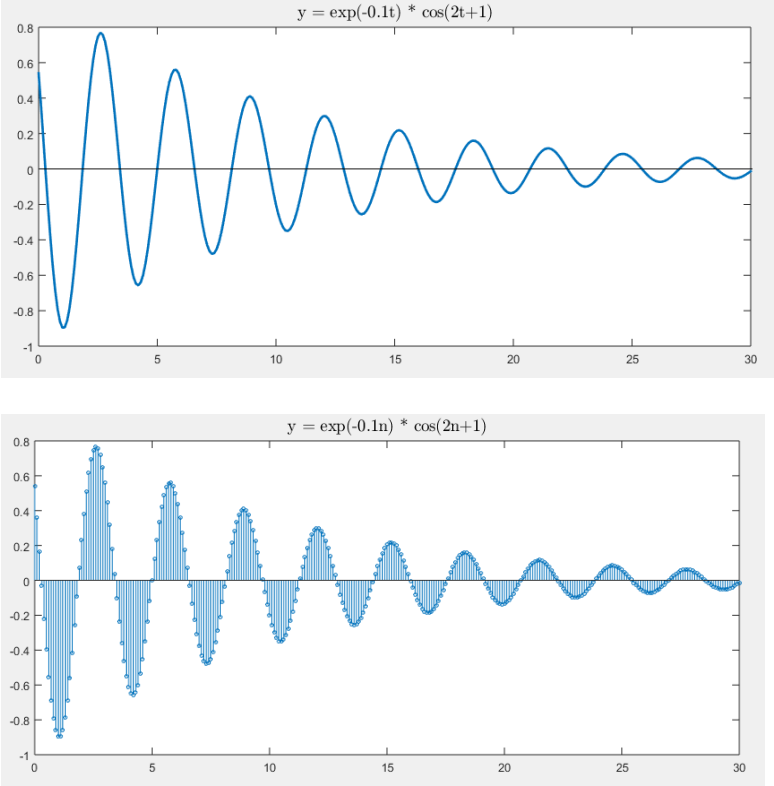



**Fig. 3.1:** a) A complex signal (the top). b) Sampling the complex of the signal (the bottom).

A combination of signal processing methods and network anomaly-detection methods is used to reduce the dimension of data series in order to detect traffic deviations. In this study, signal-processing methods are used for dimension reduction and to extract inherent features from data streams. Figure 3.2 illustrates power spectrum density of the broadband signal in Fig. 3.1.b. The 3-dB bandwidth ends at the cutoff frequency, $f_c$, is not sufficient to process a broadband signal. We must also include the signal down to the noise level, as indicated by the cutoff frequency for the broadband signal, $f_{cb}$.

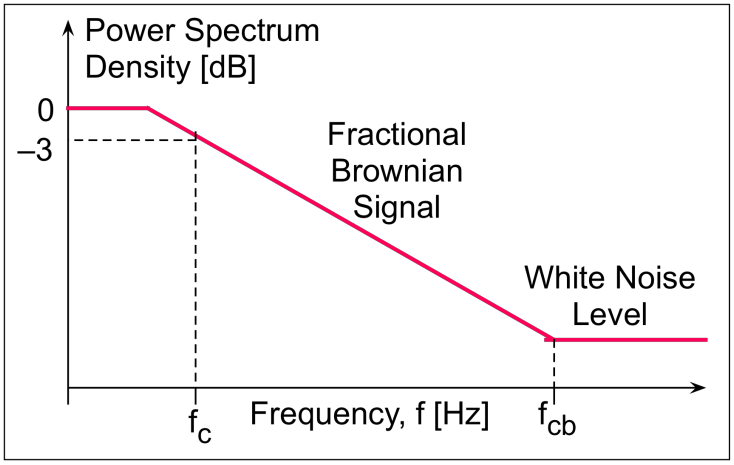



**Fig. 3.2:** Power spectrum density of the broadband signal in Fig. 3.1.b [Kins15].

3.1.2.1.2 Feature Extraction

The signal processing tools such as FT and WT can be used as feature extractors. The FT and WT are explained as follows:

Fourier transform:

FT is a mathematical method of expressing a function in terms of the sum of its projections onto a set of basis functions. Similar to decomposing a point in Euclidean space into the sum of its basis vector components, FT describes a way of decomposing a function into a sum of orthogonal basis functions. Figure 3.3 shows how Fourier analysis breaks down a complicated signal into simple waves.

In three-dimensional space, the Cartesian coordinate system is based on three mutually perpendicular coordinate axes, the x-axis, the y-axis, and the z-axis, and in this system, i, j, k are the coordinate vectors, as shown in Eq. (3.6).

$$V = xi + yj + zk \tag{3.6}$$

The orthonormal basis represents a vector $V$ in Euclidean space in the form of components.

In the same way, the basis function in FT is shown in Eq. (3.7).

$$e^{2\pi i\omega n} = cos(2\pi\omega n) + i\, sin(2\pi\omega n) \tag{3.7}$$

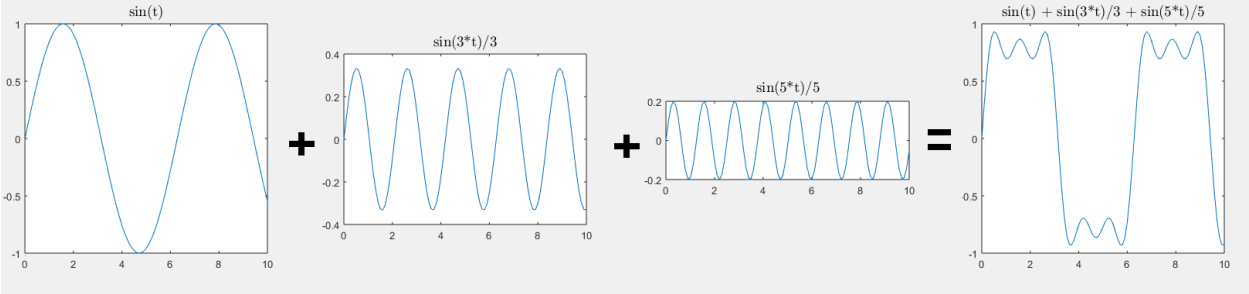



**Fig. 3.3:** Fourier analysis breaks down a complicated signal into simple waves, and FT determined how much of each simple waves are needed. (After [Sueu18])

The FT maps a time series signal into the frequency domain [AbDD02]. This representation can be used to characterize transient events and can extract hidden features from the time series signal. A sampled signal at a discrete time can be transformed into the frequency domain by using the discrete Fourier transform (DFT). Fourier transform is essentially a convolution between the discrete time series $x_k$ and a series of sin and cos functions (basis functions). The FT of a continuous signal $x(t)$ can be expressed as the inner product in Eq. (3.8), the FT of a discrete signal $x(n)$ can be expressed as the inner product in Eq. (3.9) and the DFT of a finite discrete signal $x(n)$ can be expressed as Eq. (3.10) [PrMa07].

$$X(F) = \langle x(t), e^{-j2\pi Ft} \rangle = \langle x(t), e^{-j\omega t} \rangle = \int_{-\infty}^{+\infty} x(t) e^{-j\omega t} dt \tag{3.8}$$

$$X(\omega) = \sum_{n=-\infty}^{\infty} x(n) e^{-j\omega n} \tag{3.9}$$

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N} \qquad k = 0, 1, 2, \ldots, N-1 \tag{3.10}$$

The convolution measures the similarity between $x(t)$ or $x(n)$ and the basis functions, and expresses the average frequency information during the entire period of the signal analyzed.

Fourier series analysis is not a suitable tool for analysing non-periodic signals such as transient waveform, superimposed and anomalous signals [KiAg00]. Furthermore, Fourier transform is not a suitable tool to analyse these signals because these signals are broadband. Fourier transform can deal with non-periodic signals, but these signals must have finite power. Transient waveform, superimposed and anomalous signals are broadband, so they do not have finite power which render Fourier transform outputs useless. Non-periodic and broadband signals can be overcome by *short term (windowed) Fourier transform* (STFT) because this tool uses a window function and maps a signal into two-dimensional time and frequency domains. However, STFT cannot analyse these signals because STFT has fixed time frequency size windows, and it has a fixed resolution in both domains; therefore, STFT cannot detect the location of irregularities accurately.

The non-periodicity broadband of those signal and the fixed time and frequency size windows can be overcome by WT [KiAg00] because the WT basis function can be chosen based on different application, so it can be varied. Moreover, WT is localized in both time and frequency so it can detect the location of anomalous signals. The wavelet transform multi-scale method can accelerate convergence and cope with the presence of local minima to reach global minima. The rationale behind this success is that the cost function shows stronger convexity and has fewer minima at a larger scale such that the global minimum can be achieved.

Wavelet transform:

The *wavelet transform* (WT) is a tool in signal processing, and it maps a time series signal into the two-dimensional time-scale domain [AbDD02] that represents signal details and trends. This representation can be used to characterize transient events and can extract hidden features from the time series signal, which is described in detail in the next section.

The WT is used to extract information or hidden features from a signal, so it reveals characteristics that are not observable in the time domain [GaYa11]. The WT can decompose a signal into coefficients, and it can provide this information according to different scales. The mother wavelet (wavelet) is a small wave that has an oscillating wavelike characteristic. Mother wavelets are used as basis functions like sin and cos in Fourier analysis, to represent other functions.

One of the most important properties of WT is its ability to adjust the size of a window to have a suitable resolution in the time and frequency-domains. Therefore, WT uses a narrow window function when it analyzes a high frequency, and a wide window function when it analyzes a low frequency. One of the most important properties of WT is its ability to adjust the size of a window to have a suitable resolution in the time and frequency-domains. The WT has precise time resolution for high frequency signals and precise frequency resolution for low frequency signals. The WT is a multi-scale analysis, which means that it uses different scales at different levels of decomposition. The WT is a multi-scale analysis because it represents a signal at different divisions in the time and frequency domains; therefore, it has different stages of resolution [MoKi02]. Therefore, the WT has precise time resolution for high frequency signals (with short duration) and precise frequency resolution for low frequency signals (with long duration). This property of WT makes it suitable for detecting an irregular structure and an anomaly in signals [MoKi97]. In a time interval, when a signal changes rapidly, WT can zoom in the interval to extract more characteristics of a signal. Therefore, WT is sensitive to irregularities in signals. WT coefficients with anomalous events have larger magnitudes when compared to WT coefficients without anomalous events. In addition, WT reacts to the change in the first derivative (slope) of a signal, not to the change in the amplitude of a signal.

*3.1.2.2 The Discrete Wavelet Transform*

The *discrete wavelet transform* (DWT) is a mathematical tool that represents a time series signal in time and scale domains. One of the most important properties of the DAWT is its ability to adjust the size of a window to have a suitable resolution in the time and frequency-domains. For detecting irregularities in a signal by computer, the DWT is used. In DWT, the scale parameter *j* and translation parameter *k* have logarithmic discrete values. The discrete mother wavelet is shown in Eq. (3.11), and the discrete wavelet coefficient ( $\gamma$ ) of a given signal $x(n)$ is obtained by multiplication of the signal and the basis function as shown in Eq. (3.12) [GaYa11].

$$\psi_{j,k}(n) = \frac{1}{\sqrt{2^j}} \psi\left(\frac{n - k2^j}{2^j}\right) \tag{3.11}$$

$$\gamma(j,k) = \frac{1}{\sqrt{2^j}} \sum_{n=-\infty}^{+\infty} x(n) \psi^*\left(\frac{n - k2^j}{2^j}\right) \tag{3.12}$$

where the symbol $*$ denotes the complex conjugate. There are commonly used basis functions such as: Haar, Dubachies, and symlet mother wavelets [GaYa11]. To detect an anomaly with a low amplitude, short duration, fast calculation of the DWT coefficients, fast rate of decay and rapid oscillation in a signal, the Daubechies wavelet is a proper choice as a mother wavelet for DWT [SaKi06]. In a time interval, when a signal changes rapidly, the DWT can zoom in the interval to extract more characteristics of a signal. This property of the DWT makes it suitable for detecting irregular structures and anomalies in signals [MoKi97], which is important for detecting the DDoS attacks.

*3.1.2.3 The Wavelet Transform and The Discrete Wavelet Transform Concept*

The basic idea of the WT is to represent any arbitrary function *Y* as a decomposition of wavelets [Daub92]. The function *Y* is decomposed into different scale levels, where each level is

then further deconstructed using its own mother-wavelet adapted resolution. Unlike the continuous wavelet transform, the DWT can reduce the computational time and memory size. The discrete wavelet coefficient $wt(a,b)$ of a given signal $x(n)$ is shown in Eq. (3.13) [GaYa11].

$$wt(a,b) = \frac{1}{\sqrt{a}} \sum_{n=-\infty}^{+\infty} x(n)\psi^*(\frac{n-b}{a}) \tag{3.13}$$

where $a$ is the scale parameter and $b$ is the translation parameter. Also, the symbol $*$ denotes the complex conjugate, and $\psi^*$ is the complex conjugate of the scaled and shifted wavelet function.

When the inner product of the mother wavelet is at unity with itself, it is orthogonal. An orthogonal mother wavelet decomposes a signal in non-overlapping sub-frequency components [GaYa11], so high computational efficiency can be achieved. In addition, orthogonality is valuable because it conserves the energy of the signal, which allows for perfect reconstruction of a signal from coefficients. In addition, an orthogonal mother wavelet can be implemented using simple digital filtering techniques such as a filter bank.

Any function must meet the following conditions to be a mother wavelet: (1) it must have finite energy [AbDD02]; (2) it must be able to behave as a polynomial of a certain order [GoCh11] and, (3) it must have a finite length [Mall09].

The first condition, finite energy, creates the oscillation in the mother wavelet $\psi(t)$ and is called the admissibility condition and is fundamental for any mother wavelet. The admissibility condition is shown in Eq. (3.14) [GaYa11]

$$\int_{-\infty}^{+\infty} \frac{|\Psi(f)|^2}{f} df < \infty \tag{3.14}$$

where $\Psi(f)$ is the Fourier transform of the mother wavelet function $\psi(t)$. The admissibility condition requires that the amplitude of $\Psi(f)$ is equal to zero when the frequency is zero.

The nonzero DC component (at zero frequency) violates finite energy, so with an infinite energy, it is not possible to cover its frequency spectrum and its time duration with a mother wavelet [Vale99]. Therefore, the admissibility condition requires the average value of the positive and the negative areas under the mother wavelet to be zero. As a result, the mother wavelet must oscillate and look like a ripple [GaYa11][AbDD02]. Therefore, the mother wavelet has a diversity of frequencies, unlike the basis function (*sin* or *cos*) of the Fourier transform, which is limited to just one frequency. When the admissibility condition is met, the following equation can be extracted for the mother wavelet in Eq. (3.15).

$$\int_{-\infty}^{+\infty} \psi(t)dt = 0 \tag{3.15}$$

The second condition, detecting hidden abnormalities and singularities in an input signal, can be defined using vanishing moments [AbDD02]. A useful approach to designing a mother wavelet is to require that the mother wavelet's moments $M_k$ are defined using Eq. (3.16) [SoLa05][AbDD02].

$$M_k = \int_{-\infty}^{+\infty} t^k \psi(t)dt = 0 \quad for \quad \forall \quad 0 \le k < N \tag{3.16}$$

where the mother wavelet's moments must be zero, vanish up to a certain value $k < N$ and $t^k$ is a polynomial. With these conditions this mother wavelet has $N$ vanishing moments [AbDD02]. The property of the mother wavelet vanishing moments is that the inner product between the polynomial's signal $t^k$ and $\psi(t)$ is zero, so the polynomial of degree *N-1* or less and the mother wavelet are orthogonal. The vanishing moments concept shows that the input signal can be

modelled with a specific order of a polynomial. The vanishing moments show the order of the polynomial that can estimate the mother wavelet [SaKi05].

To detect hidden abnormalities in an input signal, a proper mother wavelet with specific vanishing moments should be used [AbDD02]. For example, the Haar mother wavelet has only one moment, so it has the first order derivative [Abhy15]. Therefore, the Haar mother wavelet can discover the hidden discontinuity and the hidden jump in the input signal with the first order derivative. However, Haar mother wavelet cannot detect the hidden discontinuities with higher order. on the other hand, the *Daubechies 2* (db2) mother wavelet has two vanishing moments, so it finds out the hidden abnormalities and singularities in the input signal with the second order derivative. As a result, db2 is a more appropriate mother wavelet [Abhy15].

There is strong connection between the number of vanishing moments of the different mother wavelets and their capability to detect irregularity in a signal. Also, vanishing moments show the complexity of the mother wavelet. The higher number of vanishing moments illustrates more complexity in a mother wavelet [Chun10]. Detecting discontinuities and irregularities in the input signal is determined through vanishing moments. As the degree of a polynomial increases, there will be more turns in the shape of the polynomial. For example, with a second-degree polynomial, there is one turn in the figure of the polynomial while for a third-degree polynomial, there are two turns. In other words, depending on the degree of polynomial, there are *N-1* turns [Lect16]. Moreover, the number of mother wavelet coefficients is two times that of the vanishing moments [SaKi06].

Detecting sensitive activities in a signal requires detection of irregularities, such as spikes or jump discontinuities that have derivatives of a higher order than the regular derivative patterns. Selecting or creating a mother wavelet is necessary to detect discontinuities and anomalies that

have high degree polynomials' structures. For this reason, the concept of maximizing the order of vanishing moments was introduced. However, between maximizing the order of vanishing moments and the computation cost there is a trade-off. The high order of vanishing moments can represent a complex signal, but the computation cost is increased. The low order of vanishing moments has low computation cost, but it can represent just a simple signal.

The third condition, finite length or fast decay in time, means the mother wavelet is compactly supported. The mother wavelet is zero outside of its set polynomial coefficients [Chun10][SoLa05]. Therefore, the compact support property provides good localization in time and frequency [AbDD02]. As a result, the length of a mother wavelet is finite, and its length can characterize a mother wavelet. A mother wavelet with a short length has a low computation cost, but it represents a simple signal so the mother wavelet cannot detect discontinuities with high order of complexity. A longer mother wavelet represents a complex signal, but the computation cost is increased. Therefore, the size of support is dependent on the length of the mother wavelet, and its length is important to select or design a mother wavelet.

#### *3.1.2.4 The Multiresolution Analysis*

To construct an orthogonal mother wavelet, *multi-resolution analysis* (MRA) forms the theoretical basis of the implementation, and the Mallat algorithm realizes the wavelet transform [GaYa11]. According to MRA, a signal can be decomposed into two parts: approximate information and detailed information. In the Mallat algorithm, the signal passes through a filter bank, which splits the input signal into two hierarchical bands [Mall89]. Figure 3.4 shows a filter bank constructed using Mallat's algorithm.

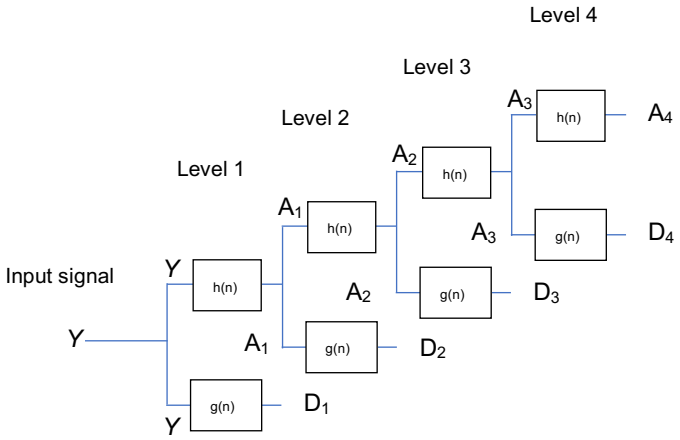



**Fig. 3.4:** The Mallat algorithm illustration with 4 resolution levels used to decompose a signal *Y* in order to compute its wavelet transform.

The Mallat algorithm decomposes signals into approximation (*A*) and detail coefficients (*D*) using a low-pass filter *h(n)* and a high-pass filter *g(n)* [Mall89]. This process is repeated with the approximation coefficients until the finest level of resolution is achieved. The low-pass filter *h(n)* must fulfil the following requirement [SoLa05]:

$$\sum_{n=1}^{N} h(n)=\sqrt{2} \tag{3.17}$$

where $N$ is the length of the low-pass filter. Also, the scaling function $\phi(t)$ is orthonormal to its translates, so the following relation must be satisfied for the low-pass filter [SoLa05]:

$$\sum_{n=1}^{N} h(n)h(n-2k)=\delta(k) \tag{3.18}$$

where $\delta(k)$ is the delta function.

If the filters are orthogonal, they are called quadrature mirror filters. In Fig. 3.4, *h(n)* and *g(n)* are the quadrature mirror filters if the condition shown in Eq. (3.19) is satisfied [GaYa11]

$$\| H^{*}x(t) \| + \| G^{*}x(t) \| = \| x(t) \| \tag{3.19}$$

where ||•|| shows the norm function, $x(t)$ shows the input signal, $H^*$ and $G^*$ form a pair of reconstruction filters. The relationship of the quadrature mirror filter coefficients is shown in Eq. (3.20) [Mall09]

$$g(n) = (-1)^{1-n} h(1-n) \tag{3.20}$$

where *g(n)* is the high-pass filter and *h(n)* is the low-pass filter. To achieve an orthonormal quadrature mirror filter, the Eq. (3.21) must satisfied in time domain [Veit05], or the Eq. (3.22) must satisfied in frequency domain [Mall09].

$$\sum_{k \in Z} | h_k |^2 = 1 \tag{3.21}$$

$$| H |^2 + | G |^2 = 2 \tag{3.22}$$

When the admissibility condition is met, the following equation for the high-pass filter can be extracted from Eq. (3.14).

$$\sum_{n} g(n) = 0 \tag{3.23}$$

### 3.1.3 Variance Fractal Dimension

A dedicated version of the *variance fractal dimension trajectory* (VFDTv2) was implemented to extract features from the general fractal data that rely on long-range dependence, for long stationary frames. The VFDTv2 was adjusted to consider all points including the boundary points of a dataset [GhKi18] [GhKi19]. Moreover, the variance equation of the data time series was adjusted to consider all the points.

The original algorithm was proposed by Kinsner in 1994 [Kins07] [Kins12] [Kins15], and applied to both deterministic and stochastic self-affine processes. The exclusion of the boundary

points is always recommended for extremely small stationary frames to reduce the inherent statistical bias. That bias becomes less significant for larger frames.

A fractal signal is a signal whose scaled-down version is similar to a part of itself [Kins15]. Also, a fractal signal is self-affine when different scales along different coordinates are using for a scale-down version of a signal. A network traffic signal is a nonstationary and a non-pure fractal signal. To analyze and measure a fractal signal complexity, a fractal dimension can be calculated. A fractal dimension interprets degree of meandering of a fractal signal [Kins15]. In this study, a polyscale analysis algorithm is used to measure the fractal dimension of the network traffic signal and extract distinguishing features of intrinsic characteristics of the signal [Kins15]. In contrast with multiscale analysis, where there is no correlation between outcomes, a polyscale analysis measures a signal with various scales, and its outcome requires all the scales be used simultaneously [Kins15]. To analyze and measure the complexity of a stochastic, non-stationary, non differentiable, broadband, dynamic, self-affine fractal, and correlated with long-range dependence signal, the *variance fractal dimension trajectory* (VFDT) is useful [Kins20]. The VFDT is used to extract features from the non pure fractal data that rely on long-range dependence as proposed originally by Kinsner [Kins07][Kins12][Kins20]. In addition, this version of the *variance fractal dimension* (VFD) is introduced because the input data is a non-pure fractal.

To analyze and measure the complexity of the signal using the VFD algorithm, the variance of its amplitude increments over a time increment has a power law relation with that time increment [Kins15].

$$\mathrm{var}[A(t_2) - A(t_1)] \sim |t_2 - t_1|^{2H} \tag{3.24}$$

where var is a variance function (the second moment), $A$ is the signal, $H$ is the Hurst exponent, and $t$ is used to represent time in a continuous signal. After sampling the data and obtaining the discrete signal, the Hurst exponent can be calculated from a log-log plot using Eq. (3.25).

$$H = \lim_{\Delta n \to 0} \frac{1}{2} \left( \frac{\log_2[\mathrm{var}(\Delta A)_{\Delta n}]}{\log_2 \Delta n} \right) \tag{3.25}$$

while $n$ shows discrete time in a discrete signal.

The output of the VFD, which is a measure of signal complexity and shown by $D_\sigma$, can be obtained using Eq. (3.26).

$$D_\sigma = E + 1 - H \tag{3.26}$$

where $E$ is the Euclidean dimension, and it is equal to one for the time series data. The result of the VFD is a parameter, $D_\sigma$, which its value is limited between one and two that is shown by the following condition

$$1 \le D_\sigma \le 2 \tag{3.27}$$

The real-time VFDv2 algorithm is characterized by a change in step 6 of the original VFD [Kins15], as described below [GhKi18] [GhKi19]:

1. Find a frame size, $T$, in which the input data series is stationary, otherwise, the algorithm's result is invalid.

2. Segment the data series according to the length of the frame size.

3. Find the number of samples in each frame, $N_T$, in the current frame by Eq. (3.28).

$$N_T = \left\lfloor \frac{T}{\delta_n} \right\rfloor \tag{3.28}$$

where $\delta_n$ is an interval between two successive sampled data.

4. Find the maximum *volume element* (vel) size, $\Delta t_{k_{\max}}$, in the current frame using Eq. (3.29).

$$\Delta t_{k_{\max}} = 2^{k_{\max}} \tag{3.29}$$

Where $2^{k_{\max}}$ is the maximum number of samples in each interval, and $k_{\max}$ can be obtained from Eq. (3.30). A relationship between parameters is shown in Fig. 3.5. As an example, it is indicated that the number of vels within the frame size 8192 is equal to 2048, when the vel size is equal to four. If the vel size is equal to eight, then the number of vels within the frame is equal to 1024, etc.

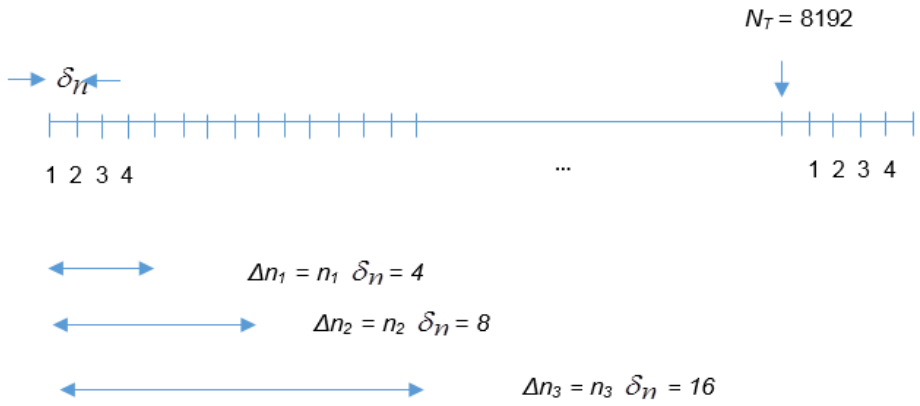



**Fig. 3.5:** Relationship between parameters $n_i$, $\delta_n$, $N_T$.

5. To find the output of the VFDv2 for the data series (signal $A$) in each frame ($N_T$), parameters from Eq. (3.30) to Eq. (3.34) are required. In the calculation program, a loop is designed to obtain the VFDv2 output within the lower bound, $k_{low}$, and the upper bound, $k_{hi}$ values as shown by Eq. (3.31) to Eq. (3.33).

$$K_{\max} = \left\lfloor \frac{\log_{10} N_T}{\log_{10} 2} \right\rfloor \tag{3.30}$$

$$K_{\text{hi}} = K_{\max} - K_{\text{buf}} \tag{3.31}$$

$$K_{buf} = \left\lceil \frac{\log_{10} 30}{\log_{10} 2} \right\rceil \tag{3.32}$$

$$K_{\text{low}} \geq 0 \tag{3.33}$$

6. To find the variance of the data series in the current frame, another nested loop is required. The parameters where described in Eq. (3.34) and (3.35) are required to find the boundaries of the second loop.

$$n_k = 2^k \tag{3.34}$$

$$N_K = \left\lceil N_T / n_k \right\rceil \tag{3.35}$$

The loop is repeated $N_K$ times, where $N_K$ represents the number of vels in the frame. Also, $N_K$ varies based on the values of $n_k$ in each step, where $n_k$ represents the size of vels in the frame. Therefore, by changing the vel size in each step, the concept of polyscale analysis is achieved using the VFDv2 algorithm. Equation (3.36) [Kins15]

$$\text{var}(\Delta A)_k = \frac{1}{N_K - 1} \sum_{j=1}^{N_k} [(\Delta A_{jk})^2 - \frac{1}{N_K} (\sum_{j=1}^{N_k} \Delta A_{jk})^2] \tag{3.36}$$

was adjusted to obtain the variance of the data series for stage *k*. Equation (3.37) is derived from Equation (3.36) for this study.

$$\text{var}(\Delta A)_k = \frac{1}{N_T - n_k} \sum_{j=1}^{N_k} [(\Delta A_{jk})^2 - \frac{1}{N_T} (\sum_{j=1}^{N_k} \Delta A_{jk})^2] \tag{3.37}$$

where the amplitude increment $\Delta A$ can be obtained from Eq. (3.38).

$$\Delta A_{jk} = A(j * n_k) - A((j-1) * n_k) \qquad \text{for } j = 1, \ldots, N_k \tag{3.38}$$

An important issue regarding Eq. (3.37) is that all points other than boundary points of the dataset, within the current frame, should be considered. Therefore, based on this concept, the variance is introduced through Eq. (3.37). Figure 3.6 (b) shows an example regarding a sliding vel with size of 16 that covers all points of the dataset within the current frame of data to calculate the VFDv2.

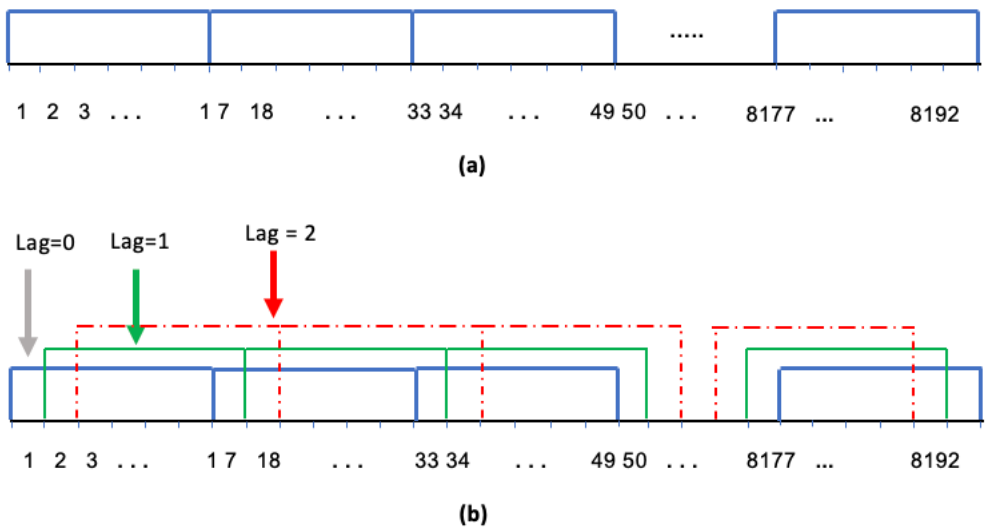



**Fig. 3.6:** (a) A graphic illustration describing the concept of boundary points of data to calculate the VFDv2.

(b) A graphic illustration describing the concept of covering all points of data to calculate the VFDv2.

Fig. 3.7 shows an example of a range of vel sizes from 1 to 4, and the number of vels in a frame size with 8192 data points.

7. Find the log-log plot values for each coordinate axes from Eq. (3.39) and Eq. (3.40).

$$X_k = \log[n_k] \tag{3.39}$$

$$Y_k = \log[\mathrm{var}(\Delta A)_k] \tag{3.40}$$

8. Compute the slope, $S$, from the log-log plot based on the least squares fit by linear regression according to Eq. (3.41).

$$S = \frac{K\sum_{i=1}^{K} X_i Y_i - \sum_{i=1}^{K} X_i \sum_{i=1}^{K} Y_i}{K\sum_{i=1}^{K} X_i^2 - (\sum_{i=1}^{K} X_i)^2} \tag{3.41}$$

9. Compute the Hurst exponent according to Eq. (3.42) from the slope of Eq. (3.41), which is extracted from Eq. (3.25)

$$H = (\frac{1}{2})S \tag{3.42}$$

10. Compute the VFDv2 based on Eq. (3.26).

11. Obtain the dimension for the next frame, until the end of the time series is reached. By a chain of the VFDv2 outputs, a VFDTv2 is calculated.

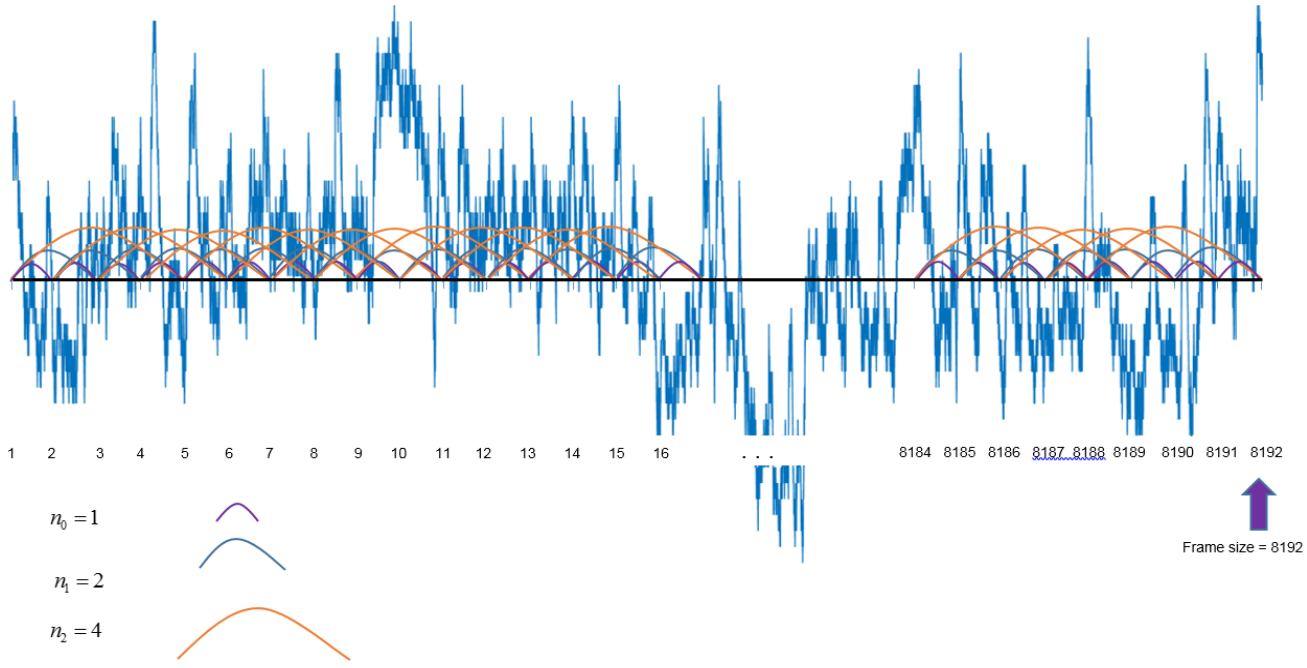



**Fig. 3.7:** Relationship between a vel size and the number of vels in each frame to calculate the VFDv2 output.

To verify the validity of the VFD algorithm, the VFD theoretical value of the white noise has to be equal to two [Kins20]. In addition, the output of the VFD is valid when time series input signals are bounded between one and two. More information about VFD verification is shown in Appendix G.

VFDTv2 has several advantages and disadvantages. First, the result of the VFD calculation is a real number between one and two. Second, the result is more accurate because all the data samples are included in the calculation of the VFD, so more valuable information can be extracted from the internet traffic that contains the DDoS attack. The disadvantage of VFDTv2 is that the proposed equation works for the Internet traffic data that contains DDoS attack datasets, and it should be tuned for another datasets.

### 3.1.4 Spectral Fractal Dimension

The *spectral fractal dimension* (SFD) can transfer a self-affine time series fractal into its power spectrum density [Kins20]. The SFD analyzes a time series signal in the frequency domain [SeKi18]. This spectrum reveals inherent properties of frequencies that are underlying characteristics behind the time series [Kins20]. For the fractal time series, the power spectrum density satisfies the following spectral power law equation [Kins20].

$$P(f) \sim \frac{1}{f^{\beta}} \tag{3.43}$$

where $\beta$ is called the power spectrum exponent. By obtaining logarithm from both side of Eq. (3.43), a linear relationship between the power spectrum and its frequency is obtained. Therefore, a regression line summarizes this relationship. The slope of the regression line is shown by the power spectrum exponent $\beta$ of the DDoS ITD as shown in Fig. 3.8.

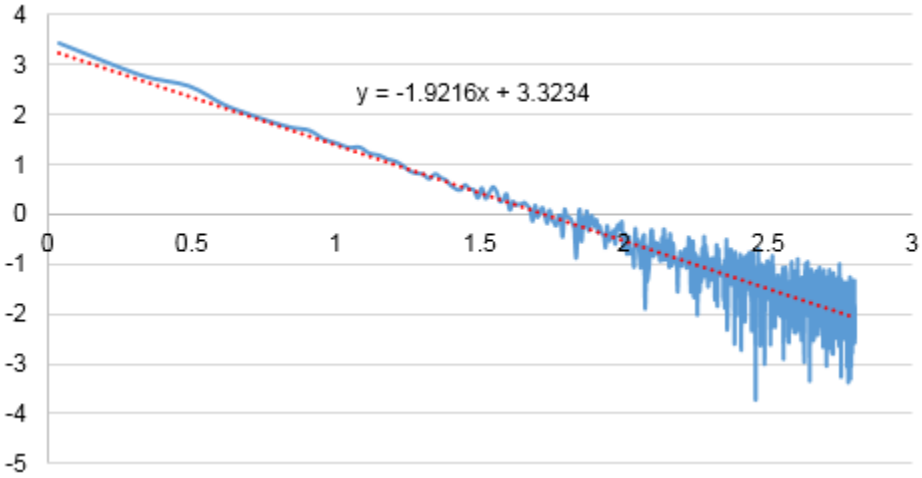



**Fig. 3.8:** The power density spectrum and the slope for the DDoS ITD.

This exponent represents the time series fractal. When the slope ( $\beta$ ) is zero, the time series is white noise. Therefore, the time series is not correlated. By increasing this slope, the correlation is increased. A complicated self-affine time series fractal such as the Internet time series data shows more than one $\beta$ , because this time series has a multi-fractal nature. The output of the SFD, which is a measure of signal complexity and shown by $D_\beta$, can be obtained using Eq. (3.44) [Kins20].

$$P(f) \sim \frac{1}{f^{\beta}} \tag{3.44}$$

Where $E$ is the Euclidean dimension, and it is equal to one for the time series data. There is relationship between the Hurst exponent and the power spectrum exponent that is shown in Eq. (3.45) [Kins20].

$$\beta = 2H + 1 \tag{3.45}$$

### 3.1.5 Zero Crossing

The zero-crossing measures the number of times that the amplitude of a signal crosses a threshold value of zero or any other threshold value within an interval [GoKi16][SeKi18]

[GhKi20a]. When the threshold value is zero, the zero crossing shows the rate of negative to positive and positive to negative mathematical sign changes of a signal within the interval. The zero crossing can find edges and rapid changes of a signal [Rang01][GhKi20a], and it can be used for feature extraction. The zero-crossing equation is shown as follows [JaBM13][SeKi18][GhKi20a]:

$$ZC = \sum_{n=-\infty}^{\infty} [|\operatorname{sgn}[x(m)] - \operatorname{sgn}[x(m-1)]|] w[n-m] \tag{3.46}$$

where represents the mathematical sign function and is defined as [JaBM13][SeKi18][GhKi20a]:

$$\operatorname{sgn}[x(n)] = \begin{cases} +1, & x(n) \geq threshold \\ -1, & x(n) < threshold \end{cases} \tag{3.47}$$

and $w[]$ represents a window containing a stationary segment of a signal and is defined as [JaBM13][GhKi20a]:

$$w[n] = \begin{cases} \frac{1}{2N}, & 0 \leq n \leq N-1 \\ 0, & otherwise \end{cases} \tag{3.48}$$

This measure is calculated in time domain, and it can be computed in real-time. Zero crossing rate is very useful for discriminating speech from noise and for determining start and end of speech segment [JaBM13][GhKi20a].

### 3.1.6 Turns Count

The turns count is a method for extracting features using changes in slope direction, rather zero crossings [Rang01][GhKi20a]. A turn occurs each time the sign of a signal's slope changes [Rang01][SeKi18][GhKi20a]. Turns count analyzes signals by determining the number of spikes occurring in a signal [Rang01]. A turn is calculated as shown in Eq. (3.49.a) or Eq. (3.49.b) [SeKi18] [GhKi20a].

$$tr_i = x(n) > x(n+1) \,\&\, x(n+1) < x(n+2) \quad (3.49.a)$$

$$tr_i = x(n) < x(n+1) \,\&\, x(n+1) > x(n+2) \quad (3.49.b)$$

where $tr_i$ is the turn occurring at a time interval that could be a stationary-window interval and $x(n)$ is an input signal. The turns count is shown in Eq. (3.50) [SeKi18][GhKi20a].

$$TC = \sum_{i=1}^{\infty} tr_i \quad (3.50)$$

### 3.1.7 Principal Component Analysis

The PCA is a statistical method used for reduction of data dimension and feature selection [Kala15][GhKi20b]. The PCA is used to process the features of input data when there is large number of features that need to be processed using a machine learning tool such as a neural network. The input can be illustrated using a lower dimension with a lower correlation. The lower correlation among the data can be obtained by mapping data to a new coordinate. The new coordinate with m-axes is obtained using the highest variance direction and the second highest variance direction until the $m^{th}$ highest variance direction of the data.

### 3.1.8 Distance

Receiving a large number of packets in a short amount of time renders the system unable to process the data, so a DDoS attack can occur. Therefore, the time difference between a packet-arrival time at $t$ and the packet-arrival time at $t-1$ is an important measurement in the detection of DDoS attacks in Internet traffic [GhKi20b]. To evaluate the neural network's ability to detect a DDoS attack in a time series, the time difference between packet-arrival times is important. In normal Internet time series, the time difference between a packet's arrival time is more than a packet's arrival time in an Internet traffic that contains DDoS. The number of packets

in the Internet traffic that contains DDoS interval is higher than in normal Internet traffic. As a result, the time difference between packets for both the normal cluster and the anomalous cluster is important. The distance between a newly arrived packet with an element from each cluster is the distance between the element and the arrived packet. This is shown in Fig. 3.9 [GhKi20b].

