Department of Information and Communication Engineering
Graduate School of Information Science and Technology
THE UNIVERSITY OF TOKYO

Master Thesis

Network Intrusion Detection Based on Distributed Trustworthy Artificial Intelligence

（信頼される分散型 AI に基づくネットワーク侵入検知）

48-196454

Yuwei SUN

Supervisor: Associate Professor Hideya OCHIAI

January 2021

Abstract

The rapidly increasing personal computers, mobile devices, IoT sensors, and other smart devices are sending trillions of bytes' data through the Internet every day. Threats underlying cyberspace are causing economic loss, information breaches, and other critical detriments. In this thesis, theoretical fundaments and practical applications of machine learning (ML)-powered network intrusion detection systems (IDS) were demonstrated against a large-scale network environment. We proposed two visualization approaches, the position-based and channel-based feature map representation, to analyze network traffic data, revealing hidden patterns of network events such as port scans, anomalous SMB, and so forth. Besides these visual analytics of utilizing extracted features, we studied the method of intrusion detection using raw network traffic. By combining these methods with a deep learning (DL) model, especially the convolutional neural networks (CNNs), we aimed to classify various types of network events thus identifying real-time threats. Moreover, we researched malicious domain name detection based on a passive and an active approach respectively. For the passive approach, we employed a variational autoencoder (VAE) to compress the lexical vectors of a domain name into latent feature vectors, followed by either a support vector machine (SVM) or a fully-connected neural network (NN) to classify them. In contrast, for the active approach, we applied a web crawler to systematically send requests to the servers of domain names, and by observing the upcoming network flows between the servers and the client, we employed a DL-based detection method for identifying malicious flows and malicious domain names. Besides, a distributed DL-based IDS was proposed based on federated learning (FL), where intrusion monitoring nodes were allowed to share real-time insights of the detection through ML model sharing, thus improving both performance and adaptability. We were the first to propose the Segmented-FL with a characteristic of automatic framework transformation for adapting to an unknown network environment. This framework could not only protect the data privacy of a client but also prevent novel cyberattacks in networks. Furthermore, to study the security and liability of a distributed DL framework such as FL, we mounted several types of attacks on the systems, endpoint data poisoning and adversarial attacks based on generative adversarial networks (GANs), which either effectively exacerbated the systems' performance or disclosed private data of a client even with slight manipulation of training data or DL model parameters. Additionally, we researched aircraft detection with CNNs to understand feature extraction and the architecture of CNNs. The research of trajectory optimization for an autonomous vehicle provided insights into the optimization theories of DL.

Contents

Part I Introduction

Chapter 1 Introduction

1.1 Problem Description

The pandemic of COVID-19 is accelerating digitalization in many walks of society, along with rapidly increasing devices and users connected to networks. Much information is stored, shared, and analyzed on the Internet every second. The current transformation is bringing us to a new era of the information society, which requires more effective and resilient network monitoring systems for safeguarding information in networks. A malware delivered in a network usually can intrude a host and further expand into other hosts in the network, causing enormous losses and affecting various aspects of people's life. For example, phishing e-mails lure and direct users to malicious websites, causing unintentional disclosure of their private information to attackers. The connections with these suspicious sources have a high risk of bringing great loss to the network users.

On the other hand, network intrusion detection strategies existing in current systems have been revealing issues of low adaptivity to network traffic in various network environments. The development of network monitoring systems that have the ability to detect malware in various networks is imperative. That's because the patterns of network traffic used for intrusion detection are varying from a network to another, with different network scales, devices with different operating systems and applications installed, and so on. For instance, some networks have more than 1,000 active users while some have only a dozen users, showing different presence information. Besides, devices in these networks are usually working on different operating systems such as Windows, macOS, and Ubuntu, which brings about different traffic patterns. Some networks also include Internet of Things (IoT) devices such as web cameras and smart home devices. Additionally, user groups could be varying from students in academic institutes to professionals in industries. These reasons contribute to the diversity of the network traffic patterns as well as the difficulty of large scale network-oriented adaptive intrusion detection.

On the other hand, the disciplinary field of artificial intelligence has been achieving great advancement recently, especially with an outstanding development in deep learning (DL). Due to the breakthroughs in 2012, when a neural network with more than three layers became feasible and several methods were proposed to overcome the former problems of neural networks such as the vanish gradients, lots of DL models have been developed and applied to counter with tasks from various fields, such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), and variational autoencoder (VAE). For that reason, there has been a lot of research on the application of DL to cybersecurity. The advancement of DL and ML provides us with insights on the design and development of distributed while highly inner-connected network monitoring systems.

Furthermore, privacy is a dominant concern in different aspects of the digital world, so is the case in DL systems. Though the current digitalization and knowledge acquisition happening in various industries is bringing us to a new era of smart society, the two essential questions have been how to protect personal data while enabling compliance with information sharing regulations. In this regard, federated learning (FL) has been suggested as a collaborative learning framework for DL users with partial and imbalanced data to pursue a shared goal; the privacy-oriented nature of FL allows users to collaborate in network intrusion detection through local model sharing, at the same time, without disclosing their personal network activities.

In addition, unfortunately, even within such a decentralized learning framework, an adversary could still perturb a system in the framework by manipulating local training data and falsifying a DL model, which effectively exacerbates the performance and security of the framework. Consequently, the main problems existing in the development of AI-based intrusion detection systems (IDS) lie in how to

effectively, accurately, and adaptively safeguard networks from cyberattacks and how to ensure the liability, security, and privacy of AI-based decision-making.

Nowadays, lots of countries' primary government agencies are acting actively towards AI-based cybersecurity. Defense Advanced Research Project Agency (DARPA) in the United States has been conducting a project on Organically Assured and Survivable Information Systems (OASIS), which focuses on increasing fault tolerance in systems and networks. The Japan Ministry of Education, Culture, Sports, Science and Technology (MEXT) has been investing in the Advanced Integrated Intelligence Platform Project. Additionally, the Ministry of Economy, Trade and Industry and, the Ministry of Internal Affairs and Communications are also actively working on the R&D and demonstration for the implementation of AI-centered IoT in various aspects of society and business.

1.2 Technical Challenges

Traditional approaches to detecting threats in networks usually show limits when it comes to frequently changing network traffic patterns. The signature-based intrusion detection has become disadvantaged compared with DL-based methods which show outstanding abilities of knowledge acquisition and feature representation thus extracting hidden patterns from enormous network traffic. Besides, though underlying threats have been observed in current network systems, network operators tend to focus on short-term fixes and tactical improvements. In summary, current network IDSs have the following disadvantages: isolated detection, data exposure while tracking, and vulnerability to high-level AI-powered attacks.

- Isolated detection: current detection algorithms are usually designed and implemented to identify an anomalous event based on the traffic patterns in a specified network. The adaptability of a network monitoring system to varying network environments is imperative.
- Data exposure while tracking: from the perspective of privacy, an IDS based on a third party's observation and analysis, involving entities such as the infrastructure administrator and the user account administrator, has the risk of exposing various extents of authority to accessing personal data.
- Vulnerability to high-level AI-powered attacks: in the future where attackers employ AI to help them with cybercrimes, current systems mainly based on experts' knowledge reveal certain vulnerability to these highly-adaptive AI-powered attacks. For example, the generative adversary networks (GANs) could be applied to generate highly imitated benign traffic patterns hiding malware to confuse an IDS or even a human expert.

For these reasons, a long-term resilient maintaining method is necessary to safeguard devices and users connected to networks. The goals are designing distributed network IDSs based on artificial liable intelligence, and implementing and validating the systems in the contexts of industrial applications. Several technical challenges include: 1) the systems should be able to defense multi-type intrusion in a network; 2) the systems are highly adaptative thus easily scalable to other environments; 3) the AI-based detection algorithms should be liable and explainable; 4) the computing should be distributed instead of relying on a central resource. These points form the base of the research in this thesis.

1.3 Constitution

To conclude, this thesis consists of five parts. In Part 2, Deep Learning in Cybersecurity, we discuss DL-based approaches to dealing with various problems in cybersecurity. Chapter 2 presents visual analytics including the poison-based approach and the channel-based approach for representing network traffic features. Chapter 3 presents a network events classifier using raw traffic data and CNNs. Chapter 4 demonstrates malicious domain name detection based on the VAE. Chapter 5 presents a web crawler on malicious websites for identifying anomalous communications with supervised learning. In Chapter 6, we focus on a distributed intrusion detection framework called the Segmented-Federated Learning

(Segmented-FL) to improve system adaptability to diverse large-scale network environments. In Part 3, we study the liability and explainability of the DL-based detection algorithms. Chapter 7 discusses a consensus-based defense framework with blockchain strategies against endpoint data poisoning attacks. In Chapter 8, we demonstrate mounting an adversarial attack on the FL to steal the personal data of a user based on GANs. In Part 4, we present research on modeling and optimization. Chapter 9 demonstrates the aircraft detection in remote sensing images based on saliency maps and CNNs. Chapter 10 presents the trajectory optimization for an autonomous vehicle driving across stochastic traffic flows based on direct collocation. Finally, in Part 5, we conclude the thesis.

Part II Deep Learning in Cybersecurity

Chapter 2 Network Intrusion Detection with Visual Analytics

2.1 Network Events Classification with Poison-Based Feature Maps and Convolutional Neural Networks

2.1.1 Introduction

Recently, the problem of cyber security has been debated by not only experts but also users of network. Especially, in the local area network (LAN), the attack known as phishing, delivering malware and spread to social media, messaging services and applications. Attack like this can affect every aspects of targets' personal and working lives. Malware invasion is also frequent. As a result, it is necessary to detect the anomaly in LAN and above that, identifying the type of attacking, which contributes to solving of the security problem.

With the development of machine learning technology, it is considerable that using machine learning to try to deal with the problem we mentioned above. As we all know, machine learning is used in the previous researches to identify the anomaly in the LAN, which tells you the network state is normal or abnormal. The previous researches with machine learning usually focus on the detecting of anomaly instead of showing the detail of the attack itself, such as the occurrence of TCP scan or ARP scan.

We propose a scheme that can learn and classify network events happening in the LAN using convolutional neural network (CNN). We propose a method of creating an image from network packet dump of a certain duration (Fig. 2.1). By putting this image through CNN, it can learn the features of the network without hardcoding type of the network events. When a malware intrudes into a LAN, and try to expand into (or steal some data from) the other hosts in the LAN, it tries to access some specific TCP or UDP ports of all the hosts. This kind of activities is a network event. In this chapter, we focus on ARP scan, TCP scan, UDP scan, and those scans to specific ports as the types of network events, which we target to classify by our scheme.

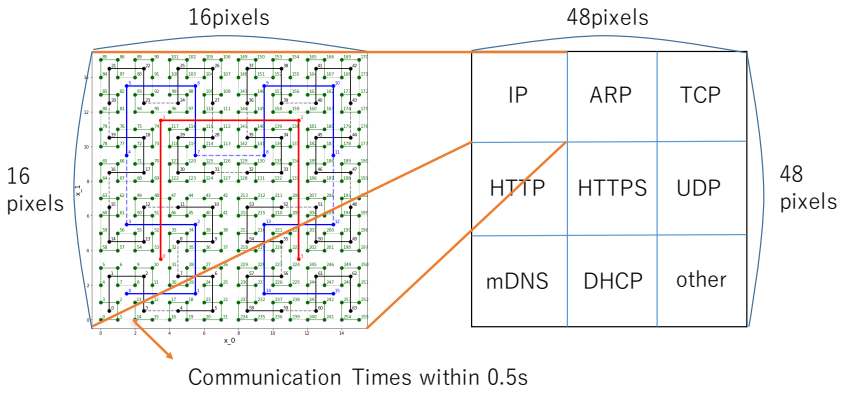


Fig. 2.1. We selected eight protocols and set different protocols at regions in one single image, with a width and height of 48 pixels.

We put the protocol information of traffic data in the LAN using Hilbert Curve (Fig. 2.1) to generate feature maps representing different types of events, which we will discuss in detail in the section 3. We

use CNN, a kind of machine learning method, usually used in tasks of object detection and classification, for recognizing the network events happened in the LAN.

After that, we can use the trained model to do inference, which is a method that constantly monitors and safeguards the network. Our model will detect and classify the types of attacks at the same time. In this way, we can provide some evidences of the occurrence of attacking. Our research is focusing on the method of visualizing features of traffic data and training CNN model using these feature maps.

This is organized as follows. Section 2 discusses related works about anomaly detection from traffic data in the LAN. Section 3 provides an overview of our algorithm including the visualization of time related features of traffic data and training of the CNN model. Section 4 presents the performance evaluation of the algorithm for events classification using a standard named recall. Section 5, we discuss the result of evaluation in different environments. Section 6, we conclude the chapter and give out the possibility of a broader practical use of our scheme.

2.1.2 Related Work

Anomaly detection in computer networks attracts many attentions, with more than 40 years of evolution [1]. With the rapid growth and increasing complexity of network infrastructures, and the evolution of attacks, identifying and preventing networks attacks are getting more and more challenging. Considering the remanding robust attribute to different network environments, in a wider perspective, many researchers have considered a semi-supervised approach, where they train the classifier with "normal" traffic data only, so that knowledge about anomalous events will be constructed and thus be detected. For example, Ugo et al. [2] described a usage of Restricted Boltzmann Machines (RBMs) to combine the expressive power of generative models with good classification accuracy capabilities. Asmaa et al. [3] presented a comprehensive discussion of using RBM model for feature learning and the classifier model for anomaly detection. Another research by Hsu and Lin [4], they used support vector machine (SVM) models, showing the ability of the proposed methodology to detect DDoS attacks in an efficient and accurate way. Moreover, in the research of Mehdi et al. [5], they used a class of advanced machine learning techniques, namely Deep Learning (DL), to facilitate the analytics and learning in the IoT domain. Kazumasa et al. [6] used the approach of machine learning to define a feature vector for detecting Command and Control (C&C) server in botnet using network traffic data.

However, the introduce of machine learning is almost aimed to detect the anomaly which is different from the purpose of our research. We proposed an approach to classify different types of events in the LAN. Different from only labeling normal data or both normal data and abnormal ones which is demonstrated in previous work using an approach of machine learning, we labeled different types of events to train the CNN model, building a classifier in a dynamic way. Furthermore, researchers have demonstrated that it is possible to apply CNN not only to image classification tasks but also to signal classification tasks. For instance, Tatsuya Harada and Yuji Tokozume [7] discussed an approach using CNN to classify different types of time-series environmental sounds. In our research, we focus on representing the protocol information of traffic data in the LAN by 2-D images data, a dataset built to train the CNN model.

2.1.3 Network Events Classification with CNN

Traffic data in the LAN can be recorded through a tool named tcpdump, which allows a user to record TCP, IP, UDP and other packets being transmitted or received in a network and also which computer is attached by the packets. After that we use a library named dpkt to analyze the big data to understand which protocol is being used in the communication. In this research, we analyze the traffic data and calculate the number of how many packets are transmitted or received for each protocol within one second to generate a feature map representing the occurrence of the specific events. Then, in order to detect an event from the traffic data captured in the LAN, a CNN model was introduced and trained

using these feature maps. After that, trained model will be used as an initialization of the detection system, preparing an environment for inference. Then, the monitor server will capture traffic data in the LAN and start data mining with a predetermined time span. At the analyzing step, as described above, the communication information within a predetermined time span (every 128 seconds) of each protocol is statistically recorded by dpkt. Then, the density data of the recorded time protocol will be converted to a time series feature map using Hilbert Curve. Here, the generated feature maps are stored on the server simultaneously with the generation. Using the generated maps, the trained CNN model will compute the probability of the occurrence of each event. The one with the highest possibility is considered occurring in the LAN. It is understood that communication information recorded in bit format can be expressed and represented as high-dimensional feature information in images.

2.1.3.1 Visualization of Time Related Features of Traffic Data

In expressing the features of traffic data, we first introduce the concept of "fineness" here. Fineness is a parameter to show how finely we should consider the information hidden in big data. And the time span we use to record the traffic data to generate each feature map is defined as in (2.1).

$$\mathrm{T} = \mathrm{Tst} \times \mathrm{fineness} \times \left(\frac{\mathrm{size}}{\mathrm{St}}\right)^2 \tag{2.1}$$

Here, Tst (time standard) is a standard time span for recording, which is defined as 60s in this research. St is a parameter of the basic recording segment, showing the standard size of a feature map. And we use 8 pixels here (thus the size of the image is 24 × 24). What's more, the parameter of size (here, exempted values are 8, 16, 32, 64, 128) is the actual one we use to generate the feature maps. And fineness (here, exempted values are 5.0, 1.0, 0.5, 0.2, 0.1, 0.05, 0.02, 0.001) is mentioned above.

As described above, it is possible to use the parameters (fineness and size) to bring features of traffic data, even different in time span and fineness into images with the same size.

Furthermore, it is necessary to comprehensively consider the adaptability to the model of deep learning how to express the time-related features of the whole event using feature maps from each protocol. Hilbert Curve is a method used to transform the structure of data so that it fills up all space in an image. As such, it is considered that traffic information in LAN can be represented by a single image on the premise that temporal characteristics will not be lost when using Hilbert Curve. Here, as a method of expressing time-related features of an event in LAN, we put features of nine types of protocols which are captured in a predetermined time span into an image (Fig. 2.1). That means, statistical information of nine types of protocols collected in the LAN can be represented in different regions of an image (48 × 48) through array exchange and projection.

