Deep Neural Networks for Network Intrusion Detection

Shiyi Yang

A thesis in fulfilment of the requirements for the degree of

Master of Philosophy

School of Computer Science and Engineering

Faculty of Engineering

The University of New South Wales

December 2021

THE UNIVERSITY OF NEW SOUTH WALES
Thesis/Dissertation Sheet

Surname or Family name: **Yang**

First name: **Shiyi** Other name/s:

Abbreviation for degree as given in the University calendar: **MPhil**

School: **School of Computer Science and Engineering** Faculty: **Faculty of Engineering**

Title: Deep Neural Networks for Network Intrusion Detection

Abstract

Networks have become an indispensable part of people's lives. With the rapid development of new technologies such as 5G and Internet of Things, people are increasingly dependent on networks, and the scale and complexity of networks are ever-growing. As a result, cyber threats are becoming more and more diverse, frequent and sophisticated, which imposes great threats to the massive networked society. The confidential information of the network users can be leaked; The integrity of data transferred over the network can be tampered; And the computing infrastructures connected to the network can be attacked. Therefore, network intrusion detection system (NIDS) plays a crucial role in offering the modern society a secure and reliable network communication environment.

Rule-based NIDSs are effective in identifying known cyber-attacks but ineffective for novel attacks, and hence are unable to cope with the ever-evolving threat landscape today. Machine learning (ML)-based NIDSs with intelligent and automated capabilities, on the other hand, can recognize both known and unknown attacks. Traditional ML-based designs achieve a high threat detection performance at the cost of a large number of false alarms, leading to alert fatigue. Advanced deep learning (DL)-based designs with deep neural networks can effectively mitigate this problem and accomplish better generalization capability than the traditional ML-based NIDSs. However, existing DL-based designs are not mature enough and there is still large room for improvement.

To tackle the above problems, in this thesis, we first propose a two-stage deep neural network architecture, DualNet, for network intrusion detection. DualNet is constructed with a general feature extraction stage and a crucial feature learning stage. It can effectively reuse the spatial-temporal features in accordance with their importance to facilitate the entire learning process and mitigate performance degradation problem occurred in deep learning. DualNet is evaluated on a traditional popular NSL-KDD dataset and a modern near-real-world UNSW-NB15 dataset, which shows a high detection accuracy that can be achieved by DualNet.

Based on DualNet, we then propose an enhanced design, EnsembleNet. EnsembleNet is a deep ensemble neural network model, which is built with a set of specially designed deep neural networks that are integrated by an aggregation algorithm. The model also has an alert-output enhancement design to facilitate security team's response to the intrusions and hence reduce security risks. EnsembleNet is evaluated on two modern datasets, a near-real-world UNSW-NB15 dataset and a more recent and comprehensive TON_IoT dataset, which shows that EnsembleNet has a high generalization capability.

Our evaluations on the UNSW-NB15 dataset that is close to the real-world network traffic demonstrate that DualNet and EnsembleNet outperform state-of-the-art ML-based designs by achieving higher threat detection performance while keeping lower false alarm rate, which also demonstrates that deep neural networks have great application potential in network intrusion detection.

Originality Statement

I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, or substantial proportions of material which have been accepted for the award of any other degree or diploma at UNSW or any other educational institution, except where due acknowledgement is made in the thesis. Any contribution made to the research by others, with whom I have worked at UNSW or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged.

Shiyi Yang
06 December, 2021

Copyright Statement

Shiyi Yang
06 December, 2021

Authenticity Statement

I certify that the Library deposit digital copy is a direct equivalent of the final officially approved version of my thesis.

Shiyi Yang
06 December, 2021

Thesis submission for the degree of Master of Philosophy

Thesis Title and Abstract | Declarations | Inclusion of Publications Statement | Corrected Thesis and Responses

ORIGINALITY STATEMENT

☑ I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, or substantial proportions of material which have been accepted for the award of any other degree or diploma at UNSW or any other educational institution, except where due acknowledgement is made in the thesis. Any contribution made to the research by others, with whom I have worked at UNSW or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged.

COPYRIGHT STATEMENT

☑ I hereby grant the University of New South Wales or its agents a non-exclusive licence to archive and to make available (including to members of the public) my thesis or dissertation in whole or part in the University libraries in all forms of media, now or here after known. I acknowledge that I retain all intellectual property rights which subsist in my thesis or dissertation, such as copyright and patent rights, subject to applicable law. I also retain the right to use all or part of my thesis or dissertation in future works (such as articles or books).

For any substantial portions of copyright material used in this thesis, written permission for use has been obtained, or the copyright material is removed from the final public version of the thesis.

AUTHENTICITY STATEMENT

☑ I certify that the Library deposit digital copy is a direct equivalent of the final officially approved version of my thesis.

Thesis Title and Abstract | Declarations | Inclusion of Publications Statement | Corrected Thesis and Responses

UNSW is supportive of candidates publishing their research results during their candidature as detailed in the UNSW Thesis Examination Procedure.

Publications can be used in the candidate's thesis in lieu of a Chapter provided:

- The candidate contributed **greater than 50%** of the content in the publication and are the "primary author", i.e. they were responsible primarily for the planning, execution and preparation of the work for publication.
- The candidate has obtained approval to include the publication in their thesis in lieu of a Chapter from their Supervisor and Postgraduate Coordinator.
- The publication is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in the thesis.

☑ The candidate has declared that **their thesis has publications - either published or submitted for publication - incorporated into it in lieu of a Chapter/s. Details of these publications are provided below.**.

Publication Details #1

Full Title:	DualNet: Locate Then Detect Effective Payload with Deep Attention Network
Authors:	Shiyi Yang, Peilun Wu, and Hui Guo
Journal or Book Name:	2021 IEEE Conference on Dependable and Secure Computing (DSC)
Volume/Page Numbers:	1-8
Date Accepted/Published:	2021
Status:	published
The Candidate's Contribution to the Work:	Shiyi Yang contributed greater than 80% of the content in the publication and is the primary author of this publication.
Location of the work in the thesis and/or how the work is incorporated in the thesis:	Chapter 3

Publication Details #2

Full Title:	Hunter in the Dark: Discover Anomalous Network Activity Using Deep Ensemble Network
Authors:	Shiyi Yang, Hui Guo, and Nour Moustafa
Journal or Book Name:	The 21st IEEE International Conference on Software Quality, Reliability, and Security (QRS)
Volume/Page Numbers:	
Date Accepted/Published:	2021
Status:	accepted
The Candidate's Contribution to the Work:	Shiyi Yang contributed greater than 80% of the content in the publication and is the primary author of this publication.
Location of the work in the thesis and/or how the work is incorporated in the thesis:	Chapter 4

Candidate's Declaration

I confirm that where I have used a publication in lieu of a chapter, the listed publication(s) above meet(s) the requirements to be included in the thesis. I also declare that I have complied with the Thesis Examination Procedure.

Abstract

Networks have become an indispensable part of people's lives. With the rapid development of new technologies such as 5G and Internet of Things, people are increasingly dependent on networks, and the scale and complexity of networks are ever-growing. As a result, cyber threats are becoming more and more diverse, frequent and sophisticated, which imposes great threats to the massive networked society. The confidential information of the network users can be leaked; The integrity of data transferred over the network can be tampered; And the computing infrastructures connected to the network can be attacked. Therefore, network intrusion detection system (NIDS) plays a crucial role in offering the modern society a secure and reliable network communication environment.

Rule-based NIDSs are effective in identifying known cyber-attacks but ineffective for novel attacks, and hence are unable to cope with the ever-evolving threat landscape today. Machine learning (ML)-based NIDSs with intelligent and automated capabilities, on the other hand, can recognize both known and unknown attacks. Traditional ML-based designs achieve a high threat detection performance at the cost of a large number of false alarms, leading to alert fatigue. Advanced deep learning (DL)-based designs with deep neural networks can effectively mitigate this problem and accomplish better generalization capability than the traditional ML-based NIDSs. However, existing DL-based designs are not mature enough and there is still large room for improvement.

To tackle the above problems, in this thesis, we first propose a two-stage deep neural network architecture, DualNet, for network intrusion detection. DualNet is constructed with a general feature extraction stage and a crucial feature learning stage. It can effectively reuse the spatial-temporal features in accordance with their importance to facilitate the entire learning process and mitigate performance degradation problem occurred in deep learning. DualNet is evaluated on a traditional popular NSL-KDD dataset and a modern near-real-world UNSW-NB15 dataset, which shows a high detection accuracy that can be achieved by DualNet.

Based on DualNet, we then propose an enhanced design, EnsembleNet. EnsembleNet is a deep ensemble neural network model, which is built with a set of specially designed deep neural networks that are integrated by an aggregation algorithm. The model also has an alert-output enhancement design to facilitate security team's response to the intrusions and hence reduce security risks. EnsembleNet is evaluated on two modern datasets, a

near-real-world UNSW-NB15 dataset and a more recent and comprehensive TON_IoT dataset, which shows that EnsembleNet has a high generalization capability.

Our evaluations on the UNSW-NB15 dataset that is close to the real-world network traffic demonstrate that DualNet and EnsembleNet outperform state-of-the-art ML-based designs by achieving higher threat detection performance while keeping lower false alarm rate, which also demonstrates that deep neural networks have great application potential in network intrusion detection.

Acknowledgement

I wish to express my deepest gratitude to my supervisor, Dr. Hui Guo, for her excellent guidance, especially in the rigor of research and the logic of writing. I am also grateful to Dr. Nour Moustafa of UNSW Canberra and Peilun Wu of Sangfor Technologies Inc.. Their contributions to the field of network intrusion detection laid a good foundation for my research and they gave me many constructive suggestions. Moreover, many thanks to my colleague, Brian Udugama of UNSW Sydney, who always gives me some useful feedbacks both in study and in life. Last but not least, I thank my parents, relatives and all my friends for their support and help all the time.

Publications

List of Publications

This thesis has led to the following first author conference publications and they are included in lieu of chapters.

- **Shiyi Yang**, Hui Guo, and Nour Moustafa, "Hunter in the Dark: Discover Anomalous Network Activity Using Deep Ensemble Network", the 21st IEEE International Conference on Software Quality, Reliability, and Security (QRS), 2021.
- **Shiyi Yang**, Peilun Wu, and Hui Guo, "DualNet: Locate Then Detect Effective Payload with Deep Attention Network", IEEE Conference on Dependable and Secure Computing (DSC), 2021.
- Peilun Wu, Nour Moustafa, **Shiyi Yang**, and Hui Guo, "Densely Connected Residual Network for Attack Recognition", IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), 2020.

Contents

List of Figures

List of Tables

Abbreviations

k-NN	k-Nearest Neighbor
ACC	Accuracy
ACSC	Australian Cyber Security Centre
AdaBoost	Adaptive Boosting
Adam	Adaptive Moment Estimation
AI	Artificial Intelligence
API	Application Program Interface
APT	Advanced Persistent Threat
ATT&CK	Adversarial Tactics, Techniques, and Common Knowledge
BiLSTM	Bidirectional Long Short-Term Memory
BN	Batch Normalization
C&C	Command & Control
CDOF	Cost Distribution-based Outlier Factor
CLSID	Class Identifier
CNN	Convolutional Neural Network
ConvNet	Primitive Convolutional Neural Network
COVID	Corona Virus Disease
CVE	Common Vulnerabilities and Exposures
DARPA	Defense Advanced Research Projects Agency
DDoS	Distributed Denial of Service
DenseBlk	Dense Block

DenseNet	Dense Neural Network
DL	Deep Learning
DMZ	Demilitarized Zone
DNN	Deep Neural Network
DNS	Domain Name System
DoS	Denial of Service
DR	Detection Rate
DSC	Depthwise Separable Convolutional Neural Network
DT	Decision Tree
FAR	False Alarm Rate
FBI	Federal Bureau of Investigation
Fintech	Financial Technology
FN	False Negative
FP	False Positive
GAP	Global Average Pooling
GPU	Graphic Processing Unit
GRU	Gated Recurrent Unit
HTML	Hypertext Markup Language
HTTP	Hypertext Transport Protocol
IBM	International Business Machine
IIoT	Industrial Internet of Things
IoT	Internet of Things
IP	Internet Protocol
LAN	Local Area Network
LB	Linear Bridging
LDAP	Lightweight Directory Access Protocol
LOF	Local Outlier Factor
LSTM	Long Short-Term Memory

MITM	Man-In-The-Middle
ML	Machine Learning
MLP	Multilayer Perceptron
MP	Max Pooling
MSIE	Microsoft Internet Explorer
NB	Naive Bayes
NIDS	Network Intrusion Detection System
OS	Operating System
OSI	Open Systems Interconnection
OWASP	Open Web Application Security Project
PHP	Professional Hypertext Preprocessor
PlainBlk	Plain Block
PRE	Precision
Probe	Probing
R2L	Remote to Local
RAM	Random Access Memory
RBF	Radial Basis Function
ReLU	Rectified Linear Unit
ResBlk	Residual Block
ResNet	Residual Neural Network
RF	Random Forest
RMSprop	Root Mean Square Propagation
RNN	Recurrent Neural Network
RPC	Remote Procedure Call
SIEM	Security Information and Event Management
SNMP	Simple Network Management Protocol
SP	Service Provider
SQL	Structured Query Language

SSL	Secure Sockets Layer
STSD	Sample Table Sample Descriptor
SVM	Support Vector Machine
Tanh	Hyperbolic Tangent
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TN	True Negative
TP	True Positive
TPU	Tensor Processing Unit
TTPs	Tactics, Techniques and Processes
U2R	User to Root
UDP	User Datagram Protocol
UNSW	University of New South Wales
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
WAN	Wide Area Network
XML	Extensible Markup Language
XSS	Cross-site Scripting

Chapter 1

Introduction

The global COVID-19 pandemic has shaped the cyber security posture of organizations in unprecedented ways. Since the pandemic began, the FBI reported a 300% increase in reported cyber-crimes [1,2]. With the urgent need to enable continuity of business through remote working and cloud services, there are new opportunities for hackers to exploit vulnerable employees' devices and networks. IBM noted that remote work has increased the average cost of a data breach by $137,000 [3]. According to Fintech News, cloud-based cyber-attacks rose 630% between January and April 2020 [4]. In addition, with the rapid development and popularization of new technologies such as 5G and Internet of Things (IoT), systems and devices are increasingly interconnected and hence the cyber attack surface is ever-expanding [5] – more and more exploitable vulnerabilities (e.g., zero-day) are emerging for hackers to launch attacks [6]. As a result, cyber-attacks are becoming more and more diverse, frequent and sophisticated.

Network intrusion detection system (NIDS) plays an important role in protecting against an ever-evolving threat landscape today and safeguarding secure sharing of online resources as well as reliable transmission of information for end-users and organizations.

Rule-based NIDS, due to its stability and dependability, has been widely used in practical network environments. The system identifies threats by matching attack patterns against

a pre-defined knowledge base of manually crafted expert rules, making it very effective to identify known threats. However, it is nearly impossible for the system to recognize novel threats due to the rules containing overly detailed descriptions of known attacks. The system also requires frequent and manual updating of the rule library by security experts, which is laborious and time-consuming. It is obvious that such a detection system is inadequate to cope with the current networked society in which the volume, frequency and complexity of unknown attacks are increasingly growing [7].

Artificial intelligence (AI) is helping cyber threat defense communities stay ahead of attacks. Security teams take advantage of the availability of massive network data and the intelligence of machine learning (ML) to develop a NIDS that can automatically learn attack behaviors from the network traffic and hence discover not only previously seen and known but also unseen and unknown cyber threats[1] [8].

1.1 Problem Statement

While ML-based NIDSs have a high capability of novel threat perception, existing designs achieve a high attack detection performance often at the cost of a large number of false alarms. Traditional ML methods [9], due to their limited scalability to the large network traffic, treat most new and unanticipated behaviors as anomalies even if some of them are legitimate traffic [10]. The high level of false positive predictions would cause alert fatigue and likely make real threats miss in the noise of false alarms and fail to get timely attention from the security team. In addition, these methods rely on handcrafted feature engineering that is a labor-intensive work to achieve good performance.

The advanced deep learning (DL) approaches with deep neural networks (DNNs) can effectively mitigate the aforementioned problems. DNN with multiple learning layers can self-learn features at various levels of abstraction from a large amount of raw network

[1] An unseen cyber threat is a threat from a category that the model has learned, but has a different attack behavior when compared to the learned threat. It is different from unknown types of network intrusion, which the model has never learned.

traffic in an end-to-end training manner, thus accomplishing much better generalization performance – the adaptability to previously unseen data – than the traditional ML methods [11]. However, existing DL-based designs are still not mature enough [12] – the attack recognition capability still needs to be improved and the false alarm rate still cannot be ignored, which leaves room for improved design solutions.

1.2 Contributions

The thesis contributions are summarized as follows.

- To achieve a high attack recognition capability while keeping a low false alarm rate, we propose a deep neural network architecture, DualNet, that consists of two asynchronous stages: 1) a general feature extraction stage to maximally capture spatial and temporal features by densely connecting a series of convolutional neural network (CNN) and recurrent neural network (RNN) subnets, and 2) a crucial feature learning stage to improve the detection efficiency by targeting important features for the final learning outcome through the self-attention mechanism.

  DualNet is evaluated on two datasets, a traditional popular dataset, NSL-KDD [13], and a modern near-real-world dataset, UNSW-NB15 [14]. It has been compared with a set of existing traditional ML-based and advanced DL-based designs. Our evaluation shows that DualNet outperforms those existing designs by achieving higher threat detection performance while maintaining lower false alarm rate.