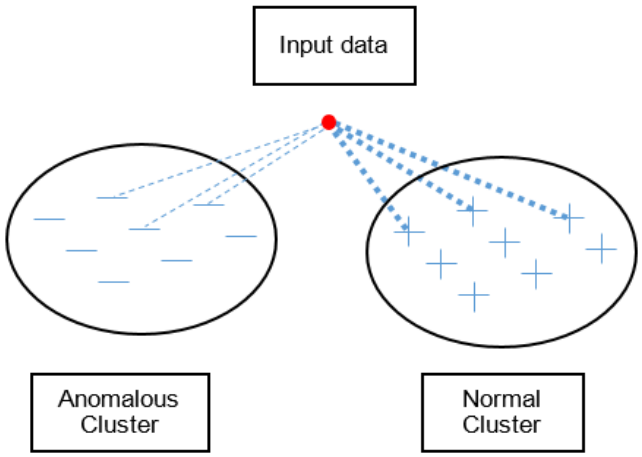



**Fig. 3.9:** The distance between a packet as input data and both the normal cluster and the anomalous cluster.

The cumulative summation of the distances between the newly arrived packets and the elements from a cluster is the total distance from the newly arrived packet and the whole cluster. Therefore, there are two total distances for a newly arrived packet: total normal distance and total anomalous distance.

## 3.2 Discussion of Features

The DDoS ITD dataset properties are important because, based on the characteristics of the dataset, a fitting feature extraction method can be selected or can be designed. The DDoS ITD is a non-stationary, stochastic, dynamic, broadband, and self-affine fractal signal that correlates with long-range dependence. Therefore, to analyze this signal, a feature extraction method should be used to extract distinguishing features of the intrinsic characteristics of the DDoS ITD. In

general, there are three categories of analysis measures to extract features: monoscale analysis, multiscale analysis, and polyscale analysis. A monoscale analysis measures a signal with one scale. This analysis is suitable for a simple signal with low complexity that has a Euclidian shape. However, the DDoS ITD is a complex signal, so the monoscale analysis is not enough for the DDoS ITD dataset to extract the hidden features from it. A multiscale analysis measures a signal with various scales, so the hidden characteristics of the DDoS ITD is extracted in each scale. An example of a multiscale analysis is DWT, so an adaptive mother wavelet is proposed and created to extract distinguishing features from the DDoS ITD. This adaptive mother wavelet for the DWT is discussed in chapter 4 With more details. A polyscale analysis measures a signal with various scales, and its outcome requires all the scales to be used simultaneously where there is a correlation between the outcomes. Therefore, the hidden features of the DDoS ITD are extracted in each scale, and the correlation between the hidden features are considered. As a result, the DDoS attacks can be detected with a higher detection rate with the polyscale analysis. An example of a polyscale analysis is VFD. Therefore, the VFD is focused on detail in this research in this chapter 3, and an improved version of the variance fractal dimension trajectory VFDTv2 is proposed and created to extract distinguishing features from the DDoS ITD.

## 3.3 Machine Learning

Machine learning is a field of study to teach computers without explicit programming, and this field of study deals with learning from tasks, gaining experience with data, and improving the performance [SoKN20]. In other word, machine learning is a type of programming code that requires to learn from its own input data. Moreover, cognitive machine learning is the science of using brain-inspired machine learning algorithms to develop a cognitive system that can think, solve problems, learn, and decide, so the cognitive system can act intelligently [SoKN20]. In

addition, cognitive machine learning is the science of using brain-inspired machine learning algorithms to develop a cognitive system that can think, solve problems, learn, and decide, so the cognitive system can act intelligently [SoKN20]. Artificial intelligence, machine learning, deep learning, neural network, decision tree and genetic algorithms are required for a computer system to build cognitive computing that mimics human thought. Moreover, cognitive machine learning is the science of using brain-inspired machine learning algorithms to develop a cognitive system that can think, solve problems, learn, and decide, so the cognitive system can act intelligently [SoKN20]. To improve the performance of cognitive computing not only artificial intelligence and machine learning algorithms are used such as deep learning, neural network, decision tree and genetic, but also polyscale analysis can be used. In this research, an analysis of DDoS attack detection is performed, using machine learning, deep learning, evolutionary algorithm, polyscale analysis and the adaptive system concept.

Machine learning programming systems are quite varied, but they can be classified in the following categories. The first category is based on supervised, unsupervised or DRL. The second category is based on online or batch learning. The third category is based on comparing new data points to known data points or detecting patterns in the input.

In supervised learning, the training data includes the desired solutions that are called labels. These labels are used for classification or regression problems. Some of the most important supervised learning algorithms, which are used in this research are: support vector machine, and artificial neural network. In unsupervised learning, the system tries to learn without any supervisor, and the training data is unlabeled. Here are some of the most important unsupervised learning algorithms: clustering, k-Means, PCA.

Supervised learning is learning from a training set of labeled examples provided by a knowledgeable external supervisor. In supervised learning, the input data must be labeled. Unsupervised learning is about finding hidden structure in collections of unlabeled data. The input data is not labeled in unsupervised learning. In these two cases, the supervised or unsupervised models have to train, then they can be used without any further changes to the models. DRL similar to unsupervised learning does not need any labeled data, but it is different from unsupervised learning. Reinforcement learning tries to maximize a reward signal instead of trying to find the hidden structure. In DRL, a model is continuously improved based on the processed data, actions and rewards from actions, so it can overcome the generalization problem. Also, the goal in unsupervised learning is to find similarities and differences between points. Thus, DRL is the third type of machine learning while it can learn from its actions based on rewards similar to the way humans learn from experience.

## 3.3.1 Supervised Learning Models

In this section, the supervised learning models that are used in this research are introduced. The models are based on: an *artificial neural network* (ANN), a *convolutional neural network* (CNN), a *support vector machine* (SVM).

### *3.3.1.1 Artificial Neural Network*

Artificial neural network (ANN) is a learning model and efficient classifier, which has been influenced by human learning [Hayk09]. ANN is shown by an interconnected network of neurons that send messages to each other. This network of neurons has weights, and the weights can be adjusted based on previous experience. Therefore, this characteristic makes ANN able to learn from its environment.

Figure 3.10 shows the structure of multilayer feed-forward ANN. The input layer obtains input from its environment and sends the input to the hidden layer. The purpose of the hidden layer is to connect the input layer and the output layer to extract more information and higher-order statistics from the input layer. The response of a group of neurons of ANN is delivered by the output layer. Each circle node represents an artificial neuron and each line represents a connection from the output of one artificial neuron to the input of another artificial neuron.

ANN has three important characteristics [MoKi02]. Firstly, a neuron is nonlinear, so interconnected networks of neurons are nonlinear. By this important property, modelling a nonlinear process is possible by ANN. For example, power consumption is a nonlinear process that can be modeled by ANN. Secondly, ANN can map between inputs and outputs, so ANN is proper for pattern classification. Thirdly, ANN can be adapted to inputs, so weights in ANN are changed based on the environment. As a result, a neural network which operates in a specific environment can deal with a new environment easily.

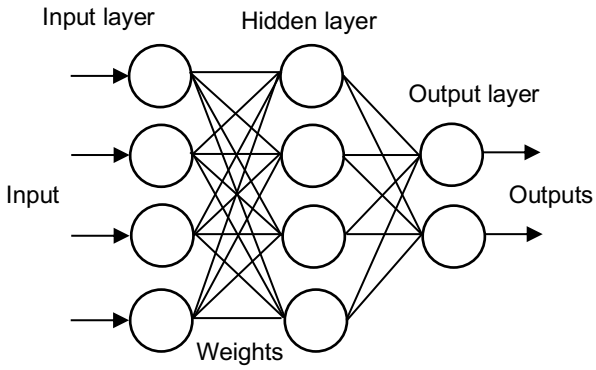



**Fig. 3.10:** A three-layer feed-forward ANN.

ANN has three important characteristics [MoKi02]. Firstly, a neuron is nonlinear, so interconnected networks of neurons are nonlinear. By this important property, modelling a nonlinear process is possible by ANN. For example, power consumption is a nonlinear process

that can be modeled by ANN. Secondly, ANN can map between inputs and outputs, so ANN is proper for pattern classification. Thirdly, ANN can be adapted to inputs, so weights in ANN are changed based on the environment. As a result, a neural network which operates in a specific environment can deal with a new environment easily. On the other hand, ANN methods such as the backpropagation algorithm can be used to classify normal and anomalous behaviors in a signal.

The input layer propagates input to the hidden layer, so the input values are multiple by a corresponding weight of each branch in the hidden layer and then summed [Kins20]. The output of the hidden layer is propagated to the output layer. Error can be calculated as difference between output and desired output. The purpose of the backpropagation algorithm is minimizing this error. Therefore, minimizing of the cost of error (cost function) can be defined as:

$$E = \frac{1}{2q} \sum_{i=1}^{q} [y_i - \hat{y}_i]^2 \tag{3.51}$$

where $y_i$ defines the desired output of the $m^{th}$ input training example, $\hat{y}_i$ represents the actual output of the neural network and $q$ represents the total number of training examples.

*3.3.1.2 Convolutional Neural Network*

*Convolutional neural networks* (CNNs) are very similar to ordinary artificial neural networks. They are made up of neurons, non-learning pathways, and learning pathways which are called weights and biases. A simple CNN is a sequence of layers, and every layer propagates one volume of activations to another layer through a differentiable function. This research study uses a one stage CNN that consists of six main layers [JKRL09] [Karp15] as shown Fig. 3.11. The six layers are introduced in the following ways [JKRL09] [Karp15] [Jiay16]:

1. The input layer holds the raw values of the data.

2. The convolution layer computes the output of neurons that are connected to local regions in the input. The convolution layer measures the similarity between the input and the coefficients of the filter. The convolution layer computes a dot product between weights of a filter and a small region of the input area. The weights of the filter in the convolution layer are shown by array *w* in Fig. 3.11. Every pathway has its own parameter. The parameters in this layer are weights of the filter *w* and its bias *b* that should be trained by the gradient descent algorithm.

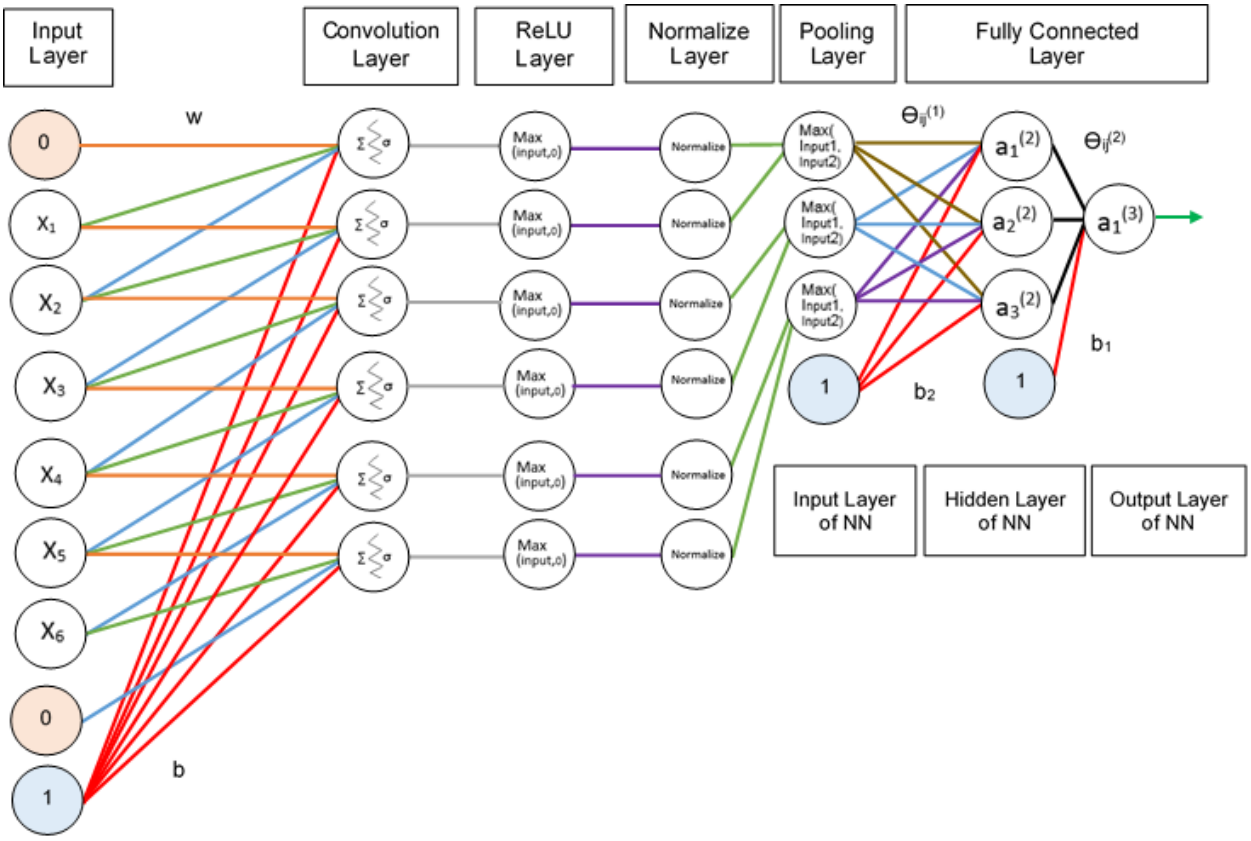



**Fig. 3.11:** One stage convolutional neural network architecture.

3. The *rectified linear unit* (ReLU) layer applies an elementwise activation function, such as the *max(0,x)*. The ReLU activation function does not have any saturation problems [KrSH12]. In contrast, the sigmoid activation function saturates and kills gradients. When a neuron's activation saturates at either tail of 0 or 1, the gradient at these regions is almost zero. For example, if the initial weights are too large then most neurons become saturated and the network barely

learns. ReLU was found to greatly accelerate (e.g. a factor of 6) the convergence of stochastic gradient descent compared to the sigmoid/tanh functions due to its linear, non-saturating form [Fere16]. The ReLU layer implements a fixed function, and it does not have any parameters for training. The ReLU activation function is shown in Eq. (3.52).

$$f(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{3.52}$$

4. The normalization layer aids generalization [KrSH12]. This layer does not have any parameters for training. Normalizing the input data for this layer causes it to be approximately at the same scale (range from 0 to 1 or –1 to +1). In general, machine learning algorithms, if the range of data values varies widely, objective functions cannot work properly without normalization [Fere16]. Also, feature scaling applied to gradient descent converges much faster when the data is normalized. Data normalization makes the training of the network faster, the memory more efficient and yields more accurate forecast results. Normalization overcomes the unlimited growth of learnable weights [Hayk09]. The normalization function [KrSH12] is shown in Eq. (3.53).

$$b^i_{x,y} = a^i_{x,y} / (2 + 0.0001 \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a^j_{x,y})^2)^{0.75} \tag{3.53}$$

5. The pooling layer performs a down sampling operation on the data received in this layer [Karp15]. The pooling layer does not have a learning function. In this study, the max pooling function is used. An example is shown in Fig. 3.12.


**Fig. 3.12:** an example of max pooling function.

6. The fully-connected layer computes the class scores. This computation is based on the labels attached to each data input from the training set. In this layer, the multilayer feed-forward

ANN is used. Similar to ordinary neural networks, each neuron in this layer will be connected to all the neurons in the network. Every pathway has its own parameter or weight. These weights are connected to the hidden (matrix of weights $\theta_{ij}^{(1)}$ and bias $b_1$) and the output sublayers (matrix of weights $\theta_{ij}^{(2)}$ and bias $b_2$) that are shown in Fig. 3.11. The following functions are used to train ANN.

6.1. The sigmoid function ( $g$ ) is used as an activation function as shown in Eq. (3.54).

$$g = \frac{1}{1+e^{-z}} \tag{3.54}$$

6.2. The logistic regression function is used as a convex cost function and this can be seen in Eq. (3.55) [Fere16].

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} -y_i \log(\hat{y}_i) - (1-y_i)\log(1-\hat{y}_i) \tag{3.55}$$

where $x_i$ is the randomly selected training sample $i$ and $y_i$ is its desired output. Also, $\hat{y}_i$ is the output of the forward propagation.

6.3. The online weight method is used to train ANN to rapidly update the weights of its neurons. The stochastic gradient descent algorithm is used to teach ANN how to update its weights. This algorithm is shown in Eq. (3.56).

$$\theta_{ij}^{(k)} = \theta_{ij}^{(k)} - \alpha \frac{\partial}{\theta_{ij}} J(\theta) \tag{3.56}$$

In this equation, the $\frac{\partial}{\theta_{ij}} J(\theta)$ is calculated using Eq. (3.57).

$$\frac{\partial}{\partial \theta_{ij}} J(\theta) = \frac{1}{m}\sum_{i=1}^{m} (\hat{y}_i - y_i) x_i^j \tag{3.57}$$

where $x_i^j$ is the $j^{th}$ feature of the selected training sample *i*.

*3.3.1.3 Support Vector Machine*

The SVM algorithm obtains an optimal hyperplane to classify the data based on a large margin separation and a kernel function. The SVM tries to find the maximum distance between that hyperplane and the closest data point on either side of the hyperplane. The optimal hyperplane or the decision boundary is shown in Fig. 3.13.

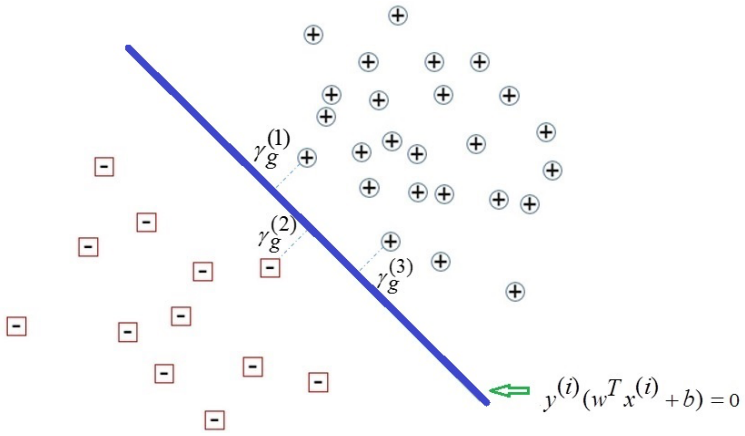



**Fig. 3.13:** Optimal hyperplane to classify the positive and negative data with the support vector machine algorithm. (After [Fere16])

The SVM formulation is the equivalent of maximizing the margin while minimizing error. The cost function can be obtained using Eq. (3.58) with the constraint shown by Eq. (3.59) [Fere16].

$$C = \min_{w,b}(\frac{1}{2}\| w \|^2) \tag{3.58}$$

$$1 - y^{(i)}(w^T x^{(x)} + b) \le 0 \quad i = 1,...,m \tag{3.59}$$

The gradient descent algorithm in Eq. (3.60) is used to minimize the cost function *C*.

$$w_j \leftarrow w_{j-1} - \eta_t \nabla_{w,b} C(w,b) \tag{3.60}$$

### 3.3.2 Deep Reinforcement Learning

The DRL is an area of machine learning. It is a computational approach in which machines learn from interacting with the environment using unlabeled data [Silv15]. The DRL purpose is to learn how to take an action to maximize the long-term rewards [SuBa16].

#### *3.3.2.1 Deep Reinforcement Learning Advantages*

The DRL has several advantages [SoKN20]. First, it can classify the unlabeled dataset. Second, it is a reward driven continuous learning process. Since it tracks previous experience, it can improve its actions on its own. Third, there is no need to extract a model from the training dataset, so the over fitting and under fitting problems are resolved. Fourth, it has to exploit what it already knows in order to obtain a reward, but it also has to explore other options in order to make better action selections in the future. Fifth, it has high performance when the data size is large. Sixth, it is used when there is a lack of domain understanding. Seventh, it is used to solve complex problems.

#### *3.3.2.2 Deep Reinforcement Learning Elements*

In reinforcement learning, a software agent makes observations and takes actions within an environment, and in return it receives rewards. The main elements of a DRL system are a set of environment states or a model of the environment, a set of actions, a policy that uses actions to transient between states, and a set of rules that determine the reward for each state [Frei15] [Silv15]. In Reinforcement Learning, a software *agent* makes *observations* and takes *actions* within an *environment*, and in return it receives *rewards*. Its objective is to learn to act in a way that will maximize its expected long-term rewards. Note that there may not be any positive rewards at all.

*3.3.2.3 Deep Reinforcement Learning Structure*

The DRL structure is shown in Fig. 3.14. A machine learning algorithm is used as an agent. However, the agent has to be an unsupervised learner, so that it can detect anomalous behaviors of unlabeled input data. The DRL learning method is based on rewards. The reward is used for finding the true value as the tag, so the approximate value is simulated as the output of the neural network. Therefore, it is essential to obtain or estimate the true value of the tag estimation methods.

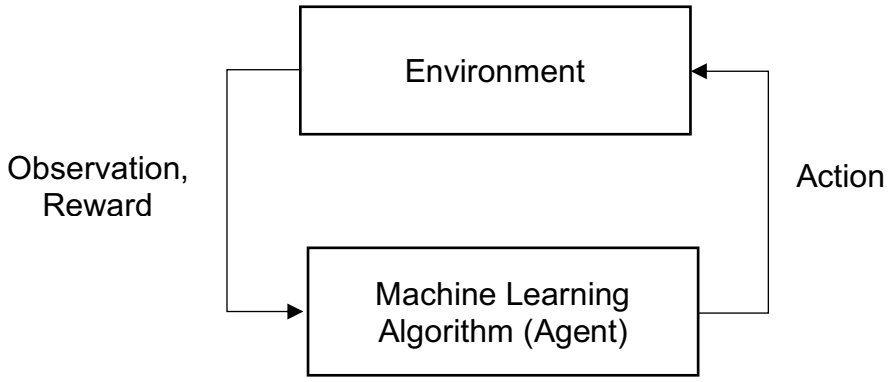



**Fig. 3.14:** Deep reinforcement learning structure. (After [SuBa16] [Silv15] [Frei15])

The agent tries to find the best policy to select an action in order to obtain the maximum reward based on whether the environment is known or unknown. When the agent knows the environment model, maximizing the reward is reduced to control [Silv15]. When the environment model is unknown to the agent, maximizing the reward is reduced to estimate [Silv15]. Also, the environment can be shown by *Markov decision process* (MDP) and the value of each state can be obtained using the Bellman equation [Silv15]. According to the MDP, a state is dependent only on the previous state and previous action in the environment that the agent is running. Therefore, the history of the system can be understood as a set of state, action and reward tuples: ($S_1$, $A_1$, $R_1$, $S_2$, $A_2$, $R_2$, ...). When the agent knows the environment model, maximizing the reward is reduced to control. In this situation, the IDSA optimizes the value function of the environment. When the

environment model is unknown to the agent, maximizing the reward is based on prediction. In this situation, the agent estimates the value function of the unknown environment, because it does not know about transitions and rewards. The prediction or estimation algorithms are *Monte Carlo* (MC), *Temporal Difference* (TD), *State Action Result State Action* (SARSA), and Q-learning to list a few, that are used for updating the state value function. The agent tries to find the best policy (or action) in order to obtain the maximum reward based on whether the environment is known or unknown.

#### *3.3.2.4 Taxonomy of Solving Deep Reinforcement Learning Problems*

There are three different techniques to solve DRL problems: Q-Learning, policy gradient and actor-critic algorithms [SuBa16][Silv15][VZHX21]. In this research, the policy gradient technique is used to design an IDS. In DRL, a policy can be parameterized using a neural network [HeZh19]. Moreover, it is possible to estimate a gradient to improve the parameterized policy. Using the policy gradient technique, the gradient estimation is implemented. Optimization of the policy tries to optimize the expected rewards. To explore the policy space there are two methods: a gradient ascent and a genetic algorithm. In this research, a gradient ascent method is used to maximize the rewards.

#### *3.3.2.5 Problem Definition*

The DRL problem in this research is detecting DDoS attacks. One of the problems with DRL is that it requires a sophisticated exploration mechanism. Randomly selecting an action without a probability distribution estimation gives a poor performance. Therefore, an appropriate training policy is important when using DRL. In this research, a novel policy structure using a CNN based on a polyscale analysis algorithm is used, which is described in detail in Ch.4.

*3.3.2.6 Policy Gradient Background*

The algorithm used by the software agent to determine its actions is called its policy. The policy is an algorithm, including a neural network, and it is a stochastic policy. The agent tries to find the best policy to determine the best actions to maximize its rewards. A policy can be any algorithm such as a neural network, and the policy can be either deterministic or stochastic. In this research, the aim is to design an *intrusion detection system agent* (IDSA) based on a policy gradient that detects DDoS attacks with the highest detection rate in a stationary frame size while the data is unlabeled. The policy used in this research, is a developed *policy gradient based deep reinforcement learning* (PGDRL) algorithm that tries to optimize the parameters of a policy using an optimization technique to gain a higher reward. The policy gradient algorithm tries looking for correlations between actions, state transitions and rewards to maximize the IDSA total reward over time.

## 3.4 Discussion of Machine Learning

The supervised learning methods that are introduced in this chapter, such as the ANN and CNN, can test the input data in real-time and adapt to various environments. However, to train the input, they need labels, which is not suitable for real-world applications. Therefore, further research is necessary to apply a non-supervised learning method. In the unsupervised learning method, the system tries to learn without any label, and the training data is unlabeled. As a result, this learning method is more suitable for real-world applications than the supervised learning method. However, unsupervised learning tries to understand patterns and discover the output; if the patterns in the environment are changed, the unsupervised learning cannot respond to new patterns. In general, the DRL deals with a large complex problem space. Also, the DRL learning does not need any label to train its input, and it has a continuous learning mechanism that is based

on maximizing the reward. Therefore, the system is capable of responding to unexpected events in the environments.

## 3.5 Summary of Chapter 3

In this chapter, the proposed feature extraction methods and machine learning models are presented. The mono-scale analysis such as PCA, statistical features (mean, median, mode, variance, standard deviation, skewness, and kurtosis), autocorrelation, ZC and TC to analysis Internet traffic carries DDoS attack. Statistical features such as kurtosis are important to analyze Internet traffic carrying DDoS attacks because positive values of kurtosis indicate heavy tail distributions. Moreover, the heavy tail distribution indicates the dataset is a fractal signal. When a signal is fractal, a multiscale analysis and especially a polyscale analysis should be used to extract hidden features from the fractal signal. In addition, the monoscale, the multiscale, and the polyscale feature extracting analysis are introduced or created to extract hidden features in the input time-series signal. Moreover, a dedicated version of the VFDTv2 was introduced in this chapter. They are introduced to increase the sensitivity of an IDS to improve the DDoS attack detection. Moreover, supervised deep learning algorithms are introduced to classify normal behaviors from anomalous behavior in frequently changing environments. To have a realistic IDS, the DRL method to detect anomalous behaviors in unlabeled data is introduced for real-world applications. These components and their combinations are used to design the anomaly detections in the following chapters.

# Chapter 4

# PROPOSED INTRUSION DETECTION SYSTEMS

In this chapter, the anomaly detection algorithms for detecting the DDoS attacks and the power consumption attacks are introduced. The designs of these algorithms are based on supervised learning and deep reinforcement learning and a combination of the machine learning algorithms.

## 4.1 The First Anomaly Detection Algorithm Design

The steps in the design of the first anomaly detection algorithm for detecting anomalous power consumption attacks are described below [GhKF16] [GhFK16]:

First, power consumption patterns of clients are obtained. Second, wavelet coefficients of power consumption are calculated by DWT. As a result, features of power consumption are extracted using the Daubechies wavelet transforms. To gain a higher anomaly detection rate,

different Daubechies wavelet transform functions are used. Third, the VFD are extracted to measure the complexity of the extracted coefficients in the second step. Fourth, the lower bound and the upper bound thresholds are used to detect anomalous behaviors in the power consumption patterns. Fifth, an ANN is used to enhance the detection rate of anomalous behaviors in a short duration of the attacks. Sixth, about 80% of the features are used to train the ANN system to extract normal and anomalous behaviors of power consumption patterns. Finally, the remaining 20% of the coefficients are used to test and classify normal power consumption and anomalous data patterns.

The proposed scheme for detecting the anomalous power consumption attack is shown in Fig. 4.1. The first proposed anomaly detection algorithm detects the power consumption attacks, but it had two shortcomings. The dataset that is used for this algorithm is simulated because real cyber domain smart-grid infrastructure data is not accessible at that time due to confidentiality and security reasons. Therefore, DDoS attacks detection data from a cyber computer system is used instead. Moreover, this algorithm produces an unsatisfactory worst-case scenario anomaly detection rate, as discussed in Chapter 6. As a result, the second anomaly detection algorithm for detecting DDoS attacks is proposed.

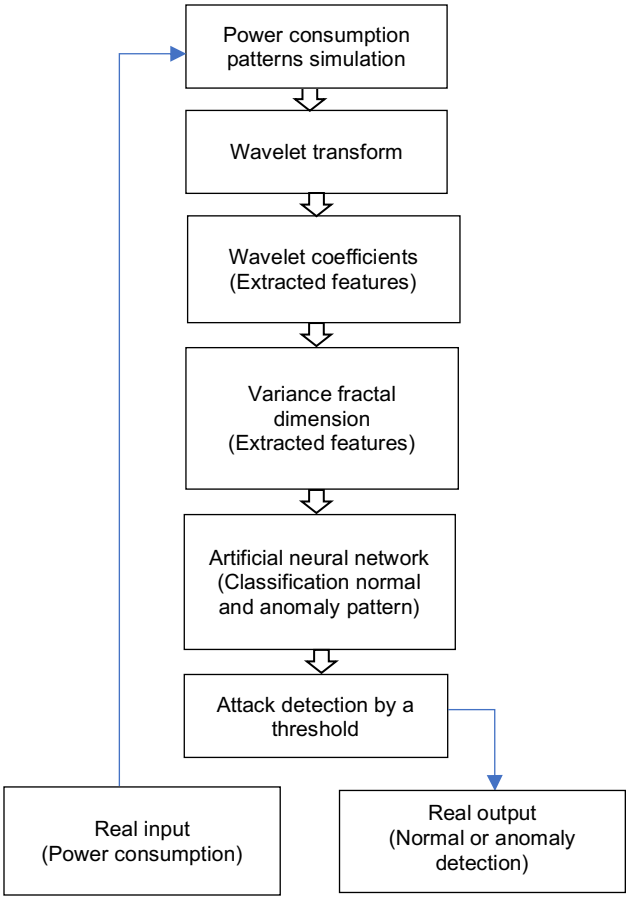



**Fig. 4.1:** Proposed scheme for detecting an anomalous power consumption attack.

## 4.2 The Second Anomaly Detection Algorithm Design

The main procedure of the second anomaly detection algorithm for detecting DDoS attacks consists of two steps: data pre-processing and data processing [GhKF17]. After the raw DDoS attack data is fetched, it is sent to the network for pre-processing. In the pre-processing step the most important and reliable underlying features are extracted using digital signal processing tools such as DWT. These features are extracted in the following way. First, the wavelet coefficients of the dataset are calculated using Daubechies wavelet transform. Second, the size of the window in

which the data is stationary is defined. Third, the output of the DWT is normalized using z-score normalization within the window size.

In the processing step, the pre-processed input is sent to the CNN where distinguishable features are extracted. Based on these extracted features the CNN is trained. Using 50% of the dataset, the CNN is trained to automatically extract the features and obtain coefficients from the dataset. Then, using about 50% of the remaining dataset, the CNN is tested with the obtained coefficients. Finally, the output of the testing phase is classified as either normal data or anomalous DDoS attack signals.

In general, a CNN deals with a large number of parameters, which decreases the training speed. In addition, the classification accuracy of a CNN is improved by increasing the number of its stages, which further increases the training time of the CNN. To overcome these problems, the pre-processing step, is used. The pre-processing step increases the detection rate and reducing the detection processing time.

This second anomaly detection algorithm improves the detection rate by approximately 30% in comparison to the ANN that is used in the first anomaly detection algorithm. However, the DDoS attacks detection algorithm should have a higher detection rate, so the third algorithm is designed to improve the detection rate further.

## 4.3 The Third Anomaly Detection Algorithm Design

The third proposed anomaly detection algorithm for detecting the DDoS attacks consists of four steps: preparing data, data pre-processing, data processing, and data post-processing. The anomaly detection algorithm is described below [GhKi18][GhKi19]:

1. Preparing data for processing

1.1. The raw data that contains a source IP address, a destination IP address, length of packet, protocol and packet- arrival time for each packet is fetched and tagged as normal or anomalous.

1.2. A new attribute called a packet *arrival time-series signal* (ATS) is calculated by using the difference between a packet- arrival time at *t* and the packet-arrival time at *t-1*.

2. Pre-processing

The ATS data is passed to the pre-processing step for feature extraction. The features are extracted in two different ways. In the first way, the ATS data is first passed to the improved version of the *variance fractal dimension* (VFDv2) algorithm described in section 3.1.3.

In the second way, the ATS data and the result of the VFDv2 algorithm are passed to the DWT. The wavelet coefficients of the ATS data and the output of the previous step are calculated by using the Daubechies wavelet transform. The wavelet coefficients can be obtained by convolution of the input signal and the Daubechies basis function within an adjustable window size. This can be achieved using Eq. (4.1) [GaYa11], where $s$ is the scaling parameter, $\tau$ is shifting parameter, $\psi(.)$ is a discrete basis function and the symbol $*$ denotes the complex conjugate.

$$wt(s,\tau) = \frac{1}{\sqrt{s}} \sum_{n=-\infty}^{+\infty} x(n)\psi^*\left(\frac{n-\tau}{s}\right) \tag{4.1}$$

In addition, the size of the window in which the data is stationary is defined. The z-score normalization within the window size is used to normalize the output coefficients of the DWT.

3. Processing

The processing step involves the CNN that was developed and described in detail in [GhKF17]. This CNN consists of six layers. The layers are the input layer, the convolution layer, the *rectified linear unit* (ReLU) layer, the normalization layer, the pooling layer and the fully-connected layer, as shown in Fig. 4.2.

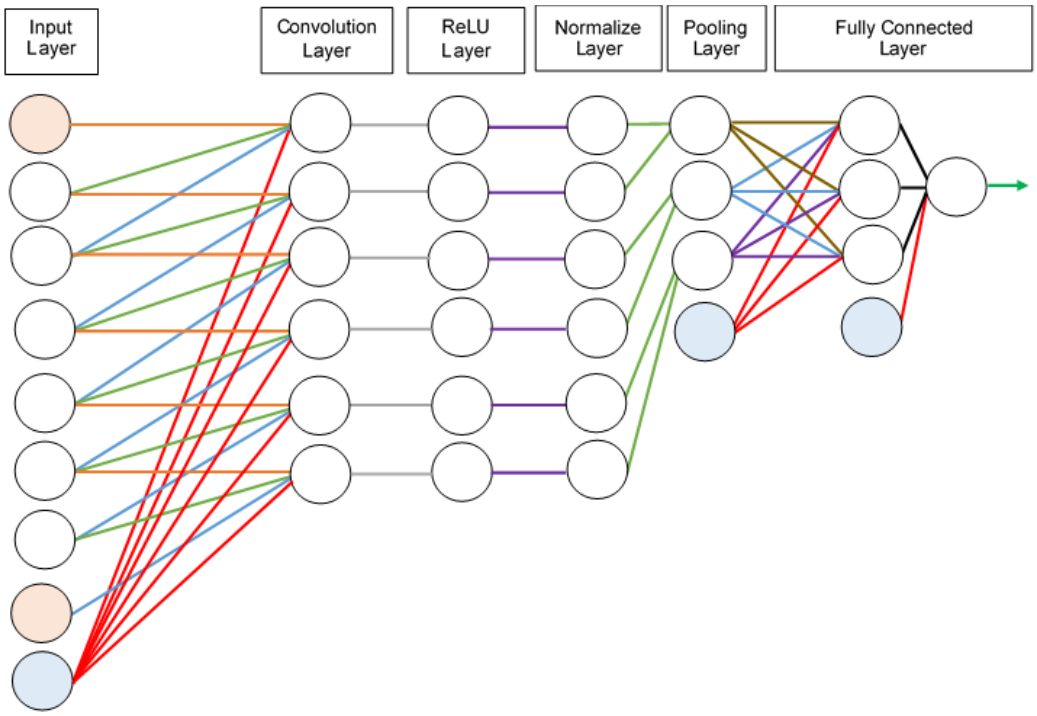



**Fig. 4.2:** A single-stage convolutional neural network architecture.

The processing step is as follows:

3.1. The raw data, the ATS data and the pre-processed data are sent simultaneously to the input layer of the CNN as one-dimensional time-series signals.

3.2. The CNN is then trained. The logistic regression function is used as a convex cost function. Furthermore, the stochastic gradient descent algorithm is used to train the CNN to update its weights and biases through a backward propagation algorithm.

3.3. Based on the learned parameters of the training phase, the output of the testing phase classifies the data into two classes: normal and anomalous behaviors.

This single-stage CNN comprises the processing step.

4. Post-processing

The post-processing step is used to overcome the duration of training the CNN. In general, a CNN involves a large number of parameters, which decreases the training speed, so this study uses an SVM as the post-processing step.

These four steps are shown in Fig. 4.3.

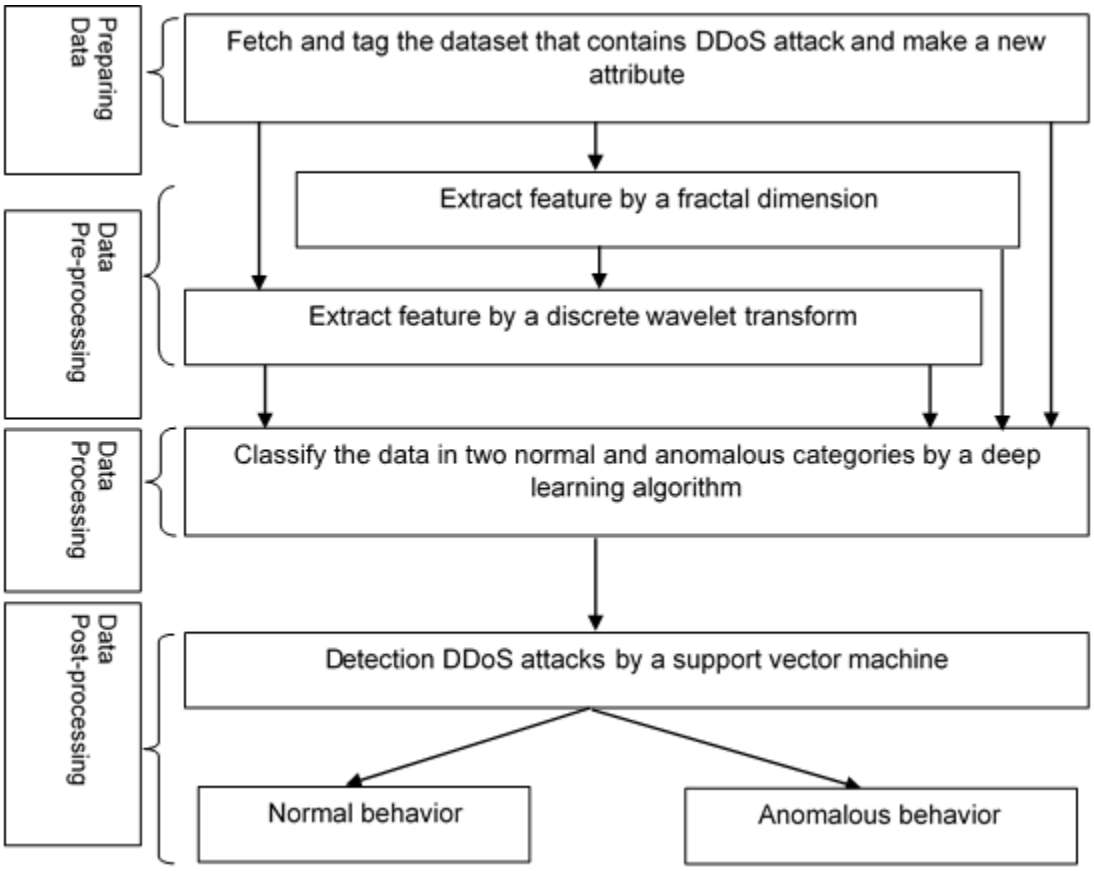



**Fig. 4.3:** The algorithm of DDoS attack detection.

This third anomaly detection algorithm improves the detection rate by approximately 7% in comparison with the second anomaly detection algorithm. However, the DDoS attacks detection algorithm should have a higher detection rate, so the fourth algorithm is designed to further improvement detection rate.

## 4.4 The Fourth Anomaly Detection Algorithm Design

The steps in the design of the fourth anomaly detection algorithm for detecting DDoS attacks are described in this section. First, a mother wavelet that adapts to its input signal is created [GhKi21a]. Second, a cost function is created for adapting to the input. As a result, the cost function is sensitive enough to find differences between the neural network output and the desired output [GhKi20b]. Third, an adaptive-wavelet convolutional neural network structure is created for the increased detection rate of the DDoS attacks [GhKi21b]. In addition, the input data features are extracted using the created mother wavelet as the preprocessing step for detecting DDoS attacks, and the created cost function is used for measuring how well the adaptive-wavelet convolutional neural network estimates its output.

### 4.4.1 Designing a Genetic Neural Network to Create an Adaptive-Mother Wavelet

The DDoS ITD relies on long-range dependence [GhKi20a] [GhKi21d]. Therefore, there is a correlation among the time series data, which can be used to extract distinguishable features and patterns in the Internet traffic. These features are fragile, and sensitive tools like the *discrete wavelet transform* (DWT) method are needed to extract these hidden features. The most challenging step in the DWT is to select its mother wavelet [GhKi21a]. To have an efficient analysis of feature extraction using DWT, a relevant mother wavelet should be considered for the given application. To choose a relevant mother wavelet, the concept of vanishing moments is used. The vanishing moment of a mother wavelet shows which polynomial has the highest correlation (similarity) with the data of the given application. The order of vanishing moments can represent the complexity of the signal. The lower the order of the vanishing moments has the smaller the

compact support (length of the mother wavelet) and the computation cost, but the lower the order of the vanishing moments represent a simple signal. The higher the order of the vanishing moments has the higher the compact support and represents a complex signal, but the computation cost is increased. Therefore, a reasonable choice is to use the smallest mother wavelet that gives satisfactory details and trends in a complex signal. In this research, a mother wavelet that adapts to its input signal most efficiently is created.

In this section a *genetic neural network* (GNN) used for creating adaptive mother wavelet. This section first describes the structure, characteristics and adaptive features of the GNN [GhKi21a]. Second, a multi-objective optimization based on genetic algorithm is explained for the GNN. Third, a cost function is applied for the GNN. The algorithm is illustrated with a dual coefficient GNN designed to create a mother wavelet for detecting the DDoS ITD as its input. The architecture of this design is shown in Fig. 4.4.

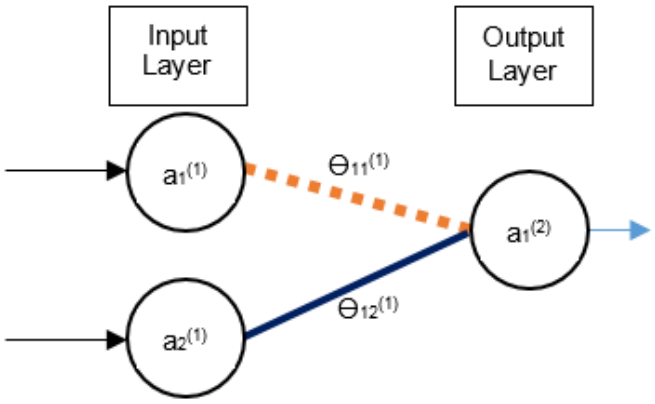



**Fig. 4.4:** The architecture of two coefficients designing adaptive-mother wavelet using genetic neural network.

Moreover, the GNN architecture with four coefficients is shown in Fig. 4.5.