2.1.3.2 Training CNN Model

When it comes to time related data, two machine learning methods are mainly used. Recurrent neural networks (RNNs) and convolutional neural networks (CNNs). For recurrent neural networks, the data along the time axis is input in order of time. On the other hand, in the case of a convolutional network, it has a character of movement invariance with respect to the input of time related data, exhibiting good performance. In this research, we use CNNs. The configuration of the convolutional network is considered to mainly include convolutional layers, some pooling layers and fully-connected layers. Through using a kernel (also called a filter) in each convolutional layer and the pooling layer in some of the layers, we can compress the information of the input data. It is thought that the information contained in the image can be expressed by combining the layers. Finally, in order to get the outcome as one-dimensional information, some fully-connected layers are combined in the convolutional neural networks, and the output can be narrowed to a specific range. In this way, a total of 2,340 data collected

by the previous steps are used as an input, and the types of events to be detected are used as labels. The considerable CNN model [8] is shown as above (Fig. 2.2).

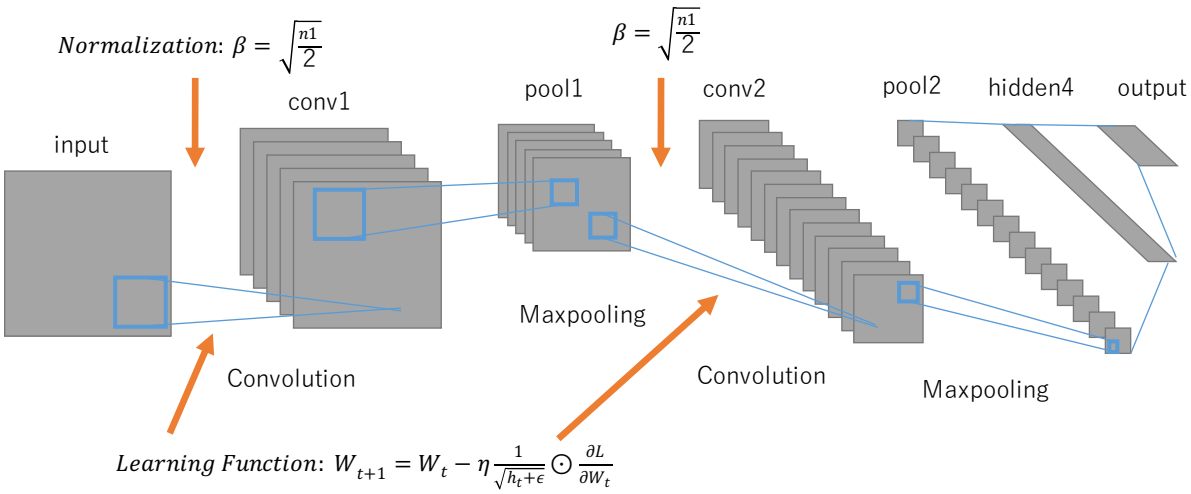


Fig. 2.2. A CNN model we used in our research: different from the former model, we used normalization before each convolutional layer to transfer the input data into a distribution of standard deviation. And we used a learning function named RMSProp, in order to attain sustainability of training and a higher accuracy.

In considering the amount of data that can be collected at this stage and the complexity of the problem as a learning model, we designed the capacity and configuration of the model. The entire model is a four-layer CNN model consisting of two convolutional layers and two fully coupled layers. The error function uses a cross entropy error (2.2). K is the number of nodes in the output layer, k is the node number in the output layer, y_k is the output value, and t_k is the correct value.

$$L = -\sum_k^K t_k \log y_k \tag{2.2}$$

Then, the parameters are updated by the learning function using the loss function. As a learning function, we choose RMSProp function (3) (4) with the following characteristics, the emphasis is placed on the latest gradient information more than the past gradient information and gradually the past gradient information is forgotten, instead, the new gradient information is greatly reflected. L is the loss function defined above, W is the weight of by zero, ρ is the decay rate which we take a value of 0.9 here.

$$h_t = \rho * h_{t-1} + (1-\rho) * \frac{\partial L}{\partial W_t} \odot \frac{\partial L}{\partial W_t} \tag{2.3}$$

$$W_{t+1} = W_t - \eta \frac{1}{\sqrt{h_t+\epsilon}} \odot \frac{\partial L}{\partial W_t} \tag{2.4}$$

In addition, it is possible to obtain more stability when learning the model through the method of making the parameter values disperse to the distribution of standard deviation at the time of initialization (2.5) [8]. And n1 is the node number of the previous layer.

$$\beta = \sqrt{\frac{n1}{2}} \tag{2.5}$$

Based on training the model in this way, the trained model is stored in the monitoring system (in this case, the personal computer on which the monitoring system is installed) and after that, using the trained model, from new traffic data the model can analyze the data and detect the anomaly in LAN.

2.1.4 Evaluation

Recall is one kind of evaluation standard, used to represent how many of true positives are recalled. In the case of network event classification, we calculate the recall rate of all the eight events as normal, default scan, and scan to some specific ports of both UDP and TCP. Through using recall, we can know how many times each event is successfully classified. And the average recall rate can be considered as the performance evaluation of the scheme.

2.1.4.1 Experimental Approach

For evaluating our algorism, we prepared three different network environments (IP A, IP B, IP C). Network A is a relatively small-scale LAN for the personal daily operating. For network B, it is a LAN of the lab, connected with many personal devices, which is used for a purpose of researching and daily operating. And network C is a large-scale network of the department building, consisting of several servers and IoT (Internet of things) devices.

Then we visualized and analyzed the communication information in these three different LANs. The constitution of the LAN in our experiment includes two terminals and a router. Specifically, it is conceivable that one terminal is connected to the LAN as a monitoring system, and the other one performs operations which recognized as attacks as follows:

scan (arp): nmap [Network]

scan (tcp): nmap -sT [Network]

scan (tcp port 23): nmap -sT -p 23 [Network]

scan (tcp port 80): nmap -sT -p 80 [Network]

scan (udp): nmap -sU [Network]

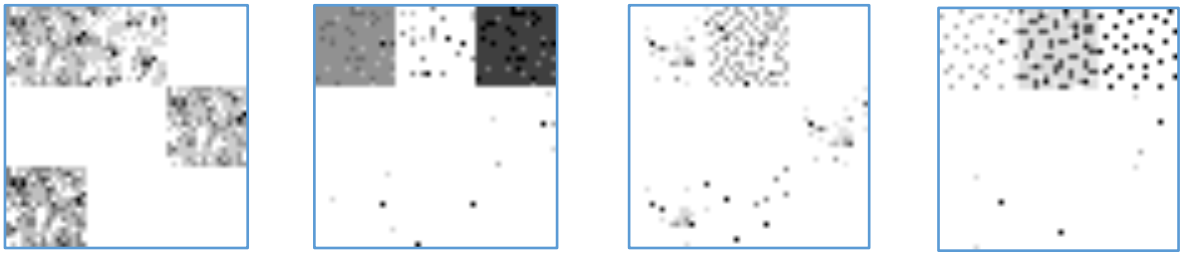
scan (udp port 137): nmap -sU -p 137 [Network]

scan (udp port 1900): nmap -sU -p 1900 [Network]

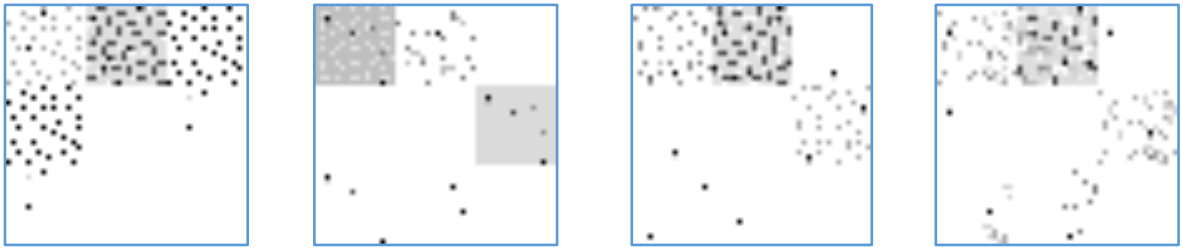
Here, we captured packets information in the LAN when each type of event occurs and saved tcpdump files on the terminal. In this research, we ran total seven different kinds of commands to simulate different events in the LAN. For each command, we ran 5 times based on a capture interval of one day. After running all commands, we got 5 capture files for each event as the dataset.

2.1.4.2 Feature Map

Through putting feature information of each protocol into a single image mentioned above. That is, feature information of nine types of protocols regarding traffic data within a specific time span is included in one image (Fig. 2.3).



(a) Normal (b) Default Scan (c) Default TCP Scan (d) TCP Port 23 Scan



(e) TCP Port 80 Scan (f) Default UDP Scan (g) UDP Port 137 Scan (h) UDP Port 1900 Scan

Fig. 2.3. Line 1, from left to right: A feature map of event as normal, default scan, default TCP scan, specific scan of TCP port 23; Line 2, from left to right: A feature map of specific scan of TCP port 80, default UDP scan, specific scan of UDP port 137, speci specific scan of UDP port 1900. (when fineness is 0.5 and size is 16)

A statistical graph and feature map obtained by adjusting the fineness discussed in the previous step shows that when the size is 16 and the fineness is 0.5, the feature representation of traffic data seems to be the best, considering the computation cost and clarity of features. Then, multiplied by the standard value of 64.000s, we can get a feature map (48 × 48) representing time-sequential information of nine types of protocols recorded in 128.0s. It is considered the most suitable for representing the features of traffic data when an event occurs.

2.1.4.3 Event Classification Accuracy

We use all three experimental environments and discuss the accuracy of model detection. As an environment, we used a personal computer separated from the space on our campus to simulate the actual operation situation. Here, two terminals connected to the same LAN are prepared, one running scan or a scan of a targeted port. Another terminal runs the tool of tcpdump simultaneously in the same LAN as a monitoring server, as a mechanism built to record traffic data of the LAN. The influence on the recorded data by operation of other facilities in the LAN or other users can be considered. We train this CNN model using a learning rate of 0.001. The training graphs of the model in different experimental environments are shown as below (Fig. 2.4). Also, the recall of each event for a trained model inferring in each network environment is shown as below (Table 2.1).

Table. 2.1. Evaluation of The Algorithm Under Different Network Environments

Event	Network A			Network B			Network C		
	Train Data	*Test Data*	*Recall*	*Train Data*	*Test Data*	*Recall*	*Train Data*	*Test Data*	*Recall*
normal	280	44	0.787	600	123	0.549	1479	202	0.786
scan (default)	215	30	0.998	1200	101	0.995	560	56	0.714
scan (tcp)	220	30	0.985	550	138	0.905	1651	300	1.00
scan (tcp port 23)	215	50	0.718	1098	250	0.590	1693	362	0.976
scan (tcp port 80)	430	100	0.665	1098	236	0.702	1190	363	0.979
scan (udp)	395	100	0.575	701	114	0.649	1015	226	0.969
scan (udp port 137)	425	100	0.988	1200	258	0.883	1190	291	0.624
scan (udp port 1900)	425	100	0.871	1200	250	0.835	1231	360	0.477

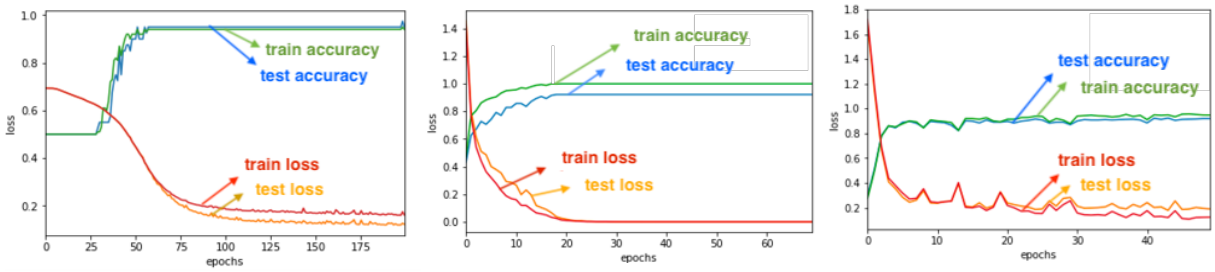


Fig. 2.4. From left to right: Training graphs in Network A, Network B, Network C environments. From the graph we can find the difference between train loss and test loss is not so large to contribute to the occurring of the over-fitting, which shows a strong flexibility of the model.

2.1.5 Discussion

We analyzed the traffic data in the LAN and realized the visualization of the feature by bringing the time related protocol information into feature maps. The parameters of adaptive fineness and size were adopted after discussing the effect of different values on traffic data feature representation. After that, a CNN model, a kind of deep learning method, was introduced and constructed according to the characteristics of the task. We discussed and compared the inference accuracy of the algorithm using recall in three different active network environments, contributing to an average recall rate of 76%.

2.1.6 Conclusion

In the level of practical use, we aim to realize a highly versatile monitoring system regardless of the maintenance status of the network environment in the LAN. In recent years, malware incidents frequently occur in Asian countries such as Thailand, Vietnam, and China. It is obvious that the introduction of a highly versatile monitoring system is required, including the cause of imperfections in the network environment. The security issues of such networks are likely to be mitigated.

Besides, as deep learning is used to deal with the task of network security, this method to safeguard the network is considerable in the field of the network security. Furthermore, with the development of technology, it seems that the control of technology will also be very significant while researchers incorporating AI technology.

In future, not only inferences will be made using trained models, but also the design of models that can be changeable in response to environmental situation in the network is considerable. It is possible to improve the expressive facility of features by further devising which protocols should be chosen according to the attribute of the network existing in the actual network. Also, in order to reduce the burden of monitoring on the administrator, other than the type of the event occurring will be announced, it is possible that system can automatically generate a log and at the same time narrow down the collected traffic data to the targeted part where the event occurred.

2.2 Network Events Classification Based on Channel-Based Feature Maps And Convolutional Neural Networks

2.2.1 Introduction

The implementation of resilient and robust networks is continually imperative. However, an effective and explainable solution to network anomaly detection has not been made clear. Especially, attacks such as phishing, delivering and spreading malware in networks by spammers have been monitored frequently in recent years. At the same time, network systems are becoming so complicated that even the manager of networks has a problem with manipulating them.

General approaches being used for detection usually have a limit to extremely changeable malware and are short of explainability. Therefore, it is considerable and meaningful to represent hidden features of network traffic using visual analytics, thus we can observe and detect anomaly from visualization results. Moreover, through a combination with machine learning, automatic classification between various abnormal events can be considered.

We adopt channel-based visual analytics based on the Hilbert curve to represent protocol information of network events. Each feature map of network events consists of three channels, which are used to save information about a specific protocol during a predetermined recording period (Fig. 2.5). The protocols used in this research are ARP, TCP, and UDP. Consequently, we use these feature maps to represent network traffic features of different network events.

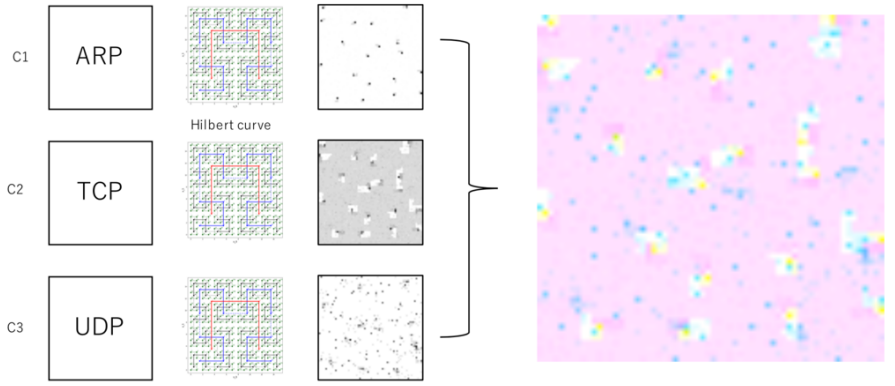


Fig. 2.5. The scheme of generating a feature map from network traffic using the channel-based visual analytics (under a case of arp scan).

We simulate eight types of known network events in LAN, using a terminal connected to this LAN as the attacker who manipulates various commands, and another terminal to collect all network traffic flowing through the LAN. Then we generate feature maps of these network events using the approach discussed above.

After that, we adopt a deep CNN model based on the VGG-16 which consists of thirteen convolutional layers accompanied by five maxpooling layers and three fully connected layers. We use several optimizing approaches to machine learning to stabilize the progress of training as well as improving performance of this model. We train this model from scratch using generated feature maps. It is supposed that this CNN model could give out confidence of each network event cluster, thus identifying the type of a network event based on confidence.

This chapter is organized as follows. Section 2 discusses related work about visualization of network traffic and anomaly detection in networks based on machine learning. Section 3 provides an overview of the scheme, consisting of generating feature maps and implementing the CNN model. Section 4 presents performance evaluation using precision, recall, and F-measure in two networks. Section 5 discusses the advantages and disadvantages of this scheme. Section 6 concludes the chapter.

2.2.2 Related Work

The research of anomaly detection in networks with machine learning has a long history. In previous research, a non-linear technique called support vector machine (SVM) is used to explore hidden relations between time-series traffic data. For instance, Palmieri et al. [9] have proposed a method to extract the transition patterns based on the analysis of non-stationary properties and hidden recurrence patterns occurring in the aggregated IP traffic flows. Moreover, Hsu and Lin [10], they demonstrated an approach of using SVM models to detect DDoS attacks in an efficient and accurate way.

Besides SVM, there are also other machine learning methods used to solve network anomaly detection. For instance, in the research of Mehdi et al. [11], they used Deep Learning (DL) to extract and analyze the features of IoT domains. In addition, Krokos et al. [12] adopted an autoencoder to process raw pcap files to Query-Space visualization, revealing potential DDoS attacks and unforeseen changes in the distribution of queries received.

The research discussed above is aimed to detect anomaly in networks from the perspective of machine learning. However, most of them lack enough explainability regarding feature representation of network traffic. In this research, we are aimed to visualize features of network traffic to achieve more intuition of anomaly in LAN, by exploring a hidden relationship among various protocols of network traffic. We adopt a supervised machine learning called the convolutional neural network for further classifying different abnormal events.