- To improve the generalization capability and enhance the understandability of detection results, we propose a deep ensemble neural network-based defense mechanism, EnsembleNet. EnsembleNet consists of a set of different DNNs and a specially designed integration algorithm: 1) each DNN is built with the CNN and RNN subnets that are connected in a way such that features learned by the subnets can be reused for good detection performance, and 2) an algorithm that efficiently integrates the detection results from the DNNs to further improve the detection performance. For

the detection results to be useful to the security team, EnsembleNet is empowered with an alert-output enhancement design. The design restores the threats (detected by the neural network) to their human-understandable raw traffic format and produces alerts of the current threats in the order of their severity so that the security team can make prompt responses and hence maximally reduce the security risk.

EnsembleNet is evaluated on two modern datasets, a near-real-world dataset, UNSW-NB15, and a more recent and comprehensive dataset, TON_IoT [15]. It has been compared with some classical ML-based designs, state-of-the-art DL-based designs and DualNet. The experimental results demonstrate that EnsembleNet outperforms those existing designs and is able to achieve a high threat detection accuracy while keeping a low false positive rate.

1.3 Thesis Organization

The rest of the thesis is organized as follows.

In Section 2.1 of Chapter 2, we introduce typical cyber-attacks in today's ever-changing threat environment and we describe the general workflow of network intrusion detection systems in Section 2.2. In Section 2.3 and Section 2.4, we review the representative work related to rule-based and ML-based network intrusion detection systems that can be found in the literature.

In Section 3.1 of Chapter 3, we briefly introduce DualNet. Its two-stage design is elaborated in Section 3.2. The evaluation of DualNet performance is detailed in Section 3.3. In Section 3.4, we provide a brief summary of Chapter 3.

In Section 4.1 of Chapter 4, we provide an overview of EnsembleNet. Its design is detailed in Section 4.2, where a set of deep neural networks and an integration algorithm are discussed. A design for user-friendly threat alerts is also introduced in this section. The evaluation of EnsembleNet performance is given in Section 4.3 and Chapter 4 is briefly concluded in Section 4.4.

In Chapter 5, we summarize the thesis and discuss some possible future research works and potential future research directions in Section 5.1.

Chapter 2

Background and Related Work

In this chapter, we first briefly introduce typical cyber-attacks and then describe the general workflow of network intrusion detection systems. Finally, we review the representative work of network intrusion detection systems that can be found in the literature.

2.1 Attack Categorization

Cyber threats can be considered as malicious activities that breach the *confidentiality, availability or integrity* of a system or the information contained within a system. Cyber threat actors often take advantage of technological vulnerabilities that exist in networks, endpoints, databases, servers, applications, websites, to name a few, or manipulate human emotions and psychology to launch crafted attacks in order to achieve their specific objectives. Typical attacks are described below.

2.1.1 Reconnaissance

In a reconnaissance attack, hackers use a set of methods, such as probing the network or through social engineering and physical surveillance, to covertly harvest as much in-

formation as possible about the victim. Such efforts include gathering general knowledge by utilizing port scanning, ping sweeping, packet sniffing, phishing, domain name system (DNS) queries, search engine retrieval and public website browsing [16].

2.1.2 Denial of Service and Distributed Denial of Service

Denial of service (DoS) attack floods the target with traffic to deplete available resources of the network or system, thereby preventing authorized users from accessing the resources. In contrast to DoS attack, in which the traffic pouring into the victim originates from the only one source, distributed denial of service (DDoS) attack requires multiple different sources – normally deploying Botnets [17] – to send excessive requests to the victim.

2.1.3 Phishing and Spear-Phishing

Phishing [18] is a type of social engineering attack usually aimed at stealing confidential information. Cyber adversaries often masquerade as trusted entities and trick victims into opening malicious links or attachments via fraudulent emails, websites, or text messages, resulting in the leakage of user credentials or the installation of malwares. Unlike phishing campaigns which are generic and commonly sent out in thousands [6], spear-phishing is a well-designed and pointed method of phishing that targets specific individuals or organizations [19]. The current typical example is COVID-19-related phishing [20], which exploits public fears about the sometimes-deadly virus.

2.1.4 Malwares

Trojan, Worm, Backdoor and Ransomware are common types of malware [21].

Trojan is a kind of malware disguised as legitimate software. It is often used to take control of the victims' computers for certain malicious purposes, such as stealing sensitive data. Unlike Computer Viruses, Trojans cannot replicate themselves. Trojan can infect

devices through social engineering, drive-by download, unauthorized access and other ways. Trojan is very covert and deceptive and is difficult to be discovered.

Worm is a stand-alone malware that can self-replicate recursively and spread itself to other computers via the network without any human intervention. In contrast to Computer Viruses that are often designed to damage the data and destroy devices, Worms are usually aimed at consuming network bandwidths and slowing down computer systems.

Backdoor is a type of malware in which hackers bypass normal authentication and security controls by spoofing and properly hiding to obtain stealthily access to the target system. Backdoor is subtly designed and is usually hidden in another form of software, such as application patches, program updates and file converters. Trojans are frequently used to create Backdoors on the target.

Ransomware is an ever-evolving form of malware that encrypts a victim's folders and files, rendering the systems that rely on those folders and files unusable. By using the malware, cybercriminals then demand a certain amount of ransom in exchange for decryption, often in the form of untraceable cryptocurrencies such as Bitcoin. According to the ACSC's Annual Cyber Threat Report July 2019 to June 2020, ransomware has become one of the most significant threats given the potential impact on the operations of governments and businesses [6].

2.1.5 Injection

Injection attack is ranked first on the Open Web Application Security Project's (OWASP) list of the top 10 most critical security risks to Web applications [22]. Scanners and fuzzers can help attackers find injection flaws. Attackers send commands or queries with embedded malicious statements to an interpreter by exploiting injection flaws, which may cause valuable data to be disclosed, tampered with, or corrupted. Typical injection attacks are Structured Query Language (SQL) injection, Operating System (OS) injection and Lightweight Directory Access Protocol (LDAP) injection.

2.1.6 Advanced Persistent Threat

Advanced persistent threats (APTs) [23] are remarkably subtle, sophisticated and long-term targeted intrusions, typically resulting in the theft of large amounts of sensitive data. APT actors target organizations across a wide variety of industries, often lurk illicitly in the target network for a prolonged period of time, many weeks, months or even years, to extend their reach before initiating devastating attacks [24]. APTs usually conform to the cyber kill chain framework [24,25] that enhances the visibility of attacks and enriches security analysts' understanding of adversarial Tactics, Techniques and Processes (TTPs). The publicly available TTPs can be found in the MITRE ATT&CK knowledge base [26].

The framework reveals seven steps that adversaries must complete in order to accomplish their goals: Reconnaissance, Weaponization, Delivery, Exploitation, Installation, Command & Control and Actions on Objectives. A typical intrusion scenario is that intruders perform reconnaissance to probe for exploitable vulnerabilities in the target network, build deliverable payloads (e.g., Microsoft Office files) by coupling tailored shellcodes with backdoors, send weaponized bundles to the victim via attachments of phishing emails, exploit the vulnerabilities to execute malicious codes on the target system, install malwares on the system such as backdoors that allow persistent access, "Hands on Keyboard" to continuously manipulate the compromised machine and finally achieve their objectives such as data exfiltration or encryption for ransom. Hence, each individual attack (e.g., from Section 2.1.1 to Section 2.1.5) often belongs to one of steps of the kill chain.

2.2 Network Intrusion Detection System

Intrusion detection systems have been studied for years and many detection systems have been developed and they have played a vital role in protecting organizations and end-users from cyber-attacks. The early detection systems identify anomalous behaviors on the OS kernel-level audit data [27] (i.e., Host-based Intrusion Detection Systems). With the increased connectivity of computer systems and hence rapid expansion of heteroge-

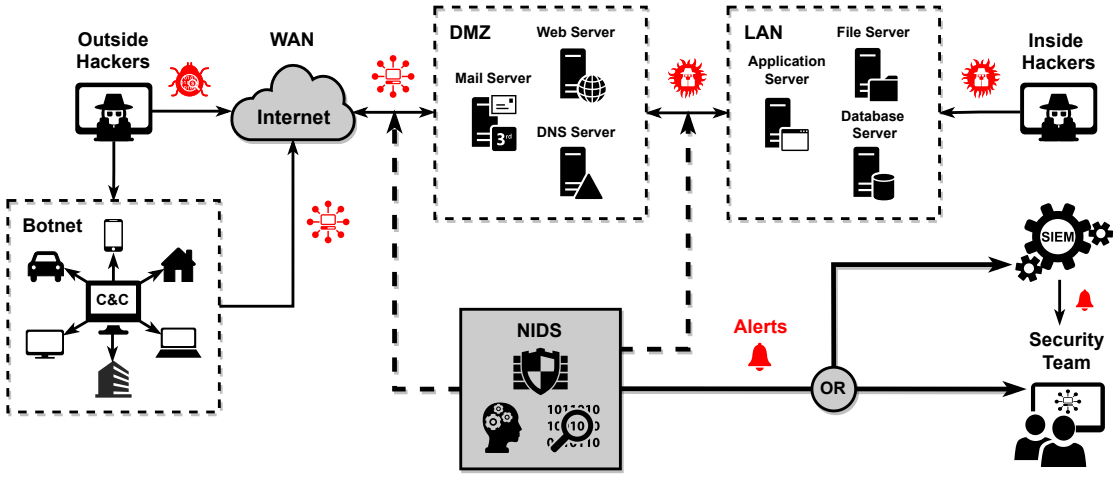


Figure 2.1: A Typical Setting of Network Intrusion Detection

neous computer networks, hackers take the network as the main carrier to launch attacks. Therefore, intrusion detection systems have been gradually expanded to cover abnormal activities on the network flow-level traffic data [28] (i.e., Network-based Intrusion Detection Systems).

Fig. 2.1 shows a typical setting of network intrusion detection. A NIDS is deployed at multiple strategic points within a network to continuously monitor the inbound and outbound network traffic of all devices connected to the network. It generates alerts of any perceived suspicious activities or policy violations and either reports them to the security team for due responses (e.g., threat investigation, containment and elimination) or collects them centrally using the security information and event management (SIEM) (e.g., Splunk [29]) – a system that provides real-time analysis of security alerts produced by multiple sources [30], as demonstrated in the figure. SIEM correlates the related security events to integrate alerts and then sends the alerts to the security team to take further action against the attacks.

Two kinds of NIDS that are currently used in the industry are rule-based and anomaly-based (i.e., ML-based) [31]. Their related work is described in the following two sections.

2.3 Rule-Based Network Intrusion Detection System

Early solutions to network intrusion detection are rule-based. The rule-based NIDS discovers threats by matching attack signatures or patterns against a pre-defined blacklist, which is effective to identify known attacks and often produces low false alarms. Due to its stability and dependability, the NIDS has been widely used in real-world business environments. Snort[1] and Zeek[2] are open-source NIDSs that can be found in the literature. These tools monitor traffic flows, especially checking for some specific features, such as a certain protocol or suspicious IP addresses or a byte pattern existing in packet payloads like some URI or USER-AGENT that may indicate some malicious activities. Once these features match their well-defined rules, an alarm is triggered. A short introduction and comparison of the two NIDSs are given below.

Snort [32] is a lightweight command-line tool for network intrusion detection created in 1998 by Martin Roesch and now developed by Cisco. It is fairly easy to use and is capable of running on Linux, Unix and Windows operating systems to perform real-time traffic analysis. Snort can detect probes and various attacks, such as operating system fingerprinting attempts, common gateway interface scans and buffer overflows. The tool would take specific actions against detected abnormal behaviors. The drawback of the command-line tool is the lack of a real graphical user interface.

Zeek [33], formerly known as Bro, was developed in 1994 by Vern Paxson. The tool is used to monitor traffic streams in real time and produce higher-level event logs in NetFlow-like format that record everything it understands from the network activities including normal traffic metadata and weird behaviors. It is more flexible than Snort in that it adds a programmatic interface, allowing users to customize traffic analysis according to different network environments. Furthermore, it can run on the application layer, making it possible to track different services from different OSI layers, such as HTTP, SNMP and DNS. A comparative study of the two systems [34] using a specific set of rules under

[1]Snort: https://www.snort.org

[2]Zeek: https://www.zeek.org

various attacks showed that Zeek was superior to Snort.

Six research groups participated in the 1998 DARPA off-line intrusion detection evaluation [35] based on more than 300 instances of 38 different attacks including probe, remote to local (R2L), user to root (U2R) and DoS attacks. The experimental results showed that rule-based systems with the best overall performance were effective to detect known attacks but failed to identify most (roughly half) of unknown attacks. The main reason is that hand-designed attack signatures contain excessively detailed descriptions of known attacks, making it almost impossible for such a NIDS to discover novel threats. Another disadvantage of the rule-based NIDS is that it requires the security team to frequently and manually update the signature and rule database, which is tedious and time-consuming.

This off-line evaluation strongly suggested that further research directions should be directed towards developing new detection methods to recognize new attacks instead of extending existing rule-based detection approaches, due to the number, frequency and complexity of unknown (zero-day) attacks are ever-increasing. Therefore, the anomaly-based NIDS comes into play.

2.4 Anomaly-Based Network Intrusion Detection System

The anomaly-based NIDS leverages the available massive learning data and the heuristics generated from machine learning to create a model of trustworthy network activities; Any activities deviated from the model can be treated as threats, hence making it possible to detect new attacks. Two typical machine learning methods can be used in building such a detection model: unsupervised machine learning and supervised machine learning [36].

2.4.1 Unsupervised Machine Learning Methods

Unsupervised learning focuses on finding patterns, structures or knowledge from unlabeled data. Outlier detection and centroid-based clustering are two representative techniques

used in the early anomaly-based NIDSs. They are briefly described below.

Outlier detection. Prakobphol et al. developed a cost distribution-based outlier factor (CDOF) scheme [37] in 2008 to identify abnormal network traffic. Compared to local outlier factor (LOF) that calculates the local density deviation of the traffic flow with respect to their neighbors as the outlier factor, the CDOF method generates a stronger outlier factor by amplifying the expressive power of the connectivity outlier factor to distinguish anomalous flows from normal flows. The evaluations using KDD Cup 1999 dataset, which contains various intrusions simulated in a military network environment, showed that although CDOF had higher detection performance than LOF, they both achieved high attack detection rates at the expense of high false positive rates.

Centroid-based clustering. Meng et al. proposed a k-means-based NIDS [38] in 2009. The model divides the given traffic data into k homogeneous and well-separated clusters, and each network traffic record belongs to the cluster with the nearest mean (cluster centroid). The proposed method was evaluated on the KDD Cup 1999, and the experimental results demonstrated that the clustering method could effectively detect novel intrusions in real network connections. They also pointed out that k-means clustering has better flexibility for large datasets, but is sensitive to noisy data and relies on feature engineering that is a labor-intensive work to achieve good performance.

Unsupervised learning, as a preliminary attempt of using machine learning for network intrusion detection, shows good performance of unknown threat perception. Another advantage of adopting unsupervised learning is that it avoids the costly and time-consuming data labeling process, hence freeing up human and computing resources for other important tasks, such as threat investigation and digital forensics. As a result, unsupervised machine learning has flourished in the past decade, and some related methods have emerged and been used to construct anomaly-based NIDSs. The most typical one is the autoencoder, as introduced in short below.

Autoencoder. Choi et al. presented an autoencoder [39] for network intrusion detection in 2019. It is a type of artificial neural network model that uses unsupervised learning. The

model is composed of an encoder sub-model and a decoder sub-model. During the training process, the encoder is to compress the input traffic data into a latent space representation (dimensionality reduction), where meaningful features will be retained and insignificant features will be ignored, and the decoder is to learn to precisely reconstruct the input from the representation provided by the encoder. In the testing phase, the new network flow is fed into the trained model to output the reconstructed data. If the reconstruction error (the error between the original data and its low dimensional reconstruction) is higher than a certain threshold, the flow can be considered as abnormality, and vice versa. Their evaluation based on the NSL-KDD dataset (an improved version of the KDD Cup 1999 dataset) showed that autoencoder performed better than the k-means and outlier detection models by achieving an accuracy of 91.70%. However, the explain-ability of the detection results of this model is low, which is also the problem existing in other unsupervised learning-based models.

Although unsupervised learning methods are capable of detecting new attacks, supervised learning approaches are more suitable to practical implementations [40] and often are superior to unsupervised learning methods in terms of detection accuracy [41], which is in line with the general trends in the machine learning community. Therefore, most recent studies focus on developing supervised machine learning-based network intrusion detection systems.

2.4.2 Supervised Machine Learning Approaches

Supervised learning constructs a predictable profile with a set of well-tagged network traffic records. The supervised learning-based designs can be basically divided into two groups: classical machine learning-based and advanced deep learning-based.

2.4.2.1 Traditional Machine Learning Models

For network intrusion detection, the classical ML methods [9] can be further considered as two kinds: individual classifiers and ensemble classifiers. Among many individual classifiers, kernel-based support vector machine (SVM) and probability-based naive Bayes (NB) are two representative designs, which are briefly introduced below.

SVM. Bhavsar et al. developed a SVM-based NIDS [42] in 2013. The model uses a kernel trick, radial basis function (RBF) kernel, to implicitly map the input data to the high-dimensional feature space, so as to find the optimal decision boundary, namely, the maximum margin hyperplane for data classification. SVM has regularization capability, which can prevent it from over-fitting. However, it takes a long training time on large datasets due to a large number of support vectors are generated. The experiments on the NSL-KDD showed that SVM was able to detect novel attacks with high detection accuracy.

NB. Gumus et al. proposed a NB [43] for network intrusion detection in 2014. The model is a special case of Bayesian network, which makes classification by applying Bayes' Theorem and assuming that features in the data are mutually independent. Their evaluations showed that NB was easy to implement and was time efficient, as compared to k-nearest neighbor (k-NN) – a classifier assigns the traffic flow to the most common category of its k nearest neighbors – on the KDD Cup 1999 dataset. However, the potential problem is that for a network flow, features are not completely independent, which may slightly limit the detection capability of NB.