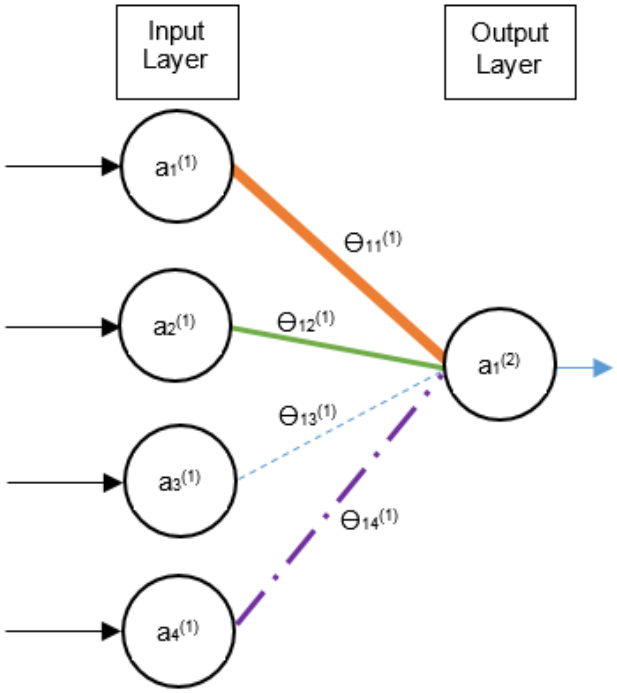



**Fig. 4.5:** The architecture of four coefficients designing adaptive-mother wavelet using genetic neural network.

#### 4.4.1.1 Designing an Architecture for the GNN

The architecture of the GNN has two layers: the input layer and the output layer. The input layer passes the input signal to the next layer (output layer). The number of nodes in this layer is adaptive, and it is equal to the number of features in the input data. Figure. 4.4 shows the input layer of the dual coefficient GNN, which uses the DDoS ITD dataset. The last layer is the output layer. The number of nodes in the output layer is equal to one.

#### 4.4.1.2 Designing a Multi-Objective Optimizer for the GNN

A *genetic algorithm* (GA) is a subset of evolutionary algorithms that is inspired by nature and biological processes such as parent selection, mutation, and crossover [Dang15][VaHC07]. In this research, a multi-objective optimizer based on a GA is used to find the best solution for creating an adaptive mother wavelet.

The GA is a universal optimizer that could be used for various type of problems. This base algorithm can search multiple parts of any large and complex search space simultaneously and is easy to implement [DiHJ17]. The GA is useful to find good sub-optimum solutions in a reasonable time in a big search space [Fere16].

The genetic algorithm is based on selecting individuals from the current population to produce the children for the next generation, and several members of the solutions in a single run of the algorithm are found. The genetic algorithm in each iteration improves a population of solutions. Over sequential generations, the population evolves toward optimal solutions or Pareto front [SHPC16].

A simple optimization algorithm cannot be used for creating adaptive mother wavelet for detecting the DDoS attacks because its architecture must satisfy several constraints or objectives (see Eq. (3.18), Eq. (3.21), Eq. (3.22) and Eq. (3.24)). Therefore, an adaptive multi-objective optimization algorithm based on a genetic algorithm such as a *Multi-objective optimization using an evolutionary algorithm* (MOEA) must be used [CoLV07][Dang15]. A MOEA is an improved and extended version of a genetic algorithm.

Since an adaptive wavelet GNN has several constraints, a MOEA can be used to optimize the adaptation of the wavelet coefficients. The elements of the MOEA as the optimizer of the GNN are explained as follows:

*1. Initialization of population*: in this MOEA structure, solutions are randomly generated for the first population.

*2. Parent selection:* one of the most important operators in the GA is parent selection because it can accelerate convergence to optimal solutions, whereas inadequate parent selection leads to inefficient solutions.

In this research, individuals are selected to crossover, and tournament selection is used. In *k*-ary tournament, *k* individuals are selected randomly from the population [Kala14][Fere16]. The selection can be with or without replacement. Therefore, an individual has a probability of *1/k* to be selected. Then, a probability mass function is assigned over *k* individuals based on their fitness, so an individual with higher fitness has higher probability to be winner among the *k* individuals.

*3. offspring generating*

3.1. Mutation: mutation operation is introduced to enhance the performance of genetic algorithms. Mutation operator in the GA introduces diversity, so the GA can escape and avoid local minima by preventing the population of solutions from becoming too similar to each other and create new solutions [JeME13].

In this research, the Gaussian mutation strategy is applied. In the Gaussian mutation, a random amount of a Gaussian distribution is added to each element of solutions (parents) to create a new offspring [Kala14]. Moreover, the selected solutions should pass a probability test. If the probability test is passed, the mutation is performed. This probability is low such as 0.008. Therefore, a parent can be altered by the mutation rate of 8 in one thousandth chance of being randomly replaced.

3.2. Crossover: it is the operator in genetic algorithms that distinguishes from other algorithms. The cross over operator is essential to the GA. If the cross over operator is deleted, the algorithm is no longer a GA. To make offspring, two solutions are selected and the crossover operation is performed. The produced offspring receive some of the characteristics of the parents. The crossover operator recombines portions of good solutions (parents), so the GA has more chance to create a better solution.

In this research, uniform crossover is used. Uniform crossover has the capability to combine any pattern. Each chromosome has continuous value instead of binary value. It is implemented based on linear combination of the first parent $P_1$ and the second parent $P_2$ to generate their offspring $x$ and $y$ as Eq. (4.2).

$$\begin{aligned} &\beta P_1 + (1-\beta)P_2 = x \\ &(1-\beta)P_1 + \beta P_2 = y \\ &\beta \sim u(0,1) \end{aligned} \tag{4.2}$$

*4. Fitness Function:* an evolutionary algorithm is made up of a population of solutions that are manipulated using a set of operators and are evaluated by several fitness functions. The fitness function depends on the nature of the given problem and is the most critical step in a genetic algorithm. The fitness function guides simulations in each iteration and determines how fit a solution and determines which solutions are transferred to the next population.

In genetic algorithms, each solution is represented as a string of numbers. A fitness function in a genetic algorithm shows how close a solution is to the aims of the desired problem or how close it is to an optimal solution. After evaluating the solutions, the best set of solutions are kept to solve a given problem [FeLo15]. In a multi-objective problem that uses multiple criteria, multiple fitness functions must be calculated. In this case, each solution should be assigned a rank (score) to indicate how close it is to the desired solution. This score is generated by applying a fitness function to each solution. Fitness functions are used in genetic algorithms to guide simulations of optimized processes. This score is generated by applying a fitness function to a solution.

In this research, the constraints on the design of this multiple objectives mother wavelet are the fitness functions. Therefore, a pair of low-pass filter coefficients *h(n)* and high-pass wavelet filter coefficients *g(n)* must be found to create an orthogonal mother wavelet. The filter coefficients

should be normalized, so their values must be between *–1* and *+1*. The fitness functions that should be satisfied through the weight training in the GNN are shown in Eq. (4.3.a) to Eq. (4.3.d) are considered.

$$\sum_n g(n) = 0 \tag{4.3.a}$$

$$\sum_n h(n) = \sqrt{2} \tag{4.3.b}$$

$$\sum_n |h(n)|^2 = 1 \tag{4.3.c}$$

$$g(n) = (-1)^{1-n} h(1-n) \tag{4.3.d}$$

*5. Updating population*: there are different strategies to update population. The *Non-dominated Sorting Genetic Algorithm* (NSGA-II) as a multi-objective optimization algorithm is used in this research to find the optimal solutions. The Pareto front is a set of the best solution for the problem that is optimized [Kala14]. The NSGA-II has three special operators: fast non-dominated sorting operator, crowded distance estimation operator and the selection operator (that is based on binary tournament selection with crowded-comparison operator) [Sing18]. The NSGA-II algorithm is based on elitist (elitism) strategy, which uses ranking and crowding distance to find the elitist elements in the population, in order to update the population.

Elitism strategy copies the best member of population for the next generation based on crowding distance and ranking criterion to improve the genetic algorithm performance [Davi91]. They use for updating population that has more than one cost functions. In crowding distance, the distance between a solution with its next neighbor is calculated. Then, the solutions are sorted based on their crowding distance value. In ranking, using the fitness function, each solution in a population has a rank. Then, the solutions are sorted based on their rank.

The NSGA-II algorithm is based on selecting individuals from the current population as parents to produce the children for the next generation. Then, analysis and evaluation of the population based on a cost (fitness) functions of multi-objective optimization will be done. Finally, the current iteration is improved based on merge, sort and truncate operations [Kala14].

In a multi-objective problem, the Pareto front provides a set of feasible solutions which do not dominate each other in terms of cost function and meeting constraints, but they do dominate other feasible solutions that are not in the Pareto front [VeKT10]. The following conditions are satisfied with any Pareto front solution that shows the non-domination concept using mathematic Eq. (4.4) [Kala14][VeKT10]:

$$\begin{aligned} f_i^x &\le f_i^y \quad for\ all\ i = 1,2,...,M \\ f_i^x &< f_i^y \quad for\ at\ least\ one\ i \in \{1, M\} \end{aligned} \tag{4.4}$$

where $x$ is a feasible solution, $y$ is another feasible solution, $i$ is the index regarding a Multi-objective cost function problem, $f_i^x$ is the value of the $i^{th}$ cost function result of the feasible solution $x$ and $f_i^y$ is the value of the $i^{th}$ cost function result of the feasible solution $y$.

The output of the multi-objective optimizer for the GNN is a Pareto front that contains a set of mother wavelets that have been chosen from among all feasible solutions. To select the best mother wavelet for detecting anomalous behavior in the DDoS ITD, a cost function is needed. Therefore, the next section introduces the steps for designing a cost function for the GNN.

#### 4.4.1.3 Designing a Cost Function Concepts for the GNN

A cost function is a measure of how well a neural network estimates its output based on its input, weights and biases. When creating a mother wavelet for the input signal, a cost function is needed to achieve an accurate function approximation to create an accurate mother wavelet. To

detect the DDoS attacks, the calculation of the GNN cost function should be based on two factors: the visual similarity between the input and the mother wavelet [SaKi05] and the fundamental properties of the original dataset. Therefore, to create an adaptive mother wavelet, it is important to make the wavelet coefficients similar to the input time series signal. To achieve this, the energy distance between the input signal and the wavelet coefficients must be minimized. Also, the complexity of data, statistics and variety must be considered. Otherwise, the detection rate of the DDoS attacks decreases because the cost function is not sensitive enough to find differences between neural network output and the desired output.

In order to design the cost function for the proposed GNN, similarity and data properties are addressed in the following ways.

Similarity: to find the similarity between the input signal and the mother wavelet, the difference between these at each point is needed. For this calculation, the energy, the mean square error, the logistic regression or an information theory criterion can be used.

To design a cost function for an anomaly-based intrusion detection system in which the DDoS ITD changes continually, a relevant concept should be considered because these Internet traffic changes are of interest. In this case, a relevance cost function is needed to extract the relevant information. Unlike an energy based cost function, which can be influenced by changes in the amplitude of the input data, a relevance cost function considers the probability of occurrence of certain events or frequencies.

To extract this hidden relevant information a sensitive tool is necessary. The relative entropy, also called the *Kullback-Leibler divergence* (KL-divergence), is an appropriate tool to extract the relevant information from a data distribution. The KL-divergence is a measure of the distance between two probability mass functions $p$ and $q$ as shown using Eq. (4.5) [CoTh06].

$$D(p \| q) = \sum_{x} p(x) \log \frac{p(x)}{q(x)} \tag{4.5}$$

where $p$ is the distribution of the desired output and $q$ is the distribution of the forward propagation output. In order to classify the data as either normal or anomalous, the KL-divergence can be used as a basis cost function for finding the performance of the GNN. Therefore, the KL-divergence for detecting DDoS attacks can be rewritten as Eq. (4.6).

$$D(y \| \hat{y}) = \sum_{i=1}^{N} y_i \log \frac{y_i}{\hat{y}_i} \tag{4.6}$$

where $y_i$ is the desired output and $\hat{y}_i$ is the output of the forward propagation (computation of the ANN output).

The KL-divergence is used to find the difference between the neural network output and the desired output. If $y_i = \hat{y}_i$, the neural network output has reached the desired output.

In addition, if the similarity between the input signal and the mother wavelet is not established, the Gibbs phenomenon happens, and the analysis needs an infinite number of coefficients to show corners, so the results are not accurate. To illustrate, if the input signal has a rectangular shape, a dual coefficient wavelet should have a rectangular shape as well. In this case, using the sin or cos as wavelets is not suitable for the rectangular input signal due to the Gibbs phenomenon. Due to the Gibbs phenomenon, in this case, a Haar mother wavelet that has rectangular shape must be used.

The data properties: the DDoS ITD is a non-differentiable, non-stationary, stochastic, dynamic, broadband, self-affine fractal signal and correlated with long-range dependence signal [Kins20]. Also, about 90% of the DDoS ITD data is normal and 10% is anomalous [SKFK17], so DDoS ITD data is not balanced. By exaggerating anomalous part of data, a better classification and data-class separation can be achieved in such a situation [SKFK17]. To design a cost function

for a fractal time series input signal, the complexity of the shape of the fractal is important [SKFK17]. To extract the complexity of a fractal time series, the *variance fractal dimension* (VFD) is useful [Kins20][GhKi18][GhKi19].

In order to extract features from the DDoS ITD with the above properties, the following feature extraction briefly demonstrated below must be followed [GhKi20b].

*The variance fractal dimension:* a fractal signal is a signal whose scaled-down version is similar to a part of itself [Kins20]. When the time series input signal is a fractal, its complexity or degree of fragmentation is its most distinguishing feature. To analyze and measure the complexity of a multifractal signal, the VFD is introduced [GhKi18][GhKi19][Kins20]. The output of the VFD equation is a number between one and two. In addition, the VFD output for normal data creates a class of data that is different from the VFD output anomalous data class, so, a decision boundary or a threshold can be used to separate the two classes.

*The distance:* the DDoS ITD considers two varieties of data: normal class and anomalous class. The distance between a newly arrived packet with an element from each class is the distance between that element and the arrived packet [GhKi20b]. The cumulative summation of the distances between the newly arrived packet that includes elements the class is the total distance.

#### 4.4.1.4 Designing a Cost Function for the GNN

A cost function for the GNN measures how well a selected mother wavelet from the Pareto front is capable of detecting and separating normal data from anomalous data. Very often, the cost function is based on either energy (i.e., squares of the differences between the predicted object and the desired object) or information-theoretic measures of the similarity between the objects. This section briefly introduces the steps for designing the combined cost function for a GNN that can

detect anomalous behavior in Internet traffic. The flowchart for designing a mother wavelet to detect DDoS attacks is shown in Fig. 4.6.

In this research, two classes are considered: normal data and anomalous data. Therefore, the following cost function in Eq. (4.7) can be extracted from Eq. (4.6). The obtained cost function separates normal data from anomalous data using two parts: entropy and extropy.

$$J(\theta) = D(y \,||\, \hat{y}) = y_i \log \frac{y_i}{\hat{y}_i} + (1 - y_i) \log \frac{(1 - y_i)}{(1 - \hat{y}_i)} \tag{4.7}$$

where $y_i$ is the desired output and $\hat{y}_i$ is the output of the forward propagation (computation of the ANN output). To have a faster convergence of the optimal coefficients of the neural network, a weighted cost function is used. Adding weights to the KL-divergence cost function can magnify the anomalous data, so this class can be bolder [GhKi20b]. The feature extraction methods can be used as weights (coefficients) of the cost functions. In essence, the KL-divergence is considered as the basis, and the extracting features are used as the weights for the basis cost function to design the weighted cost function.

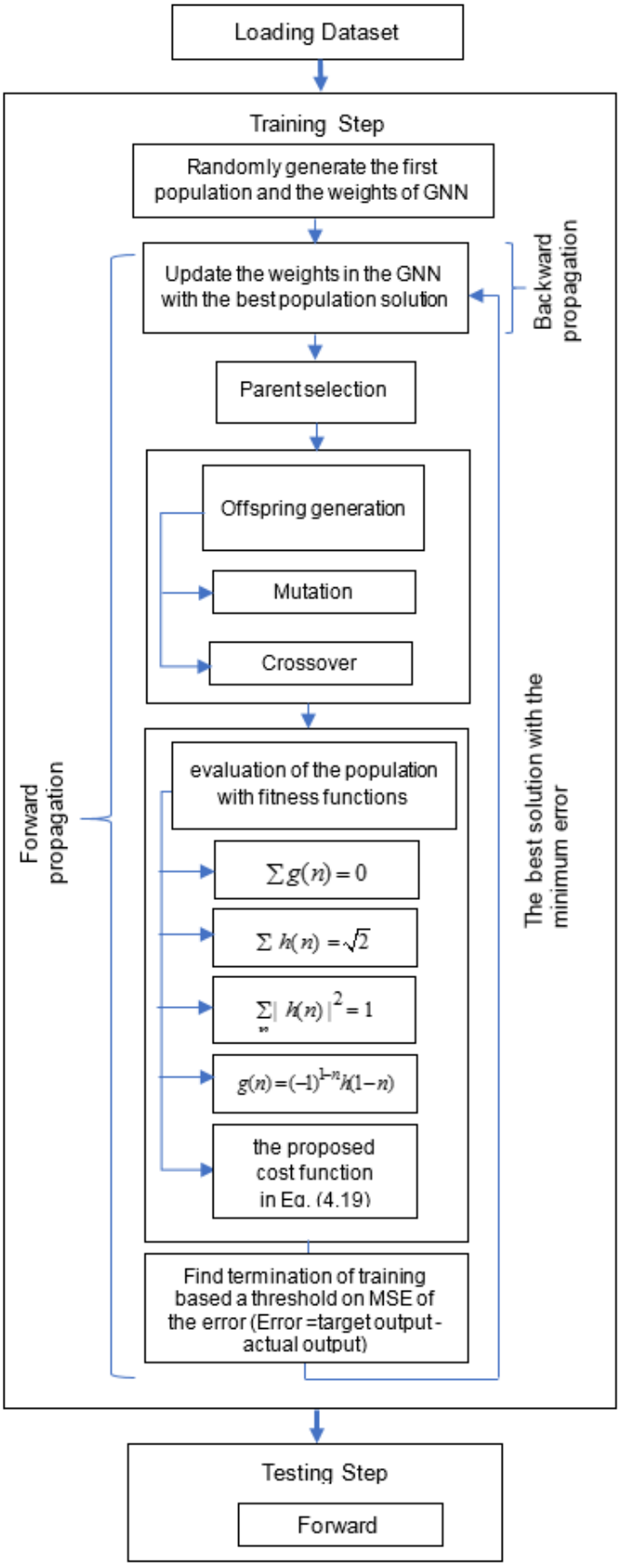

Loading Dataset
Training Step
Randomly generate the first population and the weights of GNN
Update the weights in the GNN with the best population solution
Backward propagation
Parent selection
Offspring generation
Mutation
Crossover
evaluation of the population with fitness functions
$\sum g(n) = 0$
$\sum h(n) = \sqrt{2}$
$\sum_n |h(n)|^2 = 1$
$g(n) = (-1)^{1-n} h(1-n)$
the proposed cost function in Eq. (4.19)
Find termination of training based a threshold on MSE of the error (Error =target output - actual output)
Forward propagation
The best solution with the minimum error
Testing Step
Forward

**Fig. 4.6:** A GNN flowchart for designing a mother wavelet to Detect DDoS attacks.

The weighted cost function as shown using Eq. (4.8) [GhKi20b].

$$J(\theta) = w_1.y_i \log \frac{y_i}{\hat{y}_i} + w_2.(1-y_i) \log \frac{(1-y_i)}{(1-\hat{y}_i)} \tag{4.8}$$

where $y_i$ is the desired output and $\hat{y}_i$ is the output of the forward propagation. In addition, weights are the combination of the extracted features using the VFD and the distance. To combine the features, the PCA is used to obtain the most efficient principal component with the highest variance [GhKi20b].

## 4.4.2 The Fourth Anomaly Detection Algorithm Design

In this section the adaptive mother wavelet GNN coefficients are used as a mother wavelet in the adaptive-wavelet CNN architecture to detect DDoS attacks [GhKi21a]. This section introduces the design of the structure characteristics and adaptive features of the *adaptive-wavelet convolutional neural network* (adaptive-wavelet CNN). Second, the proposed cost function for the GNN is used for the adaptive-wavelet CNN. The algorithm is illustrated with a dual coefficient adaptive-wavelet CNN designed to detect DDoS attacks and is shown in Fig. 4.7.

### 4.4.2.1 Designing the Architecture for the Adaptive-Wavelet CNN

In this section the adaptive mother wavelet GNN coefficients are used as a mother wavelet in the adaptive-wavelet CNN to detect DDoS attacks. The adaptive-wavelet CNN consists of two processing steps: data pre-processing and data processing. Moreover, the weighted cost function is used for the adaptive-wavelet CNN. This section introduces the structural characteristics and

adaptive features of the adaptive-wavelet CNN. An illustration of the adaptive-wavelet CNN with a dual coefficient to detect DDoS attacks is shown in Fig. 4.7 [GhKi21b].

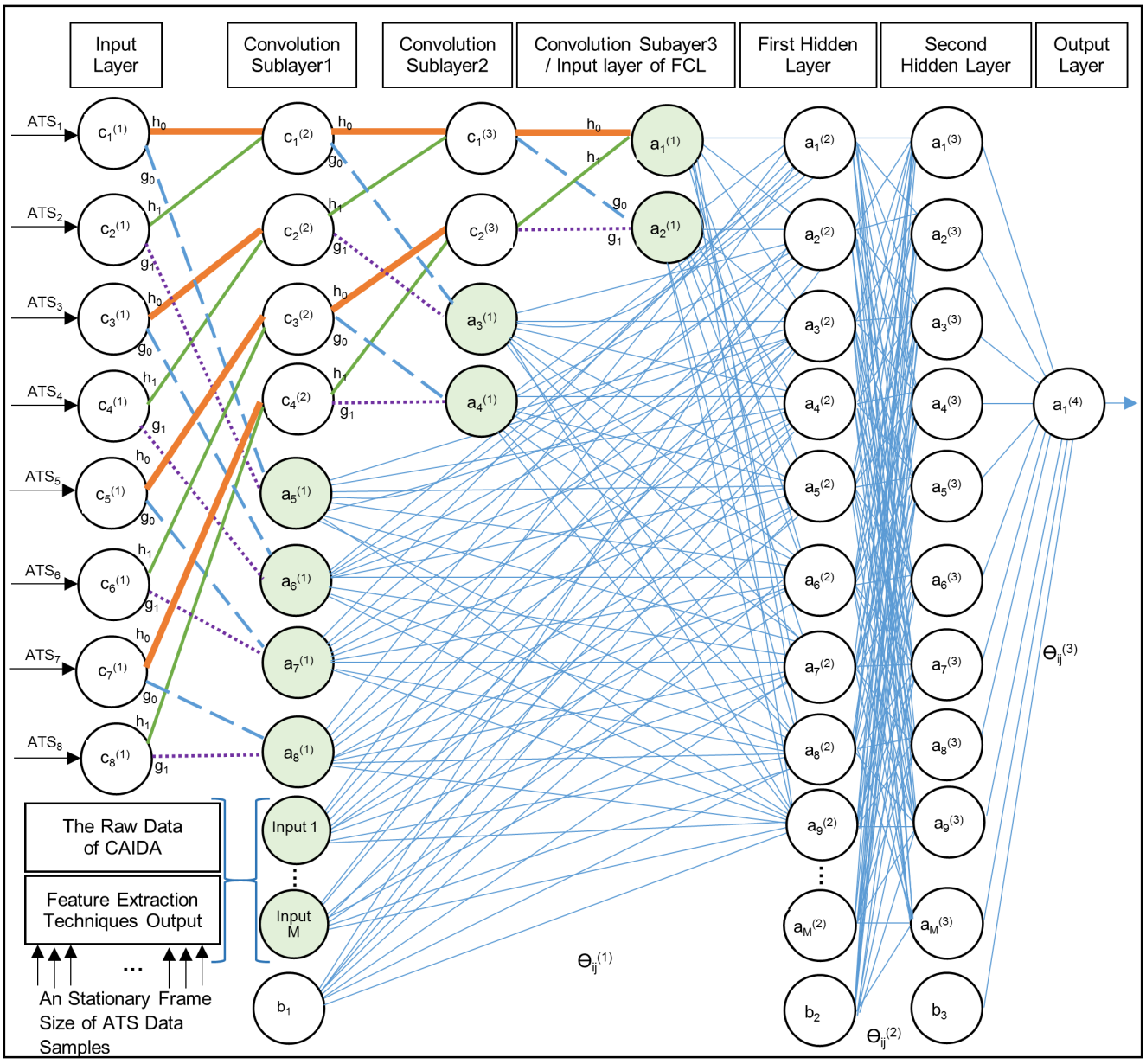



**Fig. 4.7:** The architecture of a dual coefficient adaptive-wavelet CNN designed for detecting DDoS attacks.

Pre-processing Step:

The ATS data is passed to the pre-processing step for feature extraction. Features of the input data are extracted from hidden information to increase the detection rate. The feature extraction techniques are the *Hurst exponent* (H), the DWT using 4 basis function, the SFD, the VFD, the ZC and the TC.

Designing the Architecture for the Adaptive-Wavelet CNN:

The architecture of the adaptive-wavelet CNN has three types of layers: the two input layers, the convolution layer and the fully connected layer.

The first input layer passes the input signal to the convolution layer. The number of nodes in this layer is adaptive, and it is equal to the number of features in the input data. For example, as shown in Fig. 4.7, the input layer of the dual coefficient adaptive-wavelet CNN has eight nodes. The eight nodes use a series of the ATS data as input.

The second input layer transfers two types of data to the first hidden layer: raw data and the pre-processed data, i.e. the output of feature extraction techniques.

The next layer is the convolution layer. The number of nodes in the input layer and the number of wavelet coefficients determine the number of convolution sublayers. In the dual coefficient adaptive-wavelet CNN, there are eight nodes and three convolution sublayers. This is shown in Fig. 4.7. The weights in the convolution layer are based on the weight-sharing concept, and the weights are initialized with the mother wavelet coefficients of the GNN's output. The number of input paths for each node in each sublayer varies based on the number of coefficients of the wavelet of interest. In the dual coefficient adaptive-wavelet CNN, there are two input paths for each nodes in each sublayer (Fig. 4.7). The number of sublayers in the convolution layer is

equal to the number of wavelet decomposition levels of DWT in the Mallat algorithm (Fig. 4.7). Equation (4.9) shows the number of sublayers (*N*).

$$1 < N < \log_{(nc)}^{(nf)} \tag{4.9}$$

where $nf$ is the number of features in input and $nc$ is the number of wavelet coefficients. In the dual coefficient adaptive-wavelet CNN, the number of sublayers in the convolution layer is equal to $log_{(2)}^{(8)} = 3$.

The last layer is the fully connected layer (FCL). The second input layer acts as an input layer for the FCL. The FCL has two hidden layers and one output layer. In the dual coefficient adaptive-wavelet CNN, the input layer of the FCL is shown in gray in Fig. 4.7. The number of nodes in the first and the second hidden layers equals the number of nodes in the input layer of the FCL, and the output layer of the FCL has one node as shown in Fig. 4.7.

#### 4.4.2.2 The Optimizer for the Adaptive-Wavelet CNN

The online learning is used to train the fully connected layer of the adaptive-wavelet CNN to update the weights of its neurons. The convolutional layer coefficients of the adaptive-wavelet CNN are the adaptive mother wavelet and the adaptive father wavelet (using Eq. (4.3.d)) that are obtained from the GNN. These coefficients are constant for the Adaptive-Wavelet CNN, so they do not need any learning method in this stage.

The stochastic gradient descent algorithm is used to update the weights of the fully connected layer of the adaptive-wavelet CNN. This algorithm is shown in Eq. (4.10).

$$\theta_{ij}^{(k)} = \theta_{ij}^{(k)} - \alpha \frac{\partial}{\theta_{ij}} J(\theta) \tag{4.10}$$

where $J(\theta)$ is the proposed cost function that is designed for the GNN, and it is shown in Eq. (4.8).

This fourth anomaly detection algorithm improves the detection rate by approximately 8% in comparison with the third anomaly detection algorithm. However, a DDoS attacks detection algorithm should have a higher detection rate. In addition, all the four proposed anomaly detection algorithms are based on supervised learning. As a result, they need labels to train their input, so they are not suitable for the real world. Therefore, the fifth anomaly detection algorithm for detecting DDoS attacks is proposed.

## 4.5 The Fifth Anomaly Detection Algorithm Design

The main procedure of the fifth anomaly detection algorithm for detecting DDoS attacks consists of three steps: preparing data, data pre-processing, and data processing. These three steps are shown in Fig. 4.8 [GhKi21c].

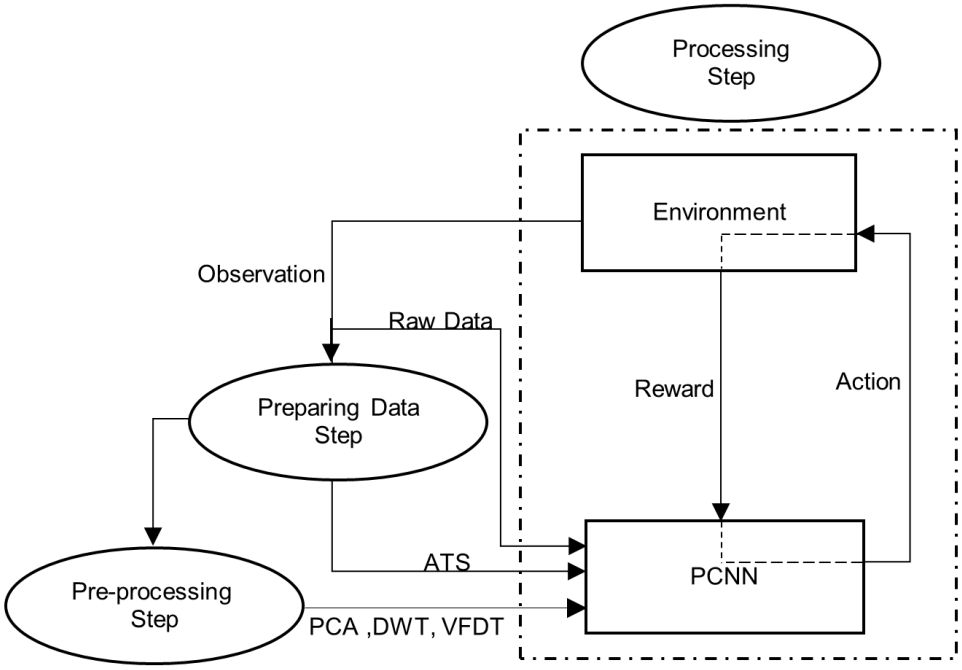



**Fig. 4.8:** Detecting DDoS attacks using the policy gradient based deep reinforcement learning algorithm.

The DDoS detection algorithm used in this study is described below:

### 4.5.1 Preparing data for processing

First, the raw data that contains a source IP address, a destination IP address, length of packet, protocol and packet- arrival time for each packet is fetched. Second, a new attribute called a *packet arrival time-series signal* (ATS) is calculated by using the difference between a packet-arrival time at *t* and the packet-arrival time at *t-1*.

### 4.5.2 Pre-processing

The ATS data is passed to the pre-processing step for feature extraction. In general, a DRL deals with unlabeled data, which decreases the accuracy of DDoS attack detection and increases detection processing time. To overcome these problems, the pre-processing step is used.

#### 4.5.2.1 Feature extraction

In the pre-processing step, the most important and reliable underlying features are extracted in three different methods: VFD, DWT and PCA. Therefore, the ATS data is passed to the VFD algorithm, DWT algorithm and PCA algorithm.

#### 4.5.2.2 The pre-processing step procedure

The pre-processing step proceeds as follows.

First, the DWT coefficients, the VFD coefficients, the ZC and the TC of the ATS data are calculated. Then, the PCA coefficients of the coefficients are extracted as the unique features. Second, the input time series must have a minimum stationary size of packets to gain a realistic result regarding the polyscale analysis algorithm. Third, based on this window size, the data is chopped into a chain of non-overlapping intervals. Fourth, within each interval, the output of step

one is normalized using z-score normalization. Fifth, the normalized interval is entered into the data processing step.

### 4.5.3 Processing

The purpose of the processing step is to classify the observed data into either normal or anomalous data. The raw data and the ATS data obtained from the preparing data for processing step, the VFD features, the DWT coefficients, the ZC, the TC and the PCA features that were extracted in the pre-processing step are all passed to the *polyscale convolutional neural network* (PCNN), which is a novel structure for policy approximation in the IDSA.

This section first describes the design of the PCNN structure and the data flow in the PCNN, and second, how the IDSA processes the output of the PCNN to detect anomalous behaviors in the input data.

#### 4.5.3.1 The Polyscale Convolutional Neural Network (PCNN) Architecture

The PCNN is a novel policy structure using a CNN based on a polyscale analysis algorithm. The output of the PCNN is the policy approximation for the IDSA. The architecture of a four coefficient PCNN algorithm is illustrated in Fig. 4.9. The four coefficient PCNN is considered because the minimum input nodes that can be illustrated in the polyscale analysis algorithm is four. If the eight coefficient PCNN is considered, the figure will be too crowded.

The architecture of the PCNN for policy approximation has the following three layers: the input layer, the convolutional layer and the fully connected layer. The PCNN architecture is adapted with its input features.

The input layer:

The number of nodes in the input layer is equal to the number of features in the input data. For convenience, the number of features in the input data follows the power of two that is shown in Eq. (4.11).

$$|A| = 2^k \qquad k = 2,3,4,... \tag{4.11}$$

where $|A|$ is the number of input layer nodes, and $k$ is an integer number greater than two. For example, the input in Fig. 4.9 has four features (coefficients), so the input layer of the PCNN has four nodes. The input layer passes the input signal to the next layer (convolutional layer).

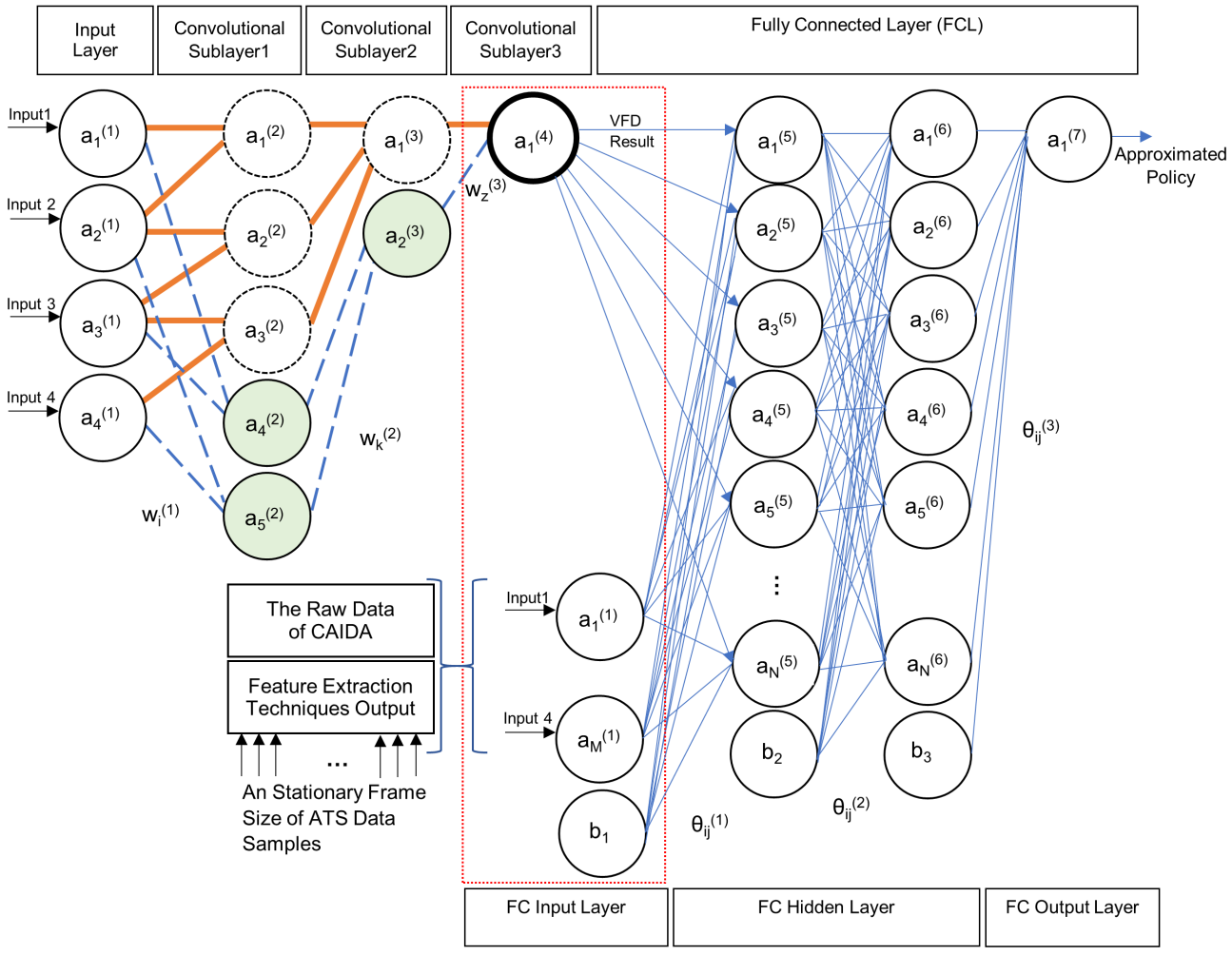



**Fig. 4.9:** The proposed polyscale convolutional neural network (PCNN) structure.

The convolutional layer:

The next layer is the convolutional layer, so the data from the input layer passes to this convolutional layer. In this study, the input data is non-pure fractal. Therefore, to detect DDoS attacks the hidden distinguishing features are extracted using the VFD algorithm. The VFD algorithm in the proposed method in this research and the conventional VFD algorithm [Kins20] [GhKi18] [GhKi19] are conceptually similar, but in the proposed method the coefficients in the first sublayer of the convolutional layer are learnable while the coefficients in the conventional VFD algorithm are constant (+1/–1). Moreover, in the proposed method the coefficients in the second sublayer of the convolutional layer share one weight for each vel size and the coefficients are learnable while in the conventional VFD algorithm the values of the previous steps are averaged. In addition, in the proposed method the coefficients in the third sublayer of the convolutional layer use one filter with learnable weight-coefficients while the conventional VFD algorithm uses the slope of the linear regression to find the VFD output. The VFD is a polyscale analysis algorithm that is used to design the architecture of the convolutional layer. In contrast with multiscale analysis, where there is no correlation between outcomes, a polyscale analysis measures a signal with various scales, and its outcome requires all the scales to be used simultaneously [Kins15].

There are three fixed convolutional sublayers, and the number of variable nodes in each convolutional sublayer is determined as follows.

For the first sublayer of the convolutional layer in Fig. 4.9, the PCNN architecture needs to find the relationship among the nodes. To extract the relationship among the nodes, the distance between the first and the last boundary node in an imagined sphere (sampling size) is considered.

The distance between these two boundary nodes is called *volume element size* (vel size) and can be calculated using Eq. (4.12)

$$vel \quad size = 2^i \qquad where \quad 0 \le i \le n \tag{4.12}$$

where $i$ is the step size of the distance between two boundary nodes, and $n$ is the number of the nodes in the input layer.

The vel size is proportional to the distance between two input nodes. The number of nodes in the input layer and the vel sizes determine the number of nodes in the first convolutional layer. The first sublayer of the convolutional layer nodes can be calculated using Eq. (4.13)

$$| C_1 | = \sum_{i=0}^{(\log_2^n)-1} (n - 2^i) \tag{4.13}$$

where $| C_1 |$ is the number of nodes in the first convolutional layer, and $n$ is the number of the nodes in the input layer. For example, the number of nodes in the first sublayer of the convolutional layer when there are four sources of input is five. This is shown in Fig. 4.10.

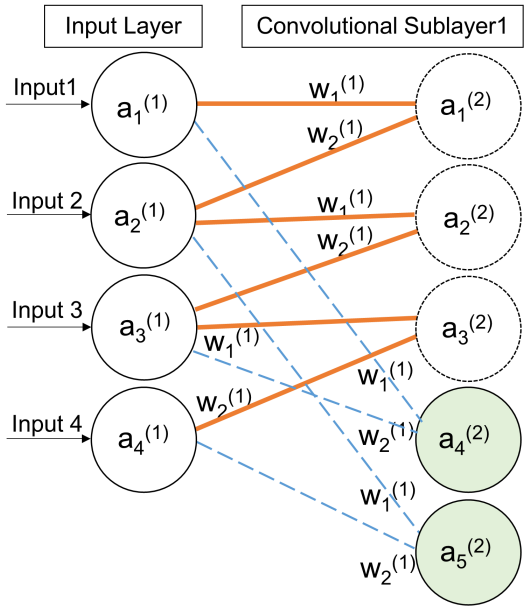



**Fig. 4.10:** The number of nodes in the first sublayer of the convolutional layer.

To extract the features for the first sublayer of the convolutional layer, the following steps are taken. First, the relationship between input $a_{(1)}^{(1)}$ and input $a_{(2)}^{(1)}$ is extracted using a vel size of $2^0$ , and the node $a_{(1)}^{(2)}$ is created. Also, the relationship between input $a_{(2)}^{(1)}$ and input $a_{(3)}^{(1)}$ is extracted and the node $a_{(2)}^{(2)}$ is created. Then, the relationship between input $a_{(3)}^{(1)}$ and input $a_{(4)}^{(1)}$ is extracted and the node $a_{(3)}^{(2)}$ is created. Second, the relationship between input $a_{(1)}^{(1)}$ and input $a_{(3)}^{(1)}$ is extracted using a vel size of $2^1$, and the node $a_{(4)}^{(2)}$ is created. Also, the relationship between input $a_{(2)}^{(1)}$ and input $a_{(4)}^{(1)}$ is extracted and the node $a_{(5)}^{(2)}$ is created.

The next vel size is $2^2$, which is bigger than the input layer size. Therefore, extracting nodes from the first sublayer of the convolutional layer is not necessary. In addition, there is one weight sharing filter for each node in the first sublayer of the convolutional layer. The size of the weight sharing filter equals two.

In the second sublayer of the convolutional layer in Fig. 4.9, the PCNN architecture needs to first find the relationship among the nodes with the same vel size. Each node in the second sublayer represents the vel size of the equidistant nodes in the first convolutional sublayer. Also, the number of inputs regarding each node corresponds to the number of nodes in the first convolutional sublayer with the same step size. The second sublayer of the convolutional layer nodes can be calculated using Eq. (4.14)

$$C_2 = \log_2^n \tag{4.14}$$

where $|C_2|$ is the number of the second convolutional sublayer nodes, and $n$ is the number of the nodes in the input layer. Moreover, the number of input pathways to each node corresponds to the number of equidistant nodes in the previous convolutional sublayer.

For example, in a PCNN with four input nodes such as the one shown in Fig. 4.9, there are two nodes in the second sublayer of the convolutional layer. The first node has three input pathways, the second node has two input pathways as shown in Fig. 4.11.