2.2.3 Visual Analytics for Feature Maps Generation

2.2.3.1 Visualization of network traffic in LAN with Hilbert curve

We adopt two active network environments here and implement simulations of various network events in each network for collecting traffic data. In detail, one terminal is used as a simulator for generating various network events in LAN and another one is used to collect traffic data when simulating a specific event (Fig. 2.6).

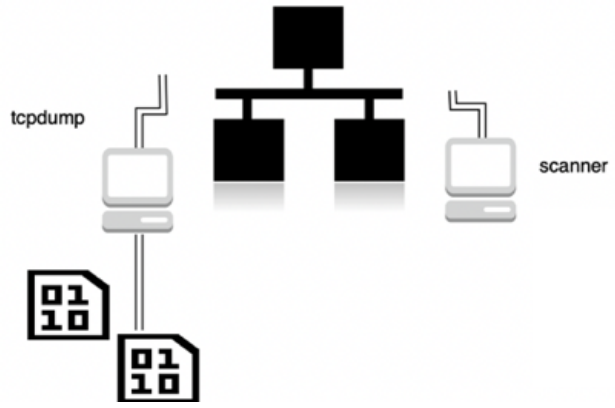


Fig. 2.6. The network events simulating and network traffic collecting system.

In this research, we use a tool called tcpdump for collecting the traffic data in LAN. And for simulating these network events, we adopt a tool called nmap with several types of manipulating commands. Moreover, simulations of each type of network event is conducted continually for two whole days. From this collected traffic data, we extract all protocol information and time stamp information to further visualize features of network traffic.

We compute how many communications of each protocol have been monitored in LAN during a fixed recording time unit, which is influenced by the value of fineness, as discriminators for representing features of network traffic. Considering the network events we experiment on in this research, from all protocol information, communication information about ARP, TCP, and UDP is extracted and computed to represent features. Then, we convert frequency information of these protocols into pixel values using (2.6), for visualization with feature maps.

$$p_i = \frac{c_i}{Max(c)} \times 255 \tag{2.6}$$

Where c_i is the frequency of a specific type of protocols' communication, and the denominator is the maximum from all frequency values within the duration for generating a feature map, and p_i is the pixel value of a pixel point in an image.

Here, the parameter of fineness mentioned above is used to show how precisely we compute the frequency of protocol communication. For a discussion about the influence of fineness on the representation result of network traffic's features, we verify results of visualization when fineness has a value of 5.0, 1.0, 0.5, 0.2, 0.1, 0.05, 0.02, and 0.001(Fig. 2.7). In this research, we found that when adopting fineness with a value of 0.5, in which case the fixed recording time unit is 0.5 seconds, the representation of network traffic's features shows the best result.

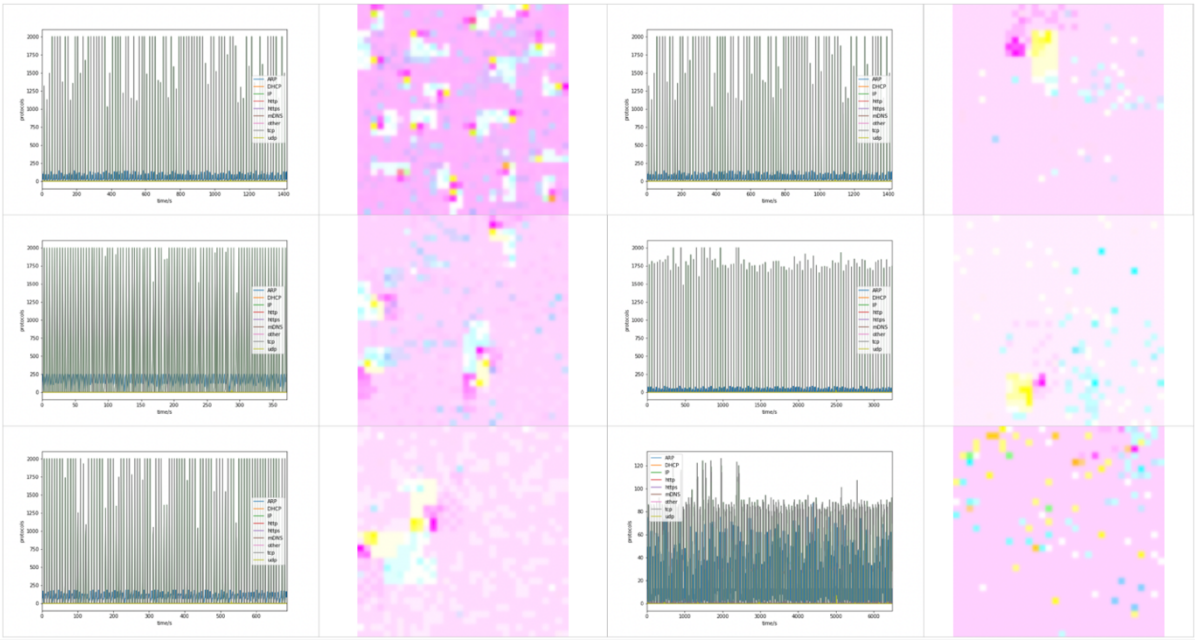


Fig. 2.7. The record of traffic data under arp scans and the corresponding feature maps when adopting different fineness. Col 1: from top to bottom, when the fineness is 5, 1, and 0.5; col 2: from top to bottom, when the fineness is 0.1, 0.05, and 0.01.

Then, we adopt a geometric structure called Hilbert curve to project these pixel values converted from communication frequency information into different pixel points in a feature map. Here, the merit of Hilbert curve is that we can suppress time-related data into a 2-Dimension image, amplifying the hidden relationship between data. As discussed above, a fineness value of 0.5 is adopted, thus each feature map consists of 1024 records, each of which shows features of a specific protocol's communication within 0.5 seconds.

After projecting every pixel value to the corresponding positions in a feature map image, we combine these three feature maps of ARP, TCP and UDP into one feature map through putting each of them into different channels of an image, merging all three types of protocol information and representing the features of a network event's traffic data with this combined feature map.

2.2.3.2 Dataset

Network traffic from two network environments, LAN A and LAN B, was collected and used for generating feature maps. Here, LAN A is a network with a variable-length subnet mask with a length of 25 digits. And it is a network of the institute's critical infrastructures. On the other hand, LAN B is a network serving for general purposes in several labs, such as research and other daily operations.

In this research, we simulated eight types of network events in LAN A and LAN B separately, including normal, arp scan, tcp scan, scan of tcp port 23, scan of tcp port 80, udp scan, scan of udp port 137, and scan of udp port 1900. Then we collected network traffic from these two networks, generating feature maps of network events based on the approach discussed above (Fig. 2.8). At last, we achieved 2000 feature maps in LAN A and 2800 feature maps in LAN B in total.

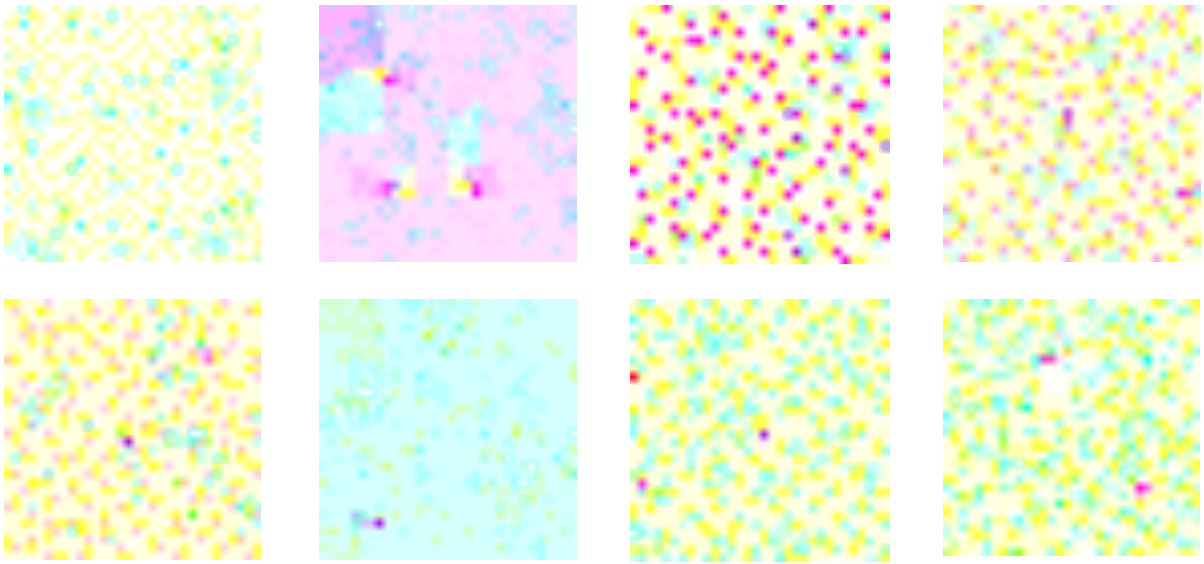


Fig. 2.8. Feature maps of network events generated with a fineness value of 0.5. Line 1, from left to right: feature maps of normal, arp scan, tcp scan and scan of tcp port 23; Line 2, from left to right: feature maps of tcp port 80, udp scan, scan of udp port 1900. (in LAN A)

2.2.4 Network Events Classification using Deep Convolutional Neural Network

2.2.4.1 Adaptivity of convolution operation to feature maps

The convolution operation corresponds to the filter operation in image processing. It computes the convolution of the input data and the filter as the output. As a result, it has great adaptivity to the geometric structure of Hilbert curve. Furthermore, results from multiple channels are added up to obtain the output, adaptable to this channel-based feature representation of network events (Fig. 2.9).

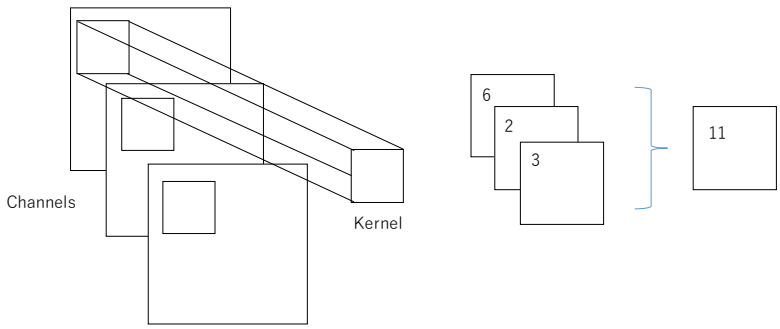


Fig. 2.9. The function of convolution operation, where a kernel is used to compute the convolution of each area of the input data and the filter.

2.2.4.2 The structure of the deep CNN model

Considering the task of multiclass classification between eight types of network events using generated feature maps, we adopt a deep CNN model in order to identify subtle differences hidden in the visualized network traffic data. At the same time, we utilize several optimization methods of deep learning to stabilize the progress of model training, as well as improving the performance of this model.

Moreover, CNN is one kind of supervised learning. Usually, a CNN model consists of several convolution layers, sometimes accompanied by pooling layers, and several fully connected layers at the end of the model. Moreover, the pooling operation in the pooling layer is used for aggregating features and reducing the size of the input in the vertical and horizontal directions. Furthermore, fully connected layers are used to narrow the output to a specific range, which consists of numerical labels of the input data. In this research, we transfer the eight types into numerical vectors as the labels to train the model.

We build a deep CNN model based on the VGG-16 model [13], consisting of thirteen convolution layers accompanied by five maxpooling layers and three fully connected layers, with an output of eight values (Fig. 2.10). By adopting this deep CNN model, the hidden features of network events inside a feature map can be extracted. Then three fully connected layers are used to flatten the output matrix of the convolution layers and maxpooling layers to a one-dimension array and compress information to a matrix, which gives out the confidence percentage of this model's prediction regarding each type of network events.

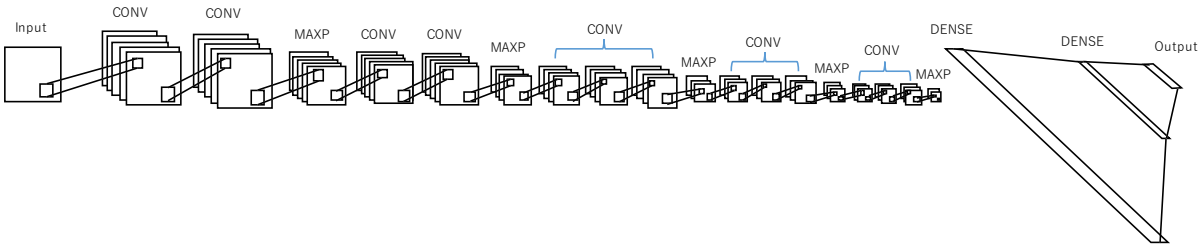


Fig. 2.10. The structure of the deep CNN model we adopt in this research, consisting of thirteen convolutional layers accompanied by five maxpooling layers and three fully connected layers.

Moreover, the activation function used in this research is ReLU which is defined as (2.7). After the computation of each layer, we use ReLU function to convert the output to a non-linear distribution. And at the last layer, for this multiclass problem, we use an activation function called softmax, which is defined as (2.8). Through using softmax instead, each component including negative, greater than one, or might not sum to 1, will be in the interval (0, 1), with a sum of 1.

$$f(x) = max(0, x) \tag{2.7}$$

Where x is input, f(x) is an output of the node, and "max" is a function used to get a maximum between 0 and x.

$$y_i = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x^j}} \tag{2.8}$$

Where x_i represents each element of the input vector x. Softmax is used to normalize these values through dividing by the sum of all the exponentials of these elements.

In order to speed up and stabilize the training progress, the learning function called RMSprop is used, where the emphasis is placed on the latest gradient information more than the past gradient information

and gradually the past gradient information is forgotten, instead, the new gradient information is greatly reflected. This learning function is defined as (2.3) (2.4).

This deep CNN model was trained from scratch based on the datasets from LAN A and LAN B separately, with a batch size of 40 and a learning rate of 0.00002. The early stopping was adopted while training to prevent the overfitting, by monitoring the validation loss for every five epochs. After training, the model is supposed to identify eight types of network events in LAN based on confidence scores of the model prediction. The corresponding training graphs in the two network environments are shown below (Fig. 2.11) (Fig. 2.12).

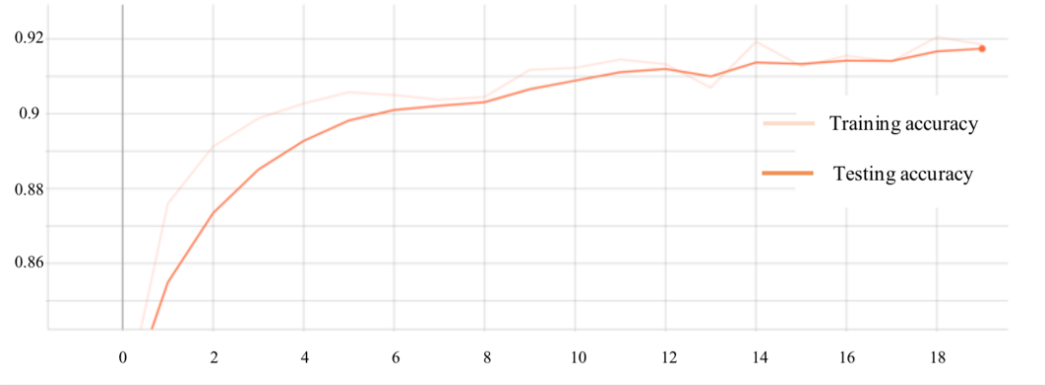


Fig. 2.11. The training accuracy and the testing accuracy in LAN A.

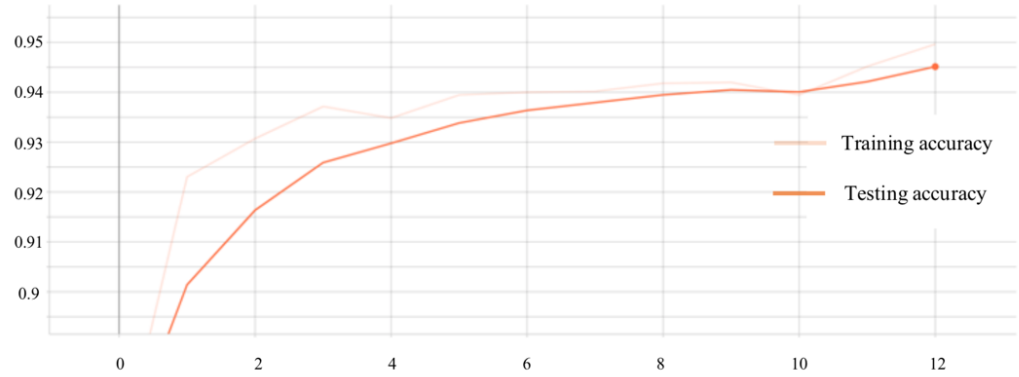


Fig. 2.12. The training accuracy and the testing accuracy in LAN B.

2.2.5 Performance Evaluation

We evaluate the scheme in two active networks, LAN A and LAN B, each of which includes the system structure we mentioned before, with a terminal for simulation of network events and another one for collecting network traffic in LAN. We evaluate the performance of it using precision, recall, and F-measure.

The precision is a parameter used to show how many events are successfully classified in all test data; the recall is a parameter that is used to show how many times a specific event is successfully classified in all test data of that event. They are defined as (2.9) and (2.10).

$$Precision = \frac{TP}{TP+FP} \tag{2.9}$$

$$Recall = \frac{TP}{TP+FN} \tag{2.10}$$

Where TP (True Positives) indicates the number of events successfully detected by the scheme, and FN (False Negative) represents the number of events unsuccessfully classified.

On the other hand, the F-measure is a parameter used to show the comprehensive performance of a model. And it is defined as (2.11).

$$F-measure = \frac{2\times Precision\times Recall}{Precision+Recall} \tag{2.11}$$

While training, we adopted the evaluation methods above for each epoch. We compute the precision, recall, and F-measure scores of each event as the measure of the classification ability of the scheme. The corresponding result of the evaluation is shown above (Table 2.2).