Ensemble classifier leverages an integration scheme to integrate several weak learners into a stronger learner in order to improve generalizability and robustness over a single learner. Decision tree (DT) [9] is a tree-like structure model that generates classification rules employing information gain, which is highly interpretable but prone to over-fitting. Adaptive boosting (AdaBoost) [44] and random forest (RF) [45] are two typical ensemble classifiers. They use DT as the base classifier, but adopt different aggregation strategies to enhance the detection performance of individual base classifiers. AdaBoost and RF are described

in short below.

AdaBoost. Gudadhe et al. generated an AdaBoost-based NIDS [44] in 2010. The model consists of many decision trees and is based on Boosting aggregation algorithm. The trees are created sequentially and errors of each tree affect the weights of the training data for the next tree. For testing, depending on the performance, each tree obtains a different weight in the final classification decision (weighted majority voting). AdaBoost decreases the bias and hence is more generalizable, but it lacks the explain-ability. The evaluations on the KDD Cup 1999 dataset showed that the proposed method achieved higher generalization accuracy than the k-NN classifier.

RF. Farnaaz et al. developed a RF [45] to detect network intrusions in 2016. The model is also composed of multiple decision trees but is based on Bagging aggregation algorithm, in which the trees are independently generated from subsets of the dataset and their predictions are combined by the majority voting. RF overcomes the over-fitting problem of decision trees and reduces the variance, thus improving the accuracy. However, compared with decision trees, the interpret-ability of the model is reduced. The experimental results demonstrated that RF was more accurate than decision trees in identifying DoS, Probe, R2L and U2R attacks, and produced lower false alarm rates.

As the scale and complexity of network traffic increase rapidly in recent years, the traditional machine learning techniques have suffered from the performance optimization bottleneck, due to so-called "the curse of dimensionality" issue [46]. Consequently, even though the classical machine learning-based design is able to discover new attacks, it achieves high detection rates at the cost of high false alarms [47], resulting in "threat alert fatigue" problem that currently plaguing the industry. Another shortcoming of traditional machine learning models is that they rely on feature engineering to obtain detectable and representative traffic patterns for good performance of learning algorithms, which is difficult and expensive in terms of time and expertise required. Deep learning offers a promising solution to the above problems.

2.4.2.2 Advanced Deep Learning Techniques

Deep learning is based on a neural network with a stack of learning layers as feature extractors. It performs intrusion detection tasks in an end-to-end manner, which can self-learn features from raw heavy network traffic and learn higher-level data representations from the learning results of previous layers, thus achieving a high detection accuracy. The typical detection models are based on multilayer perceptron (MLP), convolutional neural network (CNN), recurrent neural network (RNN) and hybrid algorithm (e.g., CNN+RNN). A short introduction of some state-of-the-art designs is given below.

MLP. Rosay et al. proposed a MLP [48] for network intrusion detection in 2020. The model is a kind of feed-forward artificial neural network with multiple layers and non-linear activation functions. Their evaluation, based on a network traffic dataset that can be representative of current network usage, showed that MLP outperformed several traditional machine learning techniques, such as k-NN classifier, with higher generalization capability. The disadvantage of the proposed model is that it has a fully connected structure, which incurs a large amount of computation and hence restricts its learning efficiency on heavy network traffic.

CNN and RNN are two paradigms that apply parameter sharing techniques to reduce computational costs and are more suitable to network intrusion detection.

CNN. Azizjon et al. developed a CNN-based NIDS [10] in 2020 to extract spatial-oriented features from network traffic. CNN uses weight-shared filters to conduct convolution calculations and produces translation equivariant responses known as feature maps, where the convolution performs element-wise multiplication and addition. The proposed model was compared with SVM and RF methods on a set of serial TCP/IP packets in a predetermined time range, and the experimental results demonstrated that CNN outperformed these methods in terms of detection accuracy and was effective to network intrusion detection.

In contrast to CNN, RNN establishes loop connections to capture the temporal-oriented

features in the traffic data. However, vanilla RNN fails to learn the long-term dependencies and suffers from the vanishing-gradient problem. To address these problems, long short-term memory (LSTM) [49] has been proposed.

LSTM. Boukhalfa et al. adopted LSTM [11] in 2020 to recognize network intrusions. The model is a special kind of RNN, which consists of a cell and three gating mechanisms: a forget gate, an input gate and an output gate. Through the protection and control of the cell state by the three gates, the cell can remember benign and anomalous behaviors in network traffic. As a result, the model has a great ability to differentiate between normal traffic and network intrusions. Their experiments on NSL-KDD dataset showed that LSTM was superior to SVM, k-NN and DT models with higher attack detection rate and lower false alarm rate.

The aforementioned neural network models are shallow. Intuitively, to unleash the learning potential of deep learning, the neural network is required to be deeper in order to learn all the intermediate features between the raw data and the high-level classification to improve its generalization capability. However, as the depth of the neural network increases, the neural network will suffer from "performance degradation" problem [47]. Feature reuse is an effective strategy to solve this problem by making features learned from the shallow layer available at deeper layers. Densely-ResNet is a state-of-the-art design that adopts this strategy to handle performance degradation in deep neural networks.

Densely-ResNet. Wu et al. developed a deep densely connected residual network [12] in 2020, which is constructed with several basic residual units, where each unit consists of two CNN-RNN subnets by wide connections. The model alleviates the vanishing-gradient problem and reduces feature loss. But with the increase in width, the computational cost of the model increases greatly. As a result, Densely-ResNet is very data-hungry, in the sense that it requires sufficient data to fully support its rich parameterization, which slightly limits its generalizability. The model was deployed at a detection surface that includes Cloud layers (Network), Fog layers (Windows and Linux) and Edge layers (IoT) for attack recognition, and accompanied by a module that uses timestamps to correlate alerts from the three computing layers to eliminate redundant alerts and hence mitigate

alarm fatigue issue.

The development of open-source neural network frameworks such as Keras[3], Tensorflow[4] and PyTorch[5] makes deep learning technologies highly flexible and easily scalable. Most of these tools are developed in a modular manner, making it easy to use, modify and consolidate their modules, thereby accelerating the building of powerful deep learning-based detection models and the deployment of these models into real-world network environments. Furthermore, the success of deep learning is partially attributed to the rapidly developing computational resources such as GPUs and TPUs, which greatly speed up the training process for constructing high-quality models.

Through the above review of the existing related works, we find that *supervised deep learning* is the most effective method to develop NIDSs adapted to the current networked society. As a summary, the reasons of using supervised deep learning for network intrusion detection are listed below.

- Rule-based NIDSs have been unable to cope with the current threat environment in which the number, frequency and sophistication of unknown attacks are constantly increasing. It is suggested that rule-based solutions should be replaced with anomaly detection-based NIDSs to enhance the capability of identifying previously unseen and unanticipated attacks, and to avoid the laborious and time-consuming updating process of the rule library.
- Machine learning is a technology to build the anomaly detection-based NIDS, which can be divided into unsupervised machine learning and supervised machine learning. Supervised machine learning-based designs often outperform unsupervised machine learning-based designs in terms of detection accuracy and are more suitable for practical applications. Hence, supervised machine learning is an optimal choice.

[3]Keras: https://keras.io

[4]Tensorflow: https://www.tensorflow.org

[5]PyTorch: https://pytorch.org

- With the ever-increasing scale and complexity of network traffic, the weaknesses of traditional machine learning methods have been gradually exposed. The traditional methods, due to their limited scalability to the large network traffic, achieve high attack detection rates at the cost of high false alarm rates, leading to alert fatigue, and they also rely on labor-intensive feature engineering to obtain good performance. Deep learning approaches can self-learn features from the raw heavy traffic data and accomplish better generalization performance than the traditional machine learning methods.

Deep learning has revolutionized many fields in recent years, from computer vision to natural language processing, but in the field of network intrusion detection, deep learning-based designs are largely at the research investigation stage. The learning potential of deep learning has not been fully unleashed yet. The attack detection capability in the existing designs still needs to be improved and the false alarms are still not ignorable. As such, we propose two designs using supervised deep learning, DualNet and EnsembleNet. DualNet is a deep neural network architecture that can reuse features to achieve a high detection performance, which is elaborated in Chapter 3. To improve the detection performance of DualNet, we introduce ensemble learning, which is to be described in detail in Chapter 4 (EnsembleNet).

Chapter 3

DualNet: A Deep Neural Network for Network Intrusion Detection

3.1 Introduction

As discussed in previous chapters, neural network-based models can detect both known and unknown threats in response to an ever-evolving threat environment and are superior to traditional machine learning-based designs with higher generalization performance.

It is intuitive that a deep neural network could have a much better detection accuracy than its shallower counterpart that possesses fewer learning layers, due to the higher-level features could be learned and hence the mapping relationship between the raw data and correct classification results could be enhanced. Unfortunately, as the neural network goes deeper, the problem of vanishing gradient occurs [50,51], which makes that the parameters of the initial layers of the neural network cannot be tuned properly [52] and hence the learning performance of the network decreases – specifically the detection accuracy tends to saturate and then declines rapidly. Such a problem is called "performance degradation" [47] and it greatly limits the learning potential of deep learning. The problem can also be observed in other deep neural network application areas, such as image recognition [53].

ResNet [54] and EfficientNet [55] have been proposed to address this issue though finding an optimal network architecture still remains a difficult problem.

To alleviate performance degradation problem to improve the generalization capability of neural networks, we propose a special learning method, densely connected learning, which can reuse features to enhance the detection accuracy when neural networks are deeply extended. We use this learning method to develop a specially designed deep dense neural network (DenseNet), which is the first stage of our design DualNet. And we leverage the self-attention mechanism to effectively locate then detect the most valuable payloads from network packets to further improve the detection accuracy, which is the second stage of DualNet. DualNet is elaborated in the next section.

3.2 DualNet

Our goal is to build a deep learning model that has a high detection capability (*model quality*) and is easy to train (*training efficiency*), and the trained model is small in size and incurs short execution time (*model cost*).

We consider that the model quality is closely related to the features extracted from the security data and how the extracted features are effectively used for the final prediction outcome. To this end, we propose a two-stage deep neural network architecture, DualNet: a *general feature extraction stage* to maximally capture spatial-temporal features from the network traffic records; and a *crucial feature learning stage* to focus more on important features to further improve the detection efficiency.

In terms of training efficiency and model cost, they are relevant to the number of trainable parameters, and a small trainable parameter number is desired. We, therefore, take this into account in our design.

An overview of our system is given in Fig. 3.1. The DualNet mainly performs two stages for attack recognition. The construction of two stages is elaborated in the next two sub

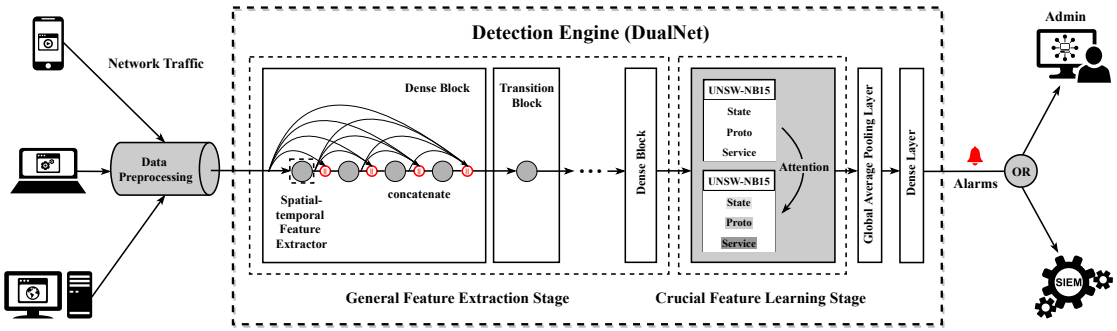


Figure 3.1: DualNet System Overview

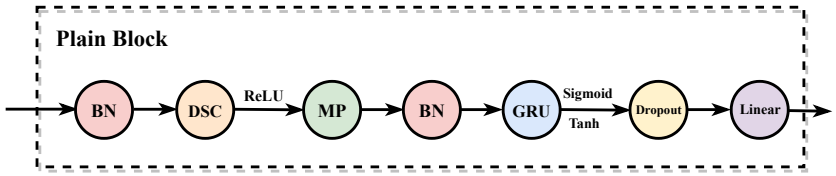


Figure 3.2: Plain Block of DualNet (Spatial-temporal Feature Extractor)

sections.

3.2.1 General Feature Extraction Stage

We consider that the multi-sourced security data has both spatial and temporal correlations. Hence, we present a special learning method, *densely connected learning*, which can learn as many spatial-oriented and temporal-oriented features at various levels of abstraction as possible from the input representations, and allow to build deeper neural network without performance degradation. The densely connected learning is to establish an interleaved arrangement pattern between two types of particularly designed blocks, *dense blocks* and *transition blocks*, where the number of dense blocks is one more than the number of transition blocks, as shown in Fig. 3.1. To construct dense blocks and transition blocks effectively, we introduce plain block as the basic building block.

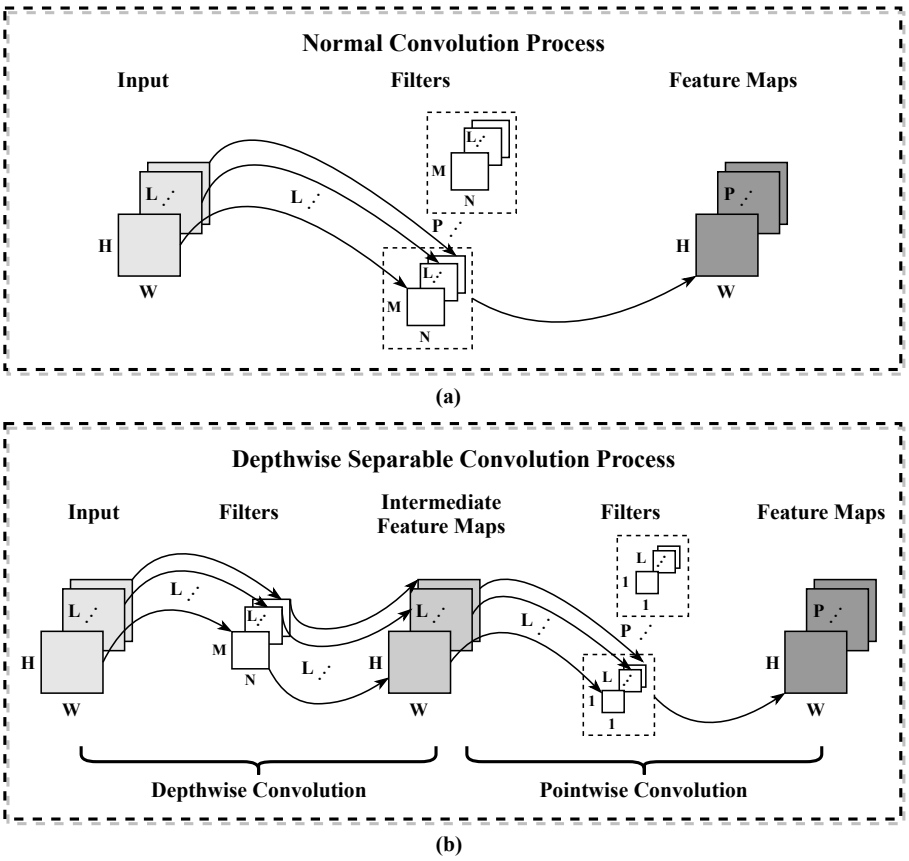


Figure 3.3: Comparison of Normal Convolution and Depthwise Separable Convolution

3.2.1.1 Plain Block of DualNet

The plain block is a seven-layer (four parameter layers) spatial-temporal feature extractor, as shown in Fig. 3.2. The block is built upon CNN and RNN subnets, where CNN is to extract the spatial-oriented features from the learning dataset and RNN is to capture the temporal-oriented features in the dataset.

Compared to primitive CNN (ConvNet) that uses the normal convolution process, *depthwise separable CNN (DSC)* [56] divides the whole convolution process into two simplified steps: depthwise convolutions and pointwise convolutions, as such the number of multiplications can be decreased and hence the number of trainable parameters can be reduced [57].

Fig. 3.3 demonstrates a comparison of the two convolution processes. Assume that the

shape of the input tensor is $H \times W \times L$, where H is height, W is width and L is depth (the number of input feature maps), the convolutional kernel size is $M \times N$, where M is height and N is width, and the expected number of output feature maps is P. To facilitate subsequent design, all convolutional operations adopt the same padding. As a result, the output size is the same as the input size, which is also $H \times W$.

For the normal convolution process, the number of kernels in each filter should be the same as the number of input feature maps, which is L, and the number of filters should be the same as the number of output feature maps, which is P. As can be seen from Fig. 3.3(a), each filter is convolved with the input, where each kernel is convolved with an input feature map, and then an output feature map is generated. Since weights are shared in the same filter, the trainable parameters of this convolution process can be calculated by $L \times M \times N \times P$. Fig. 3.3(b) shows the process of the depthwise separable convolution. In the depthwise convolution step, the number of filters should be the same as the number of input feature maps, which is L, and each filter has only one kernel to convolve with an input feature map to generate an intermediate feature map for the next step. So, the trainable parameters in this step are $M \times N \times L$. In the pointwise convolution step, it is similar to the normal convolution process, but the kernel size is 1×1. Hence, the trainable parameters in this step are $L \times P$. As a result, the total trainable parameters of the depthwise separable convolution process are $L \times (M \times N + P)$. For example, when $L = 196, M = 1, N = 10$ and $P = 196$ (These are hyper-parameter settings using the UNSW-NB15 dataset, which will be detailed in the next section), the trainable parameters of depthwise separable convolution are 40,376, while the trainable parameters of normal convolution are 384,160, which is about 10 times that of depthwise separable convolution.

As mentioned in Section 2.4.2.2, LSTM [49] with three gates solves the gradient vanishing problem and the inability of acquiring long-term dependencies in the vanilla RNN. *Gated recurrent unit (GRU) [58]* is a simplified LSTM with fewer number of gates and much lower trainable parameters.