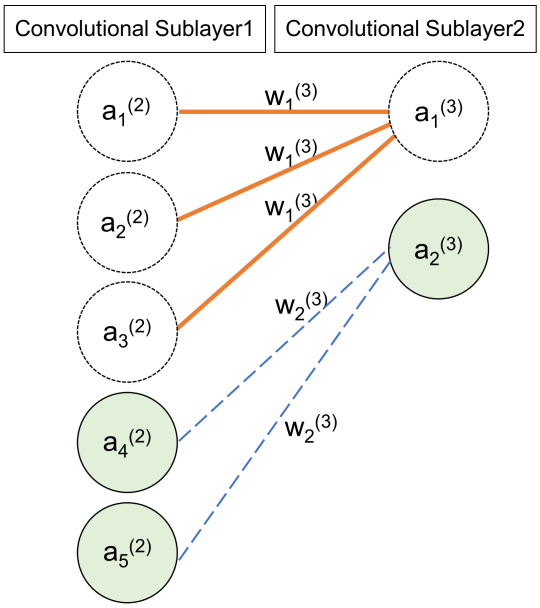



**Fig. 4.11:** The number of nodes in the second sublayer of the convolutional layer.

First, the output of the first convolutional sublayer nodes with a vel size of $2^0$ is passed to the $a_{(1)}^{(3)}$ node in the second convolutional sublayer. Second, the output of the first convolutional sublayer nodes with a vel size $2^1$ is passed to the $a_{(2)}^{(3)}$ node in in the second convolutional sublayer. In addition, the size of the weight sharing filter in the second sublayer of the convolutional layer varies for each filter size. In this sublayer, for each vel size there is one filter. Each filter has one value for its coefficients, so node $a_{(1)}^{(2)}$, node $a_{(2)}^{(2)}$ and $a_{(3)}^{(2)}$ share one filter and one coefficient.

To calculate the final result for the other vel sizes (such as $2^0$ and $2^1$), the third sublayer of the convolutional layer is needed. In the third sublayer of the convolutional layer in Fig. 4.9, the PCNN architecture needs just one node. The output of this node is the final result of the polyscale analysis and a final hidden feature is extracted using a method similar to the VFD analysis. This feature is needed to enhance detection rate.

The third sublayer of the convolutional sublayer contains only one node. This single node receives the final results of the polyscale analysis for each vel size $2^i$ from the second convolutional sublayer. For example, the output of the $a_{(1)}^{(3)}$ and $a_{(2)}^{(3)}$ nodes in the second convolutional sublayer are passed to the $a_{(1)}^{(4)}$ node in the third convolutional sublayer as shown in Fig. 4.12. In addition, there is one weight sharing filter in the third sublayer of the convolutional layer, and the number of coefficients equals the number of vel sizes.

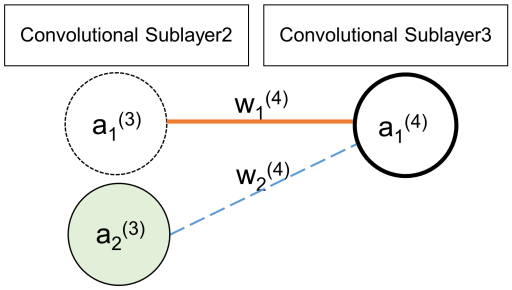



**Fig. 4.12:** The number of nodes in the third sublayer of the third convolutional sublayer.

The fully connected layer (FCL):

The data from the third convolutional sublayer passes to the FCL layer. The FCL has three sublayers: the input sublayer, the hidden sublayer and the output sublayer. The number of nodes in the fully connected input sublayer is equal to the number of elements from the input layer plus the output of the pre-processing step (the VFD, the DWT, the ZC, the TC and the PCA) plus the node in the third convolutional sublayer. The input layer elements are used in the fully connected input sublayer to improve the policy approximation for the IDSA. By considering this input layer elements, no hidden feature is missed. The first hidden sublayer of the fully connected receives input from the fully connected input sublayer nodes, so the number of nodes in this hidden sublayer is equal to the fully connected input sublayer elements. For example, when there are five input elements in the input sublayer of the FCL, there are five nodes in the hidden sublayer of the FCL, as shown in Fig. 4.9. The second hidden sublayer receives input from the first hidden sublayer,

and the number of nodes in this hidden sublayer is equal to the number nodes in the first hidden sublayer. The output sublayer of the FCL receives input from the hidden sublayer nodes, and this output sublayer has one node. The output of this node is the most precise approximation of the probability that the input data is normal or anomalous. This is the policy approximation.

#### 4.5.3.2 The detection of anomalous behavior steps for the IDSA that is based on the policy gradient based DRL

The IDSA detects anomalous behaviors by processing the output of the PCNN as shown in the following steps:

1) The IDSA uses the output (the probability) of the PCNN as its input. The output of the PCNN shows the probability that the data is normal or anomalous, and a strong PCNN produces a high confidence probability estimation. Based on this probability, the IDSA labels the input data as normal or anomalous. If the probability is above 0.5, the data may be labeled as normal, and if the probability is below 0.5, the data may be labeled as anomalous. The closer to one or zero, the higher the confidence for labeling the input as normal or anomalous respectively.

2) The IDSA chooses an action based on the probability that the data is normal or anomalous. Since the PCNN is used to approximate the IDSA policy, this policy is parameterized with the weights of the PCNN. Then, using the IDSA's experience of detecting the DDoS attacks, the PCNN learns the optimal weights. The IDSA tries to maximize its own performance over time using an unknown cost function, which is based on the weights ($\theta$) of the PCNN.

3) Then, the IDSA selects an action according to the softmax activation function as shown in Eq. (4.15).

$$\pi(a \mid s, \theta) = \frac{\exp(CNN(s, a, \theta))}{\sum_{b} \exp(CNN(s, b, \theta))} \tag{4.15}$$

where $CNN(s,a,\theta)$ is the output of the PCNN, and $a$ in $\pi(a \mid s,\theta)$ is the selected action, and $b$ is all possible actions.

4) The policy function $\pi$ is unknown. Mathematically the policy $\pi$ is a function of the weights ($\theta$) in the PCNN. In addition, the IDSA maximizes its performance by using a policy cost function, which is unknown, but can be estimated as follows

$$J = E_{\pi}[r(t)] = \int \pi(t) r(t) dt \tag{4.16}$$

The policy gradient multiplied by the discounted future returns at one time $t$ in one given state is the intermediate policy cost function unit $\pi(t)r(t)$. All of these units are summed to get the final policy cost function ($J$).

5) The input data is unlabeled, which makes the IDSA training difficult, so the policy gradient theorem is used to overcome this problem. This theorem suggests that it is possible to train the IDSA when the data is unlabeled by using the policy gradient algorithm shown in Eq. (4.17) [SuBa16][Tabo19a] [Silv15].

$$\nabla J(\theta) = \mathrm{E}_{\pi}\left[ G_t \frac{\nabla_{\theta} \pi(A_t \mid S_t,\theta)}{\pi(A_t \mid S_t,\theta)} \right] \tag{4.17}$$

where $G_t$ is the discounted future returns calculated using the MC algorithm for its simplicity, and the stochastic samples are generated by running episodes using the IDSA to detect DDoS attacks. Moreover, $\pi(A_t \mid S_t,\theta)$ is the output of the PCNN. Each time the IDSA encounters the state $S_t$, the IDSA increases the probability of taking an action to maximize the return of $G_t$. The gradient of the policy (or the gradient of the output of PCNN) is divided by the policy to achieve a vector that tells the IDSA the direction in the policy space for maximizing the chance of selecting the same action. The inverse proportionality of the policy then helps to prevent the possibility that frequently selected actions dominate the updates (prevent sampling bias). Thus, these actions

become more frequent even if they are less advantageous. By multiplying $G_t$ by $\nabla_\theta \pi(A_t \mid S_t, \theta)$ a vector is obtained that increases the probability of taking actions with high expected future returns (or reward). It is reinforcement learning that is the strength and power of the DRL algorithm.

These actions are normal labels, and they do not cause overflow yet and not normal. These normal labeled actions cause to accumulate packets in the buffer, has less than ideal result, but they are still kept. These actions that result in fewer rewards are still needed to explore new actions that might result in earlier and more accurate detection of DDoS attacks. The exploring new actions are explained with details in step 8.

6) To update the weights $\theta$ of the PCNN, the $G_t$ is used as the discounted future returns (that is rewards). The IDSA uses the gradient ascent to perform the function $\pi$ in the reinforcement learning algorithm as shown in Eq. (4.18) [Silv15].

$$\theta_{t+1} = \theta_t + \alpha V_t \nabla_\theta \ln \pi(A_t \mid S_t, \theta) \tag{4.18}$$

where $\alpha$ is the learning rate that determines the step size at each iteration while moving toward a maximum of the unknown cost function, and the value function ($V_t$) is the expected return that can be calculated using the MC algorithm as shown in Eq. (4.19) [HeZh19].

$$V_t = E[G_t = r_2 + \gamma r_3 + \gamma^2 r_4 + \gamma^3 r_5 + ...] \tag{4.19}$$

where $G_t$ is the discounted future returns and $E[G_t]$ is the expected value of the discounted future returns.

The parameter $\theta$ is updated to maximize the probability of selecting the action $A$ for a given state $S$ at time $t$. Also, the policy function $\pi(A_t \mid S_t, \theta)$ is the output of the PCNN, which passes to the $\ln$ (natural logarithm) function to create $\ln \pi(A_t \mid S_t, \theta)$. The second term in Eq. (4.18) is

comprised of the discounted future returns $G_t$ (or $V_t$) multiplied by the gradient of the natural logarithm of the probability of taking each action in each state.

The discounted future returns and the PCNN as the probability distribution are used to update the weights. The IDSA uses a reward system to indicate how well the selected action facilitate detection of DDoS attacks. The ultimate goal of the IDSA is maximization of expected cumulative reward.

7) To obtain the derivative of a function, the analytical derivative and the finite differences techniques can be used. The analytical derivative technique finds the differentiation of the function with respect to the parameter of interest. The finite differences technique finds the approximation of the differentiation using the *central finite difference* (CFD) equation as shown in Eq. (4.20) [BaED17].

$$\frac{\partial J}{\partial x_i} = \frac{J(x_1,x_2,...,x_i+\Delta x_i,...,x_n)-J(x_1,x_2,...,x_i-\Delta x_i,...,x_n)}{2\Delta x_i} \tag{4.20}$$

This approximation involves perturbing the $i^{\text{th}}$ parameter. The CFD is one of the most accurate methods of approximating derivative because the results of the CFD are close to the analytical derivative [GhKi20b]. When there is missing a function, CFD is used to approximate the gradient of the function.

In this research, the function $J$ is unknown, so the CFD is used to approximate the gradient of the function $J$ to update the weights of the PCNN accordingly. Therefore, the CFD equation can be rewritten as Eq. (4.21) to calculate the gradient of the policy.

$$\nabla \log \pi(\theta) = \sum_{k=1}^{M} \frac{\ln \pi(\theta_k+\Delta)-\ln \pi(\theta_k-\Delta)}{2\Delta} \tag{4.21}$$

where $k$ is a trajectory of states in one episode.

8) Evaluating actions is an important step in training the IDSA. In this study the input data that the IDSA uses is unlabeled, so the IDSA is guided through rewards while the rewards are based on delayed feedback [Gero17]. For example, if the IDSA manages to detect the DDoS attacks in a stationary data set, it is not possible to know which data has been labeled as normal or as anomalous nor is it possible to know which actions were correct and which ones were incorrect. The IDSA knows only that the Internet traffic queue in a DNS server (this queue is modeled as a repository for this application) has surpassed the buffer, so it crashes. This crash is visible only after the last action, which is not necessarily the action responsible for the crash, the IDSA does not know which action has been rewarded (or punished). This phenomenon is called the credit assignment problem. To overcome this problem, one strategy is to evaluate an action based on the sum of all the rewards using a *discount factor* [Gero17]. The discount factor represents the relative value of delayed versus immediate rewards. The discount factor determines how much the IDSA considers accepting the future rewards as opposed to the immediate reward. Eq. (12) shows how the sum of all the rewards is calculated using the return value ($G_t$) at time $t$, where the reward is $r_i$ at step $i$, and $\gamma$ is the discount factor at step $i$.

$$G_t = r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 r_4 + \ldots = \sum_{k=1}^{n} \gamma^{k-1} r_k \qquad (4.22)$$

If the discount factor is close to zero, then future rewards will have little value when compared to immediate rewards. If the discount factor is close to one, then rewards far into the future will have almost the same weight as immediate rewards. In general, discount factors are 0.95 or 0.99. With a discount factor of 0.95, rewards 13 steps into the future count approximately half weight of immediate rewards (since $0.95^{13} \approx 0.5$), while with a discount factor of 0.99, rewards 69 steps into the future count approximately half the weight of immediate rewards. In the DDoS

environment, actions have long-term effects, so choosing a discount factor of 0.99 is a good discount factor.

8) Evaluating actions that are selected using the PCNN is an important step in training the IDSA. If the best action at each step is known, the training of the PCNN in DRL is considered supervised learning. However, in this study the input data that the IDSA uses is unlabeled, so learning involves rewards that are based on delayed feedback. For example, if the IDSA manages to detect the DDoS attacks in a stationary dataset, it is not possible to know which data has been labeled as normal or as anomalous nor is it possible to know which actions were correct and which ones were incorrect. The IDSA knows only that the Internet traffic queue in a DNS server (this queue is modeled as a repository for this application) has surpassed the buffer, so it crashes. This crash is visible only after the last action, which is not necessarily the action responsible for the crash, the IDSA does not know which action has been rewarded. This phenomenon is called the credit assignment problem. To overcome this problem, a common strategy is to evaluate an action based on the sum of all the rewards as shown in Eq. (4.22).

9) After evaluating actions, an action can be selected. An evaluated action can be selected according to the maximum probability calculated by the PCNN, but the IDSA selects an action according to ϵ-greedy algorithm while $\epsilon$ is a small value. The ϵ-greedy algorithm is used to discover which actions led to the best rewards over time to select an action. In the ϵ-greedy algorithm, the highest probability between $P_1$ and $P_2$ is selected with a probability of $(1- \epsilon)$ while this step is called the exploitation technique. In addition, with the probability of $\epsilon$, a probability is chosen from the two probabilities $P_1$ and $P_2$ randomly while this step is called the exploration technique. Using the ϵ-greedy algorithm leads to the most valuable probability for choosing a known action with a good result while rarely selecting an unknown action with the probability $\epsilon$ for discovering a new

action. The IDSA learns incrementally using a tread off between exploration and exploitation techniques. If a known action with the highest probability is always selected, the IDSA never explores other unknown actions, so the action selection is limited to suboptimal policy [Sang21]. By exploring a new action, the IDSA may try a better result than the existing actions that have been explored already.

The proposed deep reinforcement learning structure based on a polyscale measure is shown in Fig. 4. 13. The convolutional neural network layer in this structure has several sublayers, and the sublayers follow the polyscale analysis of the VFD algorithm. The structure of the proposed DRL algorithm is shown in detail in Fig. 4.13, and this Fig. 4.13 has a similar concept and the same meaning as Fig. 4.8. Figure 4.8 shows a summarized version of the proposed deep reinforcement learning algorithm. However, Fig. 4.9 shows only a specific part of the proposed deep reinforcement learning algorithm, and Fig. 4.9 just shows the proposed PCNN based on the polyscale analysis that approximates the policy of deep reinforcement learning.

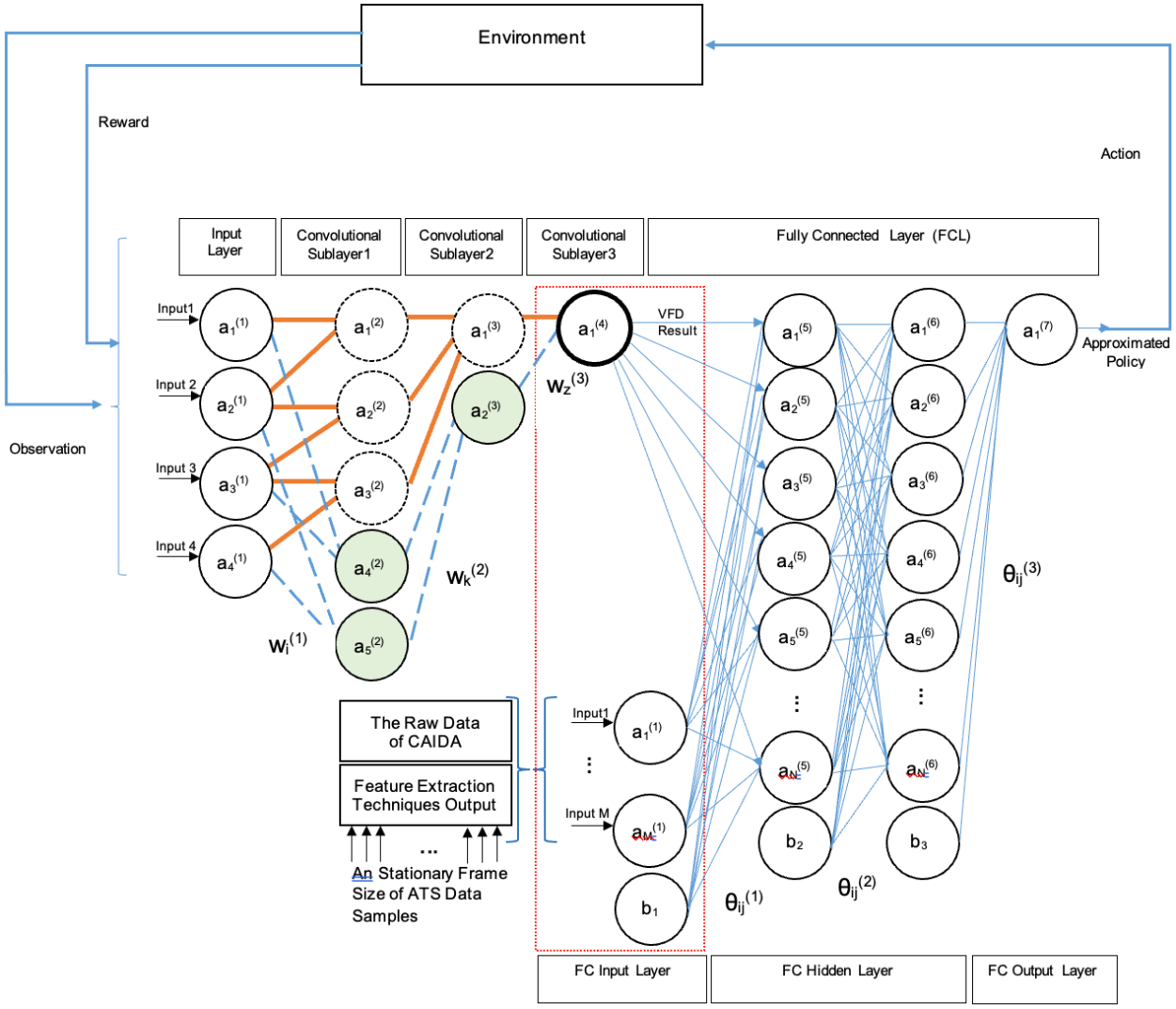



**Fig. 4.13:** The proposed deep reinforcement learning structure based on a polyscale analysis algorithm.

## 4.6 Summary of Chapter 4

In this chapter, the designs of the anomaly detection algorithms, based on supervised learning and DRL, are presented. The adaptive mother wavelet using GNN is designed and created to enhance the detection rate. The IDSs based on supervised learning of ANN, CNN, and SVM algorithms are designed to improve the accuracy of detection rate. The IDS based on the proposed

structure of PCNN is designed to detect anomalous behaviors in unlabeled data for real-world applications.

The proposed anomaly detection algorithms could be consisted of three steps: the pre-processing, the processing step and the processing step. The pre-processing was feature extraction methods could be included three types of analysis: the mono-scale analysis, the multiscale analysis and the polyscale analysis. The mono-scale analysis was PCA, statistical features (mean, median, mode, variance, standard deviation, skewness, and kurtosis), autocorrelation, ZC and TC. The multiscale analysis was the wavelet transform, and the polyscale analysis was VFDT. The processing step was machine learning algorithms that detect anomalous behaviors. The machine learning algorithms were the supervised method such as ANN, CNN, and the unlabeled learning method such as DRL. The post-processing step was a machine learning algorithm that improved the detection rate, and the algorithm was the SVM that was used in the third anomaly detection algorithm.

The simulation and the results of the proposed algorithms are introduced in Ch. 6.

# Chapter 5

# DESIGN OF EXPERIMENTS

The two complete time series datasets, the CAIDA dataset and the power consumption dataset, are introduced in this chapter. In addition, the software and the hardware are used in this research are explained in this chapter.

## 5.1 The Datasets

Two time series datasets that are used in this study are introduced. The first dataset is the power consumption that is simulated. The second dataset is the Internet traffic data that contains DDoS attacks, and it is downloaded from the *Center for Applied Internet Data Analysis* (CAIDA). The properties of the datasets are described in this chapter.

Each complete dataset is considered an epoch ($T_E$) while one stationary piece of data within each epoch is considered a frame. A window is an arbitrary part of a time series.

## 5.2 Power Consumption Dataset

### 5.2.1 Power Consumption Dataset Background

In an anomalous power consumption attack, such as one in which an electricity bill is changed for example, hackers try to manipulate data or signals by inserting, deleting, and changing control commands between consumers and controllers.

Smart grids contain smart meters that are access point for bidirectional communications between controllers and consumers who can control their power consumption. This smart meter is a key vulnerability factor in the smart grid infrastructure. There are two perspectives on the smart grid: the power electric perspective and the communications perspective [MMMK12]. In the power electric perspective, the smart grid consists of generation, transmission, distribution and customers. In the communications perspective, the smart grid consists of *wide area networks* (WAN), *neighborhood area networks* (NAN), and *home area networks* (HAN).

The utility center (in WAN level) shows the amount of utility for each subsystem NAN. The proposed method introduces an anomalous power consumption attack method for different levels of subsystems within the *advanced metering infrastructure* (AMI). When input usages of power in utility centers are not proportional with a consumer's payments, a further technique is necessary to detect and locate the attack (in HAN level). The anomalous power consumption detection software (create for this study) was developed on routers at the HAN level of the smart grid infrastructure. The software on the router monitors the smart meters. Normal energy usage patterns are different from hacker energy usage patterns (less energy usage in an anomalous power consumption). For example, a hacker injects data to a smart meter to evade payment for power usage.

### 5.2.2 Power Consumption Dataset Simulation

In this section, simulation of the power consumption dataset in the smart grid infrastructure is demonstrated. Simulation of the dataset can be described as below.

First, the power consumption signals are generated synthetically (generated by computer). Therefore, an initial normal pattern of the power consumption of a customer is generated. The normal pattern has 640 samples per day (640 samples per 24 hour). Then, elements of the normal pattern are permuted randomly between –0.3 and 0.4. Next, a set of normal patterns of power consumptions are created based on the first normal pattern of power consumption. A random perturbation was used in the power consumption simulation to slightly make differs of the simulated data from the original model. As a result, the simulated dataset was close to a real dataset. Figure 5.1 shows one sample of a normal pattern of daily power consumption.

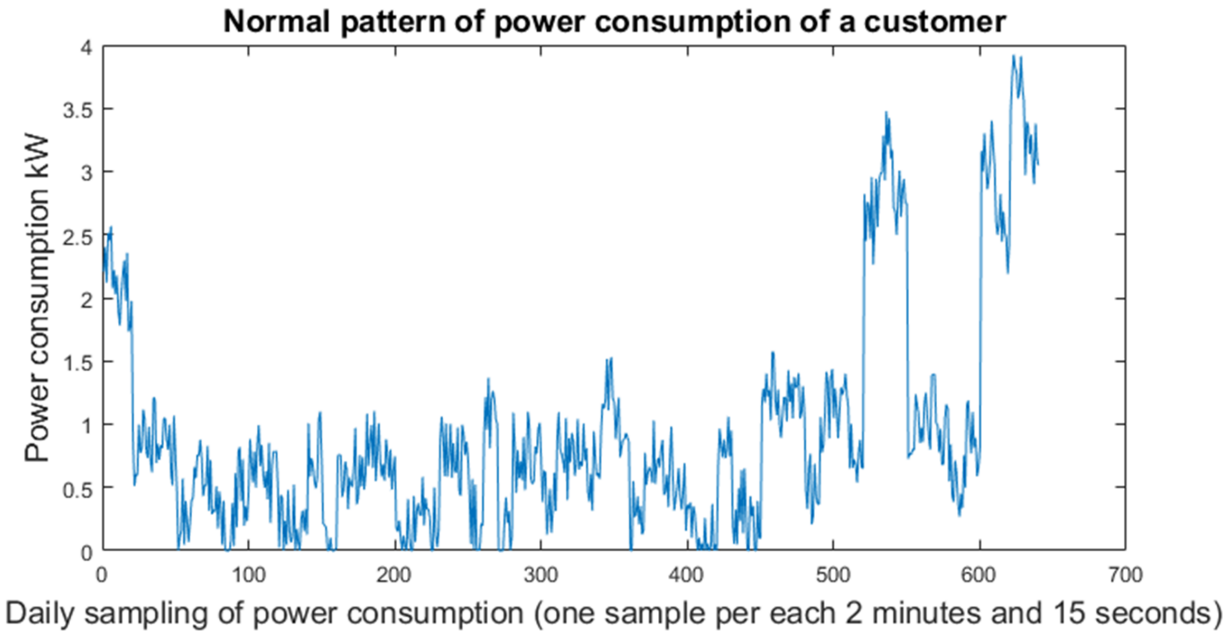



**Fig. 5.1:** A normal pattern of power consumption with 640 samples per day.

Next, a set of anomalous patterns of power consumption are created based on the first normal pattern. Anomalous patterns are simulated by manipulating a smart meter with a random

duration of attack of 2.15 minutes, 5 minutes, 15 minutes and 75 minutes with the random starting point of attack (for example attack starts at 8:00 p.m.). Each set of normal and anomalous patterns of power consumption contained 1000 patterns. Next, the normal dataset is tagged with zero, and the anomalous dataset is tagged with one. Finally, the normal dataset and the anomalous dataset are shuffled. As a result, the gained dataset contains 2000 patterns for evaluating and detecting the power consumption attacks. Figure 5.2 shows one sample of an anomalous pattern of daily power consumption.

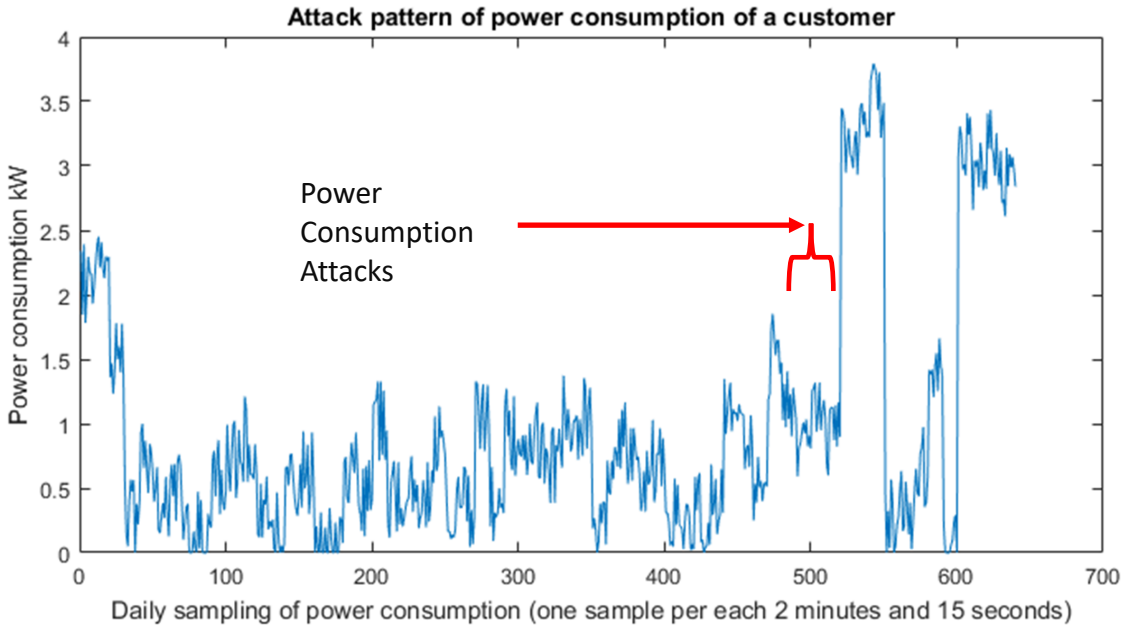



**Fig. 5.2:** An anomalous pattern of power consumption with 640 samples per day.

The power consumption dataset is a self-affine fractal signal. The characteristic of a normal pattern of power consumption is similar with an anomalous pattern of power consumption. Therefore, it is difficult to separate two classes of normal and anomalous dataset. The data that contains anomalous power consumption does not differ just in peaks, so simple algorithm cannot classify the dataset to normal class and anomalous. To extract the anomalous behaviors in a fractal signal, a proper feature extraction method, such as multiscale analysis and polyscale analysis

should be used. To have more accurate classification, machine learning algorithms should be used. As a result, proper feature extraction methods and machine learning algorithms are necessary to detect anomalous power consumption attacks.

### 5.2.3 The Power Consumption Attack Dataset Characteristics

The trajectory of the means and the variance regarding the power consumption dataset are shown in Fig. 5.3.

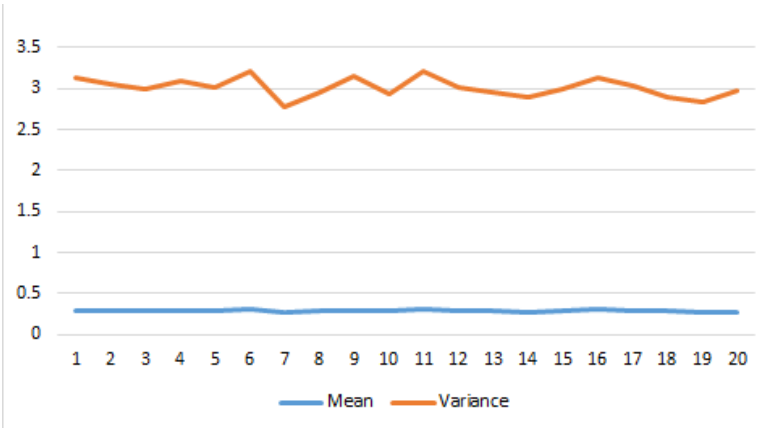



**Fig. 5.3:** Trajectories of mean and variance within a stationary frame of the power consumption dataset.

The trajectory of the skewness and kurtosis regarding the power consumption dataset are shown in Fig. 5.4.

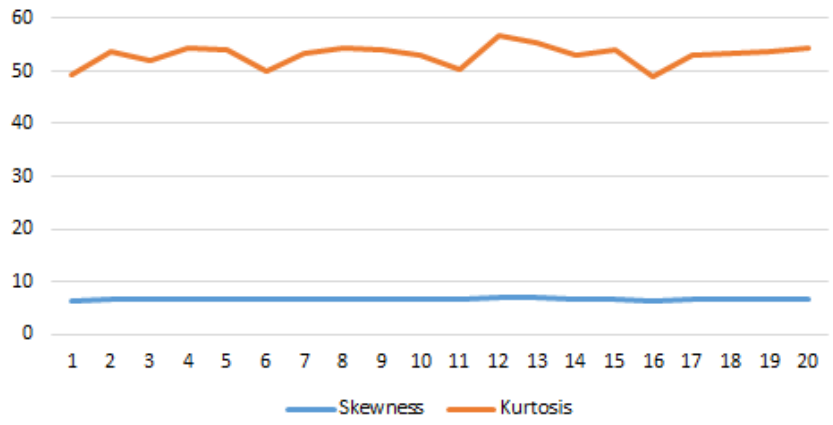



**Fig. 5.4:** Trajectories of skewness and kurtosis within a stationary frame of the power consumption dataset.

The PMF of the power consumption dataset is shown in Fig. 5.5.

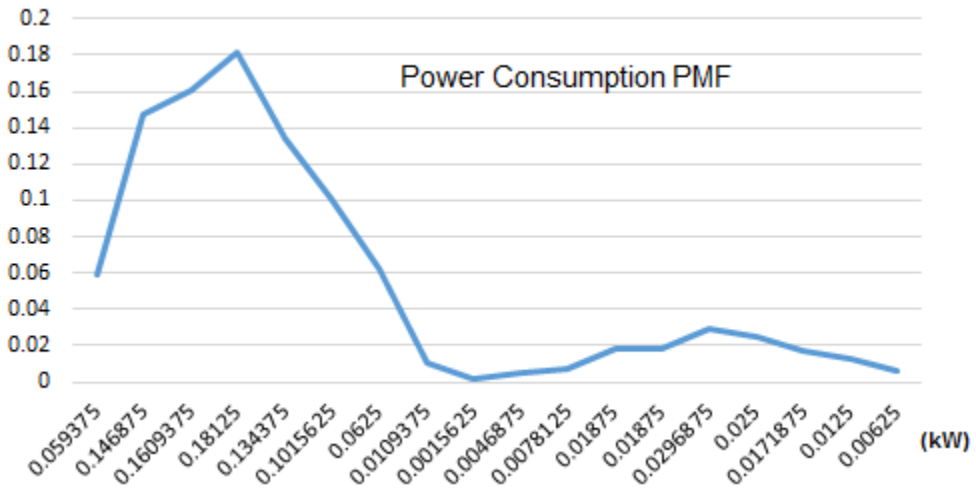



**Fig. 5.5:** The PMF of the power consumption dataset.

### 5.2.4 The Power Consumption Attack Dataset Weaknesses

This power consumption simulation is provided because it is the simplest dataset that can be provided to run an IDS algorithm. The sampling of the power consumption simulation is 640 samples per day. In other word, daily sampling of power consumption was one sample every 2

minutes and 25 seconds. If a hacker learns about the sampling rate, the hacker finds that if the hacker steals power within 2 minutes and 25 seconds, the algorithm cannot sample the dataset. As a result, the algorithm cannot detect any power consumption within 2 minutes and 25 seconds. In this situation, the detection rate is zero percent.

Moreover, if the duration of the attack is 2 minutes and 25 seconds or a little more, the detection of attack is about 50% showing that the proposed detection algorithm is random.

In addition, if the hacker steals power from the minimum power consumption duration and manipulates the data during these times, the proposed algorithm has about 50% of detecting the attack.

On the other hand, the dataset has no diversity, and the dataset does not have seasonal variations or any faults in the system. Thus, this simulation is not close to the real power consumption. Moreover, if a client's power consumption usage does not follow a proposed normal power consumption pattern, a power consumption attack is detected, but it is not an anomalous behavior. Therefore, this simulation is not close to the real power consumption, and this dataset is not accurate. Thus, the dataset was switched from power consumption in smart grid infrastructure to the DDoS attacks detection in the cyber computer system.

## 5.3 Distributed Denial of Service Attack Dataset

### 5.3.1 Distributed Denial of Service Attack Dataset Background

The dataset that was used for training and testing in this research was downloaded from the *Center for Applied Internet Data Analysis* (CAIDA). The CAIDA obtains different types of real time network traffic from around the world, and it has become one of the most reliable networks for downloading datasets. Commercial organizations, research sectors, and governments

donate and collaborate with the CAIDA while their privacy is protected [CAIDA15]. This center aims to support large-scale data collection for the scientific research community.

The dataset used in this research was chosen from CAIDA's 2007 DDoS-attack traffic, and it contained UDP Flood, TCP, ICMP (Ping) Flood, and SYN Flood packets. Each packet in the dataset contains a source IP address, a destination IP address, length of packet, protocol and packet-arrival time. The number of packets with 0.1ms duration within the stationary frame size is shown in Fig. 5.6.

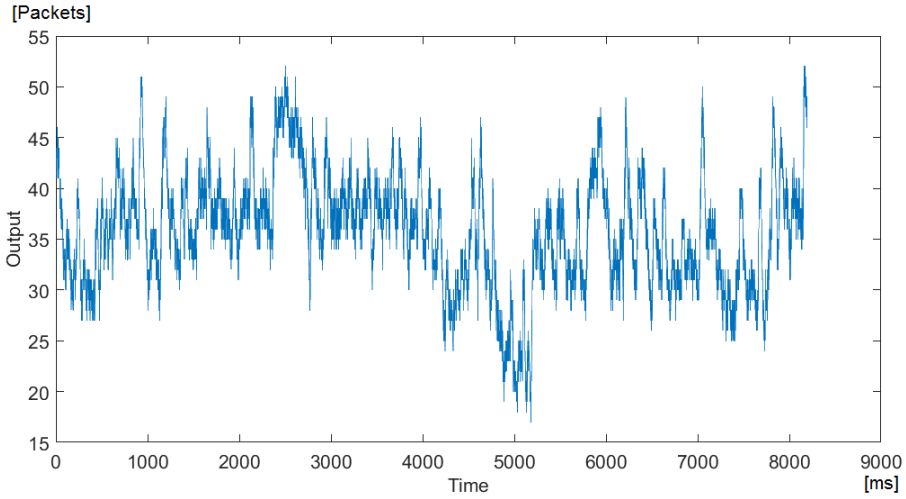



**Fig. 5.6:** The number of packets within a stationary frame size.

A *packet arrival time-series signal* (ATS) is calculated by using the difference between a packet arrival time at *t* and the packet arrival time at *t-1*. The ATS is used for data augmentations in this research. The ATS attribute of the CAIDA dataset has the following characteristics:

It is stochastic, stationary within a window size 8192, non-differentiable, broadband, self-affine fractal; and it follows a levy distribution, so it has a long-range dependence. In addition, the CAIDA dataset within a frame has a minimum amplitude of $0.002\times10^{-3}$ *Second* (s) and a maximum

amplitude of 0.0161 *s*; also, its energy is 0.1328. Moreover, more characteristics of CAIDA dataset are shown in-detail as the raw data column in Table 5.1 [GhKi20a].

**TABLE 5.1:** The DDoS ITD statistics and the outputs regarding eight mono-scale and poly-scale measures for the first stationary frame of trajectories.

| Measures | Result |
|---|---|
| Mean | 0.0029 |
| Median | 0.0020 |
| Mode | 5.0000e–06 |
| Variance | 7.7418e–06 |
| Standard deviation | 0.0028 |
| Skewness | 1.2928 |
| Kurtosis | 1.2582 |
| Autocorrelation | –3.4694e–18 |
| VFD | 1.9931 |
| Hurst exponent | 0.0069 |
| Slope ($\beta$) | 1.0138 |
| Zero crossing | 1250 |
| Turn count | 5349 |

### 5.3.2 Distributed Denial of Service Attack Dataset Characteristics

The *Internet traffic data* (ITD) *with distributed denial of service* (DDoS) *attacks* (DDoS ITD) is a time series that is a stochastic, non-stationary, non-differentiable, broadband, dynamic, self-affine fractal, and correlated with long-range dependence signal [Kins20].

Stationarity:

A data has weak stationarity when its first two moments fall within two ranges, a 95% confidence interval [Kins20]. The DDoS ITD had weak stationarity because the minimum window size of 8192 caused the trajectory of the means to fall within a range of 0.000782 to 0.003041, and

the variance trajectory to fall within a range of 1.64E–06 to 8.21E–06. The trajectory of the means and the variance are shown in Fig. 5.7.

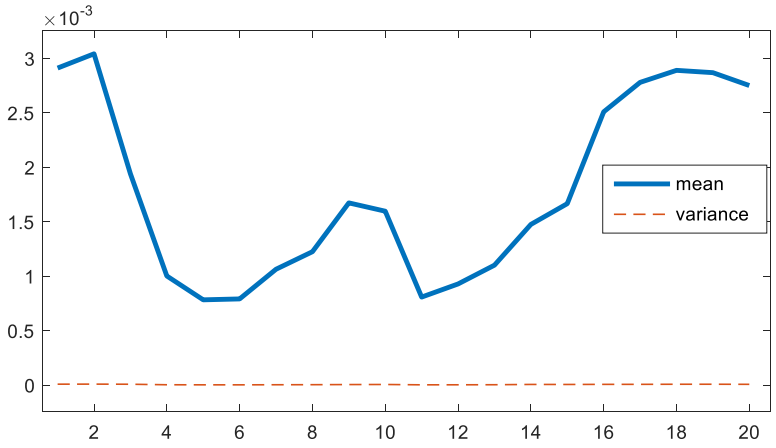



**Fig. 5.7:** Trajectories of mean and variance within a stationary frame of samples regarding the DDoS ITD.

The skewness trajectory fell within a range of 1.076 to 2.94, and the kurtosis trajectory was 1.8. The kurtosis indicates the impact of the heavy tail of the distribution. Positive values of kurtosis indicate heavy tail distributions. The trajectory of the skewness and kurtosis are shown in Fig. 5.8.

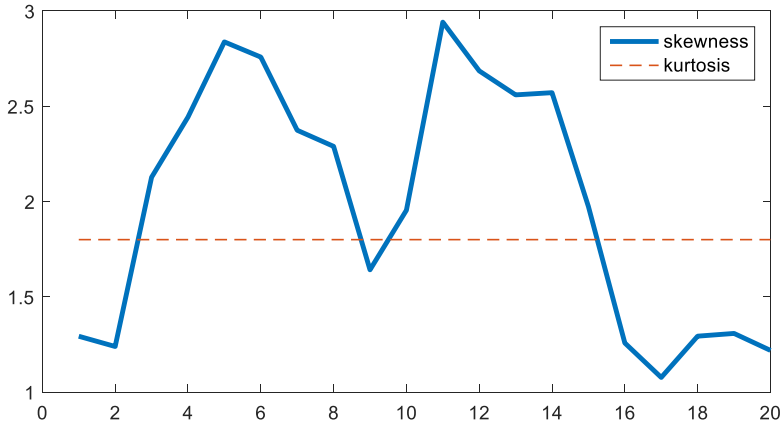



**Fig. 5.8:** Trajectories of skewness and kurtosis within a stationary frame of samples regarding the DDoS ITD.

Probability Mass Function:

To fit a probability distribution to the time series of the two datasets maximum likelihood estimation method and polynomial regression [Lect16] are used. The PMF of the DDoS ITD follows a levy distribution with the location parameter ( $\mu$ ) equal to 0.0019223631and the scale parameter ( $\sigma$ ) equal to –0.0004226877 as shown in Eq. (5.1). These parameters are achieved using maximum likelihood estimation.

$$f(x) = \sqrt{\frac{\sigma}{2\pi}} . \frac{e^{-\frac{\sigma}{2(x-\mu)}}}{(x-\mu)^{3/2}} \quad x > \mu, -\infty < \mu, \sigma > 0 \tag{5.1}$$

The heavy tail of the DDoS ITD is visible when comparing its levy distribution with a normal distribution. The PMF of the DDoS ITD is shown in Fig. 5.9.

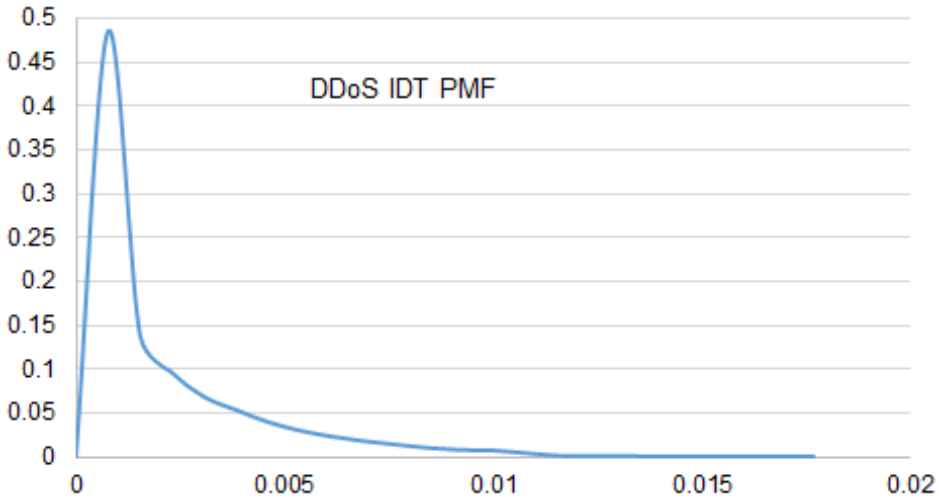



**Fig. 5.9:** The PMF of the DDoS ITD.

The CAIDA dataset features are 1. arrival time 2. source IP address 3. destination IP address 4. protocol 5. length of input. To improve the detection of DDoS attacks, this research adds another feature, which is named arriving time between adjacent packets. Therefore, “Arriving time between adjacent packets” is not used for labeling data, and it is used to increase the detection rate.