Table. 2.2. Evaluation of the scheme in two different network environments

Events	LAN A			LAN B		
	Precision	*Recall*	*F-measure*	*Precision*	*Recall*	*F-measure*
normal	0.8384	0.7829	0.7938	0.8563	0.8108	0.8273
scan (arp)	0.9935	0.9922	0.9921	0.8276	0.8084	0.8161
scan (tcp)	0.9123	0.7122	0.7822	0.9945	0.9964	0.9952
scan (tcp port 23)	0.5790	0.6184	0.5686	0.7578	0.5239	0.6059
scan (tcp port 80)	0.6519	0.6307	0.6122	0.3613	0.3324	0.3103
scan (udp)	0.6340	0.8442	0.7058	0.9900	1.0000	0.9945
scan (udp port 137)	0.9097	0.4766	0.6023	0.4294	0.8875	0.5600
scan (udp port 1900)	0.6145	0.9370	0.7236	0.8669	0.7308	0.7664
Average	0.7667	0.7493	0.7576	0.7605	0.7613	0.7597

From the result above, we can see that the classification between normal, arp scan, tcp scan, and udp scan shows relatively great performance. Whereas, the classification between scans of some specific ports such as tcp port 80 and udp port 137 comes out relatively low F-measure scores, which shows that it is more difficult to classify between scans of specific ports than to classify between the normal and the abnormal in LAN. At last, an average F-measure score of 0.76 is achieved for this anomaly classification in LAN.

2.2.6 Discussion

The visualization of network traffic allows some explainability to anomaly detection in LAN. Furthermore, a deep CNN model is adopted to classify these reoccurring patterns in feature maps of various network events. And simulation experiments under two different networks are conducted to evaluate the scheme.

On the other hand, it is still possible that an adversary could forge these features inside a feature map by adjusting the communication frequency. A more delicate and complete experiment in a real-world setting should be verified. Furthermore, besides the proposed eight types of network events in this research, how additional, non-explicit network events would influence the classification result should be considered.

2.2.7 Conclusion

In this research, we propose channel-based visual analytics, which extracts protocol information of ARP, TCP, and UDP from network traffic, generating feature maps of eight types of network events in LAN using this information. We are aimed to visualize features of network traffic by representing a relationship between communications of protocols based on Hilbert curve. We adopt a deep CNN model to approach this multiclass classification, conducting performance evaluation using the precision, recall, and F–measure in two different network environments. As a result, the scheme achieves an average F-measure score of 0.76 for the anomaly classification in LAN.

Chapter 3 Network Intrusion Detection Based on Raw Network Traffic

3.1 Introduction

With the development of information science, more and more devices are connected to networks, communicating, and sharing data with each other. As a result, there is an imperative need for automation and digitalization in various industries to build robust and efficient network monitoring systems for detecting an anomaly in networks in a timely manner.

Traditional methods of anomaly detection in a network are usually dependent on experts' experience and knowledge to design handcrafted features of network traffic. Unfortunately, this process is time-consuming and venerable to changes in network traffic patterns. The proposed approach of employing raw network traffic for anomaly detection allows more flexibility in feature design and representation.

We applied the dataset including two weeks' network traffic in LAN collected from a monitor device. Then, for representing features of the network traffic within different time windows, we analyzed the traffic data and converted it to small chunks of packets consisting of the source IP address, the destination IP address, and other concerned protocol information.

We employed several knowledge-based labeling methods, targeting on four types of an anomaly in LAN, ARP flooding, TCP SYN flooding, malicious SMB, and malicious UDP unicast. We labeled all the generated chunks of traffic data as benign or abnormal with each labeling method (Fig. 3.1).

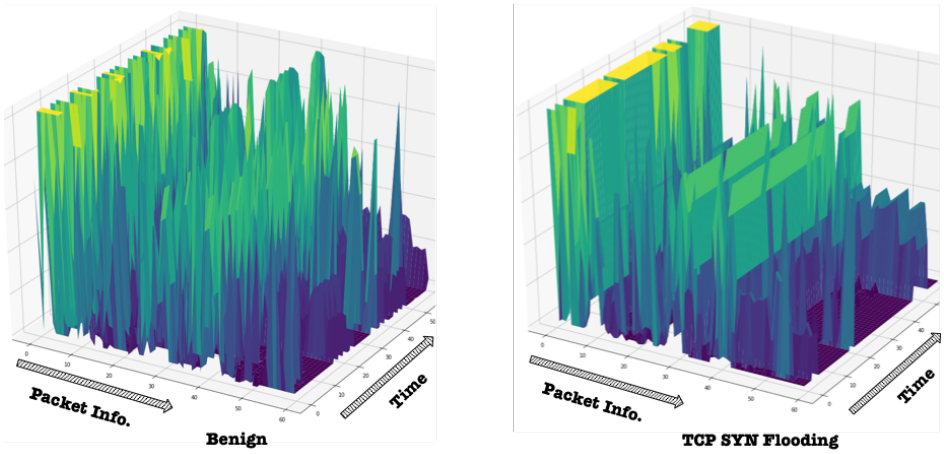


Fig. 3.1. Samples of the labeled raw network traffic chunks (left: benign; right: TCP SYN flooding).

Then, we studied the performance and resilience of the scheme by applying a four-layer supervised convolutional neural network (CNN) trained on the collected first week's traffic data and conducting an evaluation based on the collected second week's traffic data. Different from former research where the validation set and training set are generated from the same period, we separated the two sets into different periods, thus showing the adaptability of the method.

This chapter is organized as follows. Section 2 discusses related works about anomaly detection with neural networks. Section 3 provides an overview of the scheme including regulated segments generation using raw network traffic, knowledge-based labeling for multi-type anomalies, and event classification based on the supervised convolutional neural network. Section 4 presents the performance evaluation of the scheme against the precision. In Section 5, we conclude the chapter and give out the future work.

3.2 Related Work

Traditional approaches to anomaly detection are commonly related to the matching between known abnormal behavior patterns and network traffic observation [14][15]. A network-based activity study is usually conducted to determine if a node is compromised, such as traffic or frequency analysis, and deep packet inspection. Unfortunately, these approaches appear to be insufficient with underlying issues of detection efficiency and adaptivity. Deep learning is attracting lots of attention due to its ability to manipulate enormous traffic data in a network. For instance, Marín et al. [16] presented a method of training a deep learning model on a dataset consisting of raw data in a packet and a flow, instead of adopting carefully handcrafted expert-knowledge based features. Moreover, Iglesias et al. [17] demonstrated a method of compressing 41 handcrafted features into only 16 features, thus reducing the computation cost of the model. Besides, they utilized several machine learning methods for evaluating the scheme, such as an artificial neural network and support vector machine. Bhuyan et al. [18] presented a comprehensive survey on categorizing the existing network anomaly detection methods based on the underlying computational techniques adopted. Chumachenko et al. [19] presented a comparison among the state-of-the-art methods of network traffic feature extraction and representation and malware detection. They evaluated these methods based on a dataset covering nine types of malware. Different from the former research, instead of designing and employing the handcrafted features of network traffic, we adopted a method of representing the features with chunks of raw network traffic data in LAN. By further combining with a supervised neural network, we aim to build a scheme for efficient multi-type anomaly detection in networks.

3.3 Anomaly Detection with Raw Network Traffic

3.3.1 Regulated segments of raw network traffic

To process raw network traffic for anomaly detection, we employed a standard segment length of 480 bits (60 bytes), based on the middle value of packets' lengths in the dataset. We added a binary 0 when a packet had a length shorter than the standard, and cut the excess from a packet when it had a length longer than the standard. Besides, we converted each byte information in a packet to a decimal number within a range of 0 and 255. As such, we obtained the network traffic packets, each of which has a length of 60 decimal units. Instead of observing and measuring features of a single packet, we studied the feature representation and measurement of dozens of continual packets. The size of the packet chunk depends on the network scale, the activity frequency, and so on. We applied a size of 60 for representing the features of network traffic within a period of around 30 seconds in the network. As a result, for every 60 packets, we generated a segment of network traffic and represented the features of it using the regulated packets (Fig. 3.2). The information included in a segment is source IP addresses, destination IP addresses, other protocol information such as port numbers, payload lengths, and so on.

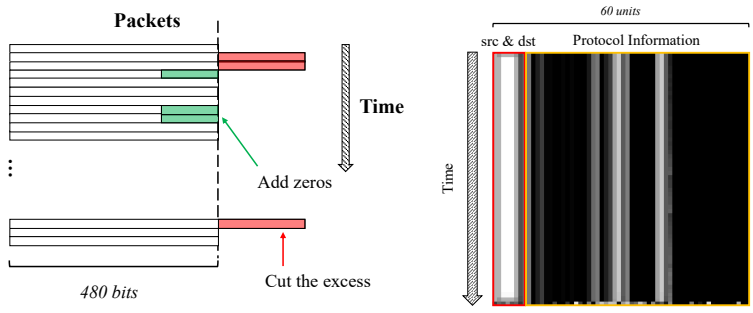


Fig. 3.2. Left: regulating network traffic packets into the standard length of 480 bits (60 bytes). Right: a sample of the regulated segment of raw network traffic, including time-series information on the source and destination IP addresses and protocol.

3.3.2 Knowledge-based labeling

We exploited and analyzed several types of abnormal behaviors in LAN from the collected traffic, covering ARP flooding, malicious SMB, TCP SYN flooding, and malicious UDP unicast, as defined in Table 3.1 ARP flooding and TCP SYN flooding was observed when there was a spike of ARP/TCP SYN packets within a specified period of network communication, while malicious SMB and malicious UDP unicast was observed when a SMB/UDP unicast packet was sent to the monitoring device. Considering the monitoring device doesn't have a function of file sharing, the received packets are highly suspicious, consequently defined as abnormal behavior in LAN.

Table. 3.1 Abnormal Behaviors in LAN

Abnormal Behaviros	Content	Explanation
ARP Flooding	Sending ARP requests to more than three IP addresses in a session	The Address Resolution Protocol (ARP) is responsible for mapping a computer's IP address with its MAC address. In ARP flooding, the offender sends ARP packets to all machines connected in a network, causing the affected machine to be unable to resolve IP and MAC addresses.
Malicious SMB	Sending any packet to TCP Port 445 (SMB) of the monitoring device	TCP port 445 is used to run SMB directly over TCP/IP, without the extra layer of NetBT, for file sharing. Usually, this port is blocked or disabled, TCP port 445 of the monitoring device is open for attracting any possible attacks to it. Since the monitoring device doesn't have a purpose for file sharing, an SYN 445 to the device has a high possibility of being malicious.
TCP SYN Flooding	Sending more three TCP SYN packets in a session	TCP SYN flooding is one type of Distributed Denial of Service (DDoS) attack that affects the normal TCP three-way handshake, thus consuming resources on the target. The offender sends TCP connection requests faster than the targeted machine can process them (more than three times in a defined session), causing network saturation.
Malicious UDP unicast	Sending any UDP unicast packet to the monitoring device (except NTP packets with a source port of 123 and DNS packets with a source port of 53)	Different from broadcast, unicast represents a connection where information is sent from one point to another point directly in a network. And UDP unicast is used for sending datagrams to a targeted machine directly. Since the authentic UDP unicast connections to the monitor device only include DNS for domain name translation, and NTP for synchronizing the clock, any other UDP unicast connection requests would be possibly malicious.

We employed the knowledge-based methods to label these generated segments of raw network traffic. In detail, we applied a tool called dpkt to parse the collected network traffic data, thus extracting the protocol information such as ARP, TCP, UDP, and so on for further measurement. We computed the total packet numbers of each concerned protocol for every network traffic segment and labeled the ones matched with each of the aforementioned rules as abnormal respectively. Besides, we also labeled the ones with any of the four rules matched as abnormal. Finally, we obtained five differently labeled datasets based on the segments.

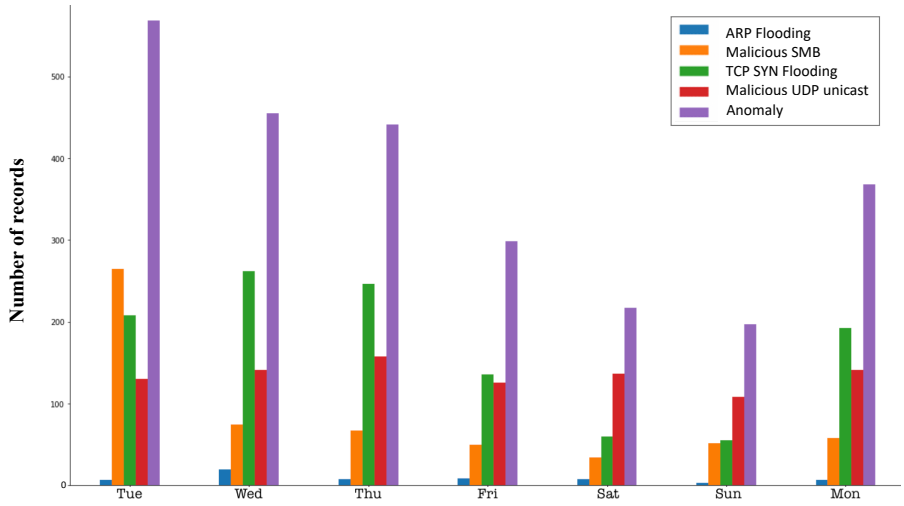
The network traffic was captured in two continual weeks from an academic institution, where the LAN functions for daily operations with lots of devices connected such as web cameras, office Internet of Things (IoT) devices, mobile devices, and so on. A monitoring device based on a raspberry pi was developed and connected to the network for collecting real-time traffic data using the tcpdump [20].

We prepared a dataset including two weeks' network traffic in the LAN. Then we applied the knowledge-based methods to label the collected data with the five different rules. We further separated the dataset into the training set and the validation set. The data from the first week was employed to train a supervised neural network, and the one from the second week was employed to evaluate the performance of the trained model. The constitution of the dataset is shown in Table 3.2.

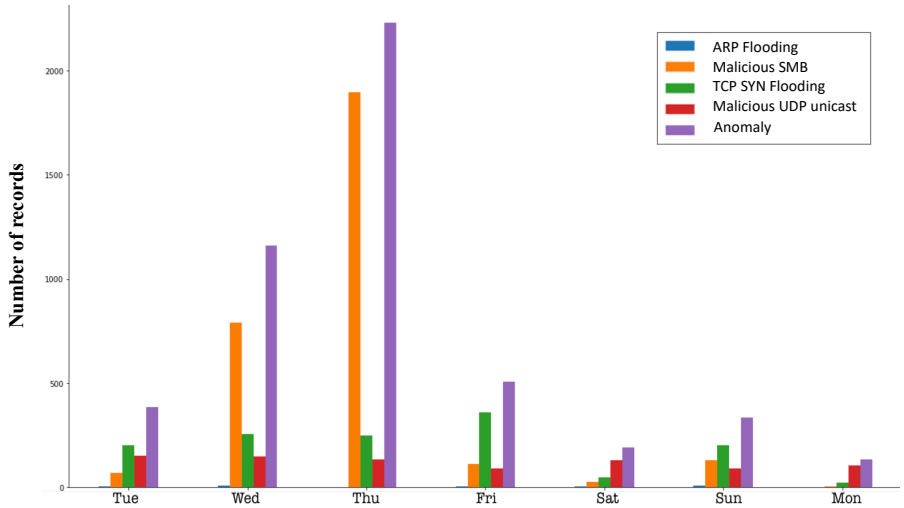
Table. 3.2. Dataset Profile

Labeling Methods	Training Set		Validation Set	
	Benign	Abnormal	Benign	Abnormal
ARP Flooding	28729	62	45395	37
Malicious SMB	28191	600	42402	3030
TCP SYN Flooding	27632	1159	44089	1343
Malicious UDP unicast	27850	941	44578	854
Anomaly	26247	2544	40485	4947

And the distribution of each type of anomaly within these two weeks is shown below (Fig. 4).



(a) The distribution of the various types of abnormal network behaviors within the 1st week.



(b) The distribution of the various types of abnormal network behaviors within the 2nd week.

Fig. 3.3. The distribution of each type of anomaly within these two weeks.

3.3.3 Anomaly detection with a supervised neural network

We combined the generated raw network traffic segments with a supervised neural network, using the results of knowledge-based labeling as the ground truths to train the model. The first week's data was used as the training set and the second week's data was used as the validation set. Based on the comparison between the inference results of the model and the ground truths, we could evaluate the performance and adaptability of the scheme for detecting various types of anomalies (Fig. 3.4).

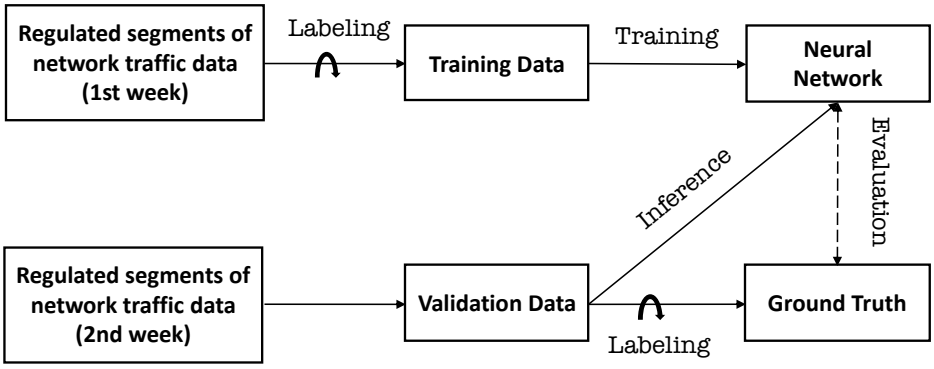


Fig. 3.4. The scheme of anomaly detection with a supervised neural network and performance evaluation.