GRU has two input tensors: one representing the current information is x_t and the other

representing the previous information is h_{t-1}, and its output tensor is h_t. Fig. 3.4 shows the GRU internal structure with two gating mechanisms: a reset gate r_t and an update gate u_t. The reset gate uses sigmoid activation function: $sigmoid(z) = \frac{1}{1+e^{-z}}$ to determine how much the previous information to be ignored. The output values range from 0 (completely discarded) to 1 (completely retained). The formula is given below.

$$r_t = sigmoid(W_r[h_{t-1}, x_t] + b_r), \tag{3.1}$$

where W_r and b_r are parameter matrix and parameter vector respectively.

The update gate decides what previous information to be forgot and what new information to be added. The output values are also compressed between 0 and 1, as specified below.

$$u_t = sigmoid(W_u[h_{t-1}, x_t] + b_u), \tag{3.2}$$

where W_u and b_u are parameter matrix and parameter vector respectively.

A hyperbolic tangent (Tanh) activation function: $tanh(z) = 2sigmoid(2z) - 1$ is used to scale the values of the tensor that combines the previous information adjusted by the reset gate with the current information in the range of -1 to 1 to create a tensor of new candidate values $\tilde{h}_t$, as shown by the following formula.

$$\tilde{h}_t = tanh(W_h[r_t * h_{t-1}, x_t] + b_h), \tag{3.3}$$

where W_h and b_h are parameter matrix and parameter vector respectively.

The previous and new current information are modulated by the update gate to form the output, as given below.

$$h_t = (1 - u_t) * h_{t-1} + u_t * \tilde{h}_t \tag{3.4}$$

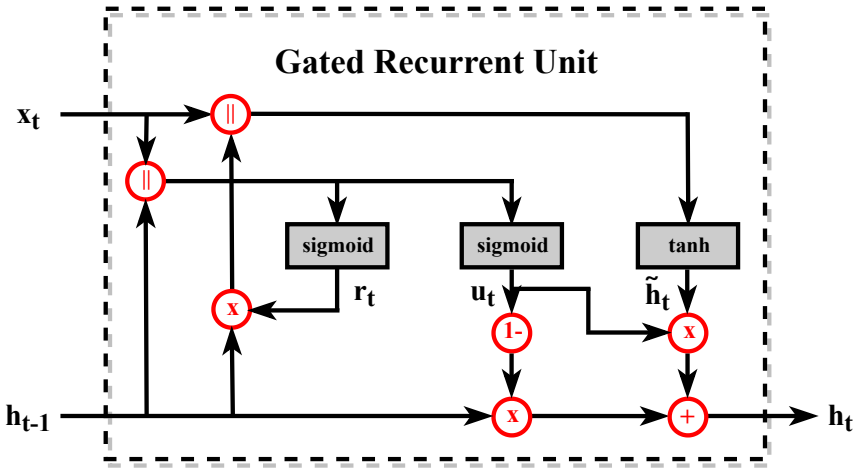


Figure 3.4: Gated Recurrent Unit

In consequence, the GRU can remember the normal patterns and anomalous patterns of network traffic.

As mentioned earlier, to efficiently leverage the feature extraction capability of both CNN and RNN for the security data and reduce the potentially high computational cost of our densely connected learning, we combine DSC with GRU to build the plain blocks, where DSC uses rectified linear unit (ReLU) [59]: $relu(z) = max(0, z)$ as the activation function to accelerate convergence rate. As can be seen from Fig. 3.2, apart from DSC and GRU subnets, we also add five layers (including two parameter layers) to further enhance the learning ability. We introduce *batch normalization (BN)* [60] to standardize the data of each mini-batch in training before feeding it into DSC and GRU to reduce the internal covariate shift, thus accelerating model fitting and decreasing the final generalization errors. In addition, there is a *max-pooling (MP)* layer after DSC to provide basic translation invariance for the internal representations and reduce the computational cost by down-sampling its inputs. And a regularizer *dropout* [61] after GRU is used to prevent overfitting and further decrease the computational cost by randomly removing several neurons. We also add a *linear bridging* layer [12] to transform non-linear parameter layers into a linear space, which is helpful for stabilizing the learning process.

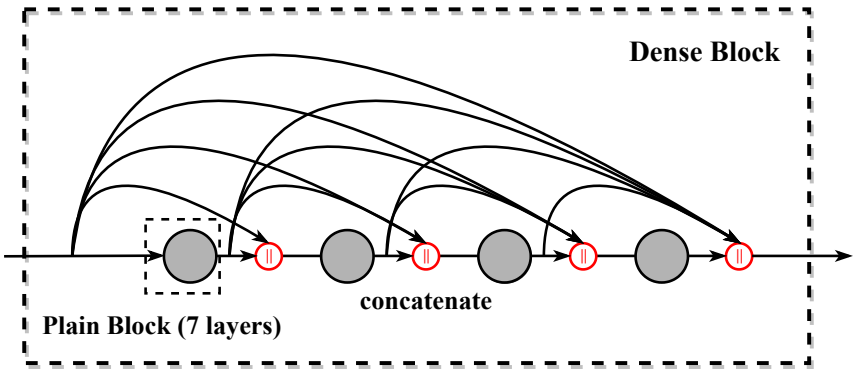


Figure 3.5: A Dense Block of DualNet with a Growth Rate of g = 4

3.2.1.2 Dense Block of DualNet

Dense block is formed with a set of densely connected plain blocks. Fig. 3.5 shows a dense block containing four plain blocks, where each plain block receives the concatenation of the input data and the output of all the preceding plain blocks through shortcut connections as its new inputs. The scale of the dense block increases with the plain blocks, and we define *growth rate* g as the number of plain blocks in the dense block.

The dense blocks encourage feature reuse and strengthen propagation of features and gradients within the network thanks to the dense connections[1].

3.2.1.3 Transition Block of DualNet

We can stack more dense blocks for a deeper neural network. To handle the potential decrease in accuracy due to the increased number of features (i.e., the dimensionality of feature space) when building deeper networks to fully learn the features at various levels of abstraction, we add a transition block between two dense blocks to reduce the dimensionality. Since the DSC subnet has strong down-sampling capability, we use it for the dimensionality reduction. DSC favors the spatial features. To maintain both spatial and temporal features during the dimensionality reduction, we also add GRU subnet to the

[1]The idea of building dense connections originated from densely connected convolutional networks [62], which have shown good performance for object recognition tasks.

transition block. As a result, the transition block has the same structure as the plain block presented before. Inserting the block between dense blocks limits the feature space grow, improves the generalization capability and robustness of the model and makes the model easy to train.

With the dense blocks and transition blocks, the first stage of DualNet can be formed as a very deep neural network for maximal extraction of general spatial-temporal features, as shown in Fig. 3.1. To further improve the detection capability, we present the second stage to focus attention on those features that are more important to the predicted results of the detection engine.

3.2.2 Crucial Feature Learning Stage

We apply a self-attention mechanism [63] to focus more on the crucial features that should be considered as the most effective payloads to distinguish attack from normal behavior.

In this stage, each feature would obtain an attention score. The higher the attention score is, the more important the feature is and the more influence it has on the prediction of the detection engine. The attention function can be described as mapping a query and a series of key-value pairs to an output [63] that is specified as below.

$$Attention = softmax(Similarity(Q, K))V, \tag{3.5}$$

where Q, K, V are the matrices of query, key, value respectively. The Similarity function performs dot-product calculation between the query and each key to obtain a weight: $w_i, i = 1, 2, ..., n$, which is much faster and more space-efficient in practice [63], that is, fewer trainable parameters are required. A softmax function [64]: $softmax(w_i) = \frac{e^{w_i}}{\sum_{j=1}^{n} e^{w_j}}$ is then applied to normalize these weights between 0 and 1, and finally the weights are combined with their corresponding values to obtain the final attention scores.

Self-attention mechanism can enhance the interpretability of captured features and hence

shrink the semantic gap between AI detectors and security analysts. Moreover, the mechanism can enable security analysts to identify important features of attacks to effectively respond to threats. By using the self-attention mechanism, our model can better memorize the long-term dependencies existed in the traffic flow records so that the gradient vanishing and performance degradation problems can be effectively mitigated, thereby achieving higher detection accuracy.

3.3 Evaluation and Discussion

Our evaluation is based on a cloud AI platform configured with a Tesla K80 GPU and a total of 12 GB of RAM. The designs are written in Python and built upon tensorflow backend with APIs of keras libraries and scikit-learn packages.

3.3.1 Datasets Selection

The training and testing of the designs are performed on two heterogeneous network intrusion detection datasets. One is a traditional well-known dataset, NSL-KDD [13], generated by Canadian Institute for Cyber Security in 2009, which is an improved version of the KDD Cup 1999 dataset [65]. The other is a more recent modern dataset, UNSW-NB15 [14], developed by Australian Cyber Security Centre in 2015, which is a more comprehensive representation of a contemporary low footprint attack environment. There are no duplicate network traffic records in the two datasets to ensure that the designs used in the evaluation do not favor more frequent records, and the designs with better detection rates for repetitive records will not bias their performance [13], [66]. In addition, both datasets are balanced with similar numbers of normal and abnormal records, and they do not contain missing values, which further ensures the effectiveness of the evaluation [13], [14]. Thus, NSL-KDD and UNSW-NB15 are often used as the baseline datasets for network intrusion detection related research [67–69].

The two cyber attack datasets are composed of two classes: normal and anomalous. In

Table 3.1: Ground Truth of UNSW-NB15 Dataset

Attack Category	**Attack References (Description)**
Generic	CVE-2005-0022, CVE-2006-3086, ...
Exploits	CVE-1999-0113, CVE-2000-0884, ...
Fuzzers	NULL (HTTP GET Request Invalid URI)
Reconnaissance	CVE-2001-1217, CVE-2002-0563, ...
DoS	CVE-2007-3734, CVE-2008-2001, ...
Shellcode	milw0rm-1308, milw0rm-1323, ...
Backdoors	CVE-2009-3548, CVE-2010-0557, ...
Analysis	NULL (IP Protocol Scan)
Worms	CVE-2004-0362, CVE-2005-1921, ...

NSL-KDD, the abnormal traffic includes 4 categories [13]: DoS, Probing (Probe), Remote to Local (R2L) and User to Root (U2R), where the attack samples were gathered based on a U.S. air force network environment. There are 148,516 processed traffic records with 41 features that can be used for training and testing from NSL-KDD. In UNSW-NB15, there are nine contemporary synthesized attack activities [14]: Generic, Exploit, Fuzzer, Reconnaissance, DoS, Shellcode, Backdoor, Analysis and Worm, which were collected from Common Vulnerabilities and Exposures[2], Symantec[3] and Microsoft Security Bulletin[4]. There are 257,673 processed traffic records with 42 features that can be used for training and testing from UNSW-NB15. It is worth noting that each attack event is simulated from a real-world attack scenario with a specific attack reference, as demonstrated in Table 3.1. The actual attack references used for our evaluation are based on this table but not limited to it, and they are in the range from CVE-1999-0015 to CVE-2014-6271. To perform a comprehensive evaluation, we use all of the attack types provided by the two datasets.

[2]CVE: https://cve.mitre.org/

[3]BID: https://www.securityfocus.com

[4]MSD: https://docs.microsoft.com/en-us/security-updates/securitybulletins

3.3.2 Data Preprocessing

Before training and testing, we pre-process the network traffic records in three phases, as given below.

3.3.2.1 Categorical Conversion

Since categorical data cannot be fed into neural networks directly, textual notations such as 'http' and 'smtp' are required to be converted to numerical form. Hence, we apply one-hot encoding [70] to encode nominal feature values into their dummy representations to expand the sparsity of the data to accelerate the training.

3.3.2.2 Random Shuffling

We randomly disrupt the order of the data records to prevent the selectivity of gradient optimization direction from severely declining caused by data regularity, hence reducing the tendency of overfitting and accelerating the convergence rate.

3.3.2.3 Data Normalization

Features in different dimensions cannot contribute equally to model fitting, which may give undue emphasis to the features with larger magnitudes and eventually result in biased predictions. Therefore, we use min-max normalization [70] to rebuild the values of each feature on a scale of 0 to 1 to maintain certain numerical comparability and improve the speed of backpropagation. The normalization is performed with the following formula.

$$X_{value} = \frac{X - X_{min}}{X_{max} - X_{min}}, \tag{3.6}$$

where X is the original feature value, X_{min} is the minimum value of the corresponding feature, X_{max} is the maximum value of the corresponding feature and X_{value} is the

normalized feature value.

3.3.3 Training and Testing

To investigate the effectiveness of our densely connected learning in handling performance degradation problem of deep neural networks and the effectiveness of self-attention mechanism for network intrusion detection, we create three DenseNets in different depths. They are briefly described below.

DenseNets. We apply our densely connected learning to establish these DenseNets. They are denoted as $Dense - n$, where n is the number of dense blocks used in a DenseNet, $n = 1, 2, and\ 3$. All dense blocks have a growth rate of 4. Each $Dense - n$ has n dense blocks along with $(n - 1)$ transition blocks in an interleaved arrangement pattern plus one global average pooling layer and one dense layer. Therefore, $Dense - 1$ has 31 layers including 19 parameter layers, $Dense - 2$ has 66 layers including 39 parameter layers, and $Dense - 3$ has 101 layers including 59 parameter layers.

DualNet is $Dense - 3$ enhanced with a self-attention mechanism.

3.3.3.1 Hyperparameter Settings

To ensure a fair comparison of those designs, uniform hyperparameter settings are applied to the training on each of the two datasets. For all designs, the number of filters in the convolutions and the number of recurrent units are adjusted to be consistent with the number of features in each dataset, where NSL-KDD has 122 features and UNSW-NB15 has 196 features after the data pre-processing (the number of features increases due to the one-hot encoding). Sparse categorical cross entropy loss function is used to calculate the errors, which avoids possible memory constraints incurred by the classification tasks with a large variety of labels. Adaptive moment estimation (Adam) algorithm is used as an optimizer to update trainable parameters to minimize the errors, which is especially

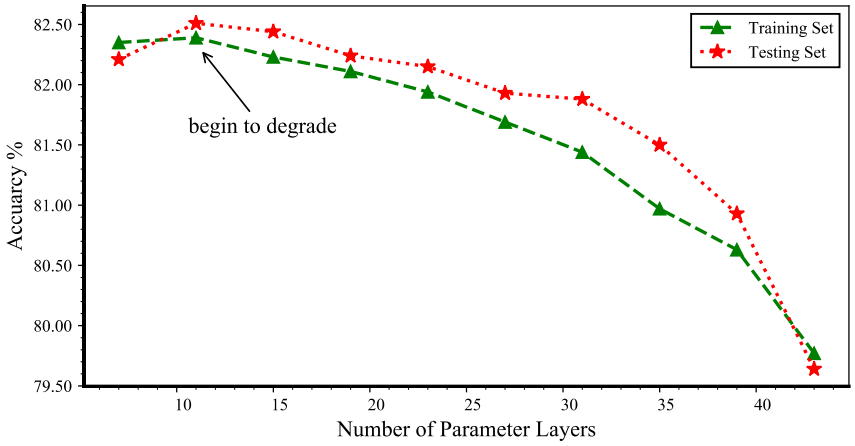


Figure 3.6: The Performance Degradation Problem of Deep Neural Network for Network Intrusion Detection on UNSW-NB15

effective for the sparse inputs [71] (the input becomes sparse after the one-hot encoding). The learning rate is set to 0.001 here.

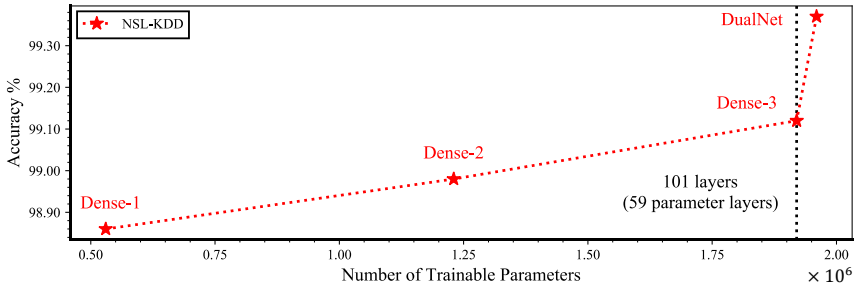
3.3.3.2 Stratified K-fold Cross Validation

We apply stratified k-fold cross validation to evaluate the generalization capability of designs. The method splits the entire dataset into k groups by preserving the same proportion of each class in original records, where k-1 groups are combined for training and the remaining one is used for testing. Here, k is set to 10 to keep non-computational advantage of bias-variance trade-off [72].

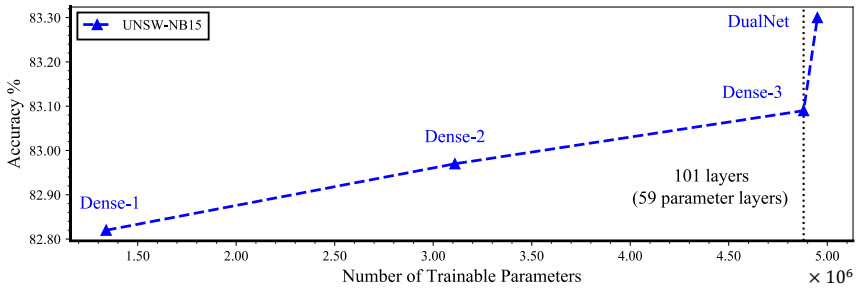
3.3.3.3 Evaluation Metrics

We use five metrics to evaluate the performance of the designs: accuracy (ACC), detection rate (DR), false alarm rate (FAR), precision (PRE) and F_1 score, as defined below.

$$ACC = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions}, \tag{3.7}$$



(a) The Detection Accuracy of Proposed Designs on NSL-KDD



(b) The Detection Accuracy of Proposed Designs on UNSW-NB15

Figure 3.7: The Detection Accuracy of Proposed Designs on Two Datasets

$$DR = \frac{TP}{TP + FN}, \tag{3.8}$$

$$FAR = \frac{FP}{FP + TN}, \tag{3.9}$$

$$PRE = \frac{TP}{TP + FP}, \tag{3.10}$$

$$F_1\ score = \frac{2 \times PRE \times DR}{PRE + DR}, \tag{3.11}$$

where TP and TN are, respectively, the number of attacks and the number of normal network traffic correctly categorized; FP is the number of actual normal traffic misclassified as attacks, and FN is the number of attacks incorrectly classified as normal network traffic.