To label DDoS attacks, network traffic parameters–such as the number of packets per 0.1 milliseconds, packet size, and source IP address–should be considered. Therefore, several parameters are involved to consider a packet normal or anomalous. As a result, the dataset cannot be easily classified as normal or anomalous using just a threshold.

Moreover, the DDoS attacks dataset is a self-affine fractal signal. The characteristic of a normal pattern of Internet traffic is similar with Internet that contains DDoS attacks. Therefore, it is difficult to separate two classes of normal and anomalous dataset. The data that contains DDoS attacks does not differ just in peaks, so simple algorithm cannot classify the dataset to normal class and anomalous. As a result, proper feature extraction method and machine learning algorithms are necessary to detect DDoS attacks. To extract the anomalous behaviors in a fractal signal, a proper feature extraction method, such as multiscale analysis and polyscale analysis should be used. Therefore, in this PhD thesis, the VFDTv2, is used to extract features from the actual fractal data that rely on long-range dependence. Moreover, an adaptive mother wavelet algorithm is introduced to extract hidden features from Internet traffic. To have more accurate classification, machine learning algorithms should be used. Therefore, in this PhD thesis a new CNN architecture for detecting the DDoS attacks is introduced. As a result, the detection of DDoS attacks using the designed mother wavelet and the proposed CNN architecture is improved. In addition, an extended version of policy gradient DRL, which uses unlabeled data, based on the PCNN is proposed to detect DDoS attacks.

Although the CAIDA dataset is used in this research, the algorithms are novel and for the first time the adaptive algorithm are used to detect DDoS attacks. Therefore, it would be quite difficult, if not impossible, to compare the proposed adaptive method for feature extraction and

machine learning algorithms with other methods that use CAIDA dataset, but they are not adaptive algorithms to detect the DDoS attacks.

Normal and anomalous data have a long-range dependency and are bursty Internet traffic. The burstiness behavior comes from human characteristics that are bursty. The number of packets should be counted within 0.1ms. If there are more than 40 packets, there are DDoS attacks. The idea behind this detection is that humans cannot produce more than 40 adjacent packets within 0.1 ms. Moreover, DDoS attacks are related to the length of arrival packets. Therefore, the DDoS attacks in a dataset are related to different issues, and only one element that is the number of packets is not involved. In addition, anomaly-based detection methods try to find a model for the system behavior and any deviation from normal behaviors should be considered malicious.

When an expert finds a DDoS attack, which is time-consuming, the IT experts try to design an automatic algorithm that mimics expert opinion while there is more than one degree of freedom.

The normal Internet traffic data has bursty behavior. Therefore, identifying normal burst-data behaviors and abnormal burst data behaviors caused by DDoS attacks is challenging. The normal traffic and DDoS attacks are stochastic, non-stationary, non-differentiable, broadband, dynamic, self-affine fractal, and correlated with long-range dependence signals.

The bursty behavior of Internet traffic and DDoS attacks are similar, but they behave differently concerning the distribution of source IP address, the access intents, and speed of the increased and decreased traffic [PrRR13]. First, the source IP addresses in the bursty behavior of Internet traffic are extremely scattered, while the distribution of IP addresses in DDoS attacks is relatively limited to the availability of zombies. Second, in the normal bursty behavior of Internet traffic, legitimate users respond to special events, such as breaking news or popular products [PrRR13]. However, there are no special events in DDoS attacks, and flooding packets that cause

the bursty behavior are launched by a hacker. Third, the number of users who accessed the server increased gradually in the bursty behavior of Internet traffic. However, the traffic increases sharply to the peak in DDoS attacks and decreases sharply when the attacker withdraws the attack. Fourth, the bursty behavior of Internet traffic comes from randomly distributed users all over the Internet, while the flow similarity among DDoS attacks is much stronger than the bursty behavior of Internet traffic [PrRR13]. DDoS attacks are illegitimate flows, and DDoS attacks should be recognized. After detecting DDoS attacks, the goal is to terminate the flood of TCP and UDP packets to a server under DDoS attacks.

The essential difference between DDoS attack packets and flash crowd packets is shown as follows [JZZC17]: the flash crowd packets are from legitimate hosts distributed in the whole network with real source IP, but the DDoS packets are from specific zombies with huge amount of fake source IP. Therefore, the mapping relationships from the source MAC to the source IP, flash crowd packets use the actual source IP with the actual MAC, while DDoS packets use many spoofed source IPs with the actual MAC.

A DDoS attack aim is to overwhelm a server with more traffic than the server or network can accommodate. The goal is to deactivate and disable (render the website or service inoperable). The traffic can consist of incoming messages, requests for connections, or fake packets. The problem that DDoS attacks cause is a minor annoyance or long-term downtime resulting in loss of business. Therefore, to overcome these problems, a machine learning algorithm is necessary to detect DDoS attacks from the bursty behavior of Internet traffic. In the first step, the machine learning algorithm requires to extract the pattern (fingerprint) of DDoS attacks from Internet traffic. In the second step, the DDoS attacks should drop malicious traffic at the network edge using the pattern of DDoS attacks extracted using the machine learning algorithm. In the third step,

the bursty behavior of Internet traffic using routing route traffic across multiple data centers to mitigate normal bursty behavior and resource exhaustion. (Internet traffic should be routed through multiple data centers using routing traffic to reduce normal behavior and resource depletion).

## 5.4 The Software

The code of this simulation regarding making dataset, writing code for designing the mother wavelet and writing code for designing the proposed anomaly detections in this research are written by software MATLAB. In addition, for some part of this research R programing language is used.

## 5.5 The Hardware

The University of Manitoba's 2014 to 2020 CC Compute Cluster was used to distribute the computationally intensive work in this research [Thin15]. It was very user-friendly environment because running and Compute Cluster were hidden from users.

## 5.6 Summary of Chapter 5

In this chapter, the datasets, the programming languages, and the hardware used in this research were introduced.

The CAIDA dataset was chosen for the following reasons. The power consumption dataset was a simulated from smart grid infrastructure power consumption data because real cyber domain data in smart grid infrastructure data was not accessible at that time due to confidentiality and security reasons. The dataset was switched from power consumption in smart grid infrastructure

to the DDoS attacks detection in the cyber computer system. The two domains are treated entirely separately.

MATLAB software, which is an academic programming language, was used mostly because the details of implementation can be written.

Moreover, the CC Compute Cluster of University of Manitoba was used as the hardware in this research to reduce the time needed for machine learning algorithms to process heavy calculations. A personal pc would have taken several weeks or months as opposed to the CC Compute cluster.

# Chapter 6

# EXPERIMENTAL RESULTS AND DISCUSSION

In this chapter, the implementation of the anomaly detection algorithms that were introduced in chapter 4 are explained. The simulation and the results of each anomaly detection are explained.

## 6.1 The First Proposed Algorithm Simulation and Results

In this section, the simulation of the proposed model for detecting the anomalous power consumption attack in the smart grid infrastructure is described. The simulation of the proposed algorithm can be described as follows.

### 6.1.1 Model Simulation

First, the power consumption signals were generated synthetically. Therefore, an initial normal pattern of the power consumption of a customer was generated. The normal pattern had 640 samples per day. Then, elements of the normal pattern were permuted randomly between –0.3

and 0.4. Next, a set of normal patterns of power consumptions were created based on the first normal pattern of power consumption. Figure 6.1 shows one sample of a normal pattern of daily power consumption.

Next, a set of anomalous patterns of power consumption were created based on the first normal pattern. Anomalous patterns were simulated by manipulating a smart meter with a random duration of attack of 2.25 minutes, 5 minutes, 15 minutes and 75 minutes with the random starting point of attack (for example attack starts at 8:00 p.m.). Each set of normal and anomalous patterns of power consumption contained 1000 patterns. Next, the normal dataset was tagged with zero, and the anomalous dataset was tagged with one. Finally, the normal dataset and the anomalous dataset were shuffled. As a result, the gained dataset contained 2000 patterns for evaluating the proposed method.

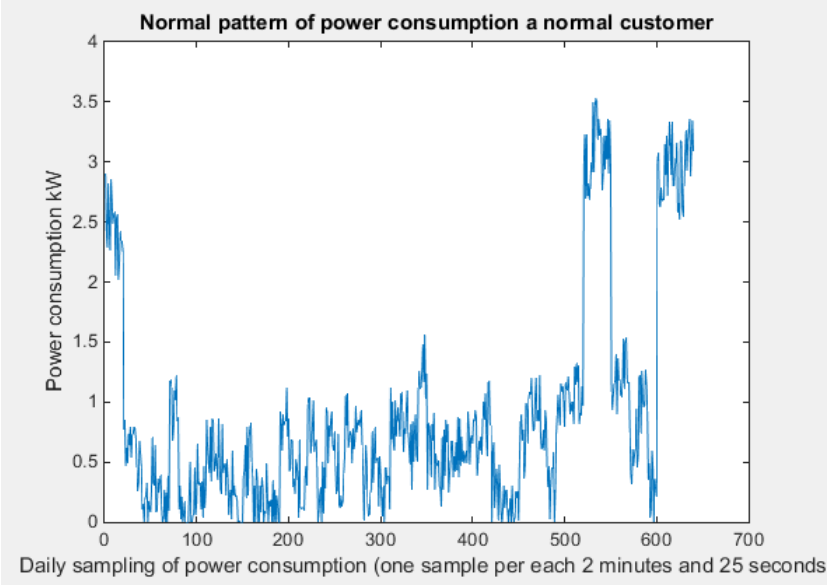



**Fig. 6.1:** A normal pattern of power consumption with 640 samples per day.

In the second step, features of power consumption patterns were extracted by the Daubechies wavelet transform 4 (db4 or D4). Different Daubechies D4 wavelet transform levels, ranging from level 3 to level 6, were applied to analyze the power consumption patterns. The

Daubechies D4 wavelet levels had different results and different sizes of output. For example, when the input of the Daubechies D4 Wavelet Transform level 3 was 640 samples, the extracted output was 660 coefficients; when the input of the Daubechies D4 Wavelet Transform level 6 was 640 samples, the extracted output was 678 coefficients.

In the third step, VFD was used for measuring the complexity of the Daubechies wavelet coefficients of power consumptions patterns. The size of the window was selected by calculating the first four moments of the time series. When the mean, variance, kurtosis, and skewness of each window are within two ranges with about a 95% confidence level, a window size can be estimated [Kins20]. In this simulation, a window size of 8192 samples was chosen in order to overcome the nonstationary problem.

At this point the VFD algorithm was calculated to find the *Hurst exponent value* (*H*) and a local VFD ($D_{\sigma i}$) for each window was determined. Finally, a VFDT that was obtained by concatenation of VFDs in each window was calculated as output at this step.

Although the purpose of the VFD was to extract hidden features of a fractal signal, the existence of thresholds was useful to classify the output of the VFD algorithm in two groups: normal and anomalous. Therefore, in the fourth step, the minimum and maximum thresholds were calculated to detect anomalous behaviors in the power consumption patterns.

If the output ( $D_{\sigma i}$ ) is between the two thresholds, the input data is normal. Therefore, the attack has not accrued or the proposed method cannot detect the attack. On the other hand, if the output is outside of the two thresholds, the proposed method detects an anomaly, or the proposed method misidentifies (wrongly suspects) the input. The minimum threshold for this algorithm is 1.969, and the maximum threshold is 1.990; this is shown in Fig. 6.2. Therefore, if the result of the VFD algorithm is less than 1.969 or above 1.990, an anomalous behavior is detected. The

VFDT in Fig. 4 is based on the extracted features of Daubechies D4 with level 5. The VFDT that is based on level 3, level 4 or level 6 of Daubechies D4 cannot distinguish between normal and anomalous behaviors in power consumption patterns (Fig. 6.3). The detection based on VFD has a crucial problem; the duration of the attack must continue for several days to change the slope in the VFD algorithm. As a result, step five must be used to detect anomalous power consumption patterns attacks of small duration (2-3 minutes).

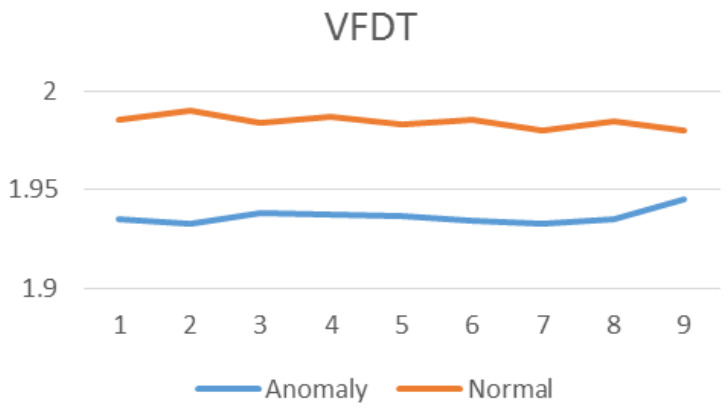



**Fig. 6.2:** Comparing of VFDT for normal data series (the upper curve) and VFDT for anomaly data series (the lower curve) based on level 5 of Daubechies D4.

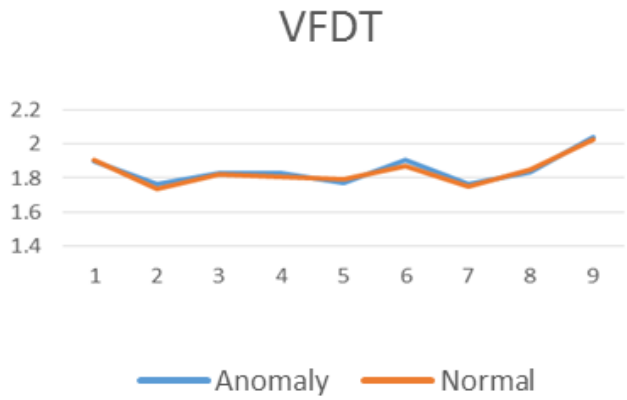



**Fig. 6.3:** Comparing of VFDT for normal data series (the upper curve) and VFDT for anomaly data series (the lower curve) based on level 4 of Daubechies D4.

Finally, the ANN parameters were learned using the ANN training phase.

### 6.1.2 ANN Training and Testing

In the ANN training phase, 80% of the coefficients were used to train the feed-forward backpropagation algorithm of ANN. The best recognition accuracy (the highest value) occurred when dataset was allocated into 80% training and 20% testing [AdOs12]. This study used the first 80% of the features (coefficients of the Daubechies wavelets and VFD for each windows) to train and the remaining 20% of the coefficients were used to test for detecting the normal and anomalous data patterns. ANN had just one output, which was either zero or one. Therefore, the existence of a threshold was necessary to classify the output in two groups of normal or anomalous. If the output was zero, the input data was deemed normal. Therefore, there was no attack, or the proposed method could not detect the attack. On the other hand, if the output was one, the proposed method detected an anomalous behavior, or the proposed method wrongly suspected the input. The last layer of the neural network (output layer) had the logistic sigmoid function, so the result of this layer was between zero and one. In programming, the different levels for the threshold are considered by a loop function. When the result of the binary decision had the least false detection (false negative and false positive), that corresponding threshold was considered the threshold of binary decision for ANN. The threshold for this algorithm was 0.5009. Therefore, if the result of the cost function in ANN was less than 0.5009, the output is zero, but if the result of the cost function in ANN was greater than 0.5009, the output was one. Also, specific parameters were used to train the feed-forward backpropagation algorithm of ANN, which is shown in Table 6.1.

**TABLE 6.1:** Specific parameters to train the feed-forward back propagation algorithm.

| Parameters | Value |
|---|---|
| Number of layers | 2 |
| epochs | 1000 |
| min_grad | 1e-06 |
| max_fail | 100000 |

In the testing phase, the remaining 20% of the coefficients are used to test and classify the normal power consumption and anomalous data patterns. Also, in the testing step, the precision of the proposed method was evaluated, and the accuracy of detection was evaluated in terms of a true positive.

### 6.1.3 Simulation Results

The true positive rate of attack detection was based on the extracted features of Daubechies D4 with level 5 (in the second step) and VFDT (in the fourth step) are shown in Fig. 6.4. The duration of the attack varied from 2.25 minutes (two minutes and 25 seconds) to 75 minutes. The weakest detection rate occurred with a 51% accuracy rate when the duration of the attack was 2.25 minutes. The highest detection occurred with a 96% accuracy rate when the duration of the attack was 75 minutes.

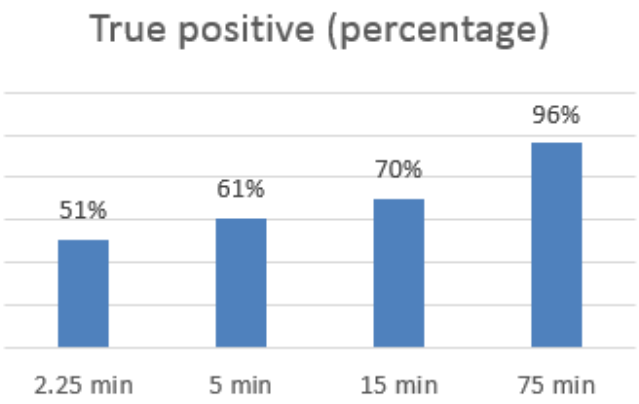



**Fig. 6.4:** True positive rate for 2.25, 5, 15 and 75 minutes anomalous power consumption attack duration.

## 6.2 The Second Proposed Algorithm Simulation and Results

In this section, the second anomaly detection algorithm simulation for detecting the DDoS attacks is described. The simulation of the second algorithm consists of two steps: processing the input data and training and testing the CNN [GhKF17]. These steps are described as follows.

### 6.2.1 Model Simulation

The simulation of the proposed algorithm followed four steps.

First, the data output was tagged based on the number of packets within 0.1 *millisecond* (ms). If there were fewer than 40 packets within 0.1 ms, the data is normal, so it is tagged as zero. If there were more than 40 packets within 0.1 ms, the data is anomalous, so it was tagged as one.

In the second step, Daubechies wavelet transform 4 (db4 or D4) and inverse Daubechies wavelet transform at each level were calculated. Different Daubechies D4 wavelet transform

levels, ranging from level 1 to level 13, were applied to analyze the raw time series to find patterns. The Daubechies D4 wavelet levels had different results and different sizes of output. Therefore, the output length of the db4 at each level was not same as the raw time series, so it was not possible to add these extracted features to the raw time series. To overcome this problem, the time series were reconstructed by inverse Daubechies wavelet transform 4 at each of the 13 levels. As a result, the raw data series was decomposed at 13 distinct signals, so low, mid and high parts of the time series signal were extracted, and then concatenated with the raw data.

Third, the stationary-window size was calculated. The size of the window was selected by calculating the first and second moments of the time series. When the mean and variance of each window were within two ranges with about a 95% confidence level, a window size could be estimated. In this simulation, a window size of 8192 samples was chosen in order to overcome the nonstationary problem.

Fourth, the CNN parameters were learned using the CNN training phase.

### 6.2.2 The CNN Training

In the training phase, the CNN learns weights and biases by the following procedure. First, a row of data from the training dataset is randomly selected. The CNN randomly initializes the weights between (–1, +1). Then, it finds the learning rate ($\alpha$) among a set of candidates: $\alpha = \{0, \ldots 0.1, \ldots, 0.5, \ldots, 1\}$. Next, to obtain the output $a_1^{(3)}$, the CNN performs the forward propagation step from the input layer to the output layer in the following way:

1) The CNN randomly chooses a training sample from dataset and temporarily removes it from the dataset.

2) Then, it computes its output by passing the data through the convolutional layer, ReLU layer, normalization layer, and finally through the pooling layer.

3) In the max pooling layer of forward propagation, $c_1$ blocks are reduced to a single maximum value, and the path of this value is found [Karp15]. Then, the pooling layer output is passed to the fully-connected layer and becomes the input of the fully-connected layer.

4) The input for each neuron $z_i^{(k)}$ in the fully-connected layer passes through the activation function of each sub-layer of the fully-connected layer to become the output of the neuron $a_i^{(k)}$. The computational output of the forward propagation step in Fig. 3.11 is shown in Fig. 6.5.

$$
\begin{aligned}
a_1^{(2)} &= g\left(z_1^{(2)}\right) = g\left(\theta_{11}^{(1)} x_1^i + \theta_{12}^{(1)} x_2^i + \theta_{13}^{(1)} x_3^i + b_1^{(1)}\right) \\
a_2^{(2)} &= g\left(z_2^{(2)}\right) = g\left(\theta_{21}^{(1)} x_1^i + \theta_{22}^{(1)} x_2^i + \theta_{23}^{(1)} x_3^i + b_2^{(1)}\right) \\
a_3^{(2)} &= g\left(z_3^{(2)}\right) = g\left(\theta_{31}^{(1)} x_1^i + \theta_{32}^{(1)} x_2^i + \theta_{33}^{(1)} x_3^i + b_3^{(1)}\right) \\
a_1^{(3)} &= g\left(z_1^{(3)}\right) = g\left(\theta_{11}^{(2)} a_1^{(2)} + \theta_{12}^{(2)} a_2^{(2)} + \theta_{13}^{(2)} a_3^{(2)} + b_1^{(2)}\right) \\
h_{\theta,b}\left(x^i\right) &= a_1^{(3)}
\end{aligned}
$$

**Fig. 6.5:** Computation of the CNN output.

The CNN computes the output-layer error based on the cost function in Eq. (3.56). The output of the forward propagation is $h_{\theta,b}(x)$ or $a_1^{(3)}$ (Fig. 6.5) and the desired output of the randomly selected training sample is $y^i$. Then, the CNN performs the backward propagation step from the output layer in the fully-connected layer to the input layer of the CNN to update each weight and bias in the CNN network. The backward propagation proceeds in the following way:

1) First, to calculate the gradient descent algorithm, the CNN computes the differentiation $\frac{\partial}{\partial \theta_{ij}} J(\theta)$ by using the chain rule on the cost function $J$.

2) The CNN uses the gradient descent algorithm to update each weight and bias in the fully-connected layer through backward propagation.

3) In the pooling layer, backward propagation entails finding a proper path to pass the error from the fully connected layer to the convolution layer. First, in the max pooling layer, the maximum value path previously discovered through forward propagation is tracked. Second, the error is passed back to the neuron that carried the maximum value in the forward propagation. In other words, the error backward propagates through this maximum value path from the fully-connected layer to the convolution layer, passing unscathed through the non-learning normalization and ReLU layers.

4) The CNN uses the gradient descent algorithm to update the weights of the filter coefficients in the convolutional layer. The backpropagation for the convolution layer is a convolution of the input and the flipped weights of the filter [Karp15].

Next, the CNN removes the row of data from the temporary dataset. Finally, another row is randomly selected until all elements have been chosen.

Table 6.2 shows the parameters that were considered to train the CNN with the one dimensional time series input. When the learning rate, $\alpha$, was 0.6, the convergence of error to zero had a suitable rate.

**TABLE 6.2:** Specific parameters to train the CNN with one dimensional input.

| Parameters | Value |
|---|---|
| Learning rate | α = [0.007,0.07,0.1, 0.2,0.5,0.55,0.6, 0.7, 1] |
| Epoch | 1500 |
| Number of features of dataset | 6 |
| Filter bank length (convolutional layer) | vector [3×1] |
| Filter bank bias | vector[1×1] |
| Weights of hidden layer in the fully connected layer | vector[6×13] |
| Weights of bias in hidden layer | vector[1×1] |
| Weights of output layer in the fully connected layer | vector[13×1] |
| Weights of bias in fully connected layer | vector[1×1] |
| Stride | 1 |
| Window size in normalization layer | 3 |
| Pooling size (pooling layer) | 2 |

The best learning rate could be gained by comparing the error of the cost function in each step with the number of iterations. When the learning rate was too small, the cost function error slowly converged to zero. When the learning rate was too large, the cost function error diverged to infinity.

### 6.2.3 The CNN Testing

In the testing phase, just the forward propagation of the CNN network is called. The trained parameters (weights and biases) that are obtained from the training phase are used in the testing phase. The testing phase proceeds in the following way: First, put testing dataset in the input layer after process it. Second, compute the output of the CNN by the trained parameters (with the testing dataset). Third, find threshold level to find the rate of false detection (false positive and false negative) of the algorithm. Fourth, find the true positive detection rate as follows: If the result is

less than threshold level, the data is normal. If the result is greater than threshold level, there is a DDoS attack.

### 6.2.4 Simulation Results

In this section, the result of the one stage CNN and the pre-processed data using DB4 with the one stage CNN are explained. As can be seen in Table 6.3, the detection rate was 56.1% when the one stage processing was applied. When the pre-processing step was added, the detection rate was 80.77% that is shown in Table 6.4. Although the processing time was increased by adding the preprocessing step, the accuracy rate increased by almost 25%.

**TABLE 6.3:** Simulation result of the one stage CNN.

| Parameters | Value |
|---|---|
| Learning rate | 1 |
| Number of features as input to CNN | 6 |
| Number of stages in the CNN (convolution layer– ReLU layer – Normalize layer– Pooling layer) | 1 |
| Training dataset size | 8192 |
| Testing dataset size | 8192 |
| Threshold | 0.91 |
| Running time of training step | 2873 second |
| Running time of testing step | 1 millisecond |
| True Positive Rate | 56.1% |

**TABLE 6.4:** Simulation result of the preprocessed data by DB4 and one stage CNN.

| Parameters | Value |
|---|---|
| Mother wavelet transform | Daubechies D4 |
| Number of features as input to CNN | 32 |
| Learning rate | 0.55 |
| Number of stages in the CNN (convolution layer– ReLU layer – Normalize layer– Pooling layer) | 1 |
| Training dataset size | 8192 |
| Testing dataset size | 8192 |
| Threshold | 0.91 |
| Running time of training step | 16260 second |
| Running time of testing step | 1 millisecond |
| True Positive Rate | 80.77% |

## 6.3 The Third Proposed Algorithm Simulation and Results

In this section, the third anomaly detection algorithm simulation for detecting the DDoS attacks is described. The simulation consists of three steps: processing the input data, training and testing the CNN and training and testing SVM [GhKi18][GhKi19]. These steps are described as follows.

### 6.3.1 Model Simulation

In this section, the simulation of the proposed DDoS attack detection algorithm is described. The simulation of the proposed algorithm followed five steps, which are described below.

First, the data output was tagged based on the number of packets within 0.1 *millisecond* (ms). If there were fewer than 40 packets within 0.1ms, the data was considered normal, so it was tagged as zero. If there were more than 40 packets within 0.1ms, the data was considered anomalous, so it was tagged as one.

Second, the VFDTv2 was used to measure the complexity of the ATS data. To overcome the nonstationary problem, the stationary-frame size was selected by calculating the first and the second moment of the ATS data. In general, a frame size can be selected when the mean and variance of each frame are within two ranges, with about a 95% confidence level [Kins15]. In this simulation, a frame size of 8192 samples was chosen, and the lower bound in each frame, $k_{low}$, was assigned a zero value. At this point, the VFDv2 algorithm was performed to find a $D_\sigma$ for each frame. In addition, an overlapping scheme of the VFDv2 algorithm was used to extract underlying features in real time, so the samples from each frame and its adjacent frame were overlapped. In this simulation, an overlapping size of 8191 samples was chosen to achieve a 99.98% overlap. Therefore, sensitivity to anomalous behaviors was increased in real time and any deviation from normal behaviors was declared. Finally, a VFDTv2, which was obtained by a chain of VFDv2s in each frame, was calculated as the output at this step. The output of this step was concatenated with the data.

Third, the *Daubechies wavelet transform 4* (db4) and the inverse Daubechies wavelet transform were calculated at each level. Different Daubechies db4 wavelet transform levels, ranging from level 1 to level 13, were applied to analyze the ATS data and the output of the VFDTv2 time series to find patterns. Different Daubechies db4 wavelet levels had different results and different sizes of output. Therefore, the output length of the db4 at each level was not the same as the input, so it was not possible to add these extracted features to the input, the ATS data and the output of the VFDTv2. To overcome this problem, the time series were reconstructed by the inverse Daubechies wavelet transform 4 at each of the 13 levels. As a result, the ATS data, the output of the VFDTv2, and the decomposed 13 distinct signals were concatenated with the raw data.

For the fourth step, the same CNN parameters described in section 6.2.2 were used to train the CNN this time. Based on the calculation described in step two, the stationary-frame size was calculated. Next, outputs of previous steps were normalized within the stationary-frame size. To meet the convergence of error to zero at early iterations, the learning rate, $\alpha$, was set to 0.6. Using 50% of the dataset, the CNN was trained to extract the features automatically, and obtain coefficients from the dataset. Next, using about 50% of the remaining dataset, the CNN was tested with the obtained coefficients of the CNN training step.

Fifth, the data, the pre-processed data and the output of the training step were passed to the SVM for training the SVM parameters. The input of the SVM is the combination of the raw data, the ATS data, the pre-processed data, and the output of the processing step. The post-processing step increases the detection rate while reducing the detection processing time. The parameters were a vector of the weight coefficients, $w$, and a constant, $b$. Table 6.5 shows the parameters to train the SVM. The remaining raw dataset, the remaining pre-processed dataset, and the output of the CNN testing phase were passed to the SVM testing step, to classify normal and anomalous behaviors.

**TABLE 6.5:** Specific parameters to train the SVM with one dimensional input.

| Parameters | Value |
|---|---|
| SVM structure | Classification |
| Kernel function | Sigmoid kernel |
| Penalizes misclassified cost | 2 |
| Vector of coefficients in SVM (w) | Vector[60×1] |
| The constant in SVM (b) | Vector[1×1] |

### 6.3.2 Simulation Results

In this section, the justification for each step and the results of the proposed anomaly detection algorithm are explained.

The VFDT was chosen for the pre-processing step because it is immune to noise, and it can be utilized in real time [Kins15]. The VFDv2 algorithm was introduced because the ATS data was not a pure fractal. Therefore, the average measure regarding all points of the ATS data with various scales should be considered. The result of the VFDv2 algorithm was between 1.1329 and 1.1828 for $k_{low} = 0$, and the result of the VFDv2 was between 2.0069 and 2.0163 for $k_{low} = 1$, which are shown in Fig. 6.6. In addition, the sliding window VFDTv2 was used to increase sensitivity of the VFDv2 algorithm to extract features from a short duration of DDoS attack.

The DWT was chosen in pre-processing step because it is sensitive to irregularities in signals. Firstly, the DWT coefficients with anomalous events have larger magnitudes when compared to the DWT coefficients without anomalous events. Secondly, the DWT reacts to the change in the first derivative (slope) of a signal, not to the change in the amplitude of a signal. The db4 as a basis function for the DWT is used because it is suitable to detect an anomaly with a low amplitude, short duration, fast calculation of DWT coefficients, fast rate of decay, and rapid oscillation in a signal [SaKi06].

Training of a CNN has a long duration. In addition, the classification accuracy of a CNN is improved by increasing the number of its stages, which decreases the speed of the training. To overcome this problem, the SVM was considered to increase the detection rate.

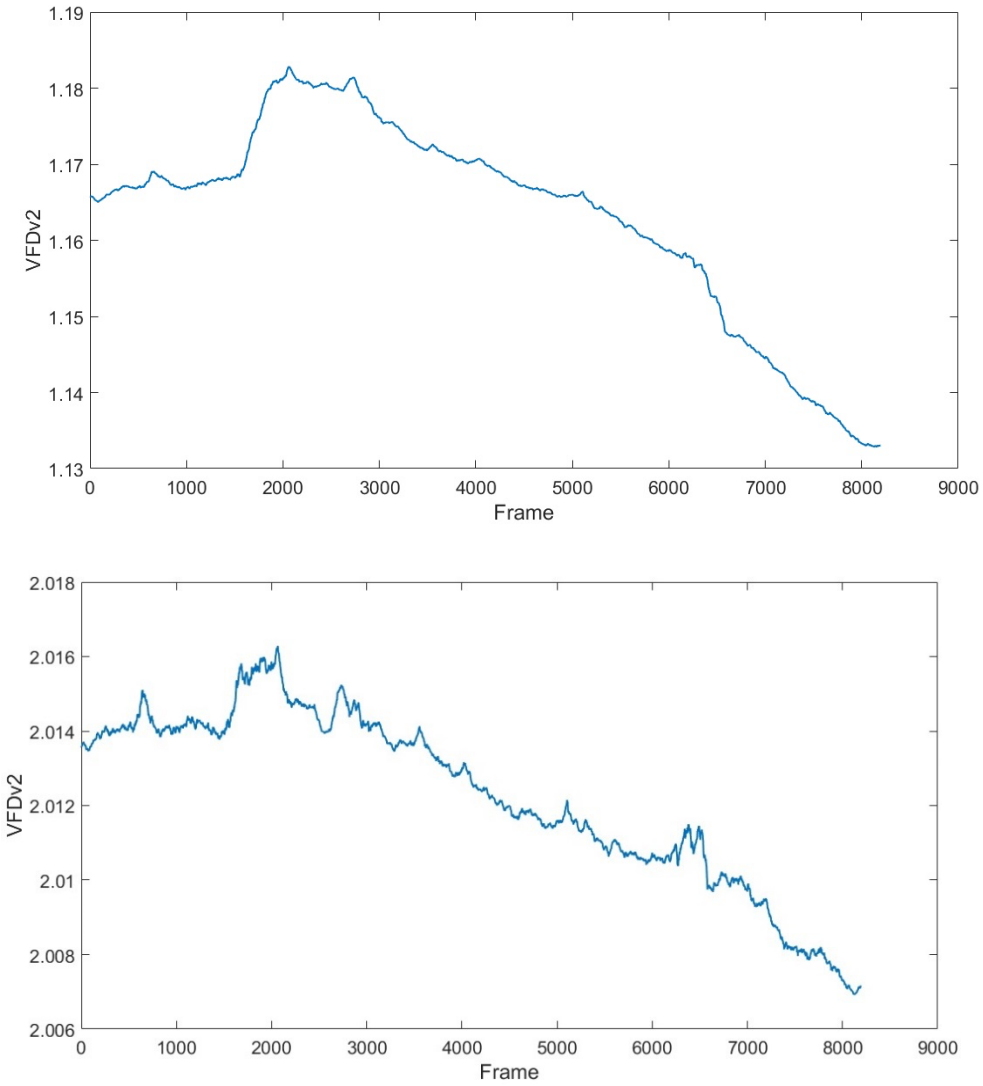



**Fig. 6.6:** The trajectory regarding VFDv2 algorithm of the ATS data within the stationary frame and $k_{low}$ =*0* (the above figure), and $k_{low}$=*1* (the below figure).

In each proposed algorithm step, detection rates of DDoS attacks are evaluated as shown in Tables 6.6 – 6.10. With only considering a single-stage CNN, the detection rate was 56.1% [GhKF17]. With considering the db4 pre-processing step and a single-stage CNN, the detection rate was 80.77% [GhKF17], which was an increase of 24.6% from the previous result. With considering the db4, the VFDTv2 and a single-stage CNN, the detection rate was 81.73%. As a

result, the distinguishing features in pre-processing step were sensitive to anomalous behaviors in comparison with normal behaviors.

Finally, after extracting coefficients with the db4 and the VFDTv2 in the pre-processing stage, a single-stage CNN, and the SVM as post-processing, the detection rate was 87.35%, which was an increase of 5.6% from preprocessed data by the db4 and the VFDTv2, and one stage of the CNN. As a result, the pre-processing step contributed significantly to the detection rate compared to the post-processing step.

**TABLE 6.6:** Simulation result of the one-stage CNN as a processing step [GhKF17].

| Parameters | Value |
|---|---|
| Number of features as input to CNN | 6 |
| True positive rate | 56.1% |

**TABLE 6.7:** Simulation result of the pre-processed data by the db4 and one stage CNN [GhKF17].

| Parameters | Value |
|---|---|
| Number of features as input to CNN | 32 |
| True positive rate | 80.77% |

**TABLE 6.8:** Simulation result of the preprocessed data using the VFDv2, the db4 and the one-stage CNN.

| Parameters | Value |
|---|---|
| $k_{low}$ (a parameter for the VFDv2 algorithm) | 1 |
| Number of features as input to CNN | 59 |
| True positive rate | 81.73% |

**TABLE 6.9:** Simulation result of using the SVM.

| Parameters | Value |
|---|---|
| Number of features as input to SVM | 6 |
| True positive rate | 28.1% |

**TABLE 6.10:** Simulation result of the pre-processed data using the db4 and the VFDv2, the one-stage CNN as a processing step and the SVM as a post-processing step.

| Parameters | Value |
|---|---|
| $k_{low}$ | 1 |
| Number of features as input to CNN | 59 |
| Number of features as input to SVM | 60 |
| False positive rate | 12.26% |
| False negative rate | 0.37% |
| True positive rate | 87.35% |

The proposed detection DDoS attacks method was based on supervised learning, so it needed tagged data as its input. To detect unlabeled data, an extension of the method is required, so a non-supervised learning method should be used. In addition, this research was conducted to detect vulnerabilities and DDoS attacks, so to enrich the paper, preventing of attacks or mitigating the effect of the attacks can be applied. Therefore, a severity score, which is a measure of the risk, regarding a vulnerability can be assigned to make a priority to response a threat for mitigating the effect of the threat [SuJo15].

### 6.3.3 Discussion

Since the proposed detection DDoS attacks method is based on supervised learning, it needs tagged data as its input. To detect unlabeled data, an extension of the method is required such as one of the non-supervised learning methods.

Moreover, one common method to generate data artificially to expand datasets for training deep learning algorithms and increasing classification rate is data augmentation. One important condition is that the data augmentation should not affect constraint of the data. For example, when applying data augmentation methods to an image, the result is still that similar image with the same tag [KSGH12] because the data augmentation preserves constraint of data. When using an image

dataset or a sound dataset, the augmented dataset has the same characteristics as the original dataset [BjGS17]. However, the current study considers time-series signals that contain DDoS attacks. Data augmentation affects constraint of this data. For example, consider an *Electrocardiography* (ECG) signal. A healthy heart beats between 60 to 100 beats per minute [Hart15], but with stretching, an ECG signal that shows one beat per minute does not belong to a healthy heart. As a result, the constraint of the signal is not preserved any more. Likewise, data augmentation cannot apply for time-series signals that contain DDoS attacks. (The time-series signals that contain DDoS attacks rely on long-range dependence that can be measured by the Hurst exponent. The closer to one the value of the Hurst exponent the greater the degree of long memory or long-range dependence [ZhHT11]. In the current study, the time-series signals have the dimension between 2.0069 and 2.0163. According to Eq. (3.42), the Hurst exponent has values between 1.00345 and 1.00815, which are close to one, so the data has a heavy tail. Internet traffic with this characteristic follows the Poisson distribution [ZhHT11] and the Levy distribution. The ATS data, which is calculated as a new attribute in the current study, is based on the difference between packet-arrival times. Also, the data output was tagged as normal or anomalous based on the number of packets within 0.1 ms. Therefore, stretching, squeezing, cutting a chunk of data and other methods of data augmentation change the ATS data and the tagged output, so the constraints are not preserved. For example, one situation in which the constraints are not preserved is when data fails to follow the Poisson distribution. Therefore, data augmentation does not apply to ECG and Internet traffic data. To increase anomaly detection, another CNN architecture should be applied.

To demonstrate this phenomenon, the effects of six augmentation methods on the long-range dependence of DDoS ITD and "Test" dataset were analyzed to show how data augmentation

methods can destroy the long-range dependence of some time series datasets by affecting their PMF [GhKi20a].

For the augmented DDoS ITD, none of the methods preserved the constraints of the original data as evidenced by the changed PMF. However, the PMF of the augmented "Test" dataset, was close to the original and the data followed the normal distribution, so the constraints were preserved. Therefore, the introduced data augmentation methods could be applied to audio time series data but not to the Internet time series data.

## 6.4 The Fourth Proposed Algorithm Simulation and Results

### 6.4.1 Model Simulation

In this section, the simulation of the proposed DDoS attack detection algorithm is described. First, a new attribute called a packet arrival time-series signal is calculated by using the difference between a packet-arrival time at *t* and the packet-arrival time at *t-1*. This research uses the packet arrival time-series signal as the input data. Second, the data output was tagged based on the number of packets within 0.1 *millisecond* (ms). If there were fewer than 40 packets within 0.1ms, the data was considered normal, so it was tagged as 0.25 (instead of zero). If there were more than 40 packets within 0.1ms, the data was considered anomalous, so it was tagged as 0.75 (instead of one). These tags are considered to overcome infinity results from operations like division by zero and overflow of the combined cost function in Eq. (4.8). Third, the MOEA's population had 100 individuals were initialized randomly. For the crossover operation, a random was used for each parent. The Eq. (3.18), Eq. (3.21), Eq. (3.22), Eq. (3.24) and the proposed cost function in Eq. (4.8) were used as the fitness function for the optimization of the MOEA. The size of the Pareto front was equal to 50. The Pareto front population had the following values: initial

weight range was between +1 and –1. Fourth, the weighted cost function consists of the KL-divergence and the combination of extracted features was analyzed. The cost function had the highest sensitivity among other variations of proposed cost function [GhKi20b] [GhKi21e]. Fifth, the training set had 8192 individuals. The network training was stopped when a pre-specified number of epochs 1000 was reached. The Pareto front of the mother wavelets is created based on the steps above. Sixth, the GNN architecture in Fig. 4.4 was used to select the best mother wavelet, but instead of the MOEA that had several constraints, the proposed cost function in Eq. (4.8), as the cost function, was considered. The best mother wavelet from the Pareto front is shown in Fig. 6.7. Seventh, the power of two for each constraint was used to overcome the cancellation of error for each constraint with other constraints [ReMa18].

### 6.4.2 Training and Testing of the GNN

To adjust the adaptive mother wavelet to the DDoS ITD as the input, the weights of the GNN should be trained. When the GNN is trained, the mother wavelet's constraints are match to the input time series signal. The multi-objective optimizer genetic algorithm can be used to find the Pareto front as the best solutions of the GNN for designing a mother wavelet. The supervised learning algorithm is used to train the GNN for obtaining coefficients of a mother wavelet. Also, the proposed weighted cost function is used to find difference between the neural network output and the desired output.

In the feed-forward neural network, each neuron in this layer will be connected to all the neurons in the network. Every pathway has its own parameter or weight. These weights are connected to the output layer (matrix of weights $\theta_{ij}^{(1)}$) that are shown in Fig. 4.4. For further details on training see [GhKi19].

The backpropagation optimization can detect the local minima with a high accuracy. However, this optimization may not detect the global minima and its convergence rate is slow. To overcome the backpropagation problems, a combination of the backpropagation optimization and the genetic algorithm can be used because a genetic algorithm creates multiple solutions to a given problem and evolves them through a number of generations. Therefore, the GA is less likely to become stuck in local minima, and it has strong capability to find global minima with a high probability. Therefore, the weights of a neural network can be initialized close to global minima with a genetic algorithm, and to find the high accuracy the backpropagation can be launched [LuXL12].