A four-layers convolutional neural network was applied, which consists of two convolution layers and two fully-connected layers. There are 50 trainable parameters at the first layer, 46 trainable parameters at the second layer, 6500 trainable parameters at the third layer, and 21 trainable parameters at the last layer. The first convolutional layer has a convolution kernel of size 3×3 and it takes one input plane and it produces five output planes. Then, for the second convolutional layer, it has a convolution kernel of size 1×1 and it takes five input planes and it produces one output plane. The first fully-connected layer has an output size of 20, followed by another fully-connected layer with an output size of one. The activation function employed at the input layer and the two hidden layers is ReLU. The activation function Sigmoid was employed at the output layer to convert the output into the range of 0 to 1. Besides, as the loss function, Crossentropy was employed for computing the prediction loss between the ground truths and the model output. The structure of the four-layers convolutional neural network is shown below (Fig. 3.5).

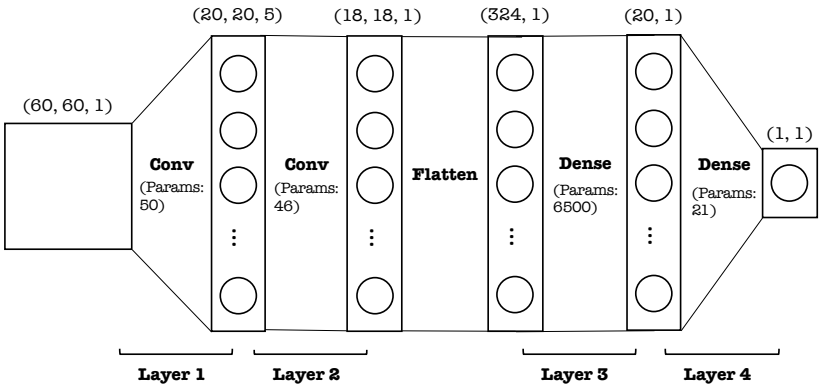


Fig. 3.5. The structure of the four-layers convolutional neural network.

A learning function Adam with decay rates of 0.9 and 0.999 was applied to update these trainable model parameters, which is defined as (3.1). For each epoch of training, the dynamics of the model is moving towards a smaller loss of prediction. Batch learning was applied to train the model with small chunks of input data, allowing more stability of training and also contributing to the improvement of model performance.

$$m_t = \rho_1 * m_{t-1} + (1 - \rho_1) * \frac{\partial L}{\partial W_t}$$

$$V_t = \rho_2 * v_{t-1} + (1 - \rho_2) * \frac{\partial L}{\partial W_t} \odot \frac{\partial L}{\partial W_t} \widehat{m_t} = \frac{m_t}{1-\rho_1^t} \tag{3.1}$$

$$\widehat{v_t} = \frac{v_t}{1-\rho_2^t}$$

$$W_{t+1} = W_t - \eta * \frac{1}{\sqrt{\hat{v}} + \epsilon} \odot \widehat{m_t}$$

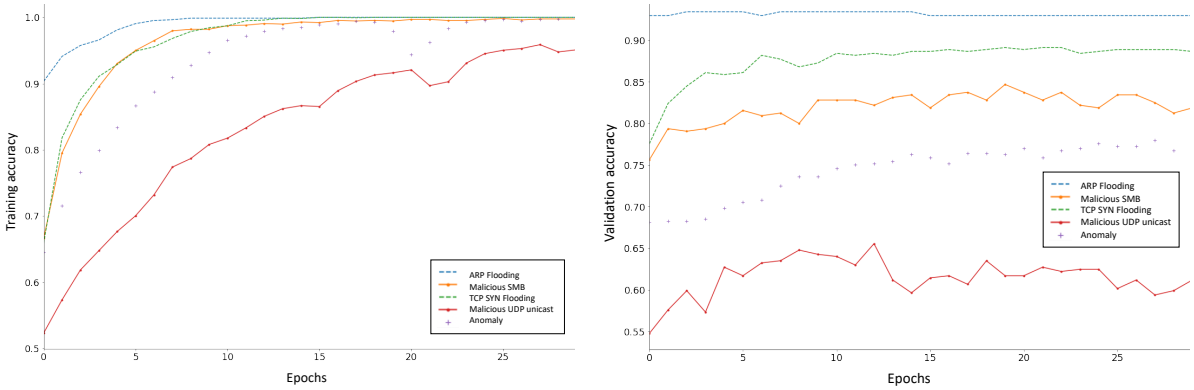
Where L is the training loss, W is weights of the node, η is the learning rate, $\rho_1$and ρ_2 are the decay rates with values of 0.9 and 0.999 individually, and ϵ is used to prevent the denominator from being zero.

3.3.4 Evaluation

For each type of anomaly, we employed a total of 1000 benign segments and all abnormal segments from the training set to train the model. We applied 30 epochs for the training with a learning rate of 0.002 and a batch size of 16. We further split the training set into two sets with a ratio of 4:1, for evaluating the training and validation performance during the learning respectively. A metric of accuracy defined in (3.2) was employed. The progress of training for each type of anomaly is shown below, with the results of the training accuracy and the validation accuracy (Fig. 3.6).

$$Accuracy = \frac{TN+TP}{TN+FN+TP+FP} \tag{3.2}$$

Where, TN (True Negative) shows the number of the benign segments correctly predicted, FN (False Negative) represents the number of the benign segments incorrectly predicted, TP (True Positive) shows the number of the abnormal segments correctly predicted, and FP (False Positive) represents the number of the abnormal segments incorrectly predicted.



(a) Training accuracy. (b) Validation accuracy.

Fig. 3.6. Training accuracy and validation accuracy at each epoch of the four-layers convolutional neural network.

As we could see from the result of the validation accuracy, the detection of the ARP flooding and the TCP SYN flooding shows a better result than the detection of the malicious SMB and the malicious UDP unicast. That's because these two types require more specific information in the raw data for pattern recognition. On the other hand, the events of flooding are comparatively revealing more hidden features related to the frequency for the neural network to identify the difference between an abnormal segment and a benign one.

Then, we applied the trained model to conduct anomaly detection using the second week's traffic data. The segments of raw network traffic were generated based on the aforementioned strategies and employed as input of the trained model. Then the model would conduct the computation based on the updated parameters to output the probability of a segment being abnormal. If the probability is larger than 0.5, then it would be classified as abnormal behavior. By comparing the inference results with the ground truths from knowledge-based labeling and observation, we studied the adaptability of the scheme for detecting an anomaly in unknown traffic data from a different period, against a metric of precision defined in (3.3).

$$Precision = \frac{TP}{TP+FP} \tag{3.3}$$

We exploited the precision scores for detecting the defined five types of anomalies in the LAN, where the precision scores for classifying the benign segments and the abnormal segments under each case were computed respectively (Table 3.3).

Table. 3.3. Adaptability Evaluation Results with Precision

Labeling Methods	Benign precision	Abnormal precision	Overall precision
ARP Flooding	0.98	0.79	0.980
Malicious SMB	0.79	0.96	0.801
TCP SYN Flooding	0.81	0.98	0.815
Malicious UDP unicast	0.56	0.90	0.566
Anomaly	0.63	0.94	0.664

As shown in the result, when applying the traffic data from a different week, we still achieved relatively high precision scores for detecting the latter four types of anomalies. While the precision scores for inferring the benign segments in these four cases didn't show great performance. For the overall averaged-precision scores, the scheme achieved a validation precision score of 0.980 for detecting ARP flooding, a score of 0.801 for detecting malicious SMB, and a score of 0.815 for detecting TCP SYN flooding respectively.

3.3.5 Conclusion

We proposed a multi-type anomaly detection method based on raw network traffic measurement. We extracted visualized features from the traffic data and by employing a supervised neural network, we trained a model for detecting these various anomalies. Then, the scheme was evaluated against precision using the validation set separated from the training set. At last, the scheme achieved a validation precision score of 0.980 for detecting ARP flooding, a score of 0.801 for detecting malicious SMB, and a score of 0.815 for detecting TCP SYN flooding respectively. This study provides insights into dealing with abundant network traffic while building the next-generation network safeguarding systems against cyberattacks.

Chapter 4 Malicious Domain Name Detection with Deep Learning

4.1 Introduction

Nowadays, a lot of malicious sources can be noticed while browsing websites or checking emails, and these malicious behaviors are usually related to the Domain Name System (DNS), which converts domain names to IP addresses for identifying various devices. Especially, a connection with these suspicious sources has a high risk of bringing great loss to network users. Therefore, domain names play a significant role in the detection of malicious sources.

However, malicious domain names have a tendency to changing frequently based on predesigned algorithms. As a result, it is considered to be difficult to distinguish malicious domain names from big data of domain names. Furthermore, the traditional approach of applying rules or setting blacklists to filter all domain names becomes more and more disadvantaged in an era of big data and a greatly increased communication amount.

We propose an approach of using a text-based method to represent domain names with numeric features and extracting the hidden feature vectors with the VAE. Then we utilize a supervised learning of CNN to classify these feature vectors between the benign and the malicious (Fig. 4.1).

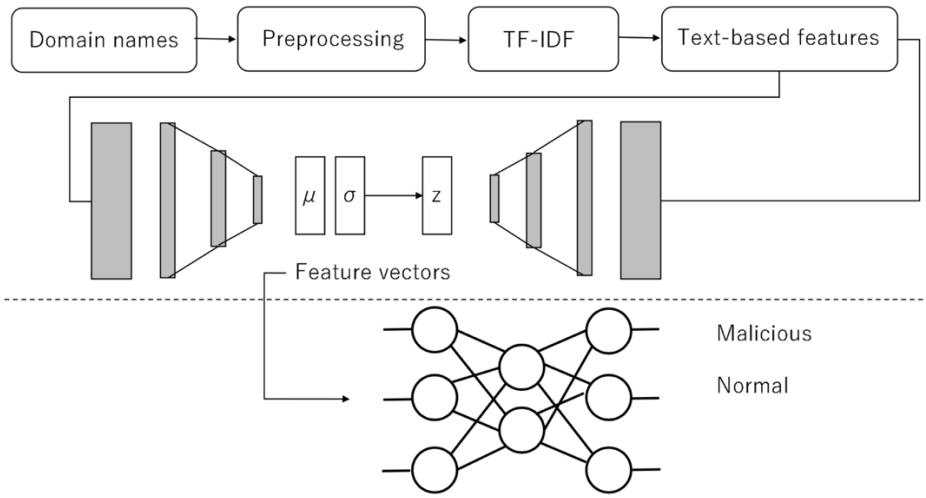


Fig. 4.1. The scheme of the malicious domain names detection system.

In detail, we use a method called TF-IDF to generate the text-based features of domain names through the term frequency and inverse document frequency, each of which includes information of 62 items in domain names. Then we use an unsupervised learning, the variational autoencoder to compress this feature information thus extracting hidden feature vectors at the hidden layer of the model, and each of these extracted feature vectors includes 25 items. In addition, we collapse the Gaussian distribution of the text-based features in a latent space to its mean, and visualize the distribution of domain names in the three-dimensional space.

At last, we adopt a supervised learning called convolutional neural network. Through training, we obtain a classifier for detecting malicious domain names based on the feature vectors of domain names.

This chapter is organized as follows. Section 2 discusses related work on anomaly detection in the domain name system. Section 3 provides an overview of our scheme consisting of converting domain

names into text-based features with the TF-IDF, extraction of latent feature vectors using the VAE, and domain names classification based on CNN. Section 4, we discuss the result of malicious domain names detection with the proposed scheme. Section 5, we conclude the chapter and give out future work of this research.

4.2 Related Work

There is some research focusing on the detection of DNS anomaly. For instance, Nazario et al. [21] mined traffic data in networks to discover new fast-flux domains and as a result tracked those botnets with active measurements for several months, finding that these botnets are associated with a broad range of online fraud and crime, including pharmacy sites, phishing, and malware distribution. Moreover, Villamarn-Salomn et al. [22] suggested an approach of detecting DNS anomaly through monitoring the abnormally high or temporally concentrated query rates in various DNS queries. Furthermore, Chatzis et al. [23] proposed a method of using similarity search over time series analysis to research on the effect of email worms on the flow-level characteristics of DNS query streams generated by user machines.

In addition, research on the application of machine learning in solving challenges in DNS anomaly detection such as the frequently changeable domain names and the difficulty in features representation of domain names is still insufficient. In one of these researches, Erman et al. [24] suggested using an unsupervised learning called the k-means clustering in order to classify DNS traffic data, thus detecting the malicious in the DNS. Furthermore, in a recent research of Krokos et al. [25], they proposed an approach of detecting malicious domain names through visualizing the distribution of DNS queries with an unsupervised learning method called the variational autoencoder. Through this approach, they were supposed to detect DNS attacks like the distributed denial-of-service (DDoS) attacks in networks.

In this research, we use a combination of the unsupervised learning and the supervised learning, in order to extract the latten features of domain names first, then identify whether a domain name is malicious.

4.3 Malicious Domain Names Detection with VAE And Supervised Learning

4.3.1 Text-based feature representation with TF-IDF

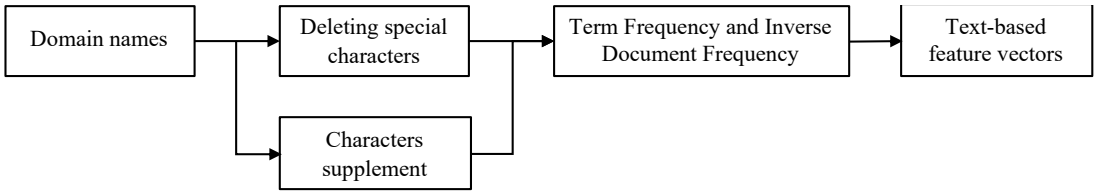


Fig. 4.2. The flowchart of generating the text-based feature vectors.

In this research, we convert domain names with various formats and lengths into the same standardized text-based representation (Fig. 4.2). First, as the preprocessing of domain names, we delete all the special characters in domain names. At the same time, in order to consider all the possible characters for the feature representation, we add one more domain name into the dataset, in which all the considered characters are used, including the lower characters of a to z, upper characters of A to Z, and all the numbers from 0 to 9.

Then we convert these characters into standardized numeric information based on the term frequency and the inverse document frequency (TF-IDF) for representing the features of domain names.

Moreover, TF (Term Frequency) indicates the frequency of word occurrence in each document (4.1). IDF (Inverse Document Frequency) gives high value to rare words with high reverse value, such as inverse document frequency (4.2). As a result, TF-IDF is an algorithm that takes into account both the word frequency and the word rarity (4.3). Considering the property of the changeable patterns hidden in domain names, the adaptation of TF-IDF, lets us represent the relationships between the words both inside specific domain names and the whole dataset.

$$tf(t,d) = \frac{n_{t,d}}{\sum_{s \in d} n_{s,d}} \tag{4.1}$$

Where $\mathrm{tf(t, d)}$ shows the TF value of a word t in document d, $\mathrm{n_{t,d}}$ is the frequency of a word t inside the document d and $\sum_{\mathrm{s \in d}} \mathrm{n_{s,d}}$ shows the total frequency of all the recorded words in document d.

$$idf(t) = log \frac{N}{df(t)} + 1 \tag{4.2}$$

Where $idf(t)$ shows the IDF value of a word t, N is the number of all the documents, and $df(t)$ shows the number of documents which include the word t.

$$w(t,d) = tf(t,d) \times idf(t) \tag{4.3}$$

Where, $w(t,d)$ shows the weight of the word t in document d, equaling the multiply of the TF and IDF values.

Furthermore, in this research, in order to consider all possible characters except the special symbols, we add one more query into the dataset consisting of 62-items characters covering the lower characters, a to z, upper characters, A to Z, and all the numbers from 0 to 9 as mentioned before. Hence, we convert these domain names in a dataset from the type of string to numeric information based on TF-IDF, each of which consists of 62 float values (Fig. 4.3).

```
0.08108 0.08108 0 0.09158 0.06212 0 0.06052 0 0 0.14250 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0.03576 0 0.06292 0.04431 0.06356 0.06245 0 0 0.10898
0 0 0.04457 0 0.07674 0.08812 0.04700 0 0 0 0.06957 0 0.11952 0 0.15376
0.06435 0
```

Fig. 4.3. A sample of generated text-based features, each of which consists of 62 numeric values.

In this research, a dataset from the Kaggle[21] is used. This dataset consists of 420461 uniform resource locators (URLs), which are all labeled as the malicious or the benign. For the preparation of the dataset used in this research, first, we separate and extract the parts of domain names from these URLs (Table 3.1). Then we eliminate all the repetitions in these domain names. As a result, we obtain 108632 benign domain names and 34826 malicious ones in total.

Table. 4.1. Sample of The Dataset (The original URLs have been modified to be shown in this table).

Labels	URLs	Domain names
malicious	officeonn.ch.ma/office.js	officeonn.ch.ma
benign	athlings.com/time.aspx?ret=racesearch&disc=2&mm=5&yy=2009&rg=US&sp=US_CA	athlings.com

After that, we randomly select 20000 items from each cluster to build the dataset. Therefore, the dataset used in this research consists of 20000 benign domain names and 20000 malicious ones. At last, we convert these domain names in the dataset to text-based features using the method discussed above and label them with the malicious or the benign.

4.3.2 Feature vectors extraction using VAE

Autoencoder (AE) is an unsupervised learning approach, consisting of two parts: the encoder and decoder. The encoder performs dimensional compression on the input vector. On the other hand, the decoder tries to reproduce the original input from the compressed vector. Since the network is trained as the original input and output are the same, latent variables retain feature quantities of the input data as much as possible. Therefore, it can be used as an approach to dimension reduction by extracting latent variables of AE.

Furthermore, Variational Autoencoder (VAE) is based on the AE. The biggest difference between these two models is the probability distribution is introduced into the latent variable part of AE, in the case of VAE. VAE uses the encoder to find the mean vector μ and the variance vector σ. And the latent variable z is sampled from the multivariate Gaussian distribution based on the mean vector and variance vector obtained (Fig. 3.3). In VAE, the vector z obtained from the sampling is the vector after the dimension reduction. What's more, the visualization of the distribution of the input data using the latent variables is called latent space.