3.3.4 DualNet Performance

We evaluate the generalization capability of DualNet by first evaluating the densely connected learning and the self-attention mechanism. To further evaluate the generalizability of DualNet, we then compare it with a set of existing designs.

3.3.4.1 Densely Connected Learning Performance

We stack plain blocks in a range from 1 to 10 to build some baseline comparison models to investigate the performance degradation problem in deeper neural networks for network intrusion detection. Fig. 3.6 shows the training and testing accuracy of the networks with different number of parameter layers on UNSW-NB15 dataset. As can be seen from the figure, with the increase of the network depth, the training and testing accuracy gets saturated at first and then decline rapidly as unexpected, namely, the performance gradually degrades. Fig. 3.7 displays the detection accuracy of three DenseNets and DualNet on two datasets. According to the figure, the accuracy of DenseNet improves with the increase of the network depth on two datasets (Dense-2 outperforms Dense-1 and Dense-3 outperforms Dense-2), which demonstrates our densely connected learning (feature reuse) can effectively handle performance degradation problem in building deeper neural networks for network intrusion detection.

3.3.4.2 Self-attention Mechanism Performance

As shown in Fig. 3.7, compared to Dense-3, DualNet presents a sharp increase in detection accuracy on two datasets, achieving 99.37% (Dense-3 achieved 99.12%) on NSL-KDD dataset and 83.30% (Dense-3 achieved 83.09%) on UNSW-NB15 dataset (multi-class classification), which demonstrates the efficiency and effectiveness of the self-attention mechanism for network intrusion detection. The self-attention mechanism helps to achieve a significant performance gain with fewer additional parameters when compared to the parameters required to improve performance from the Dense-2 to Dense-3 architecture.

Table 3.2: The Comparative Results on UNSW-NB15 Dataset

Design	**ACC %**		**DR %**	**FAR %**	**PRE %**	F_1 **Score**
	Multi	**Binary**				
RF	61.02	73.24	61.67	6.28	94.56	74.66
AdaBoost	67.67	87.46	93.31	22.90	87.83	90.48
SVM	72.86	85.78	81.76	7.11	95.32	88.02
GRU	79.22	92.94	93.02	7.22	95.80	94.39
MLP	79.27	92.57	91.58	5.67	96.62	94.03
LSTM	79.42	92.91	92.61	6.55	96.16	94.35
BiLSTM	79.43	93.06	93.90	8.42	95.18	94.53
ConvNet	79.66	93.03	93.11	7.11	95.87	94.47
DSC	80.16	93.45	92.95	5.66	96.68	94.78
DualNet	**83.30**	**94.58**	**94.46**	**5.20**	**96.98**	**95.71**

3.3.4.3 DualNet Detection Performance

DualNet is compared with a set of existing traditional machine learning designs: RF [73], AdaBoost [74] and SVM [75], and advanced deep learning designs: GRU [76], MLP [77], LSTM [78], bidirectional LSTM (BiLSTM) [79], ConvNet [80] and DSC [56] on the near-real-world dataset, UNSW-NB15. Table 3.2 shows ACC, DR, FAR, PRE and F_1 score of these designs. From the table, we can find that DualNet outperforms those designs by achieving higher ACC on both multi-class and binary classification tasks, higher DR, higher PRE, higher F_1 score and lower FAR – higher generalization performance. The comparative results further demonstrate the effectiveness of DualNet for network intrusion detection.

In addition to recognizing whether the network traffic record is normal or abnormal, DualNet can also identify a packet flow either as normal or as specific attacks. Table 3.3 shows ACC, DR, FAR, PRE and F_1 score of DualNet for the normal and each attack on two datasets. From the table, we can see that DualNet has a strong ability to recognize normal network traffic and most types of specific attacks (DoS, Probe, R2L, U2R, Generic, Exploit, Reconnaissance, Shellcode, Backdoor and Worm) with a high ACC, a high DR, a high PRE, a high F_1 score and a low FAR. Its performance on the Fuzzer attack is

Table 3.3: DualNet Performance for Each Category on Two Datasets

Datasets	**Category**	**ACC %**	**DR %**	**FAR %**	**PRE %**	F_1
NSL-KDD	Normal	99.41	99.48	0.67	99.38	99.43
	DoS	99.92	99.96	0.10	99.85	99.91
	Probe	99.71	98.93	0.14	99.22	99.07
	R2L	99.38	92.23	0.27	94.25	93.22
	U2R	99.97	91.30	0.00	100.00	95.45
UNSW-NB15	Normal	94.58	94.80	5.54	90.62	92.66
	Generic	99.98	99.98	0.02	99.97	99.97
	Exploit	98.98	97.69	0.41	99.13	98.40
	Fuzzer	89.82	66.49	4.61	77.48	71.56
	Reconnaissance	99.83	99.53	0.14	98.87	99.20
	DoS	99.80	88.11	0.01	99.21	93.33
	Shellcode	99.82	91.00	0.08	92.86	91.92
	Backdoor	99.98	92.00	0.00	100.00	95.83
	Analysis	99.53	19.23	0.00	100.00	32.26
	Worm	100.00	100.00	0.00	100.00	100.00

moderate, with a 66.49% DR. The case that especially needs to discuss is related to Analysis attacks. DualNet only achieves a DR of 19.23% on Analysis attacks. The main reason is inadequate relevant training data (only about 1% Analysis records in the UNSW-NB15 dataset), leading to insufficient learning and hence poor detection performance.

Overall, DualNet performs well in recognizing both normal and abnormal traffic flows. It can achieve 99.41% ACC, 99.33% DR, 99.44% PRE, 99.38 F_1 score while keeping 0.52% FAR on NSL-KDD dataset and 94.58% ACC, 94.46% DR, 96.98% PRE, 95.71 F_1 score while keeping 5.20% FAR on UNSW-NB15 dataset.

3.4 Conclusion

In this chapter, we propose a detection engine for network intrusion detection, DualNet, which is an extendable DenseNet with a self-attention mechanism. To allow to build deeper neural networks to achieve a high detection performance, we propose densely connected learning, which can reuse features to effectively handle performance degradation prob-

lem of deep neural networks for network intrusion detection and is applied to construct the DenseNet. We also demonstrate the effectiveness and efficiency of the self-attention mechanism for network intrusion detection.

Our experiments show that DualNet outperforms existing traditional machine learning and advanced deep learning designs in terms of accuracy, detection rate, precision, F_1 score and false alarm rate. Most importantly, its effectiveness on the near real-world UNSW-NB15 dataset demonstrates its practical value to security teams for traffic analysis and attack recognition.

Chapter 4

EnsembleNet: A Deep Ensemble Network for Network Intrusion Detection

4.1 Introduction

In order to improve the detection performance of an individual detection model, we leverage the synergy of multiple different detection models to develop an effective ensemble design for attack recognition, namely EnsembleNet. Unlike the traditional ensemble designs, which are mainly based on simple and weak ML models, EnsembleNet is constructed with the DNN models so that the high learning potential of DNNs can be utilized for good detection performance. In this design, similar to DualNet, each DNN model is built with the CNN and RNN subnets that are connected in a way such that features learned by the subnets can be reused and hence the performance degradation problem can be effectively mitigated. To efficiently integrate the detection results from the DNNs of EnsembleNet, we propose a greedy majority voting algorithm that can be applied not only to binary classification but also to multi-class classification tasks and is scalable for a large ensemble

network with many DNNs.

In addition, for the detection results to be useful to the security team, we propose an alert-output enhancement design as a user-friendly interface of EnsembleNet. The design restores the threats (detected by the neural network) to their human-understandable raw traffic format and produces alerts of the current threats in the order of their severity so that the security team can make prompt responses and hence maximally reduce the security risk. EnsembleNet is presented in the next section.

4.2 EnsembleNet

EnsembleNet consists of three modules: stream processor, detection engine and alert interface, as shown in Fig. 4.1. The design of each module is explained below.

4.2.1 Stream Processor

The stream processor is for data preprocessing. As illustrated in Fig. 4.1(a), the stream processor converts network traffic flows (shown in Fig. 4.1(d)) into corresponding statistical representation records (shown in Fig. 4.1(e)) that are suitable for neural network learning. Take the first statistical representation record shown in Fig. 4.1(e) as an example. The complete set of features of this statistical representation record (47 features) is illustrated in Fig. 4.7(a). The stream processor contains three functions for converting network traffic flows into corresponding statistical representation records, as shown in Fig. 4.1(a). Each of them is elaborated below.

4.2.1.1 Data Consolidation

For EnsembleNet, we extend our design to be more flexible and scalable. The traffic data can be collected from multiple sources and have various formats, such as .argus, .log, and .json. The data consolidation function merges the data records with a unified format.

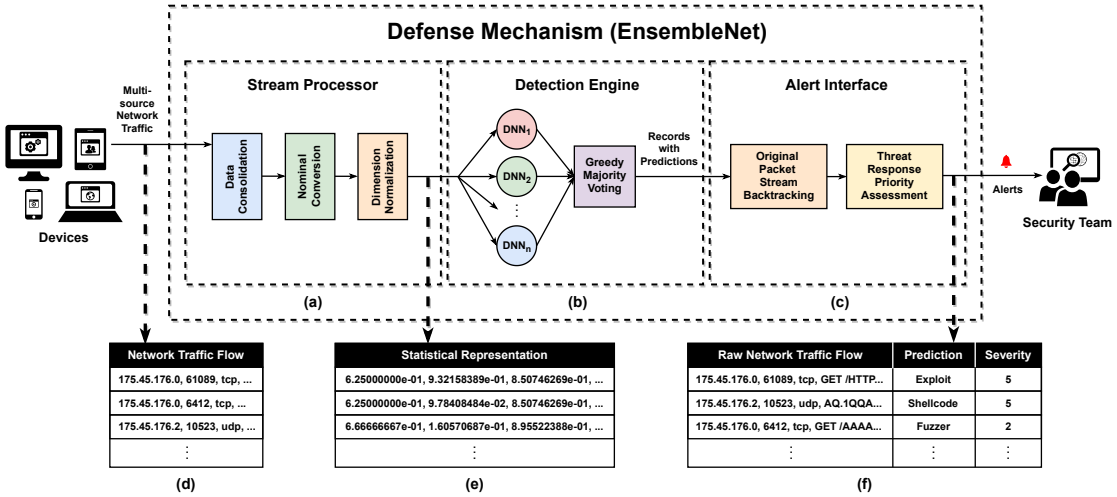


Figure 4.1: EnsembleNet System Overview

In addition, to ensure the data quality, the function also removes duplicate records and replaces missing data with the mean value of the related features.

4.2.1.2 Nominal Conversion

As discussed in Chapter 3, there are many categorical features in the traffic flow, such as IP address and protocol, which cannot be fed straight into the neural network. Moreover, there are too many types of the categorical feature values. The nominal conversion function applies label encoding technique [81] to convert the textual notation into a machine-readable form. Since the function employs digital codes to represent long textual values, the amount of computation by the neural network is reduced.

4.2.1.3 Dimension Normalization

As mentioned in Chapter 3, the features with larger magnitudes in the traffic data records may dominate in model fitting, leading to biased predictions. The dimension normalization function uses the min-max normalization [70] to reconstruct data in each feature on a scale of 0 to 1 so that all features contribute to model fitting equally. The normalization also improves the learning stability and accelerates the backpropagation in training.

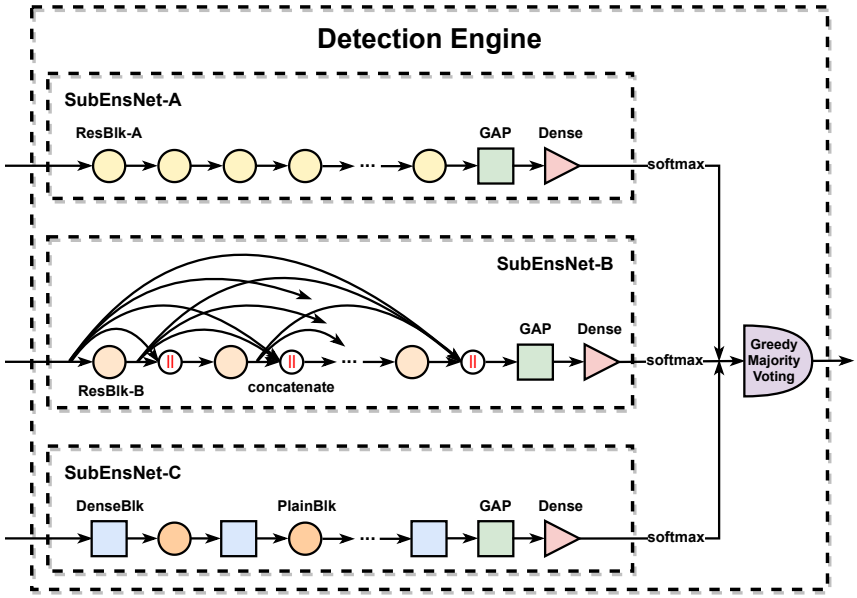


Figure 4.2: Detection Engine of EnsembleNet

4.2.2 Detection Engine

The detection engine is an ensemble neural network, as shown in Fig. 4.1(b). Unlike the traditional ensemble designs (such as RF), which are constructed with weak ML models, EnsembleNet aims for high detection performance and hence uses the strong DNN models.

Fig. 4.2 illustrates the overall architecture of a detection engine with three DNNs. For high efficiency, we want those DNN to have the following attributes: *1) capable of spatial-temporal learning, 2) able to reuse learning features, and 3) having a low computational cost.* To this end, we build the DNNs on specially designed blocks: plain blocks (PlainBlks), residual blocks (ResBlks) and dense blocks (DenseBlks). Each has a different complexity. In fact, PlainBlk is a building block of ResBlks and DenseBlks. Their designs are discussed below.

4.2.2.1 Plain Block of EnsembleNet (PlainBlk)

The plain block design is based on the plain block of DualNet, which contains the simplified versions of CNN, DSC [56] and the simplified versions of RNN, GRU [58] to extract the

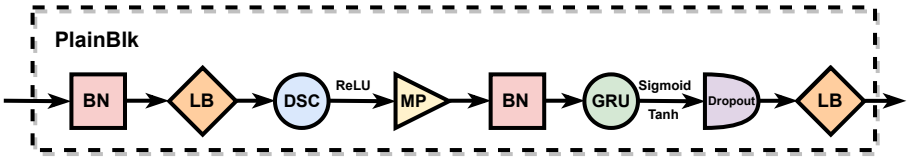


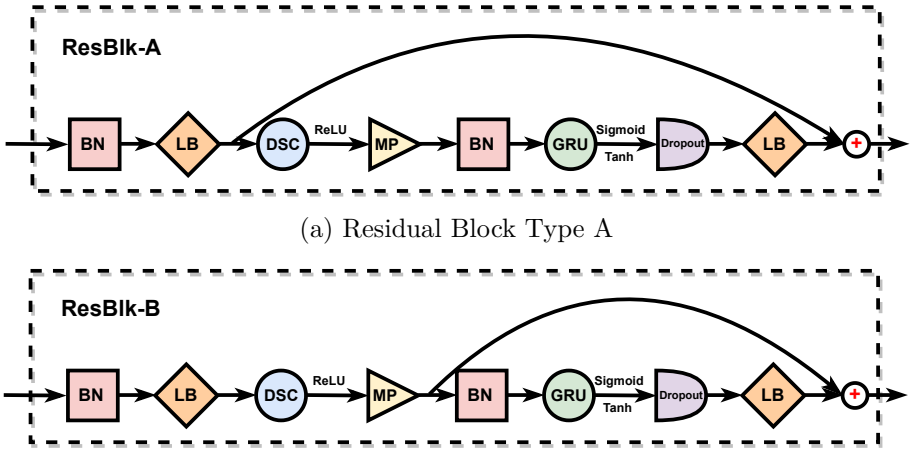
Figure 4.3: Plain Block of EnsembleNet

spatial-temporal features and reduce the computational cost, as has been discussed in Section 3.2.1.1. Fig. 4.3 shows the structure of PlainBlk. In contrast to the plain block of DualNet, we add one more linear bridging (LB) layer after the BN to enhance the learning stability. Due to the randomness of neural network training, the results of each complete training process will be slightly different. LB helps to reduce the cost of retraining required to obtain the optimal model and stabilize the learning process. Consequently, the model does not need to be retrained.

As shown in Section 3.3.4.1, with the stacking of more PlainBlks – the network goes deeper, the network would suffer performance degradation. The main reason is that as the network depth increases, the learned features gradually become extremely specific but far away from their original meanings, eventually resulting in gradient vanishing [12]. One of the effective methods to solve this optimization obstacle is *feature reuse*, as demonstrated in the previous chapter. Feature reuse keeps the features initially learned from the shallow parameter layers and makes them available at the deep layers to retain the originality of features. This is done by connecting shallow layers to deep layers, and combining the early learned features with the later learned features. The combination can be in two operation modes: *add* and *concatenation*, which leads to our other block designs: ResBlk and DenseBlk.