Adding a backpropagation algorithm to the genetic algorithm significantly reduces the amount of time for converging to the result and greatly increases the accuracy of the result [RaSe15]. The backpropagation algorithm of the GNN updates the GNN weights to converge the best mother wavelet in the Pareto front to the optimal mother wavelet coefficients. In fact, this convergence is needed for adapting the mother wavelet coefficients to the input time series.

### 6.4.3 Detecting Attacks based on Proposed Mother Wavelet

When the adaptive-wavelet CNN is trained, the supervised learning algorithm is used to train the adaptive-wavelet CNN for detecting the DDoS ITD. Also, the proposed weighted cost function is used to find difference between the neural network output and the desired output.

In the feed-forward neural network, each neuron in this layer will be connected to all the neurons in the network. Every pathway has its own parameter or weight. These weights and the architecture for feed-forward network that are connected to the output layer (matrix of weights $\theta_{ij}^{(1)}$, $\theta_{ij}^{(2)}$ and $\theta_{ij}^{(3)}$) are shown in Fig. 4.7. In the forward propagation, the output of each node is

calculating by multiplication of input of each node by the weight of the corresponding input. In the forward pass, the output of the neural network is calculated, and the computation of the adaptive-wavelet CNN output in Fig. 4.7 is shown in Eq. (6.1).

$$a_1^{(4)} = g(z_1^{(4)}) = g(\theta_{11}^{(3)} a_1^{(3)} + \ldots + \theta_{1M}^{(3)} a_M^{(3)} + b_3) \tag{6.1}$$

The output of $a_1^{(3)}$ in Fig. 4 is shown in Eq. (6.2).

$$a_1^{(3)} = g(z_1^{(3)}) = g(\theta_{11}^{(2)} a_1^{(2)} + \ldots + \theta_{1M}^{(2)} a_M^{(2)} + b_2) \tag{6.2}$$

The output of $a_1^{(1)}$ and the output of $a_2^{(1)}$ are shown in Eq. (6.3) and Eq. (6.4) respectively.

$$a_1^{(1)} = c_1^{(3)} h_0 + c_2^{(3)} h_1 \tag{6.3}$$

$$a_2^{(1)} = c_1^{(3)} g_0 + c_2^{(3)} g_1 \tag{6.4}$$

where $h_0$ and $h_1$ are the low-pass filter coefficients, and $g_0$ and $g_1$ are the high-pass filter coefficients. The neuron of $c_1^{(1)} = x_1$ to the neuron $c_8^{(1)} = x_8$ is the training sample of the input data (the DDoS ITD).

In the backward pass, the network weights are updated. To update the weights of neurons in the training step of the neural network, the backpropagation algorithm is used as shown in Eq. (6.5).

$$\theta_{ij}^{(k)} = \theta_{ij}^{(k)} - \alpha \frac{\partial}{\theta_{ij}} J(\theta) \tag{6.5}$$

where $J(\theta)$ is the proposed cost function (Eq. (4.8)) in Eq. (6.5), and the differentiation $\frac{\partial}{\theta_{ij}} J(\theta)$ is calculated while using Eq. (6.6).

$$\frac{\partial J}{\partial \theta_{ij}^{(1)}} = \frac{\partial J}{\partial a_i^{(4)}} \cdot \frac{\partial a_i^{(4)}}{\partial z_i^{(4)}} \cdot \frac{\partial z_i^{(4)}}{\partial a_i^{(3)}} \cdot \frac{\partial a_i^{(3)}}{\partial z_i^{(3)}} \cdot \frac{\partial z_i^{(3)}}{\partial a_i^{(2)}} \cdot \frac{\partial a_i^{(2)}}{\partial z_i^{(2)}} \cdot \frac{\partial z_i^{(2)}}{\partial \theta_{ij}^{(1)}} \tag{6.6}$$

Each element in Eq. (6.6) is calculated from Eq. (6.7) to Eq. (6.11) as follows:

$$\frac{\partial J}{\partial a_1^{(4)}} \cdot \frac{\partial a_1^{(4)}}{\partial z_1^{(4)}} = -w_1 . (y_i)^2 . (1 - a_1^{(4)}) + w_2 . (1 - y_i)^2 . a_1^{(4)} \tag{6.7}$$

$$z_1^{(4)} = \theta_{11}^{(3)} a_1^{(3)} + \ldots + \theta_{1M}^{(3)} a_M^{(3)} + b_3 \tag{6.8}$$

$$\frac{\partial a_1^{(4)}}{\partial z_1^{(4)}} = a_1^{(4)} (1 - a_1^{(4)}) \tag{6.9}$$

$$\frac{\partial z_i^{(4)}}{\partial a_i^{(3)}} \cdot \frac{\partial a_i^{(3)}}{\partial z_i^{(3)}} = \theta_{ij}^{(3)} . a_i^{(3)} (1 - a_i^{(3)}) \tag{6.10}$$

$$\frac{\partial z_i^{(3)}}{\partial a_i^{(2)}} \cdot \frac{\partial a_i^{(2)}}{\partial z_i^{(2)}} = \theta_{ij}^{(2)} . a_i^{(2)} (1 - a_i^{(2)}) \tag{6.11}$$

$$\frac{\partial z_1^{(2)}}{\partial \theta_{11}^{(1)}} = a_1^{(1)} \tag{6.12}$$

Once a network is trained, it is evaluated using the test case. For the testing step of the neural network, only forward propagation is used with considering the values of the weights and the bias that obtained from the training step.

### 6.4.4 Results and Discussion

First, the classification accuracy was improved by increasing the number of input features and the number of input node, so the outputs of feature extraction methods were added to the second input layer. The input signal of the extraction techniques was the ATS, and a stationary frame size of data samples was required to obtain a realistic result. The output length of all techniques was one except for the DWT technique which has an output length of 121. The output

of this preprocessing step was horizontally concatenated with the data. Moreover, the DWT technique was used twice; the first time the input was the ATS and the second time the input was the VFD. In last cases, the output of DWT in both cases was concatenated with the data vertically.

Second, the data output was tagged based on the number of packets within 0.1 *millisecond* (ms). If there were fewer than 40 packets within 0.1ms, the data was considered normal, so it was tagged as 0.25 (instead of zero). If there were more than 40 packets within 0.1ms, the data was considered anomalous, so it was tagged as 0.75 (instead of one). These tags are considered to overcome infinity results from operations like division by zero and overflow of the proposed cost function in Eq. (4.8).

Third, the weighted KL-divergence cost function using the VFD was the most sensitive cost function and had the highest eigenvalue [GhKi20b]. To calculate the VFD, the input time series must have a stationary frame size data samples to gain a realistic result. To overcome this problem, the data analysis process should be delayed until at least 8192 data samples come together to make a frame, or the input time series 8192 data samples from Internet traffic must be added before the starting point of the input. Therefore, the processing data has 8192 data samples delay in the worst-case scenario. Moreover, the weighted KL-divergence cost function had the lower running time and faster convergence to the optimal coefficients of the neural network than the non-weighted cost functions.

Fourth, the GNN parameters that adapt to generate discrete mother wavelet coefficients were trained. The GNN was tested with the obtained coefficients of the GNN training step. Table 6.11 shows the parameters to train the GNN to design the proposed adaptive wavelet algorithm, and the best mother wavelet that was created using the GNN is shown in Fig. 6.7.

**TABLE 6.11:** Specific parameters to train the Pareto front of the GNN to design the mother wavelet.

| Parameters | Value |
|---|---|
| Number of nodes in input layer | 2 |
| Weight initialization | Randomly |
| Initial weight range | { +0.25, – 0.125} |
| Learning rate | 0.001 |
| The Pareto front size | 50 |

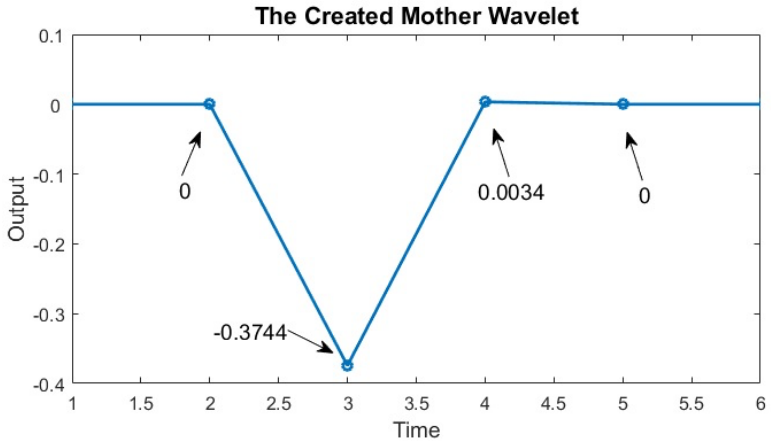



**Fig. 6.7:** The best mother wavelet for detecting DDoS attacks with coefficients [– 0.3744, 0.0034].

Fifth, the adaptive-wavelet CNN parameters were trained. The convolution layer in the adaptive-wavelet CNN structure has fixed weights that were a population of the Pareto front in each learning iteration (one at the time). Table 6.12 shows the parameters and the criteria for the adaptive-wavelet CNN training and testing. The evaluation results were based on the average of the results in different sections of the dataset. Using the adaptive mother wavelet in Fig. 6.7 and the adaptive-wavelet CNN structure, the detection rate was increased that was 95% accurate. The

detection rate was increased by almost 7% when compared with the third proposed anomaly detection algorithm [GhKi18][GhKi19].

**TABLE 6.12:** Specific parameters to train and test the adaptive-wavelet CNN.

| Parameters | Value |
|---|---|
| Learning rate | α = 0.001 |
| Epoch | 1500 |
| Number of nodes in the first input layer | 8 |
| Number of nodes in the second input layer and two hidden layers of the FCL | 173 |
| Number of nodes in the output layer of the FCL | 1 |
| Window size for normalization | 8192 |
| Dataset size | 166448 |
| Training dataset size | 8192 |
| Testing dataset size | 8192 |
| The best mother wavelet coefficients using GNN | [–0.3744,0.0034] |
| Weights initialization | Randomly |
| The activation function | Sigmoid function |
| The detection rate (Accuracy) (%) | 95% |

Sixth, due to the DDoS ITD data were not balanced, part of the dataset with more than 99% of packets contained DDoS attacks and less than 1% contained normal packets. Therefore, when the adaptive-wavelet CNN could not detect normal packets, the true positive was zero even though the detection rate and the true negative was 99.97%. Under these circumstances, the F1-Score was infinite.

### 6.4.5 Discussion

One of the most important innovative features of the fourth proposed algorithm was choosing a simple, unbiased architecture for the GNN. This shallow neural network architecture (seen in Fig. 4.4) had two layers: an input layer and an output layer. This architecture was used because a mother wavelet selected from the Pareto front that converged to be the best mother

wavelet had two coefficients. The compact support illustrated in Fig. 6.7 equals to two. If a neural network with one or several hidden layers with biases had been considered, when training this deep neural network the weights would have had to be trained as well. Using such a complex neural network would have made it difficult to find the correct coefficients for the adaptive mother wavelet. Also, the extra pathways would have had to be ignored, which means that only two pathways could be used. After training, it would have been difficult to select the mother wavelet coefficients from the neural network weights. Moreover, just using some of the pathways as weights as the mother wavelet coefficients would not have been acceptable rationally. The pathways cannot be ignored because the optimum coefficients of the mother wavelet are extracted with all the weights in the deep neural network. Therefore, if just one group of pathways in the neural network architecture were considered, the best mother wavelet coefficients could not be achieved.

One of the innovative features of the fourth proposed algorithm was the combination of the genetic algorithm and the artificial neural network GNN to achieve the best mother wavelet during simulation.

The VFD was chosen for the design of the proposed cost function because the Internet traffic that contains DDoS attacks is a multi-fractal signal [GhKi18]. To extract hidden features and exaggerate the features, the VFD, which is a poly-scale analysis, was used. Also, since the energy cost function cannot extract information from data, and it considers only the amplitude of data the KL-divergence was chosen to extract relevant features embedded in the data. To enrich the KL-divergence, the combination of the complexity of the data and the distance feature was added as a coefficient using the PCA. To evaluate the proposed cost functions and select the best one as shown in Eq. (4.8) for the current application, sensitivity analysis was used [GhKi20b].

One of the most important characteristics of the Internet datasets is that they are changing rapidly, and any dataset acquired at a given time represents the nature of the Internet at that time. Consequently, an adaptive mother wavelet has been introduced to extract hidden features from this changing environment. The adaptive-mother wavelet algorithm, GNN, extracts distinguishable features to separate normal from anomalous behaviors within a given current environment. The extraction of distinctive features plays an important role in increasing the detection rate. Thus, the proposed adaptive GNN algorithm has different mother wavelet coefficients for different datasets because the mother wavelet adapts to its input signal, so different satisfactory details and trends are extracted in each dataset. In other words, the coefficients of the mother wavelet should be changed adaptively for different datasets with different characteristics. As a result, no other dataset is used in this research because the results vary from one dataset to another. However, for the same dataset, the results are consistent. Therefore, for different datasets, different mother wavelet coefficients should be obtained through the following steps. First, the adaptive GNN algorithm should be launch for each dataset. Second, according to the obtained mother wavelet coefficients, the anomaly detection method should be conducted to detect anomalous behavior in the selected dataset.

In general, the accuracy rate of detection of the proposed mother wavelet with coefficients [–0.3744,0.0034] shown in Fig. 6.7 was increased by 0.3% making it more accurate than the Haar mother wavelet. However, the detection rate of the proposed mother wavelet had the same detection rate accuracy as the Haar mother wavelet in some scenarios. The coefficients of the created mother wavelet can be used to improve the detection of the DDoS attacks in Internet traffic.

Moreover, when the first coefficient of the created mother wavelet has the deepest value, the shape similarity between the mother wavelet and the input time series signal was increased.

Therefore, a higher DDoS attack detection rate is achieved. However, there is tread of between the detection rate and other constraints that were introduced in Eq. (4.3.a), Eq. (4.3.b), Eq. (4.3.c), Eq. (4.3.d) and Eq. (4.8). The power of two for each constraint was used to overcome the cancellation of error for each constraint with other constraints [ReRi18].

In general, the average value of the mother wavelet coefficients mother wavelet must be zero to have finite energy. With infinite energy however, it is not possible to cover the mother wavelet frequency spectrum and time duration. However, the energy of the created mother wavelet was not zero, but this was correct because the DDoS ITD data had a finite stationary frame size and the Fourier transform of the created mother wavelet within this stationary frame could cover the frequency spectrum and time duration with the created mother wavelet.

In addition, evaluating the dataset to find the normal or anomalous data is important in this application, and the result of the evaluation is important. In other words, reconstruction of the original data from the processed data using the created mother wavelet is not important. As a result, the proposed mother wavelet is valid for this application.

Moreover, it is important to make the shape of a mother wavelet similar to the shape of input time series signal to overcome the Gibbs phenomenon; otherwise, the discrete wavelet transform needs an infinite number of coefficients to show corners, and the results are not accurate. In this research, the created mother wavelet is similar to the DDoS ITD because the mother wavelet had a sharp edge exactly like the DDoS ITD.

The proposed method for creating mother wavelet showed for compact support of two. However, it can be used for other compact support to design mother wavelets as well.

In addition, this method of designing a mother wavelet is useful for people in the computer field who do not sufficient physics or mathematics background to design a mother wavelet.

Since the proposed detection DDoS attacks method is based on supervised learning, it needs tagged data as its input. The next method is expanded to detect unlabeled data, an extension of the method is required the DRL.

One of the most important characteristics of the adaptive-wavelet CNN structure is its adaptability. The input data used in this research, the DDoS ITD, were not static, but a dynamically changing environment. Therefore, a structure for adaptive anomaly-based intrusion detection system was proposed. This structure was called adaptive for the following reasons. First, extracting distinguishable features played an important role to increase the detection rate. The most challenging step in the DWT, which extracts distinguishable features, was to select its mother wavelet. In this research, the mother wavelet was used that adapts to its input signal. Therefore, satisfactory details and trends in the DDoS ITD were extracted. Second, the number of nodes in the input layer, convolution layer and the FCL were adaptive to the number of features in the input and the number of the mother wavelet coefficients. Third, adding weights to the KL-divergence cost function could magnify the anomalous data, to enhance their impact on the cost function. The extracted features from the input data were used as the weights to adapt the weighted cost function to the input data. Therefore, a faster convergence of the optimal coefficients regarding the adaptive-wavelet CNN was obtained.

Since the proposed adaptive-wavelet CNN structure is based on supervised learning, it needs tagged data as its input. Potential future research could be expanded to an adaptive detection of untagged data, with a corresponding extension of the method such as one of the deep reinforcement learning algorithms.

The characteristics of Internet dataset are varying and rapidly changing. A proper method to detect new patterns of the DDoS attacks in this changing environment is adaptive machine

learning algorithm. Moreover, an adaptive feature extraction method is introduced to extract hidden features from this changing environment. Therefore, an adaptive mother wavelet algorithm is introduced in this thesis based on a stationary frame size of input data. Extracting distinguishable features played an important role to increase the detection rate. The most challenging step in the DWT, which extracts distinguishable features, was to select its mother wavelet. In this research, the mother wavelet was used that adapts to its input signal. Therefore, satisfactory details and trends in the DDoS ITD were extracted.

Moreover, a new CNN architecture, which is adaptive, for detecting the DDoS attacks is introduced in this thesis. The number of nodes in the input layer, convolution layer and the FCL were adaptive to the number of features in the input and the number of the mother wavelet coefficients. Furthermore, adding weights to the KL-divergence cost function could magnify the anomalous data, to enhance their impact on the cost function. The extracted features from the input data were used as the weights to adapt the weighted cost function to the input data. Therefore, a faster convergence of the optimal coefficients regarding the adaptive-wavelet CNN was obtained.

Therefore, a structure for adaptive anomaly-based intrusion detection system is proposed. Thus, the adaptive algorithm has different mother wavelet coefficients and different number of layers such as the input layer, convolution layer and the FCL because these layers adapt based on the stationary size of input dataset and their weights change based on the adaptive mother wavelet coefficients. Therefore, for different dataset with different characteristics, the shape of the output of mother wavelet and the architecture of detecting DDoS attacks and the output of the VFDTv2 should be changed adaptively. As a result, another dataset has not been used in this thesis because the results vary from one dataset to another dataset.

## 6.5 The Fifth Proposed Algorithm Simulation and Results

The input signal of the extraction techniques was the ATS, and a stationary frame size of data samples was required to obtain a realistic result. The output length of all techniques was one except for the DWT technique which has an output length of 154. The output of this preprocessing step was horizontally concatenated with the data. Moreover, the DWT technique was used twice; the first time the input was the ATS and the second time the input was the VFD. In last cases, the output of DWT in both cases was concatenated with the data vertically.

### 6.5.1 Creating an Environment for the Defined Problem

To solve a DRL problem, before creating the algorithm, an environment for the IDS should be designed since the IDSA lives in the environment.

To design the environment, three entities should be included as follows: observation that is an array, a reward that is + 1 or – 1, and a done entity that is false or true. First, each new observation of the DDoS ITD from the CAIDA dataset was selected. Second, a reward entity was given at every step. The ending state of a normal state had a reward of +1 and the reward for the ending state of an anomalous state was –1. The reward of –1 was given to detect anomalous data to reduce false positive. The goal was to detect DDoS attacks as much as possible by observing the arriving packets as quickly as possible and reducing the false positive. Third, the done entity occurred when an episode (an attempt that consists of *n* consecutive steps) was over. The done entity also became active when a DDoS attack was not detected by the IDSA. The done entity was triggered in two scenarios. In the first scenario, when a DDoS attack was not detected using the IDSA, the repository was flooded with packets, a crash was triggered, and the done entity became active. In the second scenario, the observation of a window size of 8192 packets (observation) was

successfully traced by the IDSA. When a packet was labeled as normal or anomalous, the packet was stored in a repository within 0.1 *milliseconds* (ms) for further processing in the real environment. If there were more than 40 packets within 0.1 ms, the repository was flagged as overloaded, which caused the IDSA to crash. When this happened, the IDSA was punished with a reward of – 10, and the next training loop started. In other word, to realize the idea that an environment awareness was right or wrong, the concept of the overflow of the repository was considered. When a packet is considered as normal, the reward of +1 was considered. When a packet is considered as anomalous, the reward of – 1 was considered. In both cases, the packet wass put in a repository. An overflow happened when a packet was considered as a normal, and the number of packets in the repository was more than 40 packets within 0.1 ms. In this situation, the reward of – 10 is considered, and the episode is terminated. However, when a packet is considered anomalous with – 1 punishment, the overflow does not happen. The states of the IDSA are shown in Fig. 6.8.

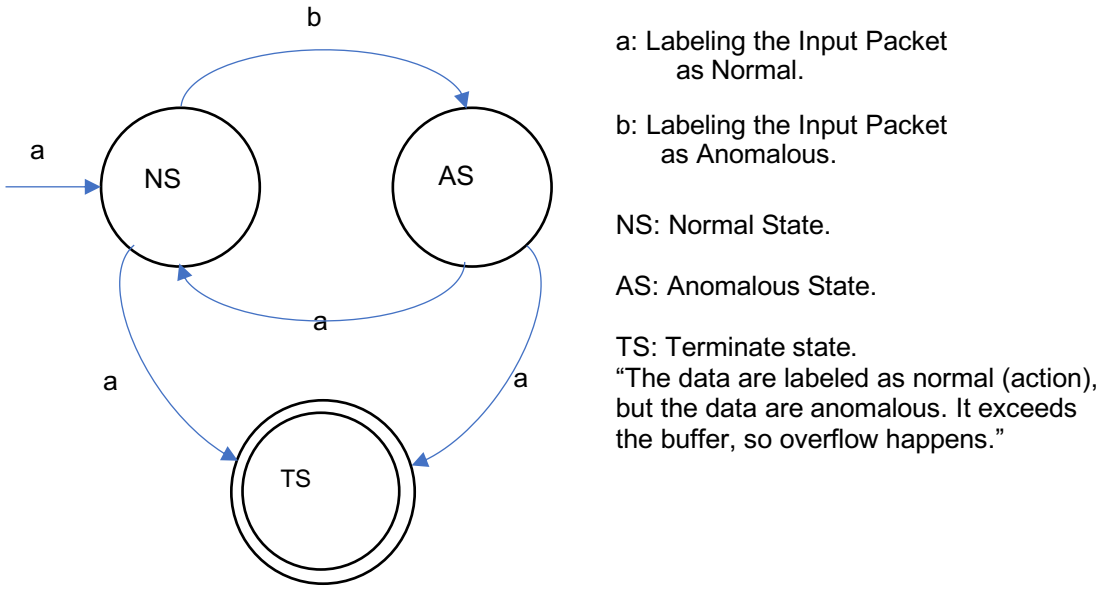



**Fig. 6.8:** The states that IDSA considered.

### 6.5.2 The Implementation of Policy and Selection of Actions

The output of the PCNN was the policy of the IDSA. The PCNN was created as a neural network policy because it was more efficient than randomly selecting a policy. This policy was used to detect the DDoS attacks using the action that received the maximum number of rewards (the minimum number of punishments) in a stationary frame size of 8192 packets from the CAIDA dataset.

In each frame size of the DDoS ITD environment for each packet there was one action. Each action was labeled as either normal or anomalous. To show the two types of actions, one neuron that had one output was sufficient. As a result, the PCNN only needed one output neuron in its output layer. The output of the neuron with a probability $P_1$ indicated that the input observation data was normal. The probability that the input observation was anomalous was obtained using $P_2=\{1-P_1\}$. For example, if its output was 0.7, the IDSA chose action 0 with 70% probability and action 1 with 30% probability. Then, according to the softmax activation function the probabilities are mapped to actions. After evaluating actions, an action can be selected according to $\epsilon$-greedy algorithm. In this step the input observation data was labeled as either normal or anomalous.

### 6.5.3 The DRL Algorithm Training Parameters

For training the PCNN using the policy gradient reinforcement learning algorithm, the following steps were considered [Tabo19a]. First, the IDSA generated an episode and tracked states, actions, and rewards in the IDSA memory. At the end of each episode, the IDSA went back through these states, actions and rewards to compute the discounted future return at each time step. Second, the actions that the IDSA took were used as labels of the input raw data. Third, the weights

of the PCNN were updated using reinforcement learning in Eq. (4.17). Fourth, this learning algorithm was repeated until the weights of the PCNN were converged to the optimum values. When the optimum values of the PCNN were obtained, the IDSA learned to select the best action. As the result of selecting the best actions in an episode, the IDSA did not crash because of overflowing its buffer through DDoS attacks. Therefore, the PCNN was trained by adjusting its weights to know what the best action in each step is.

The stationary-frame size was calculated, and outputs of previous steps were normalized within the stationary-frame size. To meet the convergence of error to zero at early iterations, the learning rate, α, was set to 0.001. Using 50% of the dataset, the PCNN was trained to extract the features automatically and obtain coefficients from the dataset. Next, using about 50% of the remaining dataset, the PCNN was tested with the obtained coefficients of the PCNN training step.

The PCNN parameters were trained to obtain the optimum weights. In the PCNN structure, the convolutional layer, which consisted of convolutional sublayers, had learnable weights. The PCNN training was stopped when the system was crashed or a window size 8192 is completed. The stationary-frame size was calculated, and outputs of previous steps were normalized within the stationary-frame size. The parameters used to train the PCNN and test the PGDRL according to the CAIDA dataset are shown in Table 6.13.

**TABLE 6.13:** The parameters to train the PCNN and PGDRL.

| Parameters | Value |
|---|---|
| The Stationary-frame size | 8192 |
| The activation function in the convolutional sublayers and activation function in the FCL | Sigmoid function |
| Zero padding in the convolutional sublayers | None |
| Learning rate | $\alpha$ = 0.0001 |
| The exploration rate | $\epsilon$ = 0.001 |
| The discount factor of delay | $\gamma$ = 0.99 |
| Epoch | 1500 |
| The number of nodes in the input layer | 8 |
| The number of nodes in the input layer of FCL | 164 |
| The number of nodes in the hidden layers of FCL | 165 |
| Dataset size for training and testing | 8192 |
| The detection rate (Accuracy) (%) | 93% |

### 6.5.4 Results and Discussion

One of the most important innovative features of the proposed fifth algorithm was the creation of a polyscale neural network called PCNN. PCNN was called a novel method because this neural network structure was first proposed. In this structure, the concept of the VFDT algorithm is embedded in a CNN. For example, the number of convolutional sublayers and nodes in the convolutional sublayers follows the concept of the VFDT algorithm. In addition, the PCNN output that approximates policy was important to detect normal and anomalous behavior because the stronger the policy approximation, the higher the detection rate.

Moreover, the policy gradient DRL based on the PCNN detected the DDoS attack with 93% accuracy. The proposed method of detecting DDoS attacks allowed detection of anomalous behaviors in various environments. In addition, the policy gradient DRL based on the PCNN used unlabeled data, so it was a realistic IDS and suitable for real-world applications.

In this study, a comparison can be made between the proposed the PGDRL and other deep Q-learning algorithms. The PGDRL directly calculated the optimal policy. Thus, the PGDRL was much faster than the deep Q-learning algorithms. As a result, the proposed PGDRL detected DDoS attacks faster than the deep Q-learning algorithms. However, the detection rate of the deep Q-learning algorithms was higher than the proposed PGDRL detection rate. To overcome this problem, the actor-critic algorithm could be used to increase the detection rate.

## 6.6 Comparison, Contrast and Discussion of the Proposed Anomaly Detection Algorithms

In this section, the sensitivity to duration of attack, detection accuracy, input, dataset, characteristics, methodology, testing phase speed, testing phase detection rate of the five proposed anomaly detection algorithms are discussed. These parameters are shown in Table 6.14.

**TABLE 6.14:** Summary of comparison and contrast regarding the proposed anomaly detection algorithms.

| | First IDS | Second IDS | Third IDS | Fourth IDS | Fifth IDS |
|---|---|---|---|---|---|
| Sensitivity to duration of attack | Not sensitive to short duration of the attack | Sensitive to any duration of the attack | Sensitive to any duration of the attack | Sensitive to any duration of the attack | Sensitive to any duration of the attack |
| Input | Labeled data | Labeled data | Labeled data | Labeled data | Unlabeled data |
| Dataset | Power consumption (Synthesize data) | DDoS attack from CAIDA (Real data) | DDoS attack from CAIDA (Real data) | DDoS attack from CAIDA (Real data) | DDoS attack from CAIDA (Real data) |
| Characteristics | Static | Static | Static | Adaptive | Adaptive |
| Methodology | Pre-processing:<br>- DB4<br>- VFD<br><br>Processing:<br>ANN | Pre-processing:<br>- DB4<br><br>Processing:<br>CNN | Pre-processing:<br>- DB4<br>- VFDTv2<br><br>Processing:<br>CNN<br><br>Post-processing:<br>SVM | Pre-processing:<br>- DB4<br>- VFDTv2<br>- SFD<br>- ZC<br>- TC<br>- statistics of the dataset<br>- auto-correlation<br><br>Processing:<br>Adaptive CNN | Pre-processing:<br>- DB4<br>- VFDTv2<br>- ZC<br>-TC<br>- PCA<br><br><br>Processing:<br>Adaptive PGDRL |
| Testing phase speed | Fast ( millisecond) | Fast ( millisecond) | Fast ( millisecond) | Fast ( millisecond) | Fast ( millisecond) |
| Testing phase detection rate | 51% ~90% | 80.75% | 87.35% | 95% | 93% |

The parameters in Table 6.14 are mentioned as follows. The first IDS was not sensitive to short durations of the attack. The second IDS, the third IDS, the fourth IDS and the fifth IDS were sensitive to any duration of attacks. The first IDS, the second IDS, the third IDS and the fourth

IDS used labeled data for training. The fifth IDS was used unlabeled data for training. The first IDS used synthesize dataset power consumption. The second IDS, the third IDS and the fourth IDS and the fifth IDS used real dataset Internet traffic contains DDoS attacks. The first IDS, the second IDS and the third IDS were static. The fourth IDS and the fifth IDS were adaptive.

The first IDS used DB4 and VFD as extracting features, and the first IDS used the ANN machine learning algorithm to detect the power consumption attacks. The second IDS used DB4 as extracting features, and the second IDS used the CNN machine learning algorithm to detect the DDoS attacks. The third IDS included a pre-processing step to extract distinguishing features using DB4 and novel VFDTv2. The processing step was used the CNN machine learning algorithm to detect the DDoS attacks. Lastly, the post-processing was step used the SVM to improve the detection of the DDoS attacks. The fourth IDS included a pre-processing step to extract distinguishing features using the novel mother wavelet, VFD, SFD, statistics of the dataset, autocorrelation, DB4, ZC and TC. The processing step was used a novel CNN architecture based on Mallat algorithm as a machine learning algorithm with the novel cost function to detect the DDoS attacks. The fifth IDS used a DRL a novel CNN architecture based on the polyscale analysis to detect the DDoS attacks, and the fifth IDS did not use labels for training. This method included a pre-processing step to extract distinguishing features using DWT DB4, VFD, ZC, TC and PCA.

All the proposed IDS methods testing phase speed were about milliseconds, so they were fast. Therefore, they could apply in real world. The first IDS detection rate was 51% in the worst-case scenario. The second IDS detection rate was 80.75%. The third IDS detection rate was 87.35%. The fourth IDS detection rate was 95%. The fifth IDS detection rate was 93%. The parameters in Table 6.14 are described in detail below.

Detection rates

The first algorithm for detecting simulated power consumption attacks was introduced. The design of this algorithm was based on supervised learning. However, real power consumption dataset was not available at the time for confidential and security reasons. Moreover, this algorithm produced an unsatisfactory worst-case scenario anomaly detection rate. As a result, the second anomaly detection algorithm for detecting DDoS attacks was introduced. To increase the detection rate, a CNN was used instead of an ANN. The main procedure of the second anomaly detection algorithm for detecting DDoS attacks consisted of two steps: the pre-processing step and the processing step. The second anomaly detection algorithm improved the detection rate by approximately 30% in comparison to the ANN used in the first anomaly detection algorithm.

The third method with a detection rate of 87.35% produce a higher detection rate than the second method with a detection rate of 80.75%. To analyze this improvement, the second method used WT as the pre-processing step and CNN as the processing step. However, the third method used WT and VFDT as the pre-processing step, CNN as the processing step and SVM as the post-processing step. Therefore, the improvement was achieved by adding VFDT algorithm and SVM.

With considering the third proposed anomaly detection algorithm simulation results that were shown in Table 6.6, Table 6.8, Table 6.10, the following facts could be observed. If only the processing step is considered, the detection rate was 56.1%, as shown in Table 6.6. If the pre-processing step and the processing step were considered, the detection rate of 81.73% was achieved as shown in Table 6.8. Therefore, about 25% improvement was achieved. If the pre-processing step, the processing step and the post-processing step were considered, the detection rate of 87.35% was achieved as shown in Table 6.10. Therefore, about 5% improvement was achieved. As a result, the pre-processing step contributed significantly to the detection rate compared to the post-

processing step. It could be concluded that as much as hidden features were extracted from the input data, the supervised machine learning algorithms and the DRL algorithms were more capable of detecting DDoS attacks. Therefore, a data pre-processing step was designed to improve the detection rate for the fourth IDS algorithm. Thus, the adaptive mother wavelet was designed, so that the features extracted from the input data in the CNN preprocessing step increased the detection rate. However, this algorithm still had a low detection, so the third algorithm was designed to improve the detection rate further.

The third proposed algorithm consisted of four steps: preparing data step, the pre-processing step, the data processing step, and the post-processing step. and improved the detection rate by approximately 7% in comparison with the second algorithm. However, the DDoS attacks detection algorithm should have a higher detection rate, so the fourth algorithm was designed to further improve detection rate. Moreover, the pre-processing step of the third proposed algorithm increased the detection rate by about 24.6% and the post-processing step increased the detection rate by about 5.6%. As a result, the pre-processing step significantly contributed to increase the detection rate compared to the post-processing step. Therefore, a data pre-processing step was added to improve the detection rate for the fourth algorithm.

The fourth algorithm was designed based on what was discovered in the previous algorithms. First, a mother wavelet that adapted to its input signal was created. The GNN was used to create the adaptive mother wavelet. An ANN and a NSGAII algorithm were used to design the GNN. A multi-objective optimizer was used because several constraints needed to be satisfied. Moreover, the NSGAII algorithm was an improved version of genetic algorithms, and the NSGAII algorithm was faster and had a good performance although using several constraints. Second, a cost function was created for adapting to the input, so a faster convergence to the optimal

coefficients of the neural network could be achieved. As a result, the cost function was sensitive enough to find differences between the neural network output and the desired output. Third, an adaptive-wavelet convolutional neural network structure was created for increasing the detection rate of the DDoS attacks. Moreover, the WT coefficients were extracted using the created adaptive mother wavelet, and the extracted WT coefficients was part of pre-processing step that that embedded on the CNN. In addition, the extracted WT coefficients were used to improve the detection of DDoS attacks, and the created cost function was used for measuring how well the adaptive-wavelet convolutional neural network estimated its output. Additionally, seven feature extractor methods were used. The fourth method had detection rate of 95% which was much higher than the detection rate of the third algorithm. As a result, the fourth anomaly detection algorithm improved the detection rate by approximately 8% compared to the third anomaly detection algorithm.

The first four proposed anomaly detection algorithms were based on supervised learning, so they needed labels for training that were unsuitable for the real world. Therefore, the fifth anomaly detection algorithm was proposed to detect DDoS attacks. The main procedure of the fifth algorithm consisted of three steps: preparing data, the pre-processing step, and the processing step. The prediction or estimation algorithm was the MC algorithm for training the policy approximation in the PGDRL because it was simple to implement. The MC algorithm was an offline learning method for training the PCNN which meant that the agent had to wait until the end of an episode. However, in the testing phase, there is no need to train the policy approximation. Therefore, training was offline, and testing was online. The TD learning algorithm or the SARSA learning algorithm could be applied for training the PCNN to have an online learning method.

Moreover, the feature extraction methods used for the last two IDSs were important for extracting hidden features to increase the detection rate.

Datasets and Application

The dataset used for the first IDS algorithm was simulated because real cyber domain smart-grid infrastructure data was not accessible at that time due to confidentiality and security reasons. Moreover, the simulated power consumption attacks had two problems.

First, if a hacker stole power within 2 minutes and 25 seconds, the simulated power consumption attacks dataset was not sampled within 2 minutes and 25 seconds interval. As a result, the first IDS algorithm could not detect any power consumption within 2 minutes and 25 seconds. In this situation, the detection rate is zero percent. To overcome this problem, the minimum sampling rate for converting the continuous signal (with a finite bandwidth) to a discrete sequence could be obtained using the Nyquist frequency sampling. However, the continuous power consumption signal was a broadband signal, so the high level of cutoff frequency was needed as shown in the following Fig. 6. 9.

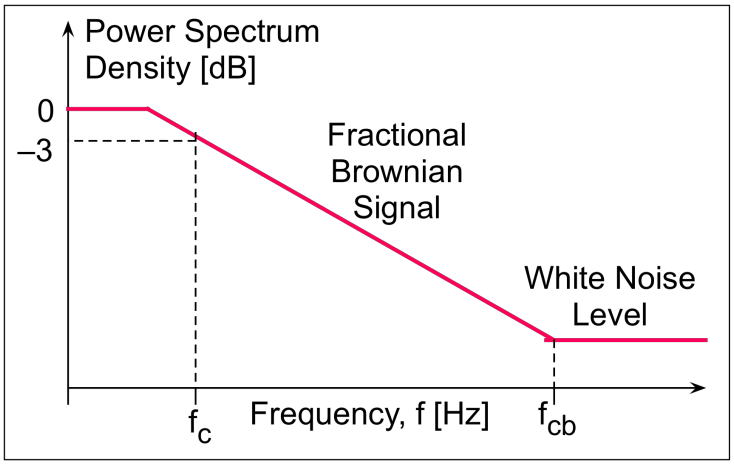



**Fig. 6.9:** The cutoff frequency concept [Kins15].

For the bandwidth of the broadband signal in Fig. 6.9, the following Eq. (6.13) could be used.

$$F_{Sampling\ rate} = f_{cb} \tag{6.13}$$

Therefore, the sampling rate shown in Eq. (6.13) was sufficient for sampling, so distortion problem could not be happened. As a result, the sampling rate in Eq. (6.13) should be used to sample the continuous power consumption signal.

The first proposed anomaly detection algorithm detected anomalous power consumption attacks on smart grid infrastructure in this PhD thesis, but the second to fifth proposed algorithms detected DDoS attacks on the DNS servers in cyber computer systems.

The most important characteristic of the DDoS ITD dataset was that it changed rapidly over time, and the dataset was not static. Therefore, the dataset should be stationary, and features were extracted within stationary frame size of the dataset. Feature extraction helped the machine learning algorithm to improve detection rate. In addition, the DDoS ITD dataset was not stationary, and it was stochastic. To make the process stationary (at least weak stationary), the statistics of the signal should be varying within two boundaries. However, the DDoS ITD dataset was stationary within a window size 8192 that was the frame size for this signal. In general, a dataset had weak stationarity when its first two moments fall within two ranges, a 95% confidence interval [Kins20]. The DDoS ITD had weak stationarity because the minimum window size of 8192 caused the trajectory of the means to fall within a range of 0.000782 to 0.003041, and the variance trajectory to fall within a range of 1.64E–06 to 8.21E–06. The skewness trajectory fell within a range of 1.076 to 2.94, and the kurtosis trajectory was 1.8. To make the dataset stationary, different parts of dataset were chosen. For example, 64 samples, 128 samples, 1024 samples, 2048 samples, 4096 samples and 8192 samples were chosen, so the dataset was divided by the above length of samples.

When the frame size was selected to be 8192 samples, the dataset was weak stationary. As a result, the first two moments of the dataset were within two ranges.

The dataset for detecting regular DDoS attacks on DNS servers was similar to DDoS attacks in smart grid infrastructure. Moreover, the proposed IDS algorithms were general algorithms in this research, so the proposed IDS algorithms could be applied for the specific security systems such as power system security and hydraulic security. This thesis addressed DDoS attacks on DNS servers, but the proposed IDS algorithms could be applied for security in other applications as well.

In general, machine learning algorithms did not have good performance when their input was not normalized. Moreover, normalization increased the convergence of weights and biases to the optimal results. As a result, normalization for the input of the machine learning algorithm was used.

If there was enough data, the dataset could be divided into 50% training and 50% testing. Otherwise, the best results were obtained using 80% of the data for training and 20% of the data for testing. In this research, the first scheme used the simulation dataset because there was not enough data. As a result, 80% of the data were used for training, and 20% of the data were used for testing regarding the first scheme. The second, third, fourth, and fifth schemes used the real dataset, so there was enough data. As a result, 50% of the data were used for training, and 50% of the data were used for testing.

Dataset analysis methods

The mono-scale analysis, multiscale analysis and polyscale analysis methods were used for two reasons in this PhD research: the feature extractor from data sets and analyzing the data sets. For analyzing the DDoS ITD data sets, the mono-scale analysis such as statistical features of the

dataset were used in this PhD research. For example, kurtosis was important to analyze the DDoS ITD because positive values of kurtosis indicate heavy tail distributions. Moreover, the heavy tail distribution indicated the dataset was a fractal signal. When a signal was fractal, a multiscale analysis and especially a polyscale analysis should be used to extract hidden features from the fractal signal.

Moreover, this complex system combined two or more components that was impossible to separate the influence of one from other components [Kins10]. In this research, the DDoS ITD at least has two components: normal data and DDoS attack data. These two components could not be separatable, and the components could not be considered independent. In other word, the decomposition of the DDoS ITD into two independent components could destroy the whole dataset. The common property of such a complex dataset was a power law distribution [Kins10]. According to the power law, the variance of its amplitude increments over a time increment had a relation with that time increment [Kins15]. As a result, the DDoS ITD was a complex dataset, and anomaly detection was a hard machine learning problem [Hard22]. As a result, a proper method to detect new patterns of DDoS attacks in this changing environment with complexity is an adaptive machine learning algorithm.

The reason for the importance of multiscale analysis and polyscale analysis was processing information only on a single scale (monoscale), which was inefficient because complex dataset processing information had different information at different scales. Moreover, using inter-scale communication between the processes at each scale was important, so to have a meaningful result, the communication between the scales was important and should be considered [Kins10]. This concept was embedded in the polyscale analysis concept, and one of the most important polyscale analyses that are based on power law distribution was the VFDT algorithm. The DDoS IDT was

not a stationarity process, so it changed over time. This characteristic of the DDoS IDT could be measured using the VFDT algorithm, and the output of this algorithm was a fingerprint for this time series. In addition, this fingerprint changed from time to time, and this change has affected the complexity of time series and complexity measures. As a result, the process was multifactorial, and a polyscale analysis was needed to analyze the DDoS IDT. The VFDT algorithm was a polyscale analysis that measured the complexity of the DDoS IDT. Therefore, the VFDT algorithm was used to design adaptive algorithms in this research because the complexity of DDoS IDT changed over time. Therefore, it was important to consider the complexity of time series for adaptability of the proposed structures (to perform adaptations). The fingerprint of non-stationarity process depended on how well a polyscale analysis measured the complexity of the process. Adapting to the changing environment was one of the goals in this research. Therefore, the adaptive cost function, the adaptive mother wavelet, and the adaptive neural network are design in this research.

To find the complexity of the dataset, the fractal dimension was used. In this research, the VFDT algorithm and the novel VFDTv2 algorithm as a polyscale fractal analysis were used to find the complexity of input data to separate a normal from the anomalous cluster.