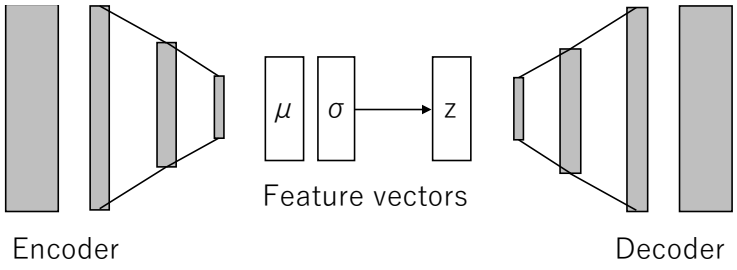


Fig. 4.4. The structure of the VAE consisting of an encoder and a decoder.

In this research, we build a four layers VAE to extract the latent features of the input feature vectors. For the part of the encoder, the first layer has an input size of 62 and an output size of 40, and the second layer has an input size of 40 and an output size of 25. For the part of the decoder, the first layer has an input size of 25 and an output size of 40, and the second layer has an input size of 40 and an output size of 62.

Moreover, for each layer, we use an activation function to transfer the liner data into the non-linear data. The functions we used are ReLU and Sigmoid which are defined as (2.7) and (4.4).

$$f(x) = \frac{1}{1+e^{-x}} \tag{4.4}$$

Where x is the input and $f(x)$ is the output of a node in the model.

Then we use the feature vectors with 62 items each, which are generated from the dataset using the TF-IDF algorithm, as both the input and the output to train the VAE model. After training, we are supposed to attain a model which can represent input vectors using compressed latent features with twenty-five values. In addition, the training graph is shown below (Fig. 4.4).

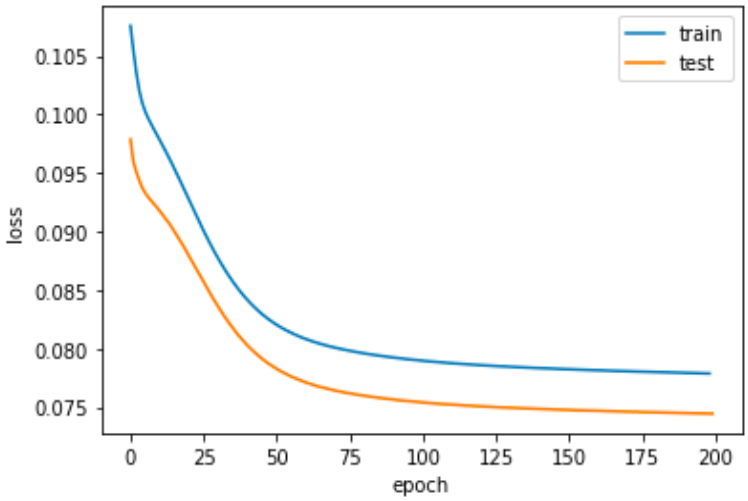


Fig. 4.4. The training graph of the VAE model using the dataset consisting of text-based features of domain names.

What's more, we project all the domain names into a 3D latent space using the latent variable z from the multivariate Gaussian distribution, which is based on the mean vector and variance vector obtained at the hidden layer of the VAE model (Fig. 4.5). From the distribution, we can find that part of the malicious domain names are separated and clustering at a specific area of the latent space. On the other hand, lots of domain names are still difficult to identify their nature with this graph.

Through this approach, we can compress the feature vectors to latent feature vectors, each of which has twenty five items. Thus, the depth of representation of domain name features is reduced from 62 to 25.

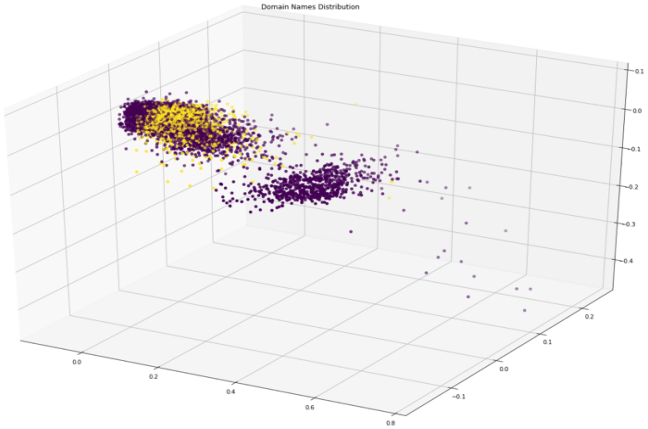


Fig. 4.5. The distribution of all domain names in the dataset: Here, malicious domain names are shown in purple and benign domain names are shown in yellow.

4.3.3 Feature vectors classification with a convolutional neural network

After extracting feature vectors of these domain names, we adopt a supervised learning model called convolutional neural network for classification. In this research, a three-layer convolutional neural network model is utilized and trained with these feature vectors, using two types of labels as the benign and the malicious (Fig. 4.6).

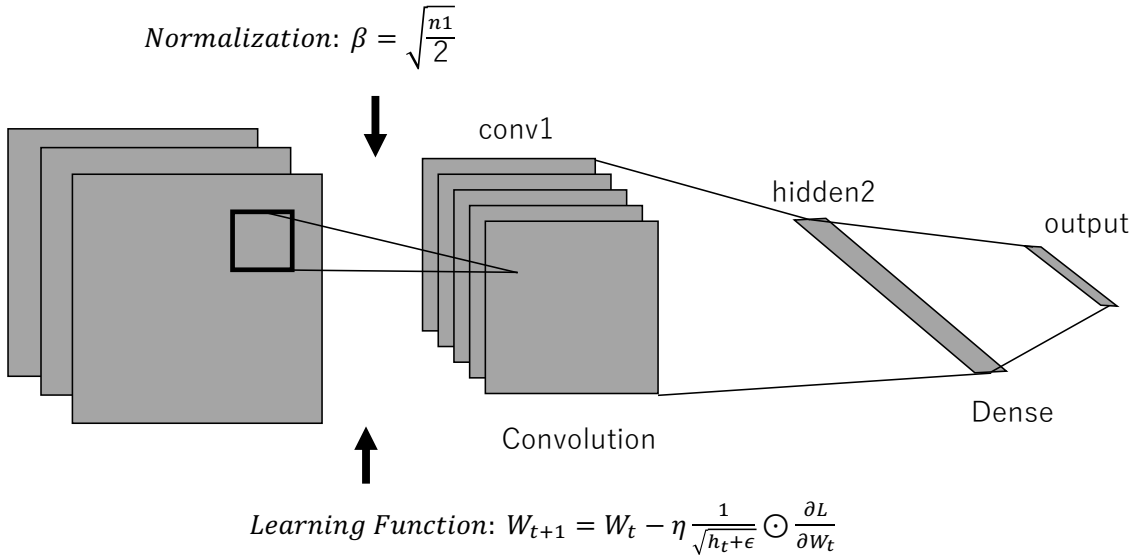


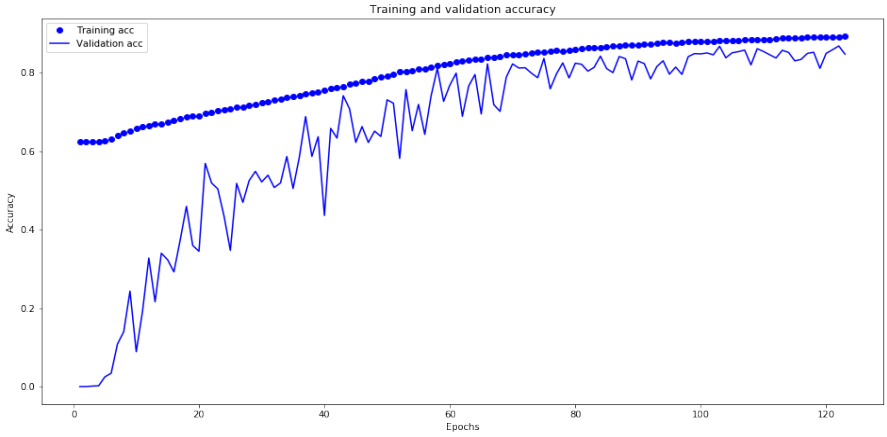
Fig. 4.6. The structure of the three-layer CNN model with the learning function of RMSProp and an adoption of normalization.

A convolutional neural network typically consists of several convolution layers, some of them with a pooling layer and several fully-connected layers at the end of the model. Moreover, CNN has a movement invariance with respect to the input data and exhibits good performance. Furthermore, for the convolution layer, we adopt the approach of normalization to transfer the input data into a normal distribution before the computation at the next layer, in order to improve the invariance of the model to the input data at each layer. In addition, we use the RMSProp as the learning function, which has the following characteristics, an emphasis is placed on the latest gradient information more than the past gradient information, and gradually forget the past gradient information so that the new gradient information is greatly reflected, thus improving the performance as well as the stability of the training progress.

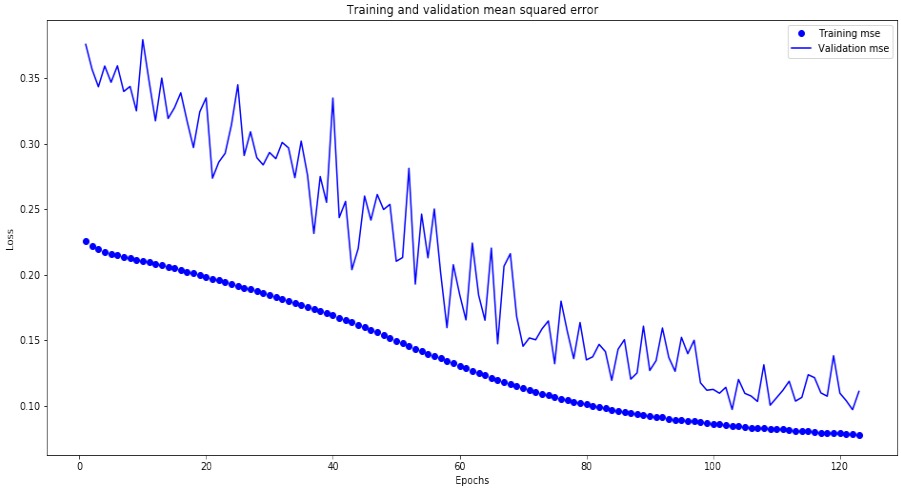
Moreover, considering the objective of a two-cluster classification, we utilize a fully-connected layer with a two-item output at the end of the model, representing the benign and the malicious individually, with an activation function called softmax, where a standard exponential function is applied to each element xi of the input vector x and these values are normalized through dividing by the sum of all these exponentials. Consequently, each component including the negative, greater than one, or might not sum to 1, will be converted into the interval (0, 1) with a sum of 1.

Then, we train this CNN model using the dataset of domain name feature vectors including 20000 benign domain names and 20000 malicious ones in total, with a batch size of 20 and a split ratio of 0.2 for the validation set. Furthermore, we adopt the early stopping to prevent overfitting, which refers to a model that models the training data too well that it causes the validation loss to increase, with a patience of twenty, monitoring the validation accuracy of every epoch.

After training, we are supposed to obtain a model which is used to classify between the benign and the malicious. Moreover, the training graphs are shown below (Fig. 4.7), from which the scheme shows a relatively stable training progress, preventing overfitting at the same time. At last, the scheme achieves a validation accuracy of 0.868 for the malicious domain names detection.



(a) The graph of the training and validation accuracy.



(b) The graph of the training and validation mean squared error.

Fig. 4.7. The training graphs of the CNN model, with the feature vectors as the input data and the two types of domain names as the labels.

4.4 Conclusion

We adopt a text-based method called TF-IDF to convert the character information of domain names to numeric features, using the term frequency and the inverse document frequency. After that, we use an unsupervised learning, the variational autoencoder to extract the latent feature vectors from these numeric features of domain names. Then, to classify between the malicious and the benign, we utilize a supervised learning of a three-layer convolutional neural network based on the generated feature vectors, each of which includes twenty-five items. Furthermore, we use a split ratio of 0.2 for the validation set in order to evaluate the performance of this model. At last, we achieve a validation accuracy of 0.868 for the malicious domain names detection problem.

For future work, a combination with analysis of traffic data in network systems can be considered to add up to the accuracy of malicious detection in the DNS. Moreover, other machine learning methods can be considered to approach malicious detection issues as well, such as the recurrent neural network and the support vector machine.

Chapter 5 Network Flow-Based Malware Detection with a Web Crawler

5.1 Introduction

The pandemic of COVID-19 is accelerating digitalization in many walks of society, along with more and more devices and people connected to networks. Much information is stored, shared, and analyzed on the Internet every second. The current transformation is bringing us to a new era of the information society, which requires more effective and resilient monitoring systems for safeguarding traffic data in networks. A malware delivered in a network usually can intrude a host and further expand into other hosts in a network, causing enormous losses and affecting various aspects of people's life. For example, phishing e-mails lure and direct users to malicious websites, causing unintentional disclosure of their private information to attackers.

In this study, we deployed a web crawler to visit various URLs covering benign websites and malware websites. In parallel, we used another workstation to collect real-time network traffic, which formed the deep learning data set. Additionally, we set a fixed time window to ensure sufficient traffic could be harvested in the interaction between the client and each site.

The collected network traffic is segmented into flows based on several features, such as the source and destination IP addresses. Then 12 network flows-based features were adopted and extracted from the traffic data. Furthermore, a deep neural network with optimized hyperparameters was implemented to classify the network traffic into malicious and benign flows (Fig. 5.1). Finally, we used the following metrics to evaluate the performance of our proposed detection approach: precision, recall, and f1 score.

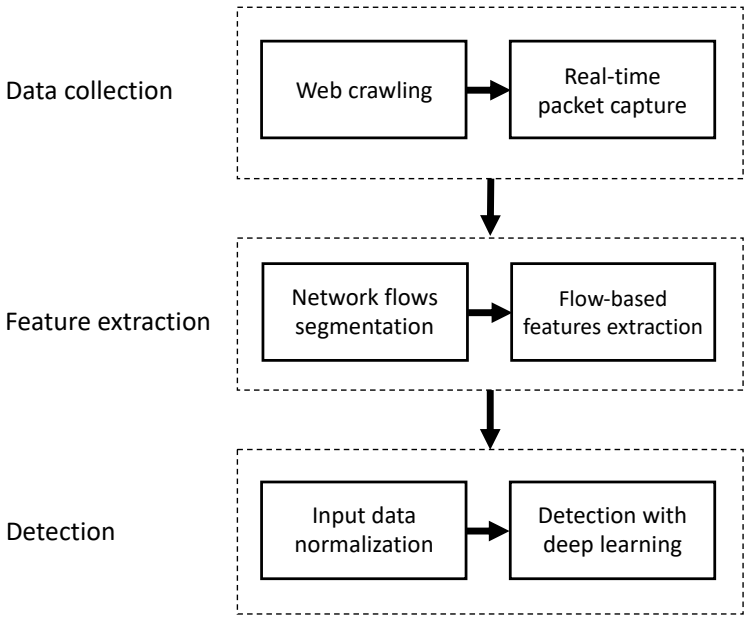


Fig. 5.1. The scheme of the network flows-based malware detection system integrating web crawling and deep learning.

This chapter is organized as follows. Section 2 discusses related work on applied machine learning for malware detection. Section 3 provides an overview of the detection system, including the web crawler, the feature extractor, and the optimized deep neural network. Section 4 presents the performance evaluation results. In Section 5, we conclude the chapter with a summary of our contributions and future work.

5.2 Related Work

Research on malware detection in networks has a long history. Especially in recent years, the implementation of AI technologies such as machine learning and artificial neural networks has fostered the development in this field. For instance, Kolias et al. [26] applied machine learning for detecting spambots and malware bots from their behaviors with distinctive communication patterns. Atienza et al. [27] presented a scheme of detecting web attacks using self-organizing maps. They visualized the data using a 2D map, thus distinguishing malicious network traffic. Moreover, Ikinci et al. [28] presented a malicious website detection scheme using low-interaction honey clients. By adopting a web crawler, they conducted a content-based analysis for examining several properties of attacks against end-users. Nunes et al. [29] demonstrated an operational system for cyber threat intelligence gathering from various social platforms, particularly on the darknet and deepnet. They implemented different machine learning methods to detect malware, achieving a recall of 0.92 for marketplaces and 0.80 for forums. Furthermore, Amit et al. [30] presented a comprehensive survey on machine learning applications in cybersecurity. Further, they discussed challenges such as the lack of labeled samples and certainty in the ground truth.

There are also research projects focusing on the lexical analysis of data. For example, Mamun et al. [31] proposed an approach of converting URLs into lexical features and combining with classifiers such as the k-nearest neighbors algorithm for detecting malicious URLs. Unlike former research efforts, we built a detection model using an optimized deep neural network based on training data consisting of network traffic collected by crawling malicious and benign URLs.

5.3 Network flows-based malware detection using crawling and deep learning

We set out to detect malicious network flows by systematically crawling various types of URLs and analyzing the associated network traffic between the client and remote servers (Fig. 5.2). We used network flows-based feature extraction algorithms to prepare the dataset for training a deep neural network. After training on these network flow features completed, the system could detect malicious traffic flows from malware URLs adaptively based on the hidden patterns of client-server interactions inside the network traffic data. The detection results show that the system can learn to discover high-level insights from network flows.