4.2.2.2 Residual Block of EnsembleNet (ResBlk)

ResBlk incorporates the residual learning [54] into PlainBlk to handle performance degradation, where the effectiveness of using the residual learning to reuse features to build DNNs for network intrusion detection has been proved in [47]. ResBlk adds a shortcut



(a) Residual Block Type A

(b) Residual Block Type B

Figure 4.4: Residual Block of EnsembleNet

connection and uses "add" to combine the features from both connected layers. ResBlk has two versions of design: ResBlk-A and ResBlk-B. Fig. 4.4(a) shows the structure of ResBlk-A, where the input of DSC is connected to the output of the last layer. While for ResBlk-B, there is a slight difference in the shallow layer to be connected, as shown in Fig. 4.4(b), which will be further discussed later. The shortcut facilitates the forward propagation of activations and the backward propagation of errors, thus avoiding gradient vanishing. It is worth noting that the summation operation requires that the tensors to be added have the same shape.

4.2.2.3 Dense Block of EnsembleNet (DenseBlk)

The DenseBlk design is based on the dense block of DualNet (Fig. 3.5), as shown in Fig. 4.5, where each PlainBlk receives a concatenation of the input and the data from all its preceding PlainBlks. In such a dense connectivity pattern, features at various levels of abstraction can be fully learned. Moreover, the flow of gradients within the network can be significantly strengthened, thus addressing the performance degradation caused by the vanishing gradient.

Based on the above basic building blocks, we develop three DNNs as sub-classifiers for

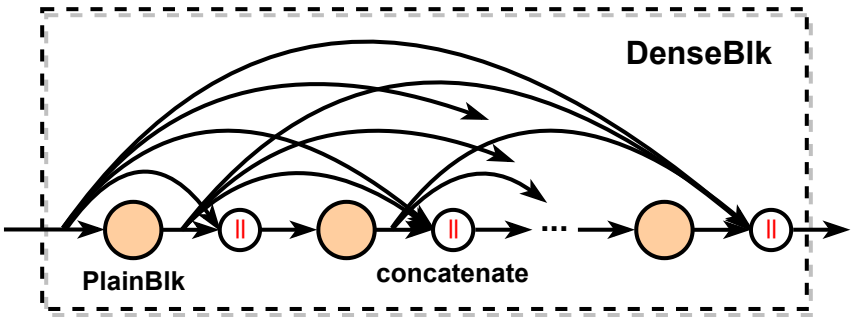


Figure 4.5: Dense Block of EnsembleNet

EnsembleNet.

4.2.2.4 Sub-classifiers of EnsembleNet

The three sub-classifiers named as SubEnsNet-A, SubEnsNet-B and SubEnsNet-C, as shown in Fig. 4.2, are presented below.

SubEnsNet-A. SubEnsNet-A is a deep residual neural network. It is constructed with a series of residual blocks, ResBlk-A blocks, followed by a global average pooling (GAP) layer and a dense layer with the softmax activation function. The GAP layer is to further strengthen corresponding relationships between features and the classification categories and the dense layer is to determine the final detection result. As can be seen from Fig. 4.2, both GAP layer and dense layer are also applied to the other two sub-classifiers (for the same purposes).

SubEnsNet-B. SubEnsNet-B is a deep densely connected residual neural network, which is also built upon a group of ResBlks but with a dense connection structure, as shown in Fig. 4.2. In addition, instead of using ResBlk-A, ResBlk-B is used. The reason is that the data dimension from the concatenation would increase, making the shape of tensors to be added inconsistent if ResBlk-A was adopted. To handle the problem, we leverage the down-sampling ability of the MP (max-pooling) and move the shortcut just after the MP (see Fig. 4.4(b)) so that the tensor shape can be adjusted while the local originality of features can be mostly retained.

SubEnsNet-C. SubEnsNet-C is a deep dense neural network that uses our densely connected learning, as detailed in Section 3.2.1. As can be seen from Fig. 4.2, SubEnsNet-C is established with DenseBlks and PlainBlks in an interleaved arrangement pattern. In this way, we can build a very deep SubEnsNet-C, which has been proved to be effective for performance improvement in Section 3.3.4.1.

In summary, we have built three extensible DNNs that allow themselves to go deeper for a good intrusion detection performance. When we put them into EnsembleNet, even better performance can be achieved, which is closely related to how to combine the prediction results from these sub-classifiers. Our design is given below.

4.2.2.5 Greedy Majority Voting Algorithm

There are some existing aggregation algorithms for ensemble classifiers, such as Boosting and Bagging, as mentioned in Section 2.4.2.1. Boosting is a sequential process on a set of sequentially performed sub-classifiers, which could result in considerable computing time if these classifiers were DNNs. Bagging, on the other hand, can be performed on the parallel sub-classifiers. But each of sub-classifier only works on a small subset of the input data, which may cause model underfitting if DNN was used. Moreover, due to sampling with replacement used in Bagging, these subsets always contain duplicate records, which may result in biased predictions of the sub-classifiers. In addition, Bagging uses random selection when the sub-classifiers come to multiple tied-voting results, which is not effective for multi-class classification, and the binary classification with an even number of voters. Here, we propose a different integration algorithm, *Greedy Majority Voting*, for EnsembleNet.

Our algorithm combines the idea of majority voting with the detection performance of each DNN sub-classifier, where each DNN learns the entire dataset and is trained and tested in parallel. The algorithm supports any number of classifiers and can handle both binary and multi-class classification tasks. Considering that for a DNN model, Accuracy indicates its generalization capability and Precision indicates whether a high threat detection capability

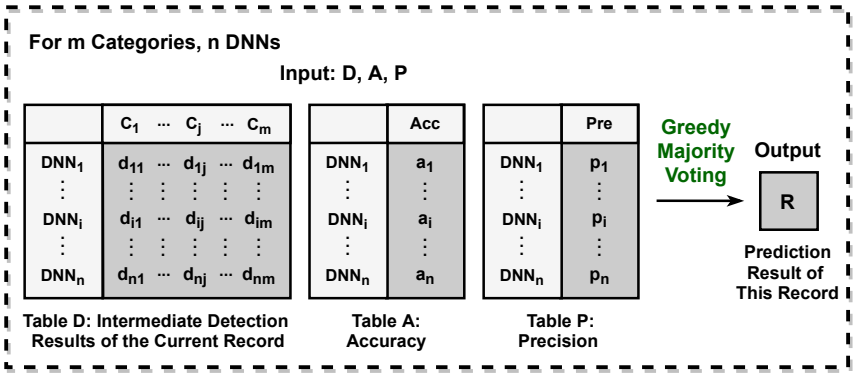


Figure 4.6: The Input and Output of Greedy Majority Voting Algorithm

of the model comes at the cost of high false alarms, we use the score obtained from these two metrics to represent the DNN performance in our algorithm. The algorithm is explained below.

Assume there are m detection categories, C_1, ..., C_m and n DNNs, DNN_1, ..., DNN_n. We use three tables to present the information available to the algorithm, as shown in Fig. 4.6. Table D holds the detection results from the n DNNs for the current network traffic record to be classified, where d_{ij} is the decision of DNN_i on whether the traffic record belongs to category C_j; if yes, d_{ij} is 1, otherwise 0. For a traffic record, each DNN will predict one and only one category to be true, therefore, the sum of each row in Table D is 1. Table A stores the accuracy of each DNN, where a_i is the accuracy of DNN_i. The precision of the DNNs is saved in Table P; on the same notion, p_i is the precision of DNN_i. The result returned from the algorithm is saved in R that holds the final prediction of the ensemble network.

Algorithm 1 describes steps of the greedy majority voting. Given the predictions from the DNNs, D, the DNNs' accuracy in A and precision in P, the algorithm first collects votes for each prediction category (line 1 - line 3). Then it selects the predictions with the highest votes and saves it in S (line 4 - line 5). If there is only one prediction in S, it returns the prediction and the algorithm stops (line 6 - line 7); Otherwise, if there is more than one prediction in S (e.g., two predictions are tied in votes), it removes those predictions from S that are generated by the DNNs with lower performance, which is

Algorithm 1 Greedy Majority Voting Algorithm

Input: D, A, P

Output: R

1: **for each** j = 1, 2, ..., m **do**
2: $v_j = \text{Sum}(d_{1j} + d_{2j} + ... + d_{nj})$
3: **end for**
4: $V = \text{Max}(\{v_j, \text{j} = 1, 2, ..., \text{m}\})$
5: $S = \text{getPredictionWithMaxVote}(V)$
6: **if** $|S| = 1$ **then** ▷ Only one element in S
7: $R = C_j, C_j \in S$
8: **else**
9: $S = \text{getPredictionWithHighAccuracy}(S, A)$
10: **if** $|S| = 1$ **then**
11: $R = C_j, C_j \in S$
12: **else**
13: $S = \text{getPredictionWithHighPrecision}(S, P)$
14: **if** $|S| = 1$ **then**
15: $R = C_j, C_j \in S$
16: **else**
17: $R = \text{getFirstPrediction}(S)$
18: **end if**
19: **end if**
20: **end if**
21: **return** R

based first on their accuracy and then on their precision if required (line 8 - line 15). After that, if there is still more than one prediction in S, return the first one (line 16 - line 17). As a result, the algorithm produces a prediction for each traffic record.

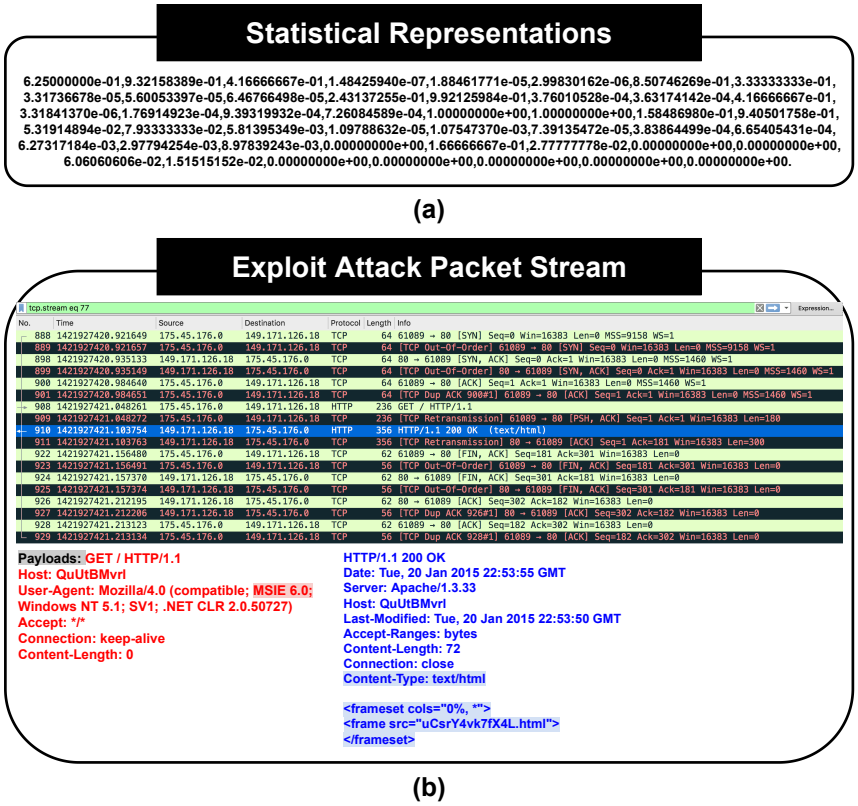


Figure 4.7: An Exploit Attack within a TCP Stream: Microsoft Internet Explorer Frameset Memory Corruption and the Attack Reference Is CVE-2006-3637

4.2.3 Alert Interface

We want the threat alerts generated from the detection engine to be understandable and useful to human security analysts. Therefore, we include a user-friendly alert interface, as shown in Fig. 4.1(c). The interface offers two functions: original packet stream backtracking and threat response priority assessment, which are presented below.

4.2.3.1 Original Packet Stream Backtracking

As shown in Fig. 4.1(e), the features used for learning are vectorized, making it difficult for security analysts to understand, interpret and analyze. Our investigation on the network traffic data shows that the network traffic streams can be uniquely identified by their source IP address, source port number, destination IP address, destination port number,

flow start time and end time, and protocol. We use this combined information to correlate a threat detected by the neural network to the related raw network flow in different source traces and restore the threat from the neural-network-used format to the human-understandable textual traffic format.

We adopt Wireshark[1] to visualize the original packet streams. Fig. 4.7 shows an example for a detected Exploit attack. Its neural-network-used format is shown in Fig. 4.7(a) and Fig. 4.7(b) is its restored raw network stream.

As can be seen from the figure, the stream contains 3-way handshake packets for establishing TCP connections, HTTP request packets, HTTP response packets and 4-way handshake packets for tearing down TCP connections. The payloads of the stream provide more valuable attack-related information. Traffic from the client to the server is colored in red, while traffic from the server to the client is colored blue. The payloads indicate *'Microsoft Internet Explorer Frameset Memory Corruption [82]' (CVE-2006-3637 [83]):* a flaw in Microsoft Internet Explorer (MSIE) 6.0 (marked in pink) that makes the browser unable to properly handle various combinations of HTML layout components. The hacker exploits the vulnerability when rendering HTML using a crafted frameset (marked in blue), which results in memory corruption.

By obtaining the raw traffic format, security analysts can find more cyber threat intelligence and specific attack behavior from the payloads to get insight into the threat so that they can effectively respond to the threat. More examples are demonstrated in Section 4.3.5.2.

4.2.3.2 Threat Response Priority Assessment

Different attacks may impose different levels of security risk. Based on the recommendation of the security analysts in the industry, we group all cyber threats into five severity levels, ranging from level 1 (of the lowest) to level 5 (of the highest): *level 1—low impact, level*

[1]Wireshark: https://www.wireshark.org

2—possible impact, level 3—medium impact, level 4—significant impact and level 5—high impact. For example, Analysis attack belongs to level 1; Reconnaissance, Scanning and Fuzzer attacks belong to level 2; DoS and Man-In-The-Middle (MITM) attacks belong to level 3; Generic, Backdoor, Password Cracking, Injection, Cross-site Scripting (XSS) and Worm attacks belong to level 4; and Exploit, Shellcode, DDoS and Ransomware attacks belong to level 5. Based on our observations, the majority of false positives produced by the anomaly-based NIDSs are the normal network traffic mis-classified as low-severity threats (e.g., Analysis). This may be due to their similar behaviors.

We output alerts of the attacks in the order of their severity, as demonstrated in Fig. 4.1(f), so that the severest attacks can get immediate attention from the security team and be contained to minimize security risks.

4.3 Evaluation and Discussion

4.3.1 Experimental Environment Settings

Our evaluation is based on a cloud AI platform configured with an NVIDIA Tesla K80 GPU and a total of 12 GB of RAM. EnsembleNet and some related ML-based designs to be used for comparison are modeled in Python, with Tensorflow as the backend and the APIs of Keras libraries and scikit-learn packages.

4.3.2 Datasets Selection

We use two modern datasets, UNSW-NB15 [14] and TON_IoT [15] as cyber threat assessment testbeds. Unlike the evaluation of DualNet that directly adopted processed datasets for evaluation, we use the testbeds including raw network traffic flows, which allows for a more complete evaluation of EnsembleNet. They were both generated by Australian Cyber Security Centre Laboratory, where UNSW-NB15 was created in 2015 but TON_IoT was produced in 2020. Similar to the UNSW-NB15 dataset, the TON_IoT dataset is also

a balanced dataset without duplicate records and missing values [15], which ensures the effectiveness of the evaluation, as discussed in Section 3.3.1. Like the other datasets used in this thesis, TON_IoT is also a widely used dataset in modern research on network intrusion detection systems [12, 84, 85].

UNSW-NB15 was generated by IXIA PerfectStorm tool[2]. The tool simulates a real-word network environment with millions of up-to-date attack scenarios that are updated regularly from the Common Vulnerabilities and Exposures (CVE) site and the normal traffic. As mentioned in Chapter 3, this testbed offers both real and synthesized anomalous network activities and covers nine typical attack types [14]: Generic, Exploit, Fuzzer, Reconnaissance, DoS, Shellcode, Backdoor, Analysis and Worm. To obtain a more comprehensive evaluation, we use the original testbed containing 2,540,044 traffic records with 47 features, where 10% of the records are randomly selected for evaluation. TON_IoT is a more recent testbed to mimic the complexity and scalability of the Industry 4.0 network that includes IoT and industrial IoT (IIoT) networks. The testbed consists of a telemetry testbed of IoT/IIoT sensors, testbeds of Linux- and Windows-based audit traces and a testbed of network traffic. Today more than 70% of network traffic is based on Transport Layer Security (TLS)/Secure Sockets Layer (SSL) cryptographic protocols to ensure secure communications. Since the network traffic testbed of TON_IoT contains TLS/SSL-related features, we use this testbed for the evaluation in order to reflect the current network traffic and intrusion behaviors. In addition to benign traffic, the testbed covers nine contemporary attack families [15]: Backdoor, Password, Injection, DDoS, Scanning, DoS, XSS, Ransomware and MITM. There are 461,043 traffic records with 44 features in the network traffic testbed. Similar to the evaluation of DualNet, we use stratified 10-fold cross validation for evaluation, as detailed in the previous chapter. And we use all of the attack types provided by the two datasets for a comprehensive evaluation.

[2]IXIA: https://support.ixiacom.com/strikes

4.3.3 Configuration of EnsembleNet

For the detection engine of EnsembleNet, we configure SubEnsNet-A with 10 ResBlk-A blocks, SubEnsNet-B with 10 ResBlk-B blocks and SubEnsNet-C with two DenseBlks interleaved by one PlainBlk, where each DenseBlk has five PlainBlks. Therefore, SubEnsNet-A has 83 layers including 53 parameter layers; Similarly, SubEnsNet-B has 83 layers including 53 parameter layers, and SubEnsNet-C has 91 layers including 58 parameter layers. EnsembleNet also requires configuring a group of hyper-parameters for model initialization. For each DNN sub-classifier of EnsembleNet, the number of filters in the convolutions and the number of recurrent units are adjusted to be consistent with the number of learning features. In the training phase, we employ sparse categorical cross entropy as loss function for performing error calculation that is used in the backpropagation. And we use root mean square propagation (RMSprop) algorithm [86] as optimizer to minimize errors and accelerate gradient descent as well as convergence rate. The learning rate is set to 0.001 here.