Training

For training the proposed IDS machine learning algorithms, an epoch equal to 1500 was considered. The error of cost function or loss function could be shown by a plot with considering the number of epochs. The error was the difference between the neural network output and the desire output. When the error was drawn in each loop, the error is dropped well around 1500th cycle. In other word, when visually the drop of error was reduced during training, the epoch size was about 1500. Therefore, the epoch size was set at 1500 for training the proposed neural

networks. In addition, the cross-validation technique was not used to analyze the training validation.

Time series management

In this research, the machine learning algorithms managed time series data well. Raw adjacent time series with the length of eight packets was used in the machine learning algorithms as IDSs. In addition, the pre-processing step used algorithms such as VFDT algorithm, which reduced the dimension of data series to detect traffic deviations. These issues are described as follows. First, a combination of signal processing methods, polyscale analysis method and network anomaly detection methods was used to reduce the dimension of data series to detect traffic deviations. In this study, signal processing methods and the VFDT algorithm were used for dimension reduction and to extract hidden features from data streams. Second, the concept of covering all points of data to calculate the VFDv2 was mentioned in Fig. 3.6 (b). In addition, the figure regarding time series with the length of 8192 packets was shown in Fig. 3.7. The time series data is stationary within this frame size of 8192, so the hidden features are extracted regarding this frame size of dataset. The features are extracted using different vel size using the polyscale analysis concept. The proposed IDSs consider this length of time series as the input of VFDT algorithm that was a pre-processing algorithm for the IDSs. The output dimension of the VFDT algorithm was just one point (one value). This point as the output of the VFDT algorithm was the fractal dimension of the time series, and the fractal dimension was passed to machine learning algorithm such as ANN, CNN and DRL. The output of the VFDT algorithm was very sensitive to any changes in the input time series, so it helped to detect traffic deviations. Third, the raw time series of the dataset with length of eight packets were passed to machine learning algorithm such as ANN, CNN, DRL. Forth, one of the most important features of Internet traffic was rapid change.

Therefore, the above feature extraction methods were used to extract hidden features, and adaptive tools for extracting more hidden features were designed as follows. The adaptive mother wavelet was designed to extract hidden features from the time series. The adaptive CNN architecture that adapts to input time series was designed. The adaptive cost function that adapts to input time series was designed to converge faster. Therefore, the adaptive CNN learned based on input characteristics. Moreover, the number of convolutional sublayers and the number of nodes in each convolutional sublayer were adapted based on the input time series characteristics. In addition, the input time series with a length of 8192 samples was a weak stationary. Therefore, during this frame size, the data characteristics did not change, and data statistics changed within a boundary. Moreover, different algorithms were used as the pre-processing step to help and increase the detection rate based on extracted features from the input time series.

Comparing with other algorithms

The first method considers a different type of attack that is a power consumption attack, so this method did not compare with other anomaly detection method. However, the second method to the fifth method consider DDoS attacks, so they can be compared. In addition, they have totally different algorithms, so comparing is challenging. Therefore, the same chunk of dataset is used for training the IDS algorithms, and the same chunk of dataset is used for testing of IDS algorithms. As a result, the IDS algorithms can be compared.

The proposed adaptive mother wavelet and the adaptive architecture of detecting DDoS attacks were introduced for the first time. Therefore, it is not possible to compare the proposed methods with other methods. These creativities were novel, so they were in the early stage of science. Therefore, other researchers should improve them to obtain higher detection rate. The experimental results demonstrate that it was critical to get the right weighted cost function and the

best mother wavelet coefficients were successfully found. This research may serve as a technical foundation for future networking security. To define a proposed method as good or not, the same piece of data was used for training for the proposed methods, and the same piece of data were used for testing. Therefore, the progress of each method can be evaluated compared to the previous method. As a result, the percentage of progress of each method could be found, so the proposed anomaly detection algorithms could be compared with each other. When the detection rate of a proposed method is higher than the previous proposed method, the result could be considered a good result.

One of the proposed methods in this research was a policy gradient-based algorithm, but most of the proposed algorithms were based on the deep Q-learning algorithm. Therefore, there was no baseline method for PGDRL, so it is not appropriate to compare the detection rate between the two methods. However, in this study, a comparison could be made between the proposed PGDRL and other deep Q-learning algorithms. The PGDRL directly calculated the optimal policy. Thus, the PGDRL was much faster than the deep Q-learning algorithms introduced in [Serv07][SeKu08] and [XuXi05]. As a result, the proposed PGDRL detected DDoS attacks faster than the deep Q-learning algorithms. However, the detection rate [Serv07][SeKu08] and [XuXi05] were more than 99%, while the proposed PGDRL detection rate was 93%. To overcome this problem, the actor-critic algorithm could be used to increase the detection rate.

The adaptive architectures

Different mother wavelets had different coefficients and different compact supports. The simplest mother wavelet was the Haar mother wavelet with compact support equal to two. In this research, an adaptive mother wavelet with compact support equal to two was designed as the simplest mother wavelet that was shown in Fig. 6.7. Then, the designed mother wavelet was

compared with the Haar mother wavelet as the gold standard. The adaptive mother wavelet improved the detection rate with 0.3 % in compare with Haar mother.

Moreover, to have a lower running time and a faster convergence to the optimal coefficients of the neural network, a weighted cost function KL-divergence was created. It was important to use measures that are reflected in a cost function. Therefore, the weight cost function was designed based on the VFDT algorithm and the KL-divergence cost function in this research. In addition, the KL-divergence was chosen as the basis for designing the weighted cost function because the highest sensitivity was achieved when the sensitivity analysis technique was used. In other word, when the input of the KL-divergence cost function was changed, the highest sensitivity could be obtained in the output of the KL-divergence cost function. Therefore, the adaptive mother wavelet was designed based on the analysis. The VFD algorithm of the input data was used as the coefficients of the KL-divergence cost function, so the cost function adapted to the input data.

The choice of mother wavelet was important for extracting features of an input data. When a mother wavelet that was not match with the input signal selected, the highest detection rate could not be achieved. One of the most challenging issues in designing an adaptive mother wavelet was to consider several constraints. The stochastic gradient descent optimization considered just one objective (cost function). To create an adaptive mother wavelet to detect DDoS attacks, four constraints on creating a mother wavelet and one constraint on detecting DDoS attacks must be met. To consider these five constraints, a multi-objective optimization was considered. One of the best multi-objective optimizations, an improved version of genetic algorithms, was the NSGAII algorithm. Therefore, an improved version of multi-objective genetic algorithm was used to consider all constraints with a fast convergence. Therefore, the GNN structure was proposed in this research. The GNN structure consists of a simple two-layer neural network and an improved

version of a genetic algorithm such as NSGAII that considers several objectives (constraints) simultaneously.

In addition, the adaptive mother wavelet and the convolutional sublayers in the architecture of a dual coefficient adaptive-wavelet CNN in Fig. 4.7 were used to extract the WT coefficients. This structure used weight sharing concept from the convolution layer in the CNN and the Mallat algorithm to decompose the input signal into approximation and detail coefficients. The created adaptive mother wavelet was used for the first convolutional sublayer. Then, the second convolutional sublayer computed the dilation version of the created adaptive mother wavelet. The dilatation version was obtained using weight sharing and architecture of the adaptive-wavelet CNN. Therefore, the output of the convolutional sublayers in the convolution layer in the adaptive-wavelet CNN were the wavelet transform coefficients of the input signal. The wavelet transform coefficients were shown from $a_1^{(1)}$ to $a_8^{(1)}$ in Fig. 4.7.

For example, if Haar mother wavelet coefficients [+1,–1] were used for the values of the shared weights of the convolutional sublayers in the adaptive-wavelet CNN, the wavelet transform coefficients of the input signal using Haar mother wavelet were extracted. There were two equations Eq. (3.11) and Eq. (3.12) that were described in detail below. First, the mother wavelet coefficients could be shown using Eq. (3.11), and the adaptive mother wavelet using the Genetic Neural Network (GNN) was designed in the PhD research. Second, the calculation of WT coefficients regarding the input data could be extracted using Eq. (3.12), and the calculation of WT coefficients (the extracted coefficients) was performed using the convolutional sublayers of the adaptive-wavelet CNN.

In addition, the number of the convolutional sublayers was adaptive, and the number of the convolutional sublayers follows Eq. (4.9). Moreover, the number of nodes in each sublayer and the connection between nodes follow Mallat algorithm.

One way to select the best compact support (length) of an adaptive mother wavelet was to create a series of adaptive mother wavelets with different compact supports (lengths) in a For Loop operator. One by one the detection rate of the DDoS attacks should be calculated for a same part of dataset. The highest detection rate obtained from the mother wavelets should be selected as the best detection rate, so the corresponding mother wavelet for the highest detection rate was the optimal mother wavelet with the best compact support for the specific dataset.

The PCNN architecture was adapted with its input features. The number of nodes in the convolutional layer was adapted with its input.

Generality of the proposed algorithms

The proposed IDS algorithms are general algorithms in this research, so the proposed IDS algorithms can be applied for the specific security systems such as power system security and hydraulic security. This thesis is addressed DDoS attacks on DNS servers, but the proposed IDS algorithms can be applied for security in other applications as well. Moreover, this research is about a classification application that classify normal data from anomalous data. Other classification applications can use the proposed algorithms in this research.

The proposed PGDRL is unsupervised and thus could be applicable to a wide range of scenarios.

## 6.7 Summary of Chapter 6

Internet traffic, the input data in this research, was a dynamically changing environment. Therefore, adaptive algorithms were needed to improve the detection rate in the stochastic Internet environment. One of the most important innovative features of this research was the implementation of the adaptive machine learning algorithms and the adaptive signal processing method. The simulation and the results of each anomaly detection were introduced. The best DDoS attacks anomaly detection method had a 95% detection rate regarding the supervise learning. In addition, the best DDoS attacks anomaly detection method had a 93% detection rate regarding the learning without tag using DRL. The anomaly detection algorithms were introduced and created in this research were general algorithms, but the algorithms were used to detect the DDoS attacks as a specific application.

# Chapter 7

# CONCLUSIONS AND FUTURE WORK

## 7.1 Conclusions

In this thesis, five anomaly detection methods were proposed to detect DDoS attacks. This thesis aimed to achieve a higher accuracy detection rate. Therefore, by using distinguishing features of the data carrying anomalous behaviors, the sensitivity of the machine learning algorithms to classify the input data was increased. A higher accuracy detection rate was achieved by adding DWT using the db4 and the designed mother wavelet, VFDTv2, and other feature extracting methods as a pre-processing step. The best proposed anomaly detection method architecture using the supervised learning method was called the adaptive-wavelet CNN. The adaptive-wavelet CNN detected the DDoS attack with 95% accuracy. The best proposed anomaly detection architecture using unlabeled data was called the policy gradient DRL based on the PCNN that detected the DDoS attack with 93% accuracy. The proposed method of detecting DDoS attacks allowed the detection of anomalous behaviors in real-time and adapted to various environments.

In addition, the policy gradient DRL based on the PCNN used unlabeled data, so it was a realistic IDS that was suitable for real-world applications.

This section answers the research questions in Sec. 1.7 as follows:

The first research question has the following answer. The reliability and availability of Internet services and network operations in secure environments are significant to obtain trustable Internet services. Identifying anomalies in Internet services is essential because cyber-attacks cause unavailability of the Internet services, deceit, data corruption shutdown and slowdown of Internet services and industrial systems. DDoS attacks are one of the most common types of cyber-attack. DDoS attacks delay or block the communication of Internet services leading to unavailability of Internet services. To overcome DDoS attacks, an adaptive IDS should be designed to identify DDoS attacks, and this IDS should be involving learning similar to the human learning.

The second question has the following answer. To design a real-time learning IDS that is similar to a cybersecurity expert, new attack patterns and behaviors must be detected in a frequently changing environment. Therefore, a machine learning method for designing the IDS to detect new attack patterns and behaviors in frequently changing environments must be used. In this research, through adopting the human learning way of experience, an extensive version of the DRL learning from its reward-based actions was designed and created. In addition, intelligent and adaptive algorithms were used to extract distinguishing features for the designed IDS to increase the DDoS attack detection. In this research, an extended version of the VFD algorithm, VFDTv2, and an adaptive mother using the GNN method were designed and created to extract distinguishable features. These feature extraction methods have represented the input signal in more detail, so the DDoS attack detection increased.

The third question has the following answer. Identifying normal burst-data behaviors and abnormal burst-data behaviors caused by DDoS attacks is challenging because both classes of network traffic have similar intrinsic characteristics. They are both stochastic, non-stationary, non-differentiable, broadband, dynamic, self-affine fractal, and correlated with long-range dependence signals. Therefore, to differentiate the two, distinguishing features must be extracted and new attack patterns and behaviors must be detected in a frequently changing environment. Therefore, a machine learning method for detecting new attack patterns and behaviors in frequently changing environments must be used. A deep learning algorithm is a good candidate for training the machine learning method and distinguishing normal from anomalous behaviors in such an environment. In addition, signal processing tools such as the DWT can represent input signal details and the trends been passed to the IDSs. The following statistical characteristics and patterns are embedded in the DDoS ITD, so the proposed IDSs were trained to extract them: The normal Internet traffic data with the bursty behavior and the DDoS ITD follow levy distributions with different parameters. The heavy tail in levy distributions is due to the presence of self-similarity in the delay of the packets. The difference in the parameters is due to the speed of traffic increases and decreases. Moreover, the normal Internet traffic data with bursty behavior and the DDoS ITD behave differently depending on access intents, distribution of source IP address, and special events such as breaking news. Finally, the fractal dimension of normal Internet traffic data with bursty behavior is different from the DDoS ITD.

The fourth question has the following answer. First, this thesis introduced the designed adaptive-mother wavelet to extract features. The designed mother wavelet was adapted to the input for feature extraction, so higher detection of DDoS attacks could be achieved. Second, the adaptive-wavelet CNN structure was introduced to detect DDoS attacks. This structure was called

adaptive because the number of nodes in the input layer, convolutional layer and the FCL are adaptive to the number of features in the input and the number of the mother wavelet coefficients. Third, the weighted cost function was introduced to detect DDoS attacks. Adding weights to the KL-divergence cost function could magnify the anomalous data, so this class could be bolder. The extracted features from input data were used as the weights to adapt the proposed cost function to the input data. Therefore, a faster convergence of the optimal coefficients regarding the adaptive-wavelet CNN was obtained. Fourth, the proposed PCNN structure was introduced as the policy approximation for the DRL. The PCNN was adapted to the number of input coefficients to increase the flexibility of the structure and the detection rate.

The fifth question has the following answer. The DDoS ITD relies on long-range dependence. Therefore, there is a correlation among the time series data, which can be used to extract distinguishable features and patterns in the Internet traffic. These features are fragile, and sensitive tools like the DWT method are needed to extract these hidden features. The DWT map a time series signal into the two-dimensional time-scale domain to achieve greater signal details and trends. This representation can be used to characterize transient events and can extract features to detect anomalous behaviors in the Internet traffic data with DDoS attacks. Therefore, using the DWT to extract the highest hidden features is important to increase the detection rate.

The most challenging step in the DWT is to select its mother wavelet. To have an efficient analysis of feature extraction using DWT, a relevant mother wavelet should be considered for the given application. To choose a relevant mother wavelet, the concept of vanishing moments is used. The vanishing moment of a mother wavelet shows which polynomial has the highest correlation (similarity) with the data of the given application. The order of vanishing moments can represent the complexity of the signal. The lower the order of the vanishing moments has the smaller the

compact support (length of the mother wavelet) and the computation cost, but the lower the order of the vanishing moments represent a simple signal. The higher the order of the vanishing moments has the higher the compact support and represents a complex signal, but the computation cost is increased. Therefore, a reasonable choice is to use the smallest mother wavelet that gives satisfactory details and trends in a complex signal. Thus, the relevant mother wavelet extracts the details and trends from input data, so the detection rate of the DDoS attacks is increased.

The sixth question has the following answer. Internet traffic, the input data in this research, is a dynamically changing environment. Therefore, an iterative algorithm such as the evolutionary algorithm is needed for adapting the design of a mother wavelet to the stochastic environment of the Internet. Thus, in this research, a mother wavelet that adapts to its input signal through similarity with the input data and using the highest correlation concept is created. The adaptive discrete mother wavelet was designed for improving detection of DDoS attacks in Internet traffic.

This design leads to have a better quality of a mother wavelet for a specific purpose. As a result, speed in convergence, improve quality of mother wavelet for pattern recognition and classification were achieved. In order to achieve this, a GNN was introduced to design a dynamic mother wavelet that adapts to its input. In addition, a multi-objective optimization based on a genetic algorithm was used to create a set of adaptive mother wavelets that best fit the weight parameters for a given input data. Moreover, to have a lower running time and a faster convergence to the optimal coefficients of the neural network, a weighted cost function KL-divergence was created. The weighted cost function was used to measure how well the GNN is able to create a mother wavelet. The best mother wavelet coefficients for detecting DDoS attacks are achieved with coefficients [–0.3744,0.0034]. The created mother wavelet increased the detection rate of the DDoS attacks by 0.3% when compared to the Haar mother wavelet.

## 7.2 Contributions

This thesis contributes to the field in several ways as follows:

1. Power consumption attacks were considered in this thesis as a specific case of existential threats to the life of a city. In particular, bad data injection in a smart grid were simulated, and this attack accurately detected for the first time in field [GhFK16], [GhKF16].
2. The previous attack was expanded to DDoS attacks that were also detected successfully [GhKF17].
3. A dedicated version of the VFDT (VFDTv2) was implemented to extract features from the general fractal data that rely on long-range dependence, for long stationary frames. The VFDTv2 was adjusted to consider all points including the boundary points of a dataset [GhKi18], [GhKi19]. Moreover, the variance equation of the data time series was adjusted to consider all the points. The original algorithms was proposed by Kinsner in 1994 [Kins07], [Kins12], [Kins15], and applied to both deterministic and stochastic self-affine processes. The exclusion of the boundary points is always recommended for extremely small stationary frames to reduce the inherent statistical bias. That bias becomes less significant for larger frames.
4. A cost function for reducing running time was designed and implemented [GhKi20b]. The basic cost function was the KL-divergence. To enrich it, two coefficients were considered: the VFD and the distance. Also, to assess the proposed cost functions, a sensitivity analysis tool was used. The results showed the weighted KL-divergence cost function was the most sensitive cost function. Also, the weighted cost function had a lower running time and a

faster convergence to the optimal coefficients of the neural network than the non-weighted cost functions, so the training time of neural networks was reduced.

5. An adaptive mother wavelet that adapts to Internet traffic input that contains DDoS attacks was designed and implemented [GhKi21a].
6. The WT coefficients considering the adaptive mother wavelet are extracted using Adaptive-Wavelet CNN architectures [GhKi21b]. As a result, Adaptive-Wavelet CNN adapts to the input and the WT coefficients are extracted.
7. A new CNN architecture for detecting the DDoS attacks was introduced [GhKi21b]. As a result, the detection of DDoS attacks using the designed mother wavelet and the proposed CNN architecture was improved.
8. An extended version of policy gradient DRL, which uses unlabeled data, based on the PCNN was proposed and implemented to detect DDoS attacks [GhKi21c].

## 7.3 Future Work

In this section, some possible solutions for improvement of the performance of detection attacks in internet traffic are proposed.

First, this research introduced the design of an adaptive mother wavelet for the wavelet transform. In the future, this research can be extended for designing shearlets for shearlet transform, and the extension can be done in the direction of both supervised and unsupervised learning. Shearlets are mathematical analysis methods used for multiscale analysis of multidimensional signals [GPLC14] and can efficiently represent multidimensional data [KuLa12].

Moreover, this research focused on DDoS attacks in internet traffic since DDoS attacks violate the availability property of computer network security field. In the future, this research can be extended to detect other threats that violate other properties of computer network security field. For example, extending the detection attacks that violate data integrity, authentication, confidentiality, and freshness of data.

Also, this application can be adapted to detect a zero-day attack that is a potentially serious software security vulnerability or a threat, which is still unknown to software developers [ZuMa12]. Including a zero-day-attack component to an anomaly detection algorithm would make this detection software more comprehensive.

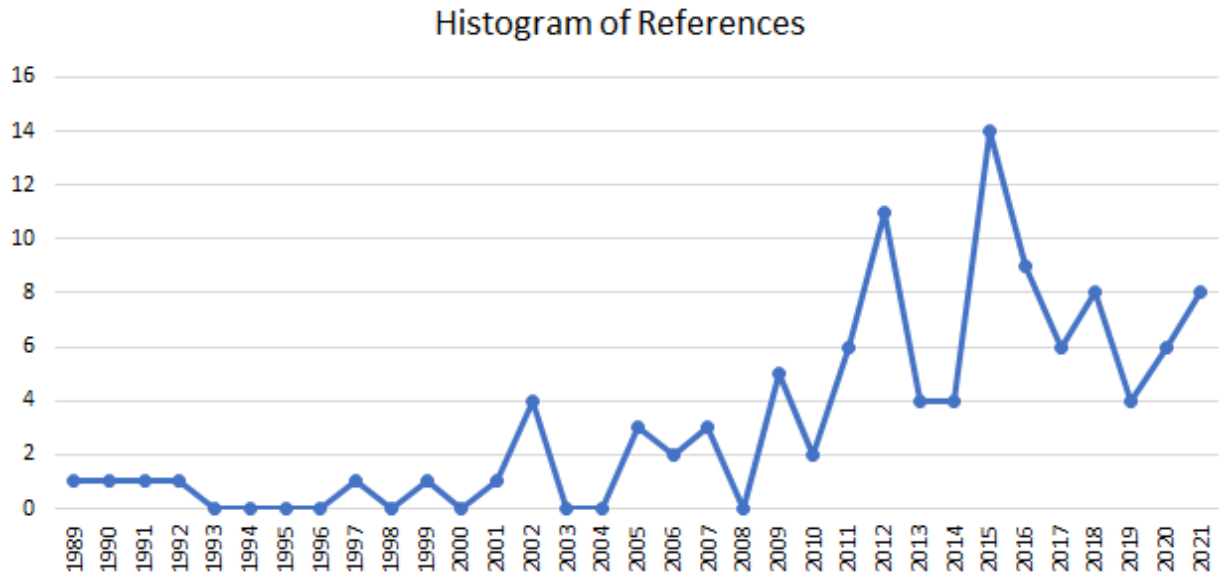
# References

In this research, Arabic numbers are not used to indicate the references such as [1], [2] and [3]. However, four letters along with two digits are used to indicate the references such as [AbDD02]. These four letters are the first letters of each author's last name and the two digits are the publication date. This method is used because by adding or deleting a reference in the body of the thesis, there is no need to update all numbering references.

References consulted for the thesis span the period from 1989 to 2021 and address the following topics: computer network security, cyber domain attacks, physical domain (power system) attacks, machine learning, convolutional neural network, deep reinforcement learning, digital signal processing, wavelet transform, statistics, fractal dimensions. The time range of references and their frequency for this research are shown in the following table.

**Table:** The time range of references and their frequency for this research.

| Year | Number of references | Year | Number of references | Year | Number of references |
|---|---|---|---|---|---|
| 1989 | 1 | 1990 | 1 | 1991 | 1 |
| 1992 | 1 | 1993 | 0 | 1994 | 0 |
| 1995 | 0 | 1996 | 0 | 1997 | 1 |
| 1998 | 0 | 1999 | 1 | 2000 | 0 |
| 2001 | 1 | 2002 | 4 | 2003 | 0 |
| 2004 | 0 | 2005 | 3 | 2006 | 2 |
| 2007 | 3 | 2008 | 0 | 2009 | 5 |
| 2010 | 2 | 2011 | 6 | 2012 | 11 |
| 2013 | 4 | 2014 | 4 | 2015 | 14 |
| 2016 | 9 | 2017 | 6 | 2018 | 8 |
| 2019 | 4 | 2020 | 6 | 2021 | 8 |



[AbDD02] Agostino Abbate, Casimer M Decusatis and Pankaj K Das, *Wavelets and Subbands Fundamentals and Applications*. Boston, MA: Birkhaüser, 2002 (1st ed.).

[Abhy15] Aditya Abhyankar, "Towards selecting wavelets through vanishing moments," *NPTEL*. 2015. Available: https://www.youtube.com/watch?v=A97jOODD4W8. [Accessed: Oct 2018].

[AdOs12] Olatunde A. Adeoti, and Peter A. Osanaiye, "Performance analysis of ANN on dataset allocations for pattern recognition of bivariate process," *Mathematical Theory and Modeling*, vol. 2, no. 10, pp. 53–63, 2012.

[BaED17] Mohamed Bakr, Atef Z. Elsherbeni, and Veysel Demir, *Adjoint Sensitivity Analysis of High Frequency Structures with MATLAB*. London, UK: Institution of Engineering and Technology: Scitech Publishing, 2017.

[BaPa12] Peter Barry and Patrick Crowly, *Modern Embedded Computing: Designing Connected Pervasive Media-Rich Systems*. Waltham, MA, USA: Morgan Kaufmann, 2012 (1st ed.).

[Beqi09] Elidon Beqiri, "Neural Networks for Intrusion Detection Systems," *Global Security, Safety, and Sustainability*, vol. 45, no. 3, pp. 156–165, 2009.

[Bhat18] Sravani Bhattacharjee, *Practical Industrial Internet of Things Security.* Birmingham, UK: Packt Publishing, 2018 (1st ed.). Available: https://www.packtpub.com/product/practical-industrial-internet-of-things-security/9781788832687

[BjGS17] Esben Jannik Bjerrum, Mads Glahder and Thomas Skov, "Data Augmentation of Spectral Data for Convolutional Neural Network (CNN) Based Deep Chemometrics," *arXiv.org (2017)*, Cornell University, NY, the USA, pp. 1–10, Oct 2017.

[BKPR02] Paul Barford, Jeffery Kline, David Plonka, and Amos Ron, "A signal analysis of network traffic anomalies," in *Proc. of the 2nd ACM SIGCOMM Workshop on Internet Measurement*, New York, NY, USA, pp. 71–82, Nov 2002.

[CAIDA15] The CAIDA UCSD "The CAIDA DDoS attack 2007 dataset," *caida.org*, 2015. [online]. Available: https://www.caida.org/data/passive/ddos-20070804_dataset.xml. [Accessed: Nov 2015].

[Chun10] Liu Chun-Lin, A Tutorial of the Wavelet Transform. Taipei, Taiwan, 2010. [online]. Available: http://disp.ee.ntu.edu.tw/tutorial/WaveletTutorial.pdf. [Accessed: Nov 2018].

[CoLV07] Carlos A. Coello Coello, Gary B. Lamont and David A. Van Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems.* New York, NY: Springer, 2007 (2nd ed).

[CoTh06] Thomas M. Cover and Joy A. Thomas, *Elements of Information Theory*. Somerset, UK: Wiley, 2006 (2nd ed.).

[Cybe21] "Cyber crime: reported damage to the IC3 2015 | Statistic," *Statista*, 2021. Available: https://www.statista.com/statistics/267132/total-damage-caused-by-by-cyber-crime-in-the-us/.[Accessed: Jun 2021].

[Dang15] Hieu V. Dang, *Adaptive Multiobjective Memetic Optimization: Algorithms and Applications.* Doctoral Thesis. Department of Electrical and Computer Engineering, University of Manitoba, Winnipeg, MB, Canada, Sep 2015.

[Daub92] Ingrid Daubechies, *Ten Lectures on Wavelets*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1992.

[Davi91] Lawrence Davis, *Handbook of Genetic Algorithms*. New York, NY:Van Nostrand Reinhold, 1991.

[DiHJ17] Ibrahim Dincer, Halil S. Hamut and Nader Javani, *Thermal Management of Electric Vehicle Battery Systems*. Hoboken, New Jersey: John Wiley & Sons, 2017 (1st ed.).

[FaMC11] Nicolas Falliere, Liam O. Murchu, and Eric Chien, "W32. Stuxnet dossier," White paper version 1.3, Symantec Corp., Security Response 5, CA, USA, Feb 2011.

[Fair16a] Peter Fairley, "Sniffing out grid attacks," *IEEE Spectrum*, vol. 53, no. 9, p. 13–15, Sep 2016.

[Fair16b] Peter Fairley, "Cybersecurity at U.S. utilities due for an upgrade: Tech to detect intrusions into industrial control systems will be mandatory [News]," *IEEE Spectrum*, vol. 53, no. 5, p. 11–13, May 2016.

[FeLo15] Paula Odete Fernandes and Rui Pedro Lopes, *Handbook of Research on Global Competitive Advantage through Innovation and Entrepreneurship: Clustering Global Entrepreneurship through Data Mining Technique*. Hershey, Pennsylvania: IGI Global, 2015 (1st ed.).

[Fere16] Ken Ferens, *Applied Computational Intelligence*, Course Notes, Winnipeg, MB; Department of Electrical and Computer Engineering, University of Manitoba, Jan 2016.

[FFSN11] Zubair Md. Fadlullah, Mostafa M. Fouda, Xuemin Shen, Yousuke Nozaki and Nei Kato, "An early warning system against malicious activities for smart grid communications," *IEEE Network*, vol. 25, no. 5, pp. 50–55, Oct. 2011.

[Frei15] Nando de Freitas, "Deep Learning Lecture 15: Deep Reinforcement Learning - Policy search," YouTube, [online], Mar. 2015. Available: https://www.youtube.com/watch?v=kUiR0RLmGCo.

[GaYa11] Robert X Gao and Ruqiang Yan, *Wavelets: Theory and Applications for Manufacturing*. New York, NY: Springer, 2011.

[Gero17] Aurelien Geron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. Sebastopol, CA: O'Reilly Media, Inc., 2017 (1st ed.).

[GhFK16] Maryam Ghanbari, Ken Ferens, and Witold Kinsner, "Anomaly detection in smart grid using wavelet transform and artificial neural network," in *Proc. of 2016 International Conference on Security and Management (SAM'16)*, Las Vegas, NV, USA, pp.15–21, Jul 2016.

[GhKF16] Maryam Ghanbari, Witold Kinsner, and Ken Ferens, "Anomaly detection in a smart grid using wavelet transform, variance fractal dimension and an artificial neural network," in *Proc. of 2016 IEEE Electrical Power and Energy Conference (EPEC)*, Ottawa, ON, Canada, pp. 1–6, Oct 2016.

[GhKF17] Maryam Ghanbari, Witold Kinsner, and Ken Ferens, "Detecting a distributed denial of service attack using a pre-processed convolutional neural network," in *Proc. of 2017 IEEE Electrical Power and Energy Conference (EPEC)*, Saskatoon, SK, Canada, pp. 1–6, Oct 2017.

[GhKi18] Maryam Ghanbari and Witold Kinsner, "Extracting features from both the input and the output of a convolutional neural network to detect distributed denial of service attacks," in *Proc. of 18th IEEE International Conference on Cognitive Informatics and Cognitive Computing (ICCI*CC'18*), Berkeley, CA, the USA, pp. 138–144, Jul 2018.

[GhKi19] Maryam Ghanbari and Witold Kinsner, "Detecting DDoS attacks using polyscale analysis and deep learning," *International Journal of Cognitive Informatics and Natural Intelligence*, vol. 14, no. 1, pp. 17–34, 2019. Available: 10.4018/ijcini.2020010102.

[GhKi20a] Maryam Ghanbari and Witold Kinsner, "Data augmentation methods and their effects on long-range dependence," in *Proc. of 20th IEEE International Conference on Cognitive*

*Informatics and Cognitive Computing (ICCI*CC'20*), Beijing, China, pp. 169–178, Sep 2020.

[GhKi20b] Maryam Ghanbari and Witold Kinsner, "Designing a cost function to assess a neural network to detect distributed denial of service attacks," in *Proc. of 20th IEEE International Conference on Cognitive Informatics and Cognitive Computing (ICCI*CC'20*), Beijing, China, pp. 131–138, Sep 2020.

[GhKi21a] Maryam Ghanbari and Witold Kinsner, "Designing a neural network and a genetic algorithm based adaptive wavelet for internet traffic containing DDoS attacks," in *Proc. of 21th IEEE International Conference on Cognitive Informatics and Cognitive Computing (ICCI*CC'21*), Banff, AL, Canada, Oct 2021. (Accepted)

[GhKi21b] Maryam Ghanbari and Witold Kinsner, "Detecting DDoS attacks using an adaptive-wavelet convolutional neural network," in *Proc. of 34th Annual IEEE Canadian Conference of Electrical and Computer Engineering (CCECE21),* Online, Canada, Sep 2021. (Accepted)

[GhKi21c] Maryam Ghanbari and Witold Kinsner, "Detecting DDoS attacks using a policy gradient based deep reinforcement learning,". in *Proc. of 21th IEEE International Conference on Cognitive Informatics and Cognitive Computing (ICCI*CC'21*), Banff, AL, Canada, Oct 2021. (Accepted)

[GhKi21d] Maryam Ghanbari and Witold Kinsner, "Data augmentation methods and their effects on long-range dependence and applications," *International Journal of Cognitive Informatics and Natural Intelligence*, 2021. (Accepted for publication 2021)

[GhKi21e] Maryam Ghanbari and Witold Kinsner, "Designing an adaptive cost function to assess a neural network to detect distributed denial of service attacks and its applications,"

*International Journal of Cognitive Informatics and Natural Intelligence*, 2021. (Accepted for publication 2021)

[GoCh11] Jaideva C. Goswami and Andrew K. Chan, *Fundamentals of Wavelets Theory, Algorithms, and Applications*, Hoboken, N.J.: John Wiley & Sons, 2011.

[GoKi16] David Terrazas Gonzalez and Witold Kinsner, "Zero-crossing analysis of Lévy walks for real-time feature extraction," in *Proc. of 2016 IEEE International Conference on Electro Information Technology (EIT)*, Grand forks, ND, the USA, pp. 0413-0421, May 2016.

[GPLC14] Xavier Gibert, Vishal M Patel, Demetrio Labate and Rama Chellappa, "Discrete shearlet transform on GPU with applications in anomaly detection and denoising", *EURASIP Journal on Advances in Signal Processing*, vol. 2014, no. 1, 2014. Available: 10.1186/1687-6180-2014-64 [Accessed 10 August 2020].

[Hart15] John Hart, "Normal resting pulse rate ranges", *Journal of Nursing Education and Practice*, vol. 5, no. 8, 2015. Available: 10.5430/jnep.v5n8p95.

[Hayk09] Simon Haykin, *Neural Networks and Learning Machines*. New York: Prentice Hall/Pearson, 2009 (3rd ed.).

[HeZh19] Ruisi He and Zhiguo Ding, *Applications of Machine Learning in Wireless Communications*, London, England: The Institution of Engineering and Technology, 2019 (1st ed.).

[HuPG14] Jiankun Hu, Hemanshu R. Pota, and Song Guo, "Taxonomy of attacks for agent-based smart grids*," IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 7, pp. 1886–1895, Jul 2014.

[JaBM13] Madiha Jalil, Faran Awais Butt, and Ahmed Malik, "Short-time energy, magnitude, zero crossing rate and autocorrelation measurement for discriminating voiced and unvoiced

segments of speech signals," in *Proc. of 2013 International Conference on Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE 2013)*, Konya, Turkey, pp. 208–212, May 2013.

[JeME13] Khalid Jebari, Mohammed Madiafi, and Abdelaziz Elmoujahid, "Parent selection operators for genetic algorithms," *International Journal of Engineering Research & Technology (IJERT)*, vol. 2, no. 11, pp. 1141–1145, Nov 2013.

[Jiay16] Yangqing Jia, "Caffe | layer catalogue berkeley vision and learning center (BVLC)," *Caffe.berkeleyvision.org*, 2016. [online]. Available: http://caffe.berkeleyvision.org/tutorial/layers.html. [Accessed: Jul 2016].

[JKRL09] Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun, "What is the best multi-stage architecture for object recognition?," in *Proc. of the 12th IEEE International Conference on Computer Vision (ICCV2009)*, pp.2146–2153, Oct 2009.

[JZZC17] Yajie Jiang, Xiaoning Zhang, Quan Zhou, and Zijing Cheng, "An entropy-based DDoS defense mechanism in software defined networks," in *Proc. of Communications and Networking: Springer International Publishing*, China, pp. 169–178, Oct. 2017, https://doi.org/10.1007/978-3-319-66625-9_17.

[Kala14] Seyed Mostafa Kalami Harris, "Multi-objective optimization educational set in Matlab," *Faradars*. 2014. [online]. https://faradars.org/courses/mvrmo9012-multiobjective-optimization-video-tutorials-pack. [Accessed: Apr 2019].

[Kala15] Seyed Mostafa Kalami Harris, "Principal component analysis," *Faradars*. 2015. [online]. https://www.youtube.com/watch?v=R17oDXTFwG0. [Accessed: Mar 2020].

[Karn11] Stamatis Karnouskos, “Stuxnet worm impact on industrial cyber-physical system security,” in *Proc. of the 37th Annual Conference on IEEE Industrial Electronics Society*, IECON 2011, pp. 4490–4494, Nov 2011.

[Karp15] Andrej Karpathy, “CS231n convolutional neural networks for visual recognition,” *Stanford University*, 2015. [online]. Available: http://cs231n.github.io/convolutional-networks/. [Accessed: Jun 2016].

[Kins07] Witold Kinsner, “A unified approach to fractal dimensions,” in *Proc. of 4th IEEE Conference on Cognitive Informatics (ICCI 2005)*, pp. 58–72, Oct 2007.

[Kins10] Witold Kinsner, "System complexity and its measures: How complex is complex.," *Advances in Cognitive Informatics and Cognitive Computing Studies in Computational Intelligence*, vol. 323, pp. 265-295, 2010.

[Kins12] Witold Kinsner, “Towards cognitive security systems,” in *Proc. of the 11th IEEE Intern. Conf. on Cognitive Informatics and Cognitive Computing* (ICCI*CC 2012), Keynote Speech, Kyoto, Japan, pp. 138–144, Aug 2012.

[Kins15] Witold Kinsner, *Fractal and Chaos Engineering*, Course Notes, Winnipeg, MB, Canada, Department of Electrical and Computer Engineering, University of Manitoba, Jan 2015.

[Kins20] Witold Kinsner, Fractal and Chaos Engineering: Monoscale, Multiscale and Polyscale Analyses. Winnipeg, MB: OCO Research, Jan. 2020, 1106 pages. {ISBN: 978-0-9939347-1-1, pbk}

[KrSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, “ImageNet classification with deep convolutional neural networks,” in *Neural Information and Processing Systems (NIPS’2012)*, pp.1097–1105, 2012.

[KSGH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, "Imagenet classification with deep convolutional neural networks," In *Advances in Neural Information Processing Systems 25 (NIPS 2012)*, pp. 1097–1105, Dec 2012.

[KuLa12] Gitta Kutyniok and Demetrio Labate, *Shearlets*, New York: Springer, 2012.

[Lect16] "Lecture: polynomial fits and splines," *Amath 301*. 2016. Available: https://www.youtube.com/watch?v=bFOTmSsDtAA. [Accessed: Mar 2019].

[LGWG13] Ting Liu, Yun Gu, Dai Wang, Yuhong Gui, and Xiaohong Guan, "A novel method to detect bad data injection attack in smart grid," in *Proc. of the IEEE INFOCOM Workshop on Communications and Control for Smart Energy Systems (CCSES),* pp. 49–54, 2013.

[Lipo19] Peter Lipovyanov, *Blockchain for Business.* Birmingham, UK: Packt Publishing, 2019 (ed. 1st).

[LuXL12] Huimin Lu, Jingbo Xu and Qiang Li, "Study on smelting reduction of coal-containing pellets of V-Ti bearing beach placers by combined rotary hearth furnace and direct current ARC furnace", in *Proc. of The Energy Technology 2012: Carbon Dioxide Management and Other Technologies*, Hoboken, NJ, USA, pp. 109–116, Mar 2012.

[MaDy14] Miroslaw Mazurek, Pawel Dymora, "Network anomaly detection based on the statistical self-similarity factor for HTTP protocol," *Przeglad Elektrotechniczny*, ISSN 0033-2097, vol. 1, pp. 127–130, Jan 2014.

[Mall89] Stephane G. Mallat, "A theory for multiresolution signal decomposition the wavelet representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, pp. 674–693, 1989.

[Mall09] Stephane Mallat, *A Wavelet Tour of Signal Processing.* London: Elsevier, 2009 (3rd ed.).

[MMMK12] Nasim Beigi Mohammadi, Jelena Misic, Vojislav B. Misic, and Hamzeh Khazaei, "A framework for intrusion detection system in advanced metering infrastructure," *Security and Communication Networks*, vol. 7, no. 1, pp. 195–205, Nov 2012.

[MoKi97] Fan Mo and Witold Kinsner, "Wavelet modelling of transients in power systems," in *Proc. of IEEE WESCANEX 97 Conference on Communications, Power and Computing*, WESCANEX 097, IEEE, pp. 132–137, May1997.

[MoKi02] Fan Mo and Witold Kinsner, *Wavelets and Artificial Neural Networks in Power System Transient Classification and Short-Term Power Load Prediction,* Master's Thesis, Department of Electrical and Computer Engineering, University of Manitoba, Winnipeg, MB, Canada, Aug 2002.

[MoMC09] David S. Moore, George P. McCabe, Bruce A. Craig, *Introduction to the Practice of Statistics*, New York: W.H. Freeman and Company, 2009 (6th ed.).

[PrRR13] K.M Prasad, A.R.M. Reddy and K.V. Rao, "Discriminating DDoS attack traffic from flash crowds on internet threat monitors (ITM) using entropy variations," *African Journal of Computing & ICT*, vol. 6, no. 2, pp. 53–62, Jun 2013.

[Proc20] Isaac R. Porche III, *Cyberwarfare: An Introduction to Information-Age Conflict*. Boston, London, UK: Artech House, 2020 (1st ed.).

[RaDe18] Pethuru Raj and Ganesh Chandra Deka, *Blockchain Technology: Platforms, Tools and Use Cases.* Elsevier, 2018. London, United Kingdom: Academic Press, 2018 (1st ed.).

[Rang01] Rangaraj Rangayyan, *Biomedical Signal Analysis*, New York: John Wiley & Sons, 2001 (1st ed.).

[ReMa18] Daniel Recoskie and Richard Mann, *Learning Sparse Orthogonal Wavelet Filters*, Master's Thesis, Department of Electrical and Computer Engineering, Waterloo, ON, Canada, 2018.

[ReRi18] Daniel Recoskie and Richard Mann, Learning Sparse Orthogonal Wavelet Filters, Master's Thesis, Department of Electrical and Computer Engineering, Waterloo, ON, Canada, 2018.

[RaSe15] Md. Mijanur Rahman and Tania Akter Setu, "An implementation for combining neural networks and genetic algorithms," *International Journal of Computer Science and Technology (IJCST)*, vol. 6, no. 3, pp. 218–222, Jul 2015.

[Rose13] Paul Rosenzweig, *Thinking about cybersecurity: from cyber crime to cyber warfare*, The George Washington University Law School, Virginia, USA: The Great Courses, pp. 3–10, 2013. Available: https://guidebookstgc.snagfilms.com/9523_Cybersecurity.pdf.

[SaKi05] Leila S. Safavian, Witold Kinsner, and H. Turanli, "A quantitative comparison of different mother wavelets for characterizing transients in power systems," in *Proc. of Canadian Conference on Electrical and Computer Engineering (CCECE2005)*, Saskatoon, SK, Canada, pp. 1453–1456, May 2005.

[SaKi06] Leila S. Safavian and Witold Kinsner, *Wavelet and Multifractal Analysis of Transients in Power Systems,* Master's Thesis, Department of Electrical and Computer Engineering, University of Manitoba, Winnipeg, MB, Canada, Jan 2006.