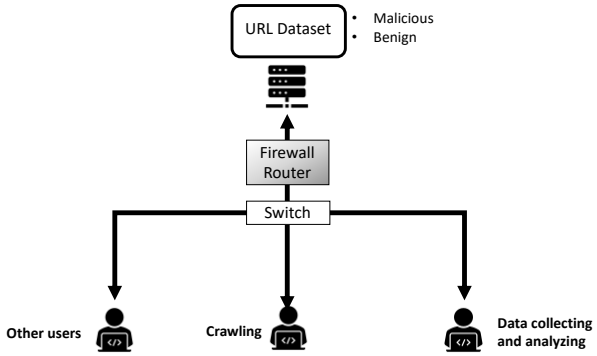


Fig. 5.2. Network flows-based malware detection: traffic data collected through crawling was used for training a deep neural network to detect malicious network flows in the network.

5.4 Web crawling

We adopted a URL dataset from the Canadian Institute of Cybersecurity [31]. The URLs were collected from Alexa top websites with the duplicate and invalid ones removed and separated into benign URLs and other types using VirusTotal.

In our settings, we launched our web crawler to visit 1400 benign URLs and 1400 malware URLs. These URLs were all confirmed to be valid at the time of the experiment (Table 5.1). Another workstation was in charge of collecting real-time network traffic data using Tcpdump for further analysis.

Table. 5.1. The constitution of URLs for web crawling (The original URLs have been modified to be shown in this table).

URL Type	Number	URL Sample
Benign	1400	http://allego.pl/royal-string-quartet-music-for-string-quartet-cd-i5262877830.html
Malicious	1400	http://www.facebok.com/plugins/like.php?href=http%3A%2F%2Fzoetegroon.nl%2F2012%2F06%2Ffc-barcelona-taart%2F&layout=button_count&show_faces=true&width=450&action=like&font=tahoma&colorscheme=light

5.4.1 Network flows-based segmentation and feature extraction

We used a tool called NFStream to perform traffic flow analysis. After converting the collected packet capture from the LAN into various flows based on the network interface, the protocol, and the IP/port source/destination, we ended up with network flows, each of which was represented with a 12-item feature vector. Then, using another tool, dpkt, we further extracted the flows related to crawling targets. Specifically, the flows with a source or destination IP address from the URLs' IP addresses in the dataset were selected as the flow feature dataset. The selected features of a network flow are shown in the following:

- Transport layer source port
- Transport layer destination port
- Transport layer protocol
- Number of flow bidirectional packets
- Number of flow bidirectional raw bytes
- Flow bidirectional duration in millisecond
- Number of flow src2dst packets
- Number of flow src2dst raw bytes
- Number of flow dst2src packets
- Number of flow dst2src raw bytes
- nDPI master protocol
- nDPI app protocol

Using the described method, we obtained 2472 flow feature vectors from the benign network traffic and 5109 feature vectors from the malicious network traffic. Samples of these extracted network flow features are shown in Table 5.2.

Table. 5.2. Samples of extracted features from network traffic flows.

id	src port	dst port	protocol	total packets	total bytes	duration	src2dst packets	src2dst bytes	dst2src packets	dst2src bytes	master protocol	app protocol
1	5938	44020	6	1	102	0	1	102	0	0	0	148
2	53162	53	17	3	321	1076	2	158	1	163	0	5
3	41027	53	17	2	182	50	1	83	1	99	0	5
4	39272	80	6	10	1286	900	6	550	4	736	7	220
5	46090	443	6	54	44930	1900	17	1763	37	43167	91	220

We analyzed the application types of these network flows used by the servers of the URLs. We found most malicious URLs the HTTP protocol-based, whereas benign URLs were more likely to adopt the TLS protocol, which is often provided by trusted sources (Fig. 5.3).

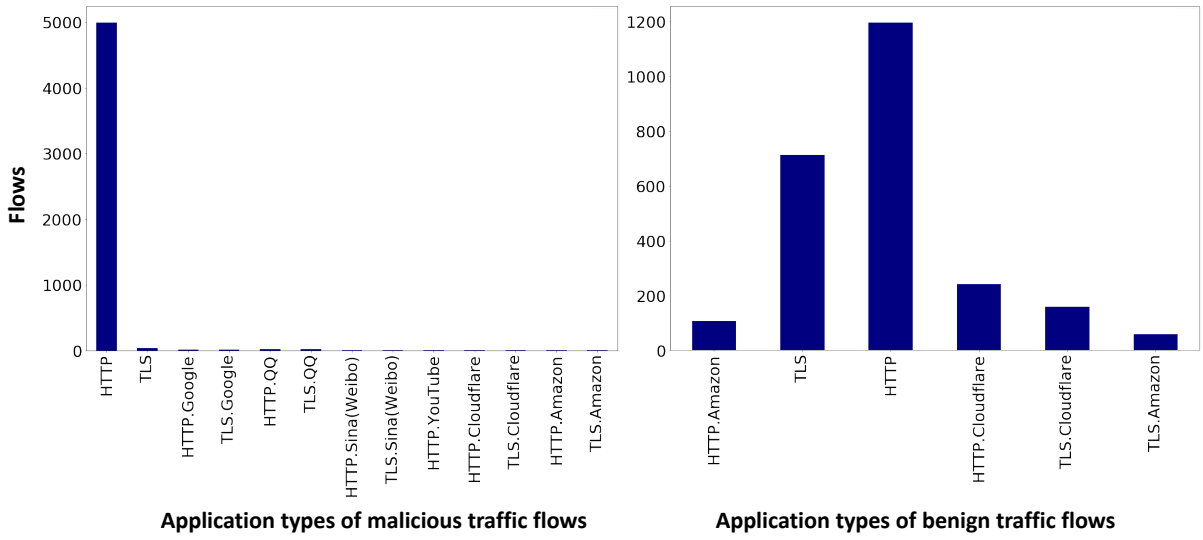


Fig. 5.3. Application types adopted by the traffic flows from the dataset.

5.4.2 Malware detection based on deep learning

We realized deep learning using a fully-connected neural network with multiple dense layers using the extracted flow features as the inputs and the URL types as the labels, where 0 represented benign, and 1 represented malicious. As a result, the input layer's size was twelve, and the output layer's size was one. To optimize the model's structure, we experimented with various hyperparameters to effect performance changes, including the composition of hidden layers, the batch size, the learning function, and the learning rate (Table 5.3). The number of hidden layers sets the depth of a neural network, and the number of neurons sets the width of each layer. The total number of variables, representing the total number of parameters that need to be updated, can be computed from each layer's depth and width. The batch learning method was applied, where the dataset was partitioned into small chunks for training purposes. A smaller batch size brings out a more satisfactory model at a slower speed. In contrast, a larger one can speed up the training process, albeit bringing out poorer training results.

Table. 5.3. Hyperparameters for finding the best ANN structure and its selection.

Hyperparameter*				
Hidden layers	**Variables**	**Batch size**	**Learning function**	**Learning rate**
100, 20	3341	8	SGD	0.00001
200, 50	12701	16	Momentum	0.0001
100, 50, 20	7391	32	RMSProp	0.001
200, 100, 50	27801	64	Adam	0.01
		128		

*The hyperparameters were combined to find the best ANN structure.

Selected parameters

Hidden layers	**Variables**	**Batch size**	**Learning function**	**Learning rate**
200, 100, 50	27801	16	Adam	0.001

We tuned a combination of hyperparameters, including hidden layers, batch size, learning function, and learning rate, to find an optimal artificial neural network (ANN) structure. The learning function affects the variable update of weights and biases at each layer of the model, with varying training stability and speed results. We compared the performance of four different learning functions: the SGD, the Momentum, the RMSProp, and the Adam. The learning rate was used to control the step length for each epoch.

We employed a function called L2 normalization as defined in (5.1) to rescale the real-valued numeric input attributes to a number between 0 and 1, as a pre-processing step for the computation within the hidden layers of our deep neural network.

$$x_{L2\,norm} = \frac{x}{\|x\|_2}$$
$$\|x\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2} \tag{5.1}$$

Where $x_{L2\,norm}$ is the L2 normalization result of the input vectors of flow features, x_i represents the *ith* vector of the input, and n represents the total number of the input vectors.

Furthermore, the activation function for the computation at the hidden layers was ReLU. The Sigmoid function as defined in (4.4) was used as the activation function at the output layer for narrowing the output into the range of (0, 1). The selected loss function was cross-entropy, which is usually utilized to measure the performance of an ANN model whose output is a probability value between 0 and 1.

5.5 Evaluation

Precision, recall, and f1 score formed the basis of our metrics for evaluating the performance of our deep learning approach to detect malicious network flows, which are defined in (5.2). The precision is the fraction of relevant network flows successfully classified among all flows in the validation set; the recall is the fraction of relevant network flows successfully classified among existing relevant network flows. Besides, the f1 score shows the overall performance of an ANN model.

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

$$Recall = \frac{TP}{TP+FN} \tag{5.2}$$

$$F1\text{-}score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

Where TP (True Positives) indicates the number of correct malicious network flows classified by the model in the data, FP (False Positives) indicates the number of incorrect malicious network flows classified in the data, TN (True Negatives) indicates the number of correct benign network flows classified by the model in the data, and FN (False Negatives) indicates the number of incorrect benign network flows classified in the data.

A total of 7581 feature vectors were generated and adopted as the input of the model. We divided the dataset into the training set and the validation set with a ratio of 4:1. The early stopping strategy was used to prevent overfitting, a situation where the training loss keeps decreasing while the validation loss keeps increasing. By monitoring the variance of the recent five epochs' validation loss rates, we could train the model with more balance between the training performance and the validation performance.

A total of 320 combinations of the various hyperparameters were used to train the model. By comparing the average validation f1 score for the last five epochs, we evaluated the performance of the ANN model for various combinations of these hyperparameters. For instance, the evaluation results of the f1 score for the four-layers ANN are shown below (Fig. 5.4), using the Adam learning function for various batch sizes and learning rates. It shows that this four-layers ANN model has the best performance when the learning rate is 0.001, and the batch size is 16.

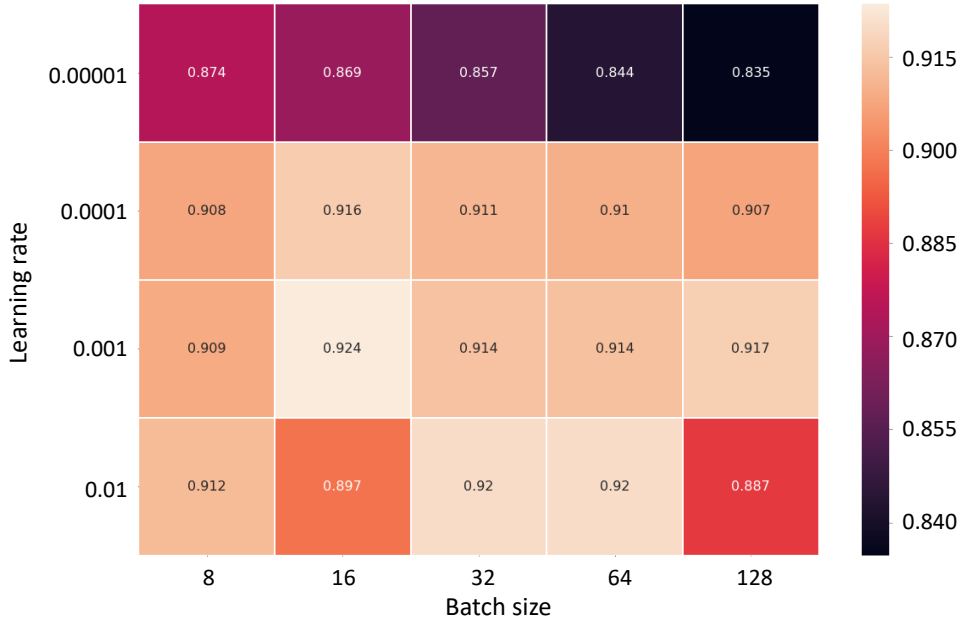


Fig. 5.4. Evaluation results of the f1 score for the four-layers ANN, using the Adam learning function for various batch sizes and learning rates.

Out of the 320 possible combinations of hyperparameters, the ANN with the following structure and learning settings delivered the best performance in network flows-based malware detection for the dataset (Table 5.3):

- Number of hidden layers: 3
- Number of neurons at each layer: 200, 100, and 50
- Batch size: 16
- Learning function: Adam
- Learning rate: 0.001

The four-layer deep neural network architecture is shown below (Fig. 5.5), using the extracted network flows-based features as the inputs and the URL types as the labels.

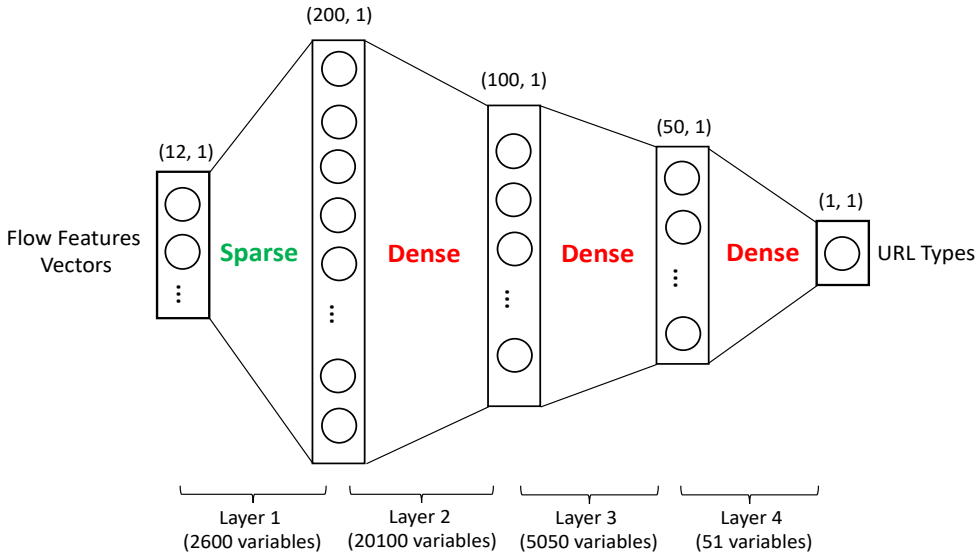
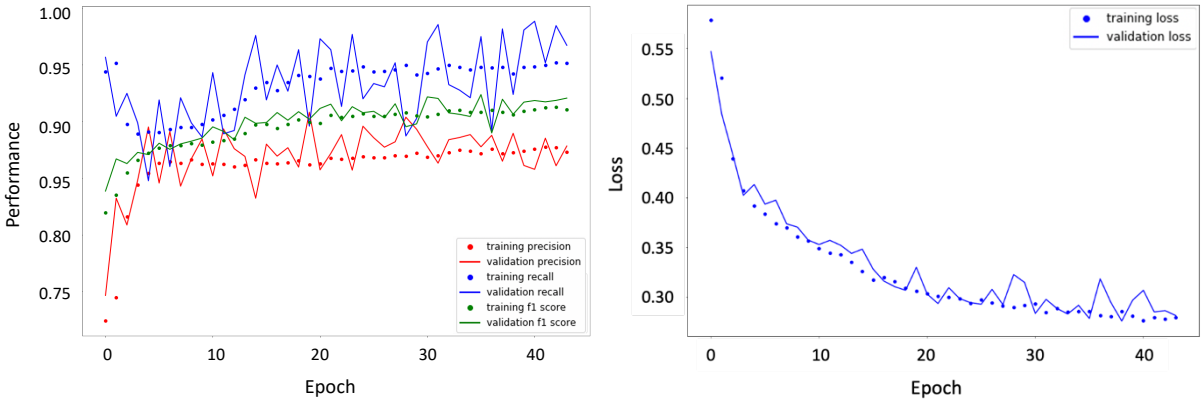


Fig. 5.5. The architecture of the four-layers deep neural network, with total 27801 variables. The flow features vectors were first dispersed into 200-items vectors and then condensed gradually into vectors with a single output for inferring the types of these flow features as benign or malicious.

Furthermore, the deep neural network worked with 27801 variables. The first layer was used to disperse the 12-dimension input data into a wider space of 200 dimensions. Then the remaining three layers were used to condense the computation result of the first layer gradually, at last, into a single numerical output. Besides, the corresponding evaluation results and the training and validation loss at each epoch of the training are shown below (Fig. 5.6).

The scheme achieved an overall validation precision of 0.880, an overall validation recall of 0.972, and an overall validation f1 score of 0.924 for the network flows-based malware detection system.

In addition, we examined various flow features' impact on the inference of the ANN model. By extracting the weights of the first layer from the trained ANN model, we obtained variables with the shape of (12, 200). Then we computed the averages of weights connected to each node at the input layer, which had a shape of (12, 1). The computed average weights were further converted into impact factors using (5.3), which represent the impact extent for each flow feature on the inference of the model.



(a) Training and validation performance. (b) Training and validation loss.

Fig. 5.6. Performance evaluation of the scheme for detecting malicious network flows based on various metrics: precision, recall, f1 score, and loss.

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$$
$$Impact\text{-}factor = \tanh(absolute(w_{average})) \tag{5.3}$$

Where $w_{average}$ represents the average weights at the first layer, $absolute$ is a function to obtain the absolute value of the average weights, and tanh is a rescaling of the logistic sigmoid, such that its outputs range from 0 to 1 with non-negative inputs.

As we can see, the features of the nDPI app protocol and the destination port had more impact on the inference of the model compared with the other features (Fig. 5.7).