4.3.4 Evaluation Metrics

Similar to other ML-based designs, we use five metrics to evaluate the performance of the designs: accuracy (ACC), detection rate (DR), false alarm rate (FAR), precision (PRE) and F_1 score, as defined below.

$$ACC = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions}, \tag{4.1}$$

$$DR = \frac{TP}{TP + FN}, \tag{4.2}$$

$$FAR = \frac{FP}{FP + TN}, \tag{4.3}$$

Table 4.1: Testing Performance of EnsembleNet and Its DNN Subnets on UNSW-NB15

Design	**ACC %**		**DR %**	**FAR %**	**PRE %**	F_1 **Score**
	Multi	**Binary**				
SubEnsNet-A	75.45	87.53	97.57	24.76	82.84	89.60
SubEnsNet-B	75.71	87.28	97.56	25.32	82.52	89.41
SubEnsNet-C	76.26	88.38	97.67	23.01	83.87	90.25
EnsembleNet	**77.27**	**88.66**	**98.12**	**22.92**	**83.99**	**90.51**

$$PRE = \frac{TP}{TP + FP}, \tag{4.4}$$

$$F_1\ score = \frac{2 \times PRE \times DR}{PRE + DR}, \tag{4.5}$$

where TP and TN are, respectively, the number of attacks and the number of normal network traffic correctly categorized; FP is the number of actual normal traffic misclassified as attacks, and FN is the number of attacks incorrectly classified as normal network traffic.

4.3.5 EnsembleNet Performance

We first evaluate the overall detection performance of the three sub-classifiers of EnsembleNet and the standalone EnsembleNet. Then, we evaluate the performance of original packet stream backtracking strategy. We finally evaluate the detection capability of EnsembleNet for each category and compare it with a set of existing typical ML-based designs to further evaluate its generalization capability.

4.3.5.1 Overall Detection Performance

Table 4.1 and Table 4.2 illustrates the testing performance of SubEnsNet-A, SubEnsNet-B, SubEnsNet-C and EnsembleNet on UNSW-NB15 and TON_IoT datasets respectively. From the tables, we can see that the three sub-classifiers of EnsembleNet achieve a high

Table 4.2: Testing Performance of EnsembleNet and Its DNN Subnets on TON_IoT

Design	**ACC %**		**DR %**	**FAR %**	**PRE %**	F_1 **Score**
	Multi	**Binary**				
SubEnsNet-A	99.13	99.14	99.42	1.01	98.15	98.78
SubEnsNet-B	99.48	99.48	99.51	0.53	99.02	99.26
SubEnsNet-C	99.78	99.78	99.58	0.11	99.79	99.69
EnsembleNet	**99.95**	**99.95**	**99.94**	**0.04**	**99.93**	**99.93**

threat detection capability while keeping a low false alarm rate, enabling EnsembleNet to have a good performance. Moreover, EnsembleNet outperforms all its sub-classifiers by achieving higher ACC on processing both multi-class and binary classification tasks, higher DR, higher PRE, higher F_1 score and lower FAR on two datasets, which testifies the effectiveness of proposed greedy majority voting algorithm. To further demonstrate the effectiveness of the algorithm, we also applied the simple majority voting on the three different neural networks. Our experiment results show that Greedy Majority Voting algorithm improves the detection accuracy by 1.24% on UNSW-NB15 dataset and 0.46% on TON_IoT dataset. Specifically, the detection accuracy of using the simple majority voting on UNSW-NB15 dataset is 76.32%, which is lower than 77.27% generated by our algorithm and the detection accuracy of using the simple majority voting on TON_IoT dataset is 99.49%, which is lower than 99.95% produced by our algorithm. Hence, the proposed algorithm is effective.

From the tables, we can also find that our designs accomplish better overall detection performance on TON_IoT dataset, as compared to UNSW-NB15 dataset. The main possible reason is that the TON_IoT dataset has more payload-related typical features that can help accurately distinguish anomalous traffic flows from legitimate traffic flows, such as 'http_method', 'http_uri' and 'http_user_agent'. A detailed discussion will be given in the next sub-section.

Figure 4.8: Comparison of the Payloads of the Normal Traffic and Different Attacks

4.3.5.2 Original Packet Stream Backtracking

Since UNSW-NB15 is a near-real-world network traffic dataset with attack references, we use this dataset to evaluate the performance of original packet stream backtracking strategy. For each record, we obtain the packet stream generated by our flow tracing method and check whether the stream matches the real traffic flow of the record. There are 100% matches, which confirms the effectiveness of this strategy.

In addition to the example given in Fig. 4.7, here we include more examples of payloads, as shown in Fig. 4.8, to reveal the typical behavior of other types of attacks detected by the neural network detection engine. For each attack, we also provide a brief description of the attack nature and a short summary of representative attack features.

Normal Network Traffic. Fig. 4.8(a) shows the payloads within a legitimate TCP

stream: a normal payload of the HTTP request packet (marked in pink) and a normal payload of the HTTP response packet (marked in blue).

Generic Attack. Fig. 4.8(b) shows the payloads within a TCP stream about a Generic attack, which is *'Apple QuickTime STSD Atoms Handling Heap Overflow [87]' (CVE-2007-3750 [88]).* Apple QuickTime before 7.3 exists the heap-based buffer overflow vulnerability, which is due to boundary errors when processing Sample Table Sample Descriptor (STSD) atoms in a movie file. The hacker exploits the flaw to trick target users into opening a QuickTime movie file (marked in pink) with crafted STSD atoms (marked in blue), eventually leading to arbitrary code execution.

Fuzzer Attack. Fig. 4.8(c) shows the payloads within a TCP stream about a Fuzzer attack, which is *'HTTP GET Request Invalid URI [89]'.* The hacker continuously sends a series of HTTP GET requests with non-existent URLs (marked in pink) to the same destination address and destination port to analyze the response information to find and exploit potentially hackable vulnerabilities.

Reconnaissance Attack. Fig. 4.8(d) shows the payloads within a TCP stream about a Reconnaissance attack, which is *'Oracle 9iAS Dynamic Monitoring Services Anonymous Access Variant 6 [90]' (CVE-2002-0563 [91]).* There is a default configuration flaw in the Oracle 9i Application Server version 1.0.2.x. The hacker exploits the vulnerability by accessing sensitive services anonymously without authentication, including Dynamic Monitoring Services such as servlet/DMSDump and DMS/AggreSpy (marked in pink).

DoS Attack. Fig. 4.8(e) shows the payloads within a TCP stream about a DoS attack, which is *'Google Chrome PDF Viewer Multi-page Printing DoS HTTP [92]' (CVE-2011-0472 [93]).* Google Chrome before 8.0.552.237 (marked in pink) has a vulnerability that can be triggered when a user prints a multi-page PDF document. The hacker launches the denial-of-service attack via the document (marked in blue), which would lead to an application crash or other unspecified impacts.

Shellcode Attack. Fig. 4.8(f) shows the payloads within a UDP stream about a Shell-

code attack, which is *'OpenBSD x86 Bind Shell – noir [94]' (milw0rm-0513).* The hacker transmits a block of shellcode (marked in pink) over a UDP socket to control the compromised machine.

Backdoor Attack. Fig. 4.8(g) shows the payloads within a TCP stream about a Backdoor attack, which is *'WordPress Backdoor iz Parameter Passthru [95]' (CVE-2007-1277 [96]).* During February and March 2007, WordPress 2.1.1 downloaded from several official distribution sites that included an externally introduced malicious backdoor. The hacker exploits the backdoor by executing arbitrary operating system commands via an untrusted passthru function call in the iz parameter to the wp-includes/theme.php (marked in pink).

Analysis Attack. Fig. 4.8(h) shows the payloads within a TCP stream about an Analysis attack, which is *'Killed ActiveX Instantiation [97]'.* The hacker sends a series of HTML pages that instantiate Microsoft ActiveX controls (marked in blue) to the same destination address and destination port, where the controls have set the kill bit through SPs or patches issued by Microsoft. These class identifiers (CLSIDs) are harmful if instantiated via Microsoft Internet Explorer (MSIE) (marked in pink), which can cause either command execution or memory corruption.

Worm Attack. Fig. 4.8(i) shows the payloads within a TCP stream about a Worm attack, which is *'Lupper.A XML-RPC Propogation Request Variant 8 [98]' (CVE-2005-1921 [99]).* Eval injection vulnerability in XML-RPC For PHP 1.1 and earlier version (marked in pink), as applied in WordPress, phpWebSite and other products. The Lupper.A worm exploits the bug to infect the system by executing a block of crafted PHP code via an XML file (marked in blue).

As can be observed through these examples and the example in Section 4.2.3, *request target (URL)*, *user agent*, *content type* and *message body* are strong features of payloads within a stream, presenting the most valuable attack-related information that can be used for rapid attack identification, performing counter-attack measures and forensic analysis. Furthermore, *content length* is a weak feature that contributes to attack recognition and

Table 4.3: Testing Performance of Using EnsembleNet for Each Category on UNSW-NB15

Category	**ACC %**	**DR %**	**FAR %**	**Precision %**	F_1 **Score**
Normal	88.66	77.08	1.88	97.09	85.94
Generic	99.97	99.97	0.02	99.96	99.97
Exploit	96.70	99.71	4.37	88.96	94.03
Fuzzer	82.02	80.10	17.76	34.31	48.04
Reconnaissance	98.94	99.89	1.15	89.30	94.30
DoS	99.94	90.51	0.01	97.64	93.94
Shellcode	99.53	99.00	0.47	59.76	74.53
Backdoor	100.00	100.00	0.00	100.00	100.00
Analysis	98.13	0.00	1.87	0.00	0.00
Worm	99.99	83.33	0.00	83.33	83.33

Table 4.4: Testing Performance of Using EnsembleNet for Each Category on TON_IoT

Category	**ACC %**	**DR %**	**FAR %**	**Precision %**	F_1 **Score**
Normal	99.95	99.96	0.06	99.97	99.96
Backdoor	100.00	100.00	0.00	100.00	100.00
Password	100.00	100.00	0.00	99.95	99.98
Injection	100.00	100.00	0.00	100.00	100.00
DDoS	100.00	100.00	0.00	100.00	100.00
Scanning	100.00	100.00	0.00	100.00	100.00
DoS	100.00	100.00	0.00	100.00	100.00
XSS	100.00	100.00	0.00	99.95	99.98
Ransomware	99.94	99.55	0.03	99.50	99.53
MITM	100.00	99.04	0.00	100.00	99.52

unknown threat perception. It is worth noting that we have to be careful when the *Post* method appears. The reason is that the Post pushes the data to the server, which could be a piece of crafted shellcodes.

4.3.5.3 Detection Capability for Each Category

EnsembleNet can not only identify whether a flow is normal or abnormal but also determine its specific attack type if it is abnormal. Table 4.3 and Table 4.4 show the testing

performance of EnsembleNet on different detection categories. From the tables, we can see that EnsembleNet performs well on the detection of normal traffic and most types of attacks – with high ACC, high DR, high precision, high F_1 score and low FAR, especially for Reconnaissance, Scanning and DoS attacks (which is beneficial to early discovery of sophisticated and severe threats such as APTs and DDoS attacks), and DDoS and Ransomware attacks (both are considered the most serious threats at present, with disastrous consequences). The exception is for the Fuzzer and Analysis attacks, which are discussed below.

For the Fuzzer attacks, the detection has a high false alarm rate, a low precision and a low F_1 score. The main reason is that legitimate users may accidentally make typos when requesting valid URLs, thus confusing the classifier. The low performance on Analysis attacks (i.e., low DR, low precision and low F_1 score) may due to two reasons. One is that Analysis is to listen to and analyze network communications to capture basic cyber information. Its behavior can also be observed in the normal traffic, making it hard to distinguish the Analysis from the normal. For example, commands such as *whoami* and $ipconfig$ can come from Analysis attacks but also can come from legitimate users. The Analysis attack is not a direct attack. It is rather considered as anomalous behavior that may lead to a real attack. Another reason of the poor detection performance on the Analysis attack is that there are only around 1.04% Analysis records applied in the evaluation, which is an imbalance learning problem that often results in poor generalization performance.

In short, EnsembleNet performs well in recognizing high-severity threats but not so well in differentiating between low-severity threats (e.g., Analysis and Fuzzer attacks) and some benign network activities because they involve similar behaviors. Given the low security risk posed by such threats, the detection results are acceptable.

Table 4.5: Testing Performance of Existing Typical Machine Learning-based Designs on UNSW-NB15

Design	**ACC %**		**DR %**	**FAR %**	**PRE %**	F_1 **Score**
	Multi	**Binary**				
AdaBoost	52.29	74.30	92.56	48.06	70.23	79.86
NB	53.06	74.60	89.57	43.74	71.50	79.52
SVM	54.51	63.89	69.40	42.85	66.49	67.91
RF	56.10	76.89	89.50	38.56	73.98	81.00
LSTM	68.88	84.78	94.68	27.35	80.92	87.26
ConvNet	69.01	83.27	97.75	34.47	77.65	86.55
MLP	71.47	86.29	97.56	27.51	81.29	88.69
Densely-ResNet	72.92	85.64	95.34	26.24	81.66	87.97
DualNet	**75.79**	**87.57**	**98.10**	**25.33**	**82.59**	**89.68**
EnsembleNet	**77.27**	**88.66**	**98.12**	**22.92**	**83.99**	**90.51**

4.3.5.4 Comparative Study

To further evaluate the generalization performance of EnsembleNet, we compare EnsembleNet with a set of existing typical ML-based designs, as discussed in Chapter 2, and our DualNet, as presented in Chapter 3. Table 4.5 shows the testing performance of those designs on the near-real-world UNSW-NB15 dataset. As can be observed from the table, the traditional ML methods accomplish high detection rate at the cost of high false alarm rate (i.e., low precision). The DL-based designs outperform the traditional ML methods with higher accuracy on both multi-class and binary classification tasks, higher detection rate, higher precision, higher F_1 score and lower false alarm rate. Importantly, among these designs, EnsembleNet presents the best overall performance; It has the highest accuracy, highest detection rate, highest precision and highest F_1 score while maintaining the lowest false alarm rate. Compared with those existing DL-based designs: LSTM, ConvNet, MLP and Densely-ResNet. Though improvement by DualNet is incremental, our final enhanced design, EnsembleNet, presents considerable achievements. Specifically, EnsembleNet can recognize 1,558 more attacks while reducing 1,638 false alarms, 165 more attacks while reducing 4,274 false alarms, 253 more attacks while reducing 1,697 false alarms, and 1,257 more attacks while reducing 1,228 false alarms, respectively.

To sum up, the comparison results demonstrate that our design EnsembleNet has a high generalization capability and it significantly improves the effectiveness of network intrusion detection systems.

4.4 Conclusion

In this chapter, we propose a deep ensemble network-based defense mechanism, EnsembleNet, for network intrusion detection. EnsembleNet is constructed with three DNN detection models that can reuse spatial-temporal features to unleash the potential of deep learning for good detection performance. The DNNs' decisions are integrated by an efficient greedy majority voting algorithm to provide an even better detection solution. EmsembleNet also offers a user-friendly alert interface where alerts are prioritized based on the threat severity level and are human understandable.

We evaluate EnsembleNet on two modern datasets, one that simulates real-world network traffic and the other that can represent the current network traffic. We compare it with a set of typical existing ML-based designs including classical ML methods, state-of-the-art DL techniques and DualNet. Our experimental results on the near-real-world UNSW-NB15 dataset demonstrate that EnsembleNet is superior to those designs, with the highest capability of attack recognition while maintaining the lowest false alarm rate. Given the effectiveness and feasibility of EnsembleNet, it can be adopted by cyber threat defense communities for future intrusion detection and traffic analysis tasks.

Chapter 5

Conclusion and Future Work

Network intrusion detection systems play a pivotal role in offering the modern society a secure and reliable network communication environment, especially in the context of the global epidemic and the continuous development of new technologies. Effective detection methods are much needed in order to develop a NIDS with high capability of cyber attack recognition while producing low false alarms so that security risks can be reduced and alert fatigue problem can be mitigated.

We provide basic background and related works of network intrusion detection in Chapter 2. In this chapter, we introduce some typical cyber-attacks in the current threat landscape and demonstrate the general workflow of a NIDS. In addition, we compare and discuss the advantages and disadvantages of rule-based NIDSs and anomaly-based NIDSs. Rule-based detection systems, due to their inability to identify new attacks, are being replaced by anomaly-based detection systems to cope with the ever-evolving threat environment. Anomaly-based solutions built with traditional machine learning methods achieve high threat detection performance at the cost of high false positives, leading to alert fatigue. Deep learning approaches with deep neural networks can alleviate this problem and accomplish better generalization capability. However, the deep learning-based NIDSs are still in the development stage and there is still large room for improvement in the existing designs. As such, we propose two deep learning-based designs, DualNet and EnsembleNet.

In Chapter 3, we describe DualNet, which is an extendable DenseNet enhanced with a self-attention mechanism. We also in this chapter demonstrate the effectiveness of feature reuse in dealing with performance degradation problem of deep neural networks for network intrusion detection and apply feature reuse to DualNet to improve its generalization performance. DualNet achieves high threat detection accuracy while keeping low false alarm rates on two datasets, one is traditional well-known NSL-KDD dataset and the other is the near-real-world UNSW-NB15 dataset.

Based on the study of DualNet, we in Chapter 4 develop a defense mechanism using ensemble learning, EnsembleNet. EnsembleNet is constructed with three extendable deep neural networks that can reuse features for good detection performance and the detection results of these deep neural networks are integrated by the greedy majority voting algorithm to further enhance the detection performance. EnsembleNet can also produce user-friendly threat alerts to allow the security team to rapidly and effectively respond to threats. EnsembleNet achieves excellent detection performance on the TON_IoT dataset that is representative of the current network traffic. Our evaluation on the near-real-world UNSW-NB15 dataset also shows that EnsembleNet outperforms state-of-the-art deep learning-based designs by accomplishing higher attack detection performance while maintaining lower false positive rate. Through our investigation, we demonstrate that deep neural network is a promising and effective detection method, which can be used to construct the next generation of NIDS.