[Sang21] Nimish Sanghi, *Deep Reinforcement Learning with Python*, Berkeley, CA: Apress, 2021 (1st ed.).

[SeKi18] Sina Sedigh and Witold Kinsner, *Application of Polyscale Methods for Speaker Verification,* Master's Thesis, Department of Electrical and Computer Engineering, University of Manitoba, Winnipeg, MB, Canada, Apr 2018.

[SeKu08] Arturo Servin and Daniel Kudenko, "Multi-agent reinforcement learning for intrusion detection," In *Adaptive Agents and Multi-Agent Systems III. Adaptation and Multi-Agent Learning*, vol. 4865, pp. 211–223, Feb 2008.

[Serv07] Arturo Servin, "Towards traffic anomaly detection via reinforcement learning and data flow," Department of Computer Science, University of York, United Kingdom, pp. 81–88, 2007.

[ShCh15] Ramalingam Shanmugam and Rajan Chattamvelli, *Statistics for Scientists and Engineers*, John Wiley & Sons, Incorporated, 2015 (1st ed.).

[ShJa12] Afshin Shaabany and Fatemeh Jamshidi, "Network traffic deviation detection based on fractal dimension," *Journal of Computing and Information Technology,* vol. 20, no.1, pp. 27–32, Mar 2012.

[ShMH21] Reza Bakhtiari Shohani, Seyedakbar Mostafavi, and Vesal Hakam, "A statistical model for early detection of DDoS attacks on random targets in SDN," *Wireless Personal Communications*, vol.?, no.?, pp.1–22, Mar 2021.

[SHPC16] Hongjian Sun, Nikos D. Hatziargyriou, H. Vincent Poor, Laurence Carpanini, Miguel Angel Sánchez Fornié, *Smarter Energy: From smart metering to the smart grid.* London, UK: The Institution of Engineering and Technology, 2016 (1st ed.).

[Silv15] David Silver, "Reinforcement learning lectures," *DeepMind*. 2015. [Online]. https://deepmind.com/learning-resources/-introduction-reinforcement-learning-david-silver. [Accessed: Sep 2017].

[Sing18] Aishwarya Singh, "A gentle introduction to handling a non-stationary time series in Python", *Analytics Vidhya*, 2018. Available: https://www.analyticsvidhya.com/blog/2018/09/non-stationary-time-series-python/.

[SKFK17] Sana Siddiqui, Muhammad Salman Khan, Ken Ferens, and Witold Kinsner, "Fractal based cognitive neural network to detect obfuscated and indistinguishable internet threats, " in *Proc. 2017 IEEE 16th International Conference on Cognitive Informatics & Cognitive Computing (ICCI*CC17),* Oxford, UK, pp. 297–308, Jul 2017.

[SoKN20] Arun Solanki, Sandeep Kumar and Anand Nayyar, *Handbook of Research on Emerging Trends and Applications of Machine Learning*. Hershey, Pennsylvania: IGI Global, 2020 (1st ed.).

[SoLa05] Leif Sornmo and Pablo Laguna, *Bioelectrical signal processing in cardiac and neurological applications.* Amsterdam: Elsevier Academic Press, 2005 (1st ed.).

[SuBa16] Richard S. Sutton and andrew G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, Massachusetts, England: MIT Press, 2016 (2rd ed.).

[Sueu18] Jèrôme Sueur, *Sound Analysis and Synthesis with R*. Manhattan, New York: Springer, 2018 (1st ed.).

[SuJo15] Candace Suh-Lee and Juyeon Jo, "Quantifying security risk by measuring network risk conditions," in *Proc. of 2015 IEEE Computer and Information Science (ICIS)*, pp. 9–14, Jun 2015.

[Tabo19a] Phil Tabor, "How policy gradient reinforcement learning works," *Udemy: Modern Reinforcement Learning: Actor-Critic Algorithms*. 2019. [online]. https://www.youtube.com/watch?v=A_2U6Sx67sE. [Accessed: Nov 2019].

[Tabo19b] Phil Tabor, "How to beat lunar lander with policy gradients | TensorFlow tutorial," *Udemy: Modern Reinforcement Learning: Actor-Critic Algorithms*. 2019. [online]. Available: https://www.youtube.com/watch?v=UT9pQjVhcaU. [Accessed: Nov 2019].

[Thin15] "ThinLinc", *Ccldesktop.cc.umanitoba.ca*, 2015. [online]. Available: https://ccldesktop.cc.umanitoba.ca/main/. [Accessed: Sep 2015].

[TuLa13] Shantanu Tushar and Sarath Lakshman, *Linux Shell Scripting Cookbook*. Birmingham, UK: Packt Publishing, 2013 (2nd ed.).

[VaHC07] Frederik P. J. Vandecasteele, Thomas F. Hess and Ronald L. Crawford, *Manual of Environmental Microbiology: Using Genetic Algorithms to Optimize Functions of Microbial Ecosystems*. Washington: American Society for Microbiology (ASM) Press., 2007 (3rd ed.).

[Vale99] Clemens Valens, *A Really Friendly Guide to Wavelets*. Notes, Albuquerque, NM, US; Department of Computer Science, University of New Mexico, 1999. [online]. Available: http://iar.cs.unm.edu/search.html?q=Clemens+Valens&submit=#gsc.tab=0&gsc.q=Clemens%20Valens&gsc.page=1 [Accessed: Sep 2019].

[Veit05] David Veitch, *Wavelet Neural Networks and Their Application in the Study of Dynamical Systems*. Master's Thesis. Department of Mathematics, University of York, Heslington, York, UK, Aug 2005.

[VeKT10] Alex Van der Velden, Patrick Koch, and Santosh Tiwari, "Design optimization methodologies," *Simula*. ASM Handbook Volume 22B Application of Metal Processing Simulations, pp. 614–624, 2010. DOI: https://doi.org/10.31399/asm.hb.v22b.a0005505

[Vela20] Sathiyamoorthi Velayutham, *Handbook of research on applications and implementations of machine learning techniques*, Hershey, Pennsylvania: IGI Global, 2020 (1st ed.).

[VZHX21] Vivek Veeriah, Tom Zahavy, Matteo Hessel, Zhongwen Xu, Junhyuk Oh, Iurii Kemaev, Hado van Hasselt, David Silver, Satinder Singh, "Discovery of options via meta-learned subgoals," *arXiv preprint arXiv*, Cornell University, Feb 2021.

[Wang02] Yingxu Wang, "On cognitive informatics," In *Proc. of 1st IEEE Intern. Conf. Cognitive Informatic*, Calgary, AB, Canada, pp. 34–42, Aug 2002.

[WiLe90] Bernard Widrow and Michael A. Lehr, "30 Years of adaptive neural networks: perceptron, Madaline, and Backpropagation," *Proc. of IEEE*, vol. 78, no. 9, 1990, pp. 1415–1442.

[WrFW12] Michael Wrinch, Tarek H. M. EL-Fouly, and Steven Wong, "Anomaly detection of building systems using energy demand frequency domain anlaysis," in *Proc. IEEE Power & Energy Society General Meeting (IEEE PES GM 2012)*, San-Diego, CA, USA, pp. 1–6, Jul 2012.

[XuXi05] Xin Xu and Tao Xie, "A reinforcement learning approach for host-based intrusion detection using sequences of system calls," In *Advances in Intelligent Computing (ICIC 2005)*, Heidelberg, Germany, pp. 995–1003, Sep. 2005.

[ZhHT11] Sen Xin Zhou, Jiang Hong Han, and Hao Tang, "A Trust Evaluation Model for Industrial Control Ethernet Network," *International Journal of Wireless and Microwave Technologies*, vol. 1, no. 5, pp. 60–66, Oct 2011.

[ZuMa12] Junaid Ahmed Zubairi and Athar Mahboob, *Cyber Security Standards, Practices and Industrial Applications*. Hershey, Pa., USA: IGI Global, 2012 (1st ed.).

# Recent Pertinent References

[AAAA21] Kamal Alieyan, Ammar Almomani, Mohammed Anbar, Mohammad Alauthman, Rosni Abdullah, and B. B. Gupta, "DNS rule-based schema to botnet detection," Enterprise Information Systems, vol. 15, no. 4, pp. 545-564, 2021. https://www.tandfonline.com/doi/abs/10.1080/17517575.2019.1644673

[AbKh16] Al-Sudani Mustafa Qahtan Abdulmunem, and Vyacheslav S. Kharchenko, "Availability and security assessment of smart building automation systems: combining of attack tree analysis and Markov models," in 2016 Third International Conference on Mathematics and Computers in Sciences and in Industry, MCSI, (Chania, Greece; 27-29 Aug. 2016), pp. 302–307, 2016. {DOI: 10.1109/MCSI.2016.062} https://ieeexplore.ieee.org/abstract/document/7815162 https://scholar.google.com/scholar?q=Abdulmunem%2C%20A.-S.%20M.%20Q.%2C%20Kharchenko%2C%20V.%20S.%3A%20Availability%20and%20security%20assessment%20of%20smart%20building%20automation%20systems%3A%20combining%20of%20attack%20tree%20analysis%20and%20Markov%20models.%20In%3A%202016%20Third%20International%20Conference%20on%20Mathematics%20and%20Computers%20in%20Sciences%20and%20in%20Industry%20%28MCSI%29%2C%20pp.%20302%E2%80%93307

[AdDG18] Vipindev Adat; Amrita Dahiya; B. B. Gupta, (2018, January). "Economic incentive based solution against distributed denial of service attacks for IoT customers," in 2018 IEEE International Conference on Consumer Electronics, ICCE, (Las Vegas, NV; 12-14 Jan. 2018) pp. 1-5, Jan 2018. https://ieeexplore.ieee.org/abstract/document/8326280

[Chen07] Cao Lai-Cheng, "A high-efficiency intrusion prediction technology based on Markov chain," in Pro. 2007 International Conference on Computational Intelligence and Security Workshops, CISW 2007, (Harbin, China; 15-19 Dec. 2007), pp. 518–521, 2007. {DOI: 10.1109/CISW.2007.4425547}
https://ieeexplore.ieee.org/abstract/document/4425547
https://scholar.google.com/scholar?q=Cao%2C%20L.-C.%3A%20A%20high-efficiency%20intrusion%20prediction%20technology%20based%20on%20Markov%20chain.%20In%3A%202007%20International%20Conference%20on%20Computational%20Intelligence%20and%20Security%20Workshops%20%28CISW%202007%29%2C%20pp.%20518%E2%80%93521

[ChCh22] Chin-Ling Chen and Chieh-Min Chen, "An early detection of distributed denial of service attack," in Proceedings of ICACCP 2021; Tapan Kumar Gandhi, Debanjan Konar, Biswaraj Sen, and Kalpana Sharma (eds), Advanced Computational Paradigms and Hybrid Intelligent Computing. Advances in Intelligent Systems and Computing, Volume 1373. Singapore: Springer, 2022. {ISBN 978-981-16-4368-2, pbk, US$ 249.99; ISBN 978-981-16-4369-9; ebk, US$ 189.00}
https://doi.org/10.1007/978-981-16-4369-9_21

[CPGC21] Ivan Cvitić, Dragan Peraković, Brij Gupta, and Kim-Kwang Raymond Choo, "Boosting-based DDoS detection in Internet of Things systems," IEEE Internet of Things Journal (Open Access), 2021, 14 pages. {DOI: 10.1109/JIOT.2021.3090909}
https://ieeexplore.ieee.org/abstract/document/9461235

[DaGu] A. Dahiya and B. B. Gupta, "How IoT is making DDoS attacks more dangerous?" Insights2Techinfo, Dec 11, 2021. https://insights2techinfo.com/how-iot-is-making-ddos-attacks-more-dangerous/

[DDGS17] Michele De Donno, Nicola Dragoni, Alberto Giaretta, and Angelo Spognardi, "Analysis of DDoS-capable IoT malwares," in 2017 Federated Conference on Computer Science and Information Systems, FedCSIS, (Prague, Czech Republic: 3-6 Sept. 2017) pp. 807-816, 2017. {DOI: 10.15439/2017F288} https://ieeexplore.ieee.org/abstract/document/8104642

[HoBK15] Nazrul Hoque, Dhruba K. Bhattacharyya, and Jugal K. Kalita, "Botnet in DDoS attacks: trends and challenges," IEEE Communications Surveys & Tutorials, vol. 17, no. 4, pp. 2242-2270, Fourth Quarter 2015. {DOI: 10.1109/COMST.2015.2457491} https://ieeexplore.ieee.org/abstract/document/7160662

[HoVV20] Pilar Holgado, Víctor A. Villagrá, and Luis Vázquez, "Real-time multistep attack prediction based on hidden Markov models," IEEE Trans. Dependable Secure Comput., vol. 17, no. 1, pp. 134-147, Jan-Feb 2020. {DOI: 10.1109/TDSC.2017.2751478} https://ieeexplore.ieee.org/abstract/document/8031986 https://scholar.google.com/scholar?q=Holgado%2C%20P.%2C%20Villagr%C3%A1%2C%20V.%20A.%2C%20V%C3%A1zquez%2C%20L.%3A%20Real-time%20multistep%20attack%20prediction%20based%20on%20hidden%20Markov%20models.%20IEEE%20Trans.%20Dependable%20Secure%20Comput.%2017%281%29%20%282020%29

[JZAG20] Yizhen Jia, Fangtian Zhong, Arwa Alrawais, Bei Gong, and Xiuzhen Cheng. "Flowguard: An intelligent edge defense mechanism against IoT DDoS attacks," IEEE

Internet of Things Journal, vol. 7, no. 10, pp. 9552-9562, Oct 2020. {
DOI: 10.1109/JIOT.2020.2993782}
https://ieeexplore.ieee.org/abstract/document/9090824

[KaZo08] D. A. Karras and V. C. Zorkadis, "On efficient security modelling of complex interconnected communication systems based on Markov processes," in Proc. 2008 New Technologies, Mobility and Security, (Tangier, Morocco; 5-7 Nov. 2008), pp. 1–7, 2008. {DOI: 10.1109/NTMS.2008.ECP.72}
https://ieeexplore.ieee.org/abstract/document/4689126
https://scholar.google.com/scholar?q=Karras%2C%20D.A.%2C%20Zorkadis%2C%20V.C.%3A%20On%20efficient%20security%20modelling%20of%20complex%20interconnected%20communication%20systems%20based%20on%20Markov%20processes.%20In%3A%202008%20New%20Technologies%2C%20Mobility%20and%20Security%2C%20pp.%201%E2%80%937

[KoKI19] Maryna Kolisnyk, Vyacheslav Kharchenko, and Piskachova Iryna, "IoT server availability considering DDoS-attacks: Analysis of prevention methods and Markov model," in Proc. 2019 10th International Conference on Dependable Systems, Services and Technologies, DESSERT, (Leeds, UK; 5-7 June 2019), 2019. {DOI: 10.1109/DESSERT.2019.8770012}
https://ieeexplore.ieee.org/abstract/document/8770012
https://scholar.google.com/scholar?q=Kolisnyk%2C%20M.%2C%20Kharchenko%2C%20V.%2C%20Iryna%2C%20P.%3A%20IoT%20server%20availability%20considering%20DDoS-attacks%3A%20analysis%20of%20prevention%20methods%20and%20Markov%20model.

%20In%3A%202019%2010th%20International%20Conference%20on%20Dependable%20Systems%2C%20Services%20and%20Technologies%20%28DESSERT%29

[KuWY12] GuangCai Kuang, XiaoFeng Wang, and LiRu Yin, "A fuzzy forecast method for network security situation based on Markov," in Proc. 2012 International Conference on Computer Science and Information Processing, CSIP, (Xi'an, Shaanxi; 24-26 Aug. 2012), pp. 785–789, 2012. {DOI: 10.1109/CSIP.2012.6308971}
https://ieeexplore.ieee.org/abstract/document/6308971
https://scholar.google.com/scholar?q=Kuang%2C%20G.C.%2C%20Wang%2C%20X.F.%2C%20Yin%2C%20L.R.%3A%20A%20fuzzy%20forecast%20method%20for%20network%20security%20situation%20based%20on%20Markov.%20In%3A%202012%20International%20Conference%20on%20Computer%20Science%20and%20Information%20Processing%20%28CSIP%29%2C%20pp.%20785%E2%80%93789

[LeHo18] Ngoc T. Le and Doan B. Hoang, "Security threat probability computation using Markov chain and common vulnerability scoring system," in Proc. 2018 28th International Telecommunication Networks and Applications Conference, ITNAC, (Sydney, NSW; 21-23 Nov. 2018), pp. 1–6, 2018. {DOI: 10.1109/ATNAC.2018.8615386}
https://ieeexplore.ieee.org/abstract/document/8615386
https://scholar.google.com/scholar?q=Le%2C%20N.T.%2C%20Hoang%2C%20D.B.%3A%20Security%20threat%20probability%20computation%20using%20Markov%20chain%20and%20common%20vulnerability%20scoring%20system.%20In%3A%202018%2028th%20International%20Telecommunication%20Networks%20and%20Applications%20Conference%20%28ITNAC%29%2C%20pp.%201%E2%80%936

[MiRT17] Erik Miehling, Mohammad Rasouli, and Demosthenis Teneketzis, "A dependency graph formalism for the dynamic defense of cyber networks," in Proc. 2017 IEEE Global Conference on Signal and Information Processing, GlobalSIP, (Montreal, QC; 14-16 Nov. 2017), pp. 511–512, 2017. {DOI: 10.1109/GlobalSIP.2017.8308695}
https://ieeexplore.ieee.org/abstract/document/8308695
https://scholar.google.com/scholar?q=Miehling%2C%20E.%2C%20Rasouli%2C%20M.%2C%20Teneketzis%2C%20D.%3A%20A%20dependency%20graph%20formalism%20for%20the%20dynamic%20defense%20of%20cyber%20networks.%20In%3A%202017%20IEEE%20Global%20Conference%20on%20Signal%20and%20Information%20Processing%20%28GlobalSIP%29%2C%20pp.%20511%E2%80%93512

[Sing21] Dhananjay Singh, "Captcha Improvement: Security from DDoS Attack," Insights2Techinfo, 2021, pp.1
https://insights2techinfo.com/captcha-improvement-security-from-ddos-attack/

[SHSh10] Marn-Ling Shing and Chen-Chi Shing, "Information security risk assessment using Markov models," in Proc. 2010 Third International Symposium on Electronic Commerce and Security, (Nanchang, China; 29-31 July 2010), pp. 403–406, 2010. {DOI: 10.1109/ISECS.2010.97}
https://ieeexplore.ieee.org/abstract/document/5557362
https://scholar.google.com/scholar?q=Shing%2C%20M.-L.%2C%20Shing%2C%20C.-C.%3A%20Information%20security%20risk%20assessment%20using%20Markov%20models.%20In%3A%202010%20Third%20International%20Symposium%20on%20Electronic%20Commerce%20and%20Security%2C%20pp.%20403%E2%80%93406

[Sun15] Shouxin Sun, "The research of the network security situation prediction mechanism based on the complex network," in Proc. 2015 International Conference on Computational Intelligence and Communication Networks, CICN, (Jabalpur, India, 12-14 Dec. 2015), pp. 1183–1187, 2015.
https://ieeexplore.ieee.org/abstract/document/7546283
https://scholar.google.com/scholar?q=Sun%2C%20S.%3A%20The%20research%20of%20the%20network%20security%20situation%20prediction%20mechanism%20based%20on%20the%20complex%20network.%20In%3A%202015%20International%20Conference%20on%20Computational%20Intelligence%20and%20Communication%20Networks%20%28CICN%29%2C%20pp.%201183%E2%80%931187

[TGAM13] Shweta Tripathi, Brij Gupta1, Ammar Almomani, Anupama Mishra1, and Suresh Veluru, "Hadoop based defense solution to handle distributed denial of service (DDoS) attacks," Journal of Information Security, vol. 4, no. 3, Article ID: 3462, 2013, 15 pages. (Open Journal) {DOI:10.4236/jis.2013.43018}
https://www.scirp.org/html/4-7800161_34629.htm?pagespeed=noscript

[TNEC17] T. T. Teoh, Y. Y. Nguwi, Yuval Elovici, N. M. Cheung, and W. L. Ng, "Analyst intuition based hidden Markov model on high speed, temporal cyber security big data," in Proc. 2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery, ICNC-FSKD, (Guilin, China; 29-31 July 2017), pp. 2080–2083, 2017. {DOI: 10.1109/FSKD.2017.8393092}
https://ieeexplore.ieee.org/abstract/document/8393092
https://scholar.google.com/scholar?q=Teoh%2C%20T.T.%2C%20Nguwi%2C%20Y.Y.%2C%20Elovici%2C%20Y.N.%2C%20Cheung%2C%20M.%2C%20Ng%2C%20W.L.%3A%

20Analyst%20intuition%20based%20hidden%20Markov%20model%20on%20high%20speed%2C%20temporal%20cyber%20security%20big%20data.%20In%3A%202017%2013th%20International%20Conference%20on%20Natural%20Computation%2C%20Fuzzy%20Systems%20and%20Knowledge%20Discovery%20%28ICNC-FSKD%29%2C%20pp.%202080%E2%80%932083

[WSWF16] Changchun Wang, Changhui Shi, Chong Wang, and Ying Fu, "An analyzing method for computer network security based on Markov game model," in 2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference, IMCEC, (Xi'an, China; 3-5 Oct. 2016), pp. 454–45, 2016. {DOI: 10.1109/IMCEC.2016.7867253}

https://ieeexplore.ieee.org/abstract/document/7867253

https://scholar.google.com/scholar?q=Wang%2C%20C.%2C%20Shi%2C%20C.%2C%20Wang%2C%20C.%2C%20Fu%2C%20Y.%3A%20An%20analyzing%20method%20for%20computer%20network%20security%20based%20on%20Markov%20game%20model.%20In%3A%202016%20IEEE%20Advanced%20Information%20Management%2C%20Communicates%2C%20Electronic%20and%20Automation%20Control%20Conference%20%28IMCEC%29%2C%20pp.%20454%E2%80%93458

[YYGJ15] Qiao Yan, F. Richard Yu, Qingxiang Gong, and Jianqiang Li, "Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges," IEEE Communications sSrveys & Tutorials, 18(1), 602-622, 2015.

https://ieeexplore.ieee.org/abstract/document/7289347

[ZaTi13] Saman Taghavi Zargar; James Joshi; David Tipper, "A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks," IEEE Communications Surveys & Tutorials, vol. 15, no. 4, pp. 2046-2069, Fourth Quarter 2013. https://ieeexplore.ieee.org/abstract/document/6489876

[ZhLD10] Jiangtao Zhai, Guangjie Liu, and Yuewei Dai, "A covert channel detection algorithm based on TCP Markov model," in Proc. 2010 International Conference on Multimedia Information Networking and Security, (Nanjing, China; 4-6 Nov. 2010), pp. 893–897, 2010. {DOI: 10.1109/MINES.2010.190} https://ieeexplore.ieee.org/abstract/document/5671231 https://scholar.google.com/scholar?q=Zhai%2C%20J.%2C%20Liu%2C%20G.%2C%20Dai%2C%20Y.%3A%20A%20covert%20channel%20detection%20algorithm%20based%20on%20TCP%20Markov%20model.%20In%3A%202010%20International%20Conference%20on%20Multimedia%20Information%20Networking%20and%20Security%2C%20pp.%20893%E2%80%93897

[ZhNa18] Jianjun Zheng and Akbar Siami Namin, "Defending SDN-based IoT networks against DDoS attacks using Markov decision process," in Proc. 2018 IEEE International Conference on Big Data (Big Data), (Seattle, WA; 10-13 Dec. 2018), pp. 4589–4592, 2018. {DOI: 10.1109/BigData.2018.8622064} https://ieeexplore.ieee.org/abstract/document/8622064 https://scholar.google.com/scholar?q=Zheng%2C%20J.%2C%20Namin%2C%20A.S.%3A%20Defending%20SDN-based%20IoT%20networks%20against%20DDoS%20attacks%20using%20Markov%20deci

sion%20process.%20In%3A%202018%20IEEE%20International%20Conference%20on%20Big%20Data%20%28Big%20Data%29%2C%20pp.%204589%E2%80%934592

[ZHXY15] Chunjie Zhou, Shuang Huang, Naixue Xiong, Shuang-Hua Yang, Huiyun Li, Yuanqing Qin, and Xuan Li, "Design and analysis of multimodel-based anomaly intrusion detection systems in industrial process automation," IEEE Trans. Syst. Man Cybernet. Syst., vol. 45, no. 10, pp. 1345–1360, Oct 2015. {DOI: 10.1109/TSMC.2015.2415763} https://ieeexplore.ieee.org/abstract/document/7081762 https://scholar.google.com/scholar?q=Zhou%2C%20C.%2C%20Huang%2C%20S.%2C%20Xiong%2C%20N.%2C%20Yang%2C%20S.-H.%2C%20Li%2C%20H.%2C%20Qin%2C%20Y.%2C%20Li%2C%20X.%3A%20Design%20and%20analysis%20of%20multimodel-based%20anomaly%20intrusion%20detection%20systems%20in%20industrial%20process%20automation.%20IEEE%20Trans.%20Syst.%20Man%20Cybernet.%20Syst.%2045%2810%29%2C%201345%E2%80%931360%20%282015%29

.

# Appendix A

# SCADA

As mentioned in the introduction to Ch. 1, SCADA systems are physical devices that are vulnerable to cyber-attacks. SCADA systems are highly distributed systems used to control geographically distributed properties, often divided over thousands of square kilometers, by using centralized data acquisition and supervisory control [StFK06]. SCADA needs data from a network of sensors that enables it to monitor a system. The SCADA control center performs centralized monitoring and control for field sites over long-distance communications networks, including monitoring alarms and processing status data.

PLC is an industrial computer that controls industrial equipment and processes and executes control operations in a company environment. PLCs are control system components used throughout SCADA. PLCs are used in almost all industrial processes. PLC communications are usually *local area network* (LAN) technologies that are typically more reliable and with faster speed compared to the long-distance communication systems *wide area network* (WAN)

technologies that are used by SCADA systems. PLCs are generally used for discrete control for specific applications and generally provide regulatory control.

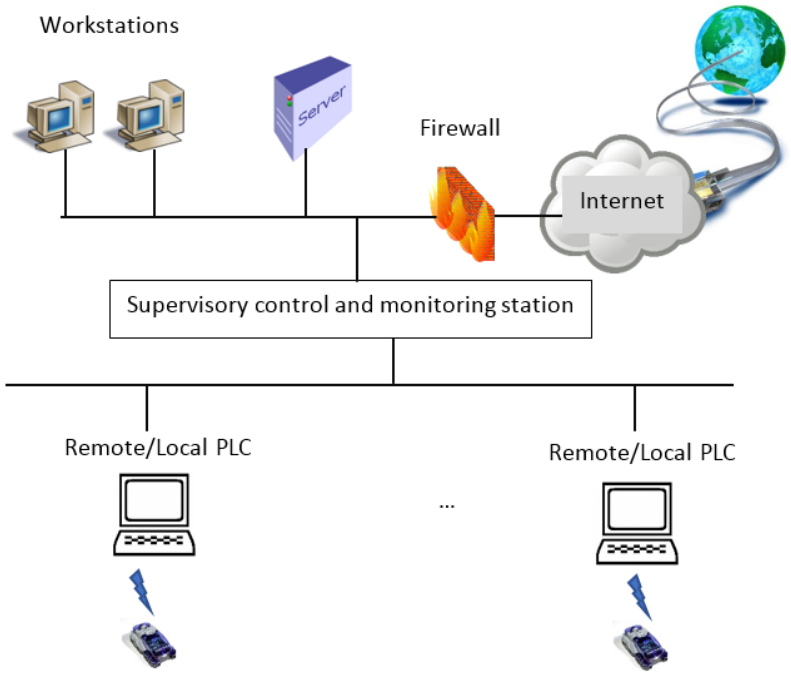



**Fig. A.1:** SCADA. (After [Unite04])

# Appendix B

# SMART GRID

As mentioned in the introduction to Ch. 1, smart grid systems are physical devices that are vulnerable to cyber-attacks. A smart grid is a large-scale power delivery network [Feng13]. The smart grid is one of the most important control systems. It uses *advanced metering infrastructure* (AMI) to provide bidirectional communication between smart meters (in consumers' side) and utilities providers (in the control system's side) [MMMK12] [YShT11]. The purposes of the smart grid are real-time control, and real-time measurement of power consumption which improve the reliability, efficiency, and the security and safety of devices while saving energy and reducing cost. Although transferred data between smart meters and utility centers are vital for smart grid infrastructure, the cyber layer in a smart grid opens up cyber threats because adversaries can manipulate the transferred data. Therefore, AMIs are vulnerable to cyber threat, so hackers can ruin a smart grid's beneficial effects.

In order to understand the vulnerabilities of the smart grid, two aspects of the smart grid need to be considered. The two perspectives regarding smart grids are: the power electric aspect and the communications aspect [MMMK12]. The electric aspect of the smart grid consists of

power generation, transmission, and distribution to customers. The communications aspect of the smart grid consists of WANs, *neighborhood area networks* (NANs), and *home area networks* (HANs). In Fig. B.1 smart grid elements are shown. In Fig. B.2 communications aspect of the smart grid are shown [MMMK12].

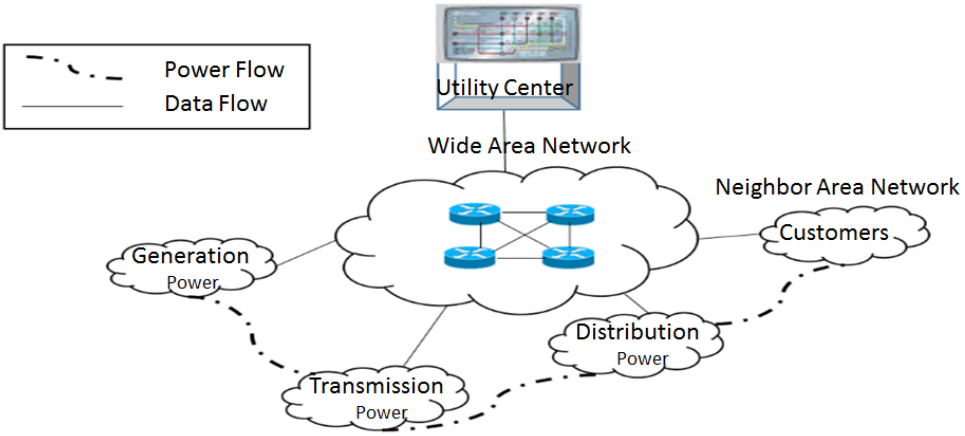



**Fig. B.1:** The power electric and communications aspects of the smart grid. (After [MMMK12])

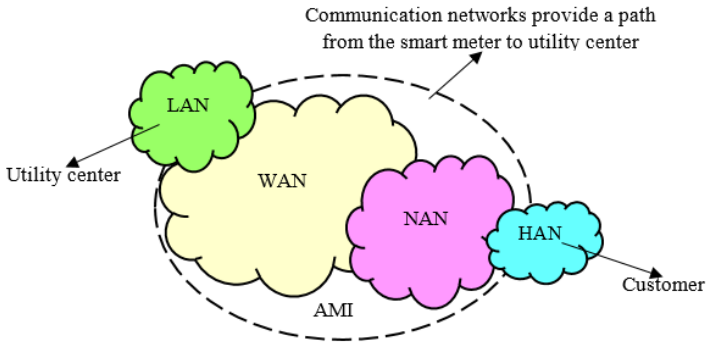



**Fig. B.2:** AMI in the smart grid. (After [MMMK12])

The utility center (in WAN level) shows the amount of utility for each subsystem NAN. However, the utility center does not indicate exactly where the power consumption takes place. In

the case of a cyber attack, the exact location of the anomalous power consumption is necessary in order to stop the attack in the smart grid.

There is a key vulnerability factor in smart grids called a smart meter. A smart meter is an access point for bidirectional communications between its controller and the consumer. Consumers can monitor their power consumption, and they can manage peak load by saving energy, reducing cost, and increasing reliability using a smart meter. In the case of a cyber domain attack, a hacker can change the data of the smart meter or deny a customer access to power.

# Appendix C

# BLOCKCHAIN

As mentioned in the introduction to Ch. 1, blockchain technology is vulnerable to cyber-attacks. Blockchain is a software protocol technology that provides secure Internet money transfer services without engaging a third party, like a central bank or a financial organization [RaDe18]. A transaction is validated, executed, and recorded in a database in the order in which the event occurred. The database is always available on the Internet for on-demand lookup and verification. Blockchain technology supports the Bitcoin (₿) cryptocurrency (Bitcoin is Fiat Currency that uses as money in e-transfer and e-commerce for trading) [RaDe18] [Lipo19] through its decentralized payment system for real-time transfer of digital currency, at any time and to anyone in the world. The bitcoin protocol is similar to SMTP, which is a simple mail transfer protocol, providing the underlying software protocol for Internet users to send emails to each other. Since bitcoin protocol uses blockchain protocol, Bitcoin can be transferred without engaging a financial organization.

# Appendix D

# DNS

As mentioned in Sec. 1.1, DNS naming system is a vulnerable protocol to cyber-attacks. In the networking world, computers do not find each other by names like how a human does, instead, they find each other through using numbers on the Internet [BaPa12]. To access a website, a web server address should be used, which is identified by the IP address. The Domain Name System (DNS) is developed to fill the gap between websites and their IP addresses making communications easier. The DNS is like a phonebook for humans. The DNS is a hierarchical naming system that translates websites and domain names to their IP addresses [TuLa13].

The method of translating a name occurs when an application or a web browser of a host machine has to obtain the IP address of a website. First, the DNS calls a library function called a Name Resolver [TuLa13]. The library function accepts the name of a website as its input parameter. Then, the library function generates a query packet using the user datagram protocol (UDP). The library function sends this UDP packet to the address of a by a default DNS server. All host machines must have at least one IP address from a DNS server. This local DNS server

returns the IP address equivalent to the website name after a hierarchical search. When the IP address is found, the application or the web browser can continue the operation.

Therefore, DNS is an important service, that efficiency and robustness are the main components of its performance. However, the DNS security is low. To improve the DDoS attack detection in a DNS, this research has been conducted. In this study, the CAIDA DNS 2006 and 2007 datasets are used [CAIDA15].

# Appendix E

# TCP/IP MODEL

As mentioned in Sec. 1.1, DNS protocol is one of the services provided by the application layer in the TCP/IP model. TCP/IP is a four-layer model used for computer network communications between two devices, such as clients and servers (TCP/IP model is shown in Fig. E.1).

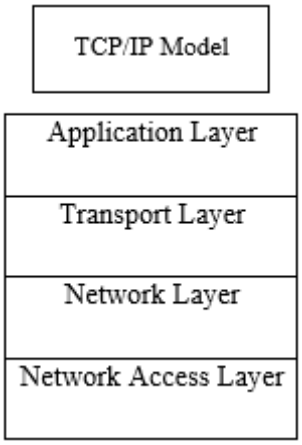



**Fig. E.1:** TCP/IP model.

In the physical layer, transmission media and mode of communication between two devices are defined. In physical layer, the data is sent to (or received from) using guided (cable or optical fiber) or wireless. In the IP layer, the most important duty is routing packets in the networks, so the best path that is not crowded is selected. In the TCP layer, the most important duty is receiving/sending packets in connection-oriented communication mode. Also, TCP layer controls the congestion. The application layer is the most attractive layer for users, and the most services are used in this layer for the users.

For communications, two devices need a common language that is called a protocol. There are several protocols in computer networks such as HTTP, SMTP and POP3; however, most protocols are not designed secure. Two methods are used to secure the networks using secure protocols and security devices. First, devices for secure communications should use secure protocols, such as HTTPS, IPsec and SSL VPN. Second, the backbone of networks and devices should use security devices for monitoring incoming and outgoing network traffic to permit or block data packets such as firewalls. The firewalls can be used in the IP layer, TCP layer, and application layer. In this thesis, the proposed IDSs detect and alert DDoS attacks. Next, firewalls perform actions such as blocking and filtering of traffic on the IDSs alerts. Therefore, by setting firewall rules in the application layer, the DDoS attacks can be detected and prevented. The application layer is used to set firewall for the DNS because DNS runs in the Application Layer (layer 4 of TCP/IP model or layer 7 of the OSI model).

# Appendix F

# DDoS Attacks Characteristics

As mentioned in the introduction to Ch. 1, DDoS attacks have specific characteristics that are introduced as follows. The normal Internet traffic data has bursty behavior. Therefore, identifying normal burst-data behaviors and abnormal burst-data behaviors caused by DDoS attacks is challenging. They are both stochastic, non-stationary, non differentiable, broadband, dynamic, self-affine fractal, and correlated with long-range dependence signals.

The bursty behavior of Internet traffic and DDoS attacks are similar, but they behave differently with respect to distribution of source IP address, the access intents, and speed of the increased and decreased traffic [PrRR13]. First, the source IP addresses in bursty behavior of Internet traffic is very scattered, while the distribution of IP addresses in DDoS attacks are relatively limited to the availability of zombies. Second, in a normal bursty behavior of Internet traffic, legitimate users respond to special events such as breaking news or popular products [PrRR13]. However, in the DDoS attacks there are not the special events and are launched by a hacker. Third, the number of users accessed the server increased gradually in bursty behavior of Internet traffic. However, the traffic increases sharply to the peak in DDoS attacks and decreases

sharply when the attacker withdraws the attack. Fourth, bursty behavior of Internet traffic comes from randomly distributed users all over the Internet, while the flow similarity among DDoS attack is much stronger than bursty behavior of Internet traffic [PrRR13]. DDOS attacks are illegitimate flows, and this attack has been a serious threat to internet security and stability [ShMH21].

# Appendix G

# VFD ALGORITHM VERIFICATION

As mentioned in in Sec. 3.1.1, VFD algorithm should be verified. To verify the VFD algorithm results, a uniform white-noise time series with the mean equal to zero, variance equal to 0.08 and epoch size equal to $2^{20}$ was generated. The white noise within a selected frame of 512 pieces of stationary data is shown in Fig. G.1.

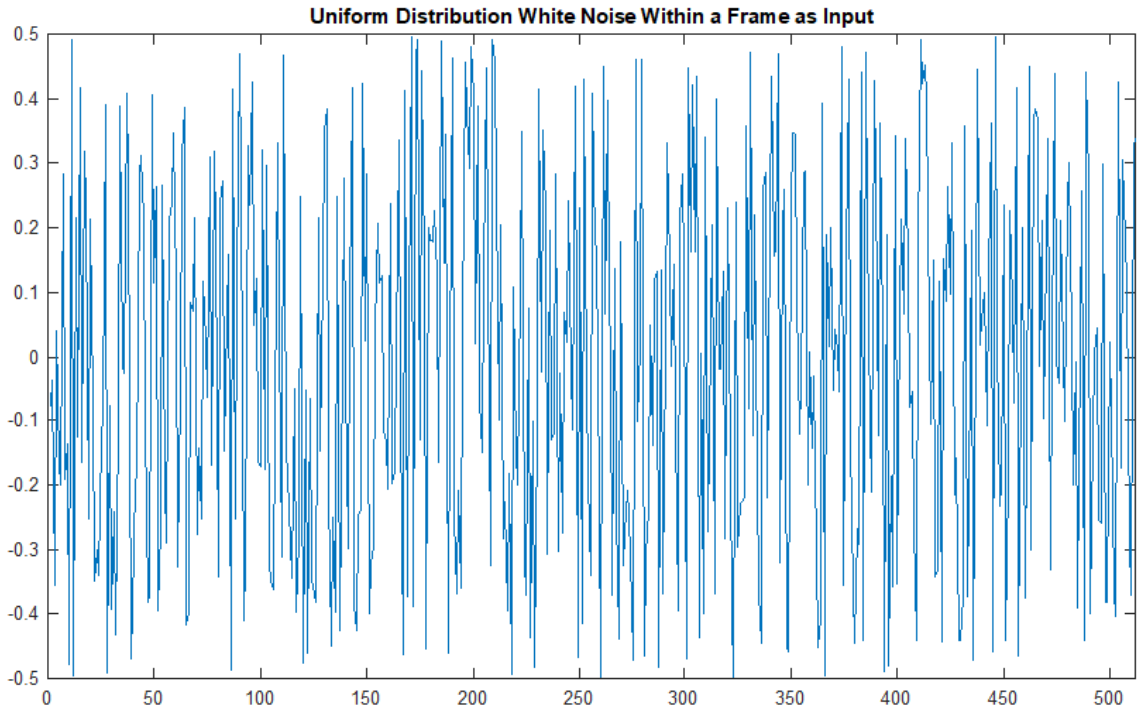



**Fig. G.1:** Uniformly distributed, generated white noise within a 512 element frame.

For an epoch of white noise, the non-overlapping window version of VFD output was between 1.955 and 1.995, calculated within an absolute error of –0.0241 and absolute error –4.15%, which shows the algorithm is correct. The VFDT of this noise is shown in Fig. G.2.

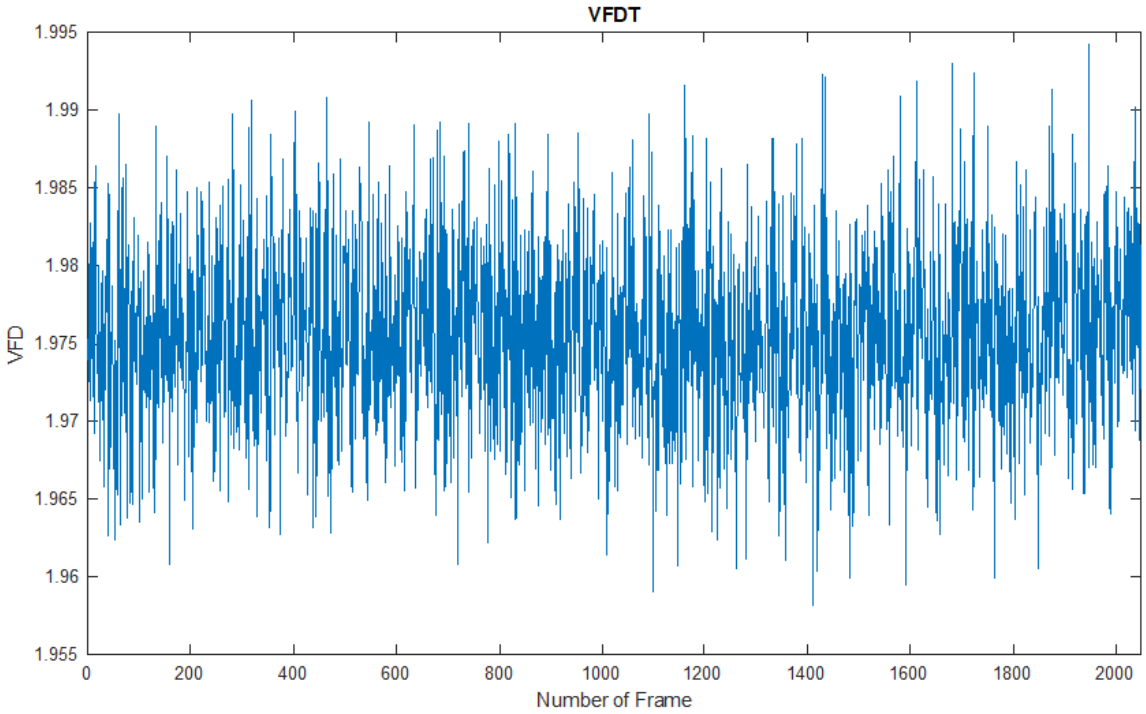



**Fig. G.2:** The uniform distribution white noise signal's VFDT with zero overlapping for the epoch.