5.1 Future Work

While EnsembleNet is superior to typical existing anomaly-based designs in identifying attacks and reducing false alarms, its detection performance is not perfect, especially in the perception of some low-severity threats, which can be consider for further improvement. In the future, more effective deep neural network architectures and more efficient integration algorithms will be investigated.

This thesis has demonstrated the effectiveness of deep learning in recognizing abnormal behaviors from network traffic data represented in the Euclidean space. The traffic data can also be represented in a non-Euclidean space, namely as graphs with complex relationships and interdependencies between objects [100]. Developing deep learning models based on the graph-structured data can enhance the correlation of threat alerts to help security analysts quickly investigate, contain and defeat attacks. Thus, graph neural networks for network intrusion detection can be an interesting investigation area.

Deep learning can also be used for other security tasks, such as provenance graph-based APT detection [30], semantic-level phishing email detection [19] and Android malware detection [101], which can be future research directions.

References

[1] Fbi sees spike in cyber crime reports during coronavirus pandemic. Accessed: 2022-02-28. [Online]. Available: https://thehill.com/policy/cybersecurity/493198-fbi-sees-spike-in-cyber-crime-reports-during-coronavirus-pandemic

[2] Imc grupo: Since the pandemic began, the fbi reported a 300% increase in reported cybercrimes. Accessed: 2021-09-24. [Online]. Available: https://www.imcgrupo.com/covid-19-news-fbi-reports-300-increase-in-reported-cybercrimes/

[3] Ibm: How much does a data breach cost? Accessed: 2021-09-24. [Online]. Available: https://www.ibm.com/security/data-breach

[4] Fintech news: Cloud-based cyber attacks rose 630% between january and april 2020. Accessed: 2021-09-24. [Online]. Available: https://www.fintechnews.org/the-2020-cybersecurity-stats-you-need-to-know/

[5] Forbes: 3 key cybersecurity trends to know for 2021. Accessed: 2021-09-24. [Online]. Available: https://www.forbes.com/sites/chuckbrooks/2021/04/12/3-key-cybersecurity-trends-to-know-for-2021-and-on-/?sh=5b2d55394978

[6] Acsc annual cyber threat report. Accessed: 2021-09-24. [Online]. Available: https://www.cyber.gov.au/sites/default/files/2020-09/ACSC-Annual-Cyber-Threat-Report-2019-20.pdf

[7] J. Ashraf, M. Keshk, N. Moustafa, M. Abdel-Basset, H. Khurshid, A. D. Bakhshi, and R. R. Mostafa, "Iotbot-ids: A novel statistical learning-enabled botnet detection

framework for protecting networks of smart cities," *Sustainable Cities and Society*, p. 103041, 2021.

[8] S. Yang, P. Wu, and H. Guo, "Dualnet: Locate then detect effective payload with deep attention network," in *2021 IEEE Conference on Dependable and Secure Computing (DSC)*, 2021, pp. 1–8.

[9] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications surveys & tutorials*, vol. 18, no. 2, pp. 1153–1176, 2015.

[10] M. Azizjon, A. Jumabek, and W. Kim, "1d cnn based network intrusion detection with normalization on imbalanced data," in *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, 2020, pp. 218–224.

[11] A. Boukhalfa, A. Abdellaoui, N. Hmina, and H. Chaoui, "Lstm deep learning method for network intrusion detection system." *International Journal of Electrical & Computer Engineering (2088-8708)*, vol. 10, 2020.

[12] P. Wu, N. Moustafa, S. Yang, and H. Guo, "Densely connected residual network for attack recognition," in *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 2020, pp. 233–242.

[13] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE symposium on computational intelligence for security and defense applications*. IEEE, 2009, pp. 1–6.

[14] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 Military Communications and Information Systems Conference (MilCIS)*. IEEE, 2015, pp. 1–6.

[15] N. Moustafa, "A new distributed architecture for evaluating ai-based security systems at the edge: Network ton_iot datasets," *Sustainable Cities and Society*, vol. 72, p. 102994, 2021.

[16] S. Winterfeld and J. Andress, "The basics of cyber warfare," *Understanding the Fundamentals of Cyber Warfare in Theory and Practice, Waltham*, 2013.

[17] N. Koroniotis, N. Moustafa, and E. Sitnikova, "Forensics and deep learning mechanisms for botnets in internet of things: A survey of challenges and solutions," *IEEE Access*, vol. 7, pp. 61 764–61 785, 2019.

[18] I. Fette, N. Sadeh, and A. Tomasic, "Learning to detect phishing emails," in *Proceedings of the 16th international conference on World Wide Web*, 2007, pp. 649–656.

[19] G. Ho, A. Cidon, L. Gavish, M. Schweighauser, V. Paxson, S. Savage, G. M. Voelker, and D. Wagner, "Detecting and characterizing lateral phishing at scale," in *28th {USENIX} Security Symposium ({USENIX} Security 19)*, 2019, pp. 1273–1290.

[20] In april 2020, google blocked 18 million daily malware and phishing emails related to coronavirus. Accessed: 2021-09-24. [Online]. Available: https://cloud.google.com/blog/products/identity-security/protecting-against-cyber-threats-during-covid-19-and-beyond

[21] J. Kaur, "Taxonomy of malware: Virus, worms and trojan," *Int. J. Res. Anal. Rev*, vol. 6, no. 1, pp. 192–196, 2019.

[22] Open web application security project top 10 (owasp top 10). Accessed: 2021-09-24. [Online]. Available: https://owasp.org/www-project-top-ten/

[23] M. K. Daly, "Advanced persistent threat," *Usenix, Nov*, vol. 4, no. 4, pp. 2013–2016, 2009.

[24] W. U. Hassan, A. Bates, and D. Marino, "Tactical provenance analysis for endpoint detection and response systems," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 1172–1189.

[25] Cyber kill chain. Accessed: 2021-09-24. [Online]. Available: https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html

[26] Mitre att&ck: a globally-accessible knowledge base of adversary tactics and techniques based on real-world observations. Accessed: 2021-09-24. [Online]. Available: https://attack.mitre.org

[27] D. Denning, "An intrusion-detection model," *IEEE Transactions on Software Engineering*, vol. SE-13, no. 2, pp. 222–232, 1987.

[28] B. Mukherjee, L. T. Heberlein, and K. N. Levitt, "Network intrusion detection," *IEEE network*, vol. 8, no. 3, pp. 26–41, 1994.

[29] Splunk: The data-to-everything platform. Accessed: 2021-09-24. [Online]. Available: https://www.splunk.com

[30] S. M. Milajerdi, R. Gjomemo, B. Eshete, R. Sekar, and V. Venkatakrishnan, "Holmes: real-time apt detection through correlation of suspicious information flows," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 1137–1152.

[31] N. Moustafa, B. Turnbull, and K.-K. R. Choo, "An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4815–4830, 2018.

[32] M. Roesch *et al.*, "Snort: Lightweight intrusion detection for networks." in *Lisa*, vol. 99, no. 1, 1999, pp. 229–238.

[33] V. Paxson, "Bro: A system for detecting network intruders in real-time," *Computer networks*, vol. 31, no. 23-24, pp. 2435–2463, 1999.

[34] K. Thongkanchorn, S. Ngamsuriyaroj, and V. Visoottiviseth, "Evaluation studies of three intrusion detection systems under various attacks and rule sets," in *2013 IEEE International Conference of IEEE Region 10 (TENCON 2013)*. IEEE, 2013, pp. 1–4.

[35] R. P. Lippmann, D. J. Fried, I. Graf, J. W. Haines, K. R. Kendall, D. McClung, D. Weber, S. E. Webster, D. Wyschogrod, R. K. Cunningham *et al.*, "Evaluating

intrusion detection systems: The 1998 darpa off-line intrusion detection evaluation," in *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00*, vol. 2. IEEE, 2000, pp. 12–26.

[36] P. Laskov, P. Düssel, C. Schäfer, and K. Rieck, "Learning intrusion detection: supervised or unsupervised?" in *International Conference on Image Analysis and Processing*. Springer, 2005, pp. 50–57.

[37] K. Prakobphol and J. Zhan, "A novel outlier detection scheme for network intrusion detection systems," in *2008 International Conference on Information Security and Assurance (isa 2008)*. IEEE, 2008, pp. 555–560.

[38] M. Jianliang, S. Haikun, and B. Ling, "The application on intrusion detection based on k-means cluster algorithm," in *2009 International Forum on Information Technology and Applications*, vol. 1, 2009, pp. 150–152.

[39] H. Choi, M. Kim, G. Lee, and W. Kim, "Unsupervised learning approach for network intrusion detection system using autoencoders," *The Journal of Supercomputing*, vol. 75, no. 9, pp. 5597–5621, 2019.

[40] J. Suaboot, A. Fahad, Z. Tari, J. Grundy, A. N. Mahmood, A. Almalawi, A. Y. Zomaya, and K. Drira, "A taxonomy of supervised learning for idss in scada environments," *ACM Computing Surveys (CSUR)*, vol. 53, no. 2, pp. 1–37, 2020.

[41] S. Gamage and J. Samarabandu, "Deep learning methods in network intrusion detection: A survey and an objective comparison," *Journal of Network and Computer Applications*, vol. 169, p. 102767, 2020.

[42] Y. B. Bhavsar and K. C. Waghmare, "Intrusion detection system using data mining technique: Support vector machine," *International Journal of Emerging Technology and Advanced Engineering*, vol. 3, no. 3, pp. 581–586, 2013.

[43] F. Gumus, C. O. Sakar, Z. Erdem, and O. Kursun, "Online naive bayes classification for network intrusion detection," in *2014 IEEE/ACM International Conference on*

Advances in Social Networks Analysis and Mining (ASONAM 2014). IEEE, 2014, pp. 670–674.

[44] M. Gudadhe, P. Prasad, and L. K. Wankhade, "A new data mining based network intrusion detection model," in *2010 International Conference on Computer and Communication Technology (ICCCT)*. IEEE, 2010, pp. 731–735.

[45] N. Farnaaz and M. Jabbar, "Random forest modeling for network intrusion detection system," *Procedia Computer Science*, vol. 89, pp. 213–217, 2016.

[46] Y. Bengio, O. Delalleau, and N. L. Roux, "The curse of highly variable functions for local kernel machines," in *Advances in neural information processing systems*, 2006, pp. 107–114.

[47] P. Wu, H. Guo, and N. Moustafa, "Pelican: A deep residual network for network intrusion detection," in *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*. IEEE, 2020, pp. 55–62.

[48] A. Rosay, F. Carlier, and P. Leroux, "Feed-forward neural network for network intrusion detection," in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*. IEEE, 2020, pp. 1–6.

[49] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[50] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[51] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[52] M. Raghu, B. Poole, J. Kleinberg, S. Ganguli, and J. Sohl-Dickstein, "On the expressive power of deep neural networks," in *international conference on machine learning*. PMLR, 2017, pp. 2847–2854.

[53] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[54] ——, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[55] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*. PMLR, 2019, pp. 6105–6114.

[56] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.

[57] R. Zhao, Z. Li, Z. Xue, T. Ohtsuki, and G. Gui, "A novel approach based on lightweight deep neural network for network intrusion detection," in *2021 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2021, pp. 1–6.

[58] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[59] A. F. Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018.

[60] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[61] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[62] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[63] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[64] B. Asadi and H. Jiang, "On approximation capabilities of relu activation and softmax output layer in neural networks," *arXiv preprint arXiv:2002.04060*, 2020.

[65] Kdd cup 1999 dataset. Accessed: 2022-02-28. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[66] M. Nour and S. Jill, "The evaluation of network anomaly detection systems: Statistical analysis of the unsw-nb15 data set and the comparison with the kdd99 data set," *Information Security Journal: A Global Perspective*, vol. 25, no. 1-3, pp. 18–31, 2016.

[67] B. A. Tama, M. Comuzzi, and K.-H. Rhee, "Tse-ids: A two-stage classifier ensemble for intelligent anomaly-based intrusion detection system," *IEEE Access*, vol. 7, pp. 94 497–94 507, 2019.

[68] P. Wu and H. Guo, "Lunet: a deep neural network for network intrusion detection," in *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2019, pp. 617–624.

[69] Y. Yu and N. Bian, "An intrusion detection method using few-shot learning," *IEEE Access*, vol. 8, pp. 49 730–49 740, 2020.

[70] S. García, J. Luengo, and F. Herrera, *Data preprocessing in data mining*. Springer, 2015.

[71] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[72] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*. Springer, 2013, vol. 112.

[73] J. Zhang, M. Zulkernine, and A. Haque, "Random-forests-based network intrusion detection systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 5, pp. 649–659, 2008.

[74] W. Hu, J. Gao, Y. Wang, O. Wu, and S. Maybank, "Online adaboost-based parameterized methods for dynamic distributed network intrusion detection," *IEEE Transactions on Cybernetics*, vol. 44, no. 1, pp. 66–82, 2013.

[75] X. Bao, T. Xu, and H. Hou, "Network intrusion detection based on support vector machine," in *2009 International Conference on Management and Service Science*. IEEE, 2009, pp. 1–4.

[76] C. Xu, J. Shen, X. Du, and F. Zhang, "An intrusion detection system using a deep neural network with gated recurrent units," *IEEE Access*, vol. 6, pp. 48 697–48 707, 2018.

[77] I. Ahmad, A. Abdullah, A. Alghamdi, K. Alnfajan, and M. Hussain, "Intrusion detection using feature subset selection based on mlp," *Scientific research and essays*, vol. 6, no. 34, pp. 6804–6810, 2011.

[78] S. A. Althubiti, E. M. Jones, and K. Roy, "Lstm for anomaly-based network intrusion detection," in *2018 28th International Telecommunication Networks and Applications Conference (ITNAC)*. IEEE, 2018, pp. 1–3.

[79] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[80] R. Vinayakumar, K. Soman, and P. Poornachandran, "Applying convolutional neural network for network intrusion detection," in *2017 International Conference on*

Advances in Computing, Communications and Informatics (ICACCI). IEEE, 2017, pp. 1222–1228.

[81] J. T. Hancock and T. M. Khoshgoftaar, "Survey on categorical data for neural networks," *Journal of Big Data*, vol. 7, pp. 1–41, 2020.

[82] Exploit: Microsoft internet explorer frameset memory corruption. Accessed: 2021-09-24. [Online]. Available: https://support.ixiacom.com/strikes/exploits/browser/ms06_042_html_frameset_memory_corruption.xml

[83] Cve-2006-3637. Accessed: 2021-09-24. [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2006-3637

[84] M. A. Ferrag, O. Friha, L. Maglaras, H. Janicke, and L. Shu, "Federated deep learning for cyber security in the internet of things: Concepts, applications, and experimental analysis," *IEEE Access*, vol. 9, pp. 138 509–138 542, 2021.

[85] I. A. Khan, N. Moustafa, I. Razzak, M. Tanveer, D. Pi, Y. Pan, and B. S. Ali, "Xsru-iomt: Explainable simple recurrent units for threat detection in internet of medical things networks," *Future Generation Computer Systems*, vol. 127, pp. 181–193, 2022.

[86] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.

[87] Generic: Apple quicktime stsd atoms handling heap overflow. Accessed: 2021-09-24. [Online]. Available: https://support.ixiacom.com/strikes/generic/ixia/apple_quicktime_stsd_atoms_handling_heap_overflow_attack.xml

[88] Cve-2007-3750. Accessed: 2021-09-24. [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-3750

[89] Fuzzer: Http get request invalid uri. Accessed: 2021-09-24. [Online]. Available: https://support.ixiacom.com/strikes/fuzzers/http/get_invaliduri.xml

[90] Reconnaissance: Oracle 9ias dynamic monitoring services anonymous access variant 6. Accessed: 2021-09-24. [Online]. Available: https://support.ixiacom.com/strikes/recon/http/oracle/oracle_dms_anonymous_access_5.xml

[91] Cve-2002-0563. Accessed: 2021-09-24. [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2002-0563

[92] Dos: Google chrome pdf viewer multi-page printing dos http. Accessed: 2021-09-24. [Online]. Available: https://support.ixiacom.com/strikes/denial/browser/chrome_pdf_multipage_printing_dos.xml

[93] Cve-2011-0472. Accessed: 2021-09-24. [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-0472

[94] Shellcode: Openbsd x86 bind shell - noir. Accessed: 2021-09-24. [Online]. Available: https://support.ixiacom.com/strikes/shellcode/openbsd/bind_x86_noir_udp.xml

[95] Backdoor: Wordpress back-door iz parameter passthru. Accessed: 2021-09-24. [Online]. Available: https://support.ixiacom.com/strikes/backdoors/wordpress_iz_passthru.xml

[96] Cve-2007-1277. Accessed: 2021-09-24. [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-1277

[97] Analysis: Killed activex instantiation. Accessed: 2021-09-24. [Online]. Available: https://support.ixiacom.com/strikes/analysis/html/activex_killbit_clsids.xml

[98] Worm: Lupper.a xml-rpc propogation request variant 8. Accessed: 2021-09-24. [Online]. Available: https://support.ixiacom.com/strikes/worms/linux_lupper_a_xmlrpc_08.xml

[99] Cve-2005-1921. Accessed: 2021-09-24. [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2005-1921

[100] W. W. Lo, S. Layeghy, M. Sarhan, M. Gallagher, and M. Portmann, "E-graphsage: A graph neural network based intrusion detection system," *arXiv preprint arXiv:2103.16329*, 2021.

[101] K. Xu, Y. Li, R. H. Deng, and K. Chen, "Deeprefiner: Multi-layer android malware detection system applying deep neural networks," in *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2018, pp. 473–487.