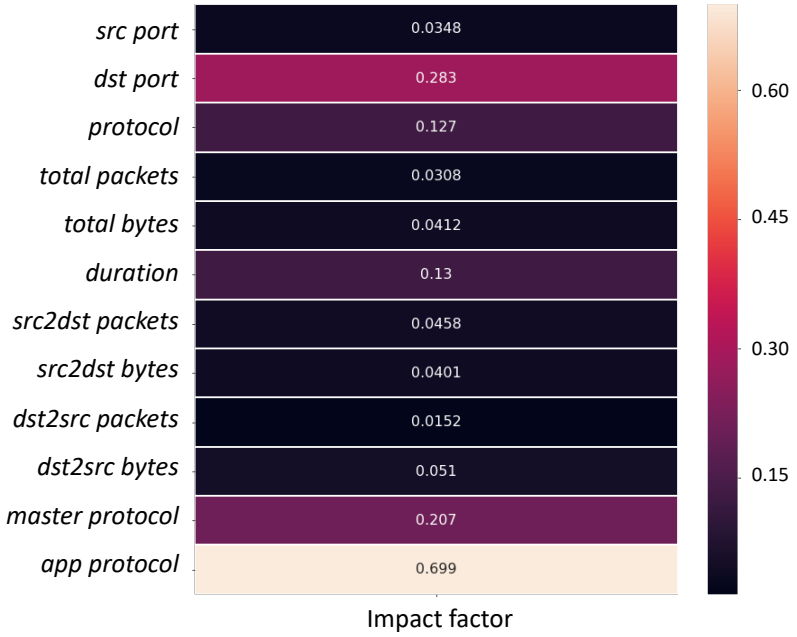


Fig. 5.7. Impact factors of the 12 flow features with respect to the inference of the ANN model.

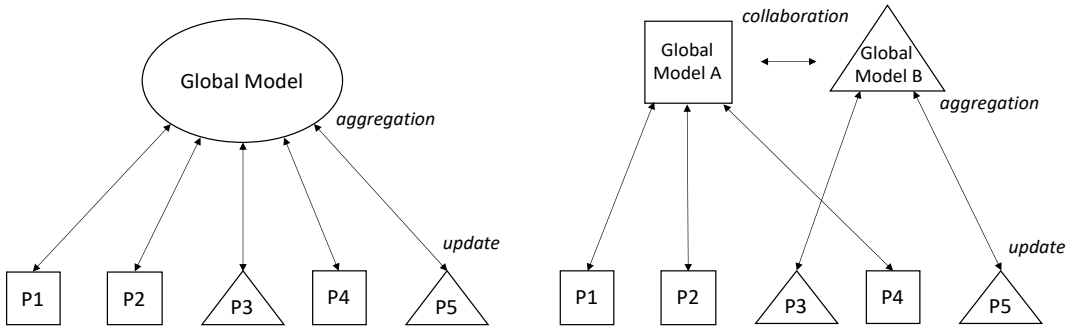
5.6 Conclusion

In this chapter, we presented a network flows-based malware detection mechanism based on web crawling and deep learning. A dataset consisting of 7581 feature vectors of network flows was built for training a deep neural network. These flow features were generated from the collected network traffic of benign and malicious URLs. A comparison of the ANN model's performance for the various hyperparameter combinations was also conducted by using the f1 scores. The optimized deep neural network showed the best performance in malicious flow detection out of all combinations considered in our study, with an f1 score of 0.924. Our future work would extend this effort to a cooperative scheme where networks share insights on malware detection in a federated learning architecture [32].

Chapter 6 Segmented-Federated Learning

6.1 Introduction

Network intrusion detection strategies existing in current systems have been revealing issues of low adaptivity to network traffic from various network environments. A network monitoring system that has the ability to detect and track malware in various networks is highly demanded. The scheme of federated learning (FL) was first proposed by Google to solve problems of data scarcity and privacy in the field of machine learning [33]. This scheme has been employed in applications such as image recognition, natural language processing, cybersecurity, and so on. It showed that by using this scheme, users could share intelligence on a machine learning task with each other, whereas without disclosing their raw data.



(a) Federated Learning (b) Segmented-Federated Learning

Fig. 6.1. Segmented-FL has multiple global models, each of which aggregates local models of similar participants. E.g. since P1, P2, and P4 are similar; P3 and P5 are similar, they are segmented into two groups belonging to Global Model A and Global Model B respectively.

Despite the success of existing FL solutions [34][35], in NIDS, a network's traffic data does not always fit into the single global model of FL; some networks have similarities with each other but other networks do not. In this article, we propose a novel adaptive learning scheme of Segmented-Federated Learning (Segmented-FL) (Fig. 6.1). We employ periodic local model evaluation and segmentation for adaptive model training. Different from all users training a model under the single global model at the central server in FL, Segmented - FL has a feature that each segmented group of users is arranged with a specified global model for adaptive learning.

The Segmented - FL is utilized for parameter sharing among participants as well as automatic segmentation of participants thus adapting to massively distributed networks. For each day, also called round, selected participants conduct local model training based on locally collected datasets. Then the trained models are uploaded and aggregated in the parameter server for updating a global model. The periodic evaluation of local models is used to validate and divide them into several groups for adaptive learning based on recent performance in tasks of intrusion detection. If a model shows a disadvantaged performance compared with the other models, it will be removed from the current group, and a new group with an independent global model will be initialized. Thus, from the next round, these participants will train their local models using respective global models.

A deep convolutional neural network (CNN) is used as the protocol for model training and sharing. We adopt a combined approach of feature representation and expert knowledge-based labeling of

benign and malicious network events to train an artificial neural network (ANN) model. The trained models are used to detect malicious network events in LANs, and enhance detection performance through ANN model parameters sharing among group members.

The patterns of network traffic data for intrusion detection are varying in large-scale network environments, with various network scales, devices with different operating systems and applications installed, and so on. For example, some LANs have more than 1000 active users while some have only a dozen users, showing different presence information. Devices in these LANs are working on different operating systems such as Windows, macOS, and Ubuntu, which reveal different traffic patterns. Some LANs include the Internet of Things (IoT) devices such as web cameras and smart home devices. Besides, the user groups could be different from students in academic institutes to professionals in industries. These reasons above contribute to the diversity of network traffic patterns as well as the difficulty of intrusion detection in large-scale distributed networks.

The research project of the LAN-Security Monitoring Project aims to collect these varying real-time network traffic data from massively distributed academic networks [35]. In this project, a monitoring device was developed and deployed in a network to collect traffic data. The collected data is saved at a server for further analysis, including complete broadcast traffic data in a local area network (LAN) and network traffic sent directly to monitoring devices. A total of 38 institutes have been participating with a monitoring device set up in an office network or laboratory network. The data from this project provided the basis of datasets used in the research of Segmented-FL.

The challenges and contributions of the research include:

- For segmenting similar participants into the same group with a respective global model, we propose Segmented-FL for adaptive model learning and sharing.
- To understand the effect of key parameters of Segmented-FL, we have conducted a comprehensive survey on 27 combinations of them for exploiting the optimized solution.
- For validating the model with diverse network traffic data, we applied massively distributed networks' real-time traffic from various academic institutes.
- To compare the performance between FL and our method, we have conducted evaluations against a range of metrics, weighted precision, recall, and F1 score. It shows that our method has better performance in intrusion detection with large-scale distributed networks.

This chapter is organized as follows. Section 2 discusses related works on network intrusion detection and the application of federated learning in cybersecurity. Section 3 provides an overview of our method consisting of fundamental federated learning, intrusion detection with a combined approach of visual analytics and CNN, and the proposed Segmented-FL. Section 4 presents experiments and performance evaluation against the metrics of weighted precision, recall, and F1 score. Section 5 discusses the pros and cons of this method. Section 6 concludes the article.

6.2 Related Work

Traditional approaches to network intrusion detection are commonly related to matching between the database of known malicious patterns based on expert knowledge and network behavior in a network [36][37][38]. Many NIDSs [39][40] use a network-based activity study to determine if a node is compromised, such as traffic or frequency analysis, and deep packet inspection. Unfortunately, these approaches appear to be insufficient, with underlying issues of adaptivity and privacy.

Recently, machine learning and neural networks have been employed to strengthen the performance of network systems for intrusion detection in personal computers and critical infrastructures, such as

support vector machine (SVM) and artificial neural network (ANN) [41][42]. Besides, due to the rapid development of deep learning in recent years, data analysis on big data of network traffic became feasible and was employed in lots of research. For instance, Salama et al. [43] presented an intrusion detection hybrid scheme using deep belief network and SVM, recognizing the malicious network events. An evaluation based on the NSL-KDD dataset was conducted, with a detection accuracy of above 0.9. Yang et al. [44] adopted a combined approach of using restricted Boltzmann machine to extract high-level feature representation of network traffic and classifying these features with SVM. Duy et al. [45] presented intrusion detection based on the NSL-KDD dataset, using a feedforward neural network. Their model achieved an F1 score of 0.962 at last. Saxe et al. [46] proposed a deep neural network-based malware detector by employing two-dimensional binary program features. Yousefi-Azar et al. [47] presented a generative feature learning-based approach for malware classification, where they extracted latent features of network traffic based on an unsupervised learning model called autoencoder.

The aforementioned research on intrusion detection aims to detect malware in limited network environments. Detection strategies and results from one network usually could not be applied to other network environments. Federated learning (FL) is a collaborative learning scheme that allows users to share intelligence on machine learning tasks with each other thus improving overall performance. Different from centralized learning [48], FL allows users to share local models instead of raw network traffic data, which might reveal private information to the learning group. There have been already lots of research on the application of FL in cybersecurity and edge computing. For example, Abeshu et al. [33] proposed a cyberattack detection model using FL to improve accuracy in detecting attacks. It showed enhanced performance and privacy of participants as well as reduced traffic load. Wang et al. [34] presented an intrusion detection in simulated environments based on FL, with a gradient descent-based learning function. Schneible et al. [49] demonstrated an application of FL in anomaly detection, where autoencoder was employed for analytics and observations. Zhao et al. [50] employed FL to tackle the data scarcity problem and to preserve data privacy, where multiple participants collaboratively train a global model. Daga et al. [51] proposed Cartel, a system for collaborative learning in edge clouds. The CICIDS2017 [52] Intrusion Detection evaluation dataset was used, consisting of benign data samples, distributed denial-of-service (DDoS) attacks, and port scan attacks. Additionally, Yu et al. [53] presented a Mobility-aware Proactive edge Caching scheme based on Federated learning (MPCF) for predicting content popularity with the private training data distributed on local vehicles. This scheme could adapt to various mobility patterns and references of vehicles and protect users' privacy.

Different from former research, we propose Segmented-FL for adaptive intrusion detection in large-scale massively distributed networks. The proposed approach has a feature that the scheme architecture is adjusted continually based on periodic evaluation results of local models thus adapting to varying traffic data in these network environments. A deep CNN model is employed as the protocol ANN for collaborative learning, using feature representation of network traffic as the input, results from knowledge-based labeling as the ground truth.

6.3 Adaptive Network Intrusion Detection Based on Segmented-FL

6.3.1 Federated Learning

Federated learning is a collaborative learning scheme for users to share intelligence on model training with each other, without sharing their private data. The intrusion detection in networks could cause exposure of private information to third-party entities while analyzing network traffic data for tracking malicious users. The implementation of FL in intrusion detection allows network operators to detect malware without accessing users' private data.

The architecture and mechanism of FL for collaborative learning by sharing local model training results is shown below (Fig. 6.2).

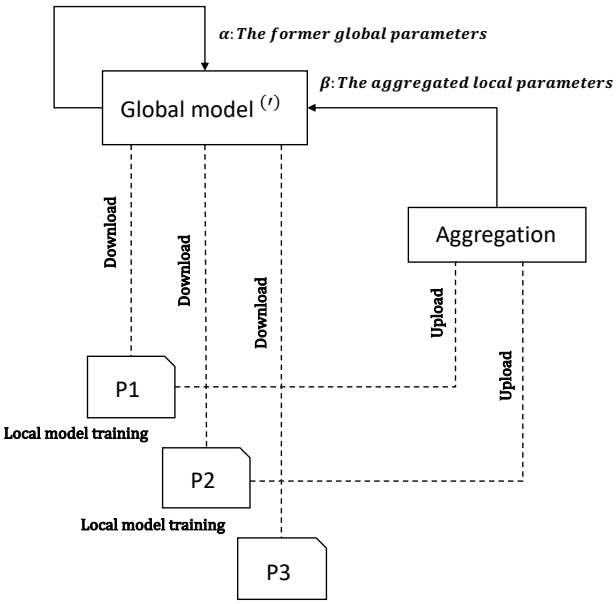


Fig. 6.2. The architecture and mechanism of federated learning.

In this graph, first, participants of FL download the latest global model from the central server and update their local models. Then, the selected users will conduct model training based on a local dataset with varying numbers of data and portions of different clusters. After training, these trained local models will be uploaded to the server for aggregation. As a result, based on the aggregated local parameters and the former global parameters, the current global model is updated for the next round's collaborative learning.

6.3.2 Intrusion detection with a combined approach of visual analytics and convolutional neural networks

To process network traffic for model training, we generated feature maps representing communication patterns of various network protocols by employing a quantitative approach based on frequency information. These extracted information of protocols includes IP, ARP, TCP, HTTP, HTTPS, UDP, mDNS, DHCP, and Others. Fineness, a time window parameter, which corresponds to generating a pixel value in a feature map, was employed when extracting protocol information from raw data. By studying varying fineness rates for the efficiency of feature representation [54], we found a larger time window would reveal more patterns of network traffic in the feature map, while a smaller one would contribute to the sensibility of the detection systems. Given the balance between the pattern representation ability and the detection sensibility, we selected 0.5 seconds as fineness for computing frequencies, recording how many packets related with a specified protocol were sent or received within the defined time window, thus generating a feature map.

Then a chunk of these records with a size of 256 for each protocol was combined to generate communication patterns related with these protocols. As a result, each generated communication pattern covers network traffic features within 128 seconds, related with a specified protocol. The reason for choosing a size of 256 is that the selected time window for generating a pattern allows a network operator to respond to these detected malicious behaviors timely. The relationship between the time window for pattern generation and the fineness rate is defined in (6.1).

$$T = T_{st} \cdot fineness \cdot s^2 \tag{6.1}$$

Where *T* represents the time window for pattern generation, Tst (Time Standard) represents the standard interval of one second for generating a pattern, fineness represents the time window for record generation thus controlling information density in a pattern, and s represents the pattern size (side length).

To combine every 256 records into a pattern, we converted frequency information into pixel values by employing (2.6). Then the Hilbert curve was used to generate a feature map, which is a geometric structure used to transform the structure of data so that it fills up all space in an image, by projecting pixel information to specified positions in the Hilbert curve.

By employing the array exchange, we aggregated these generated feature maps of nine types' protocols into one feature map representing protocol-wise network traffic features within 128 seconds in a LAN. Each generated feature map has a size of 48 pixels × 48 pixels (Fig. 6.3).

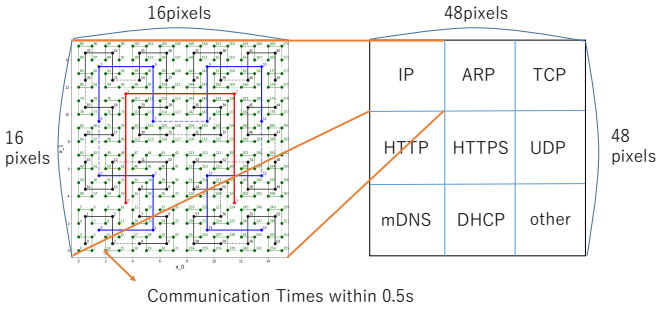


Fig. 6.3. Protocol-wise feature representation of network traffic in a LAN based on the Hilbert curve and array exchange.

To process these feature maps of network events, we employed a supervised learning method of CNN (Fig. 6.4), since the generated feature maps are two-dimensional data with time-related information encoded. Considering possible training computation and transmitting time cost while model sharing, we employed a four-layers CNN which consists of two convolution layers each of which is followed by a maxpooling layer, and two fully-connected layers. For the first convolution layer, ten kernels which are used for convolution operation with a size of 3 × 3 and a stride of one were used. For the second convolution layer, ten kernels with a size of 1 × 1 and a stride of one were used. The fully-connected layer of hidden layers consists of 200 neurons, and the output layer consists of one neuron for classification between the benign and the malicious. The padding with a value of one pixel was used in computation at each convolution layer.

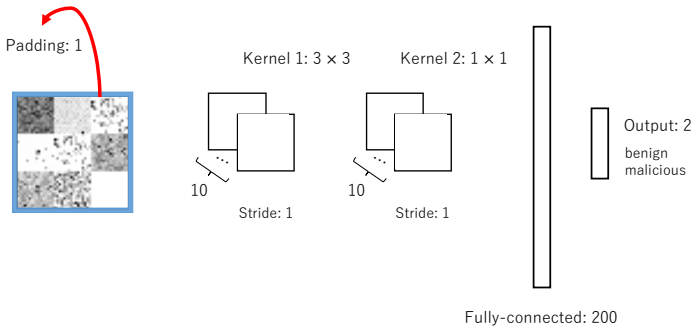


Fig. 6.4. The architecture of the four-layers CNN model.

The learning function is used to update parameters of weights and biases in the model, with the purpose of decreasing prediction loss as much as possible. The RMSProp was employed for the model training, which has a characteristic that the emphasis is placed on the latest gradient information more

than the past gradient information and gradually the past gradient information is forgotten, instead, the new gradient information is greatly reflected. The learning rate is used to control the step length for each epoch's update. Besides, the cross-entropy defined in (6.2) was employed as the loss function to compute prediction loss for updating.

$$L = -\sum_{k}^{K} t_k \log y_k \tag{6.2}$$

Where K is the number of nodes at the output layer, k represents the kth node at the output layer, y_k is the prediction result, and t_k is the ground truth.

The Sigmoid function defined in (6.3) was used as the activation function at the last layer for narrowing the output into the range of (0, 1).

$$Sig(x) = \frac{1}{1+e^{-x}} \tag{6.3}$$

Where x is the computation results of the last layer, and $Sig(x)$ is the output of the Sigmoid function with numerical values in the range of (0, 1).

6.3.3 Segmented-Federated Learning

Due to network environment diversity, a network's traffic data does not always fit into the single global model of FL; some networks have similarities with each other but others do not. We propose a novel adaptive learning scheme of Segmented-FL (Fig. 6.4), to safeguard the data privacy of participants as well as adapting to the various network environments by employing multiple global models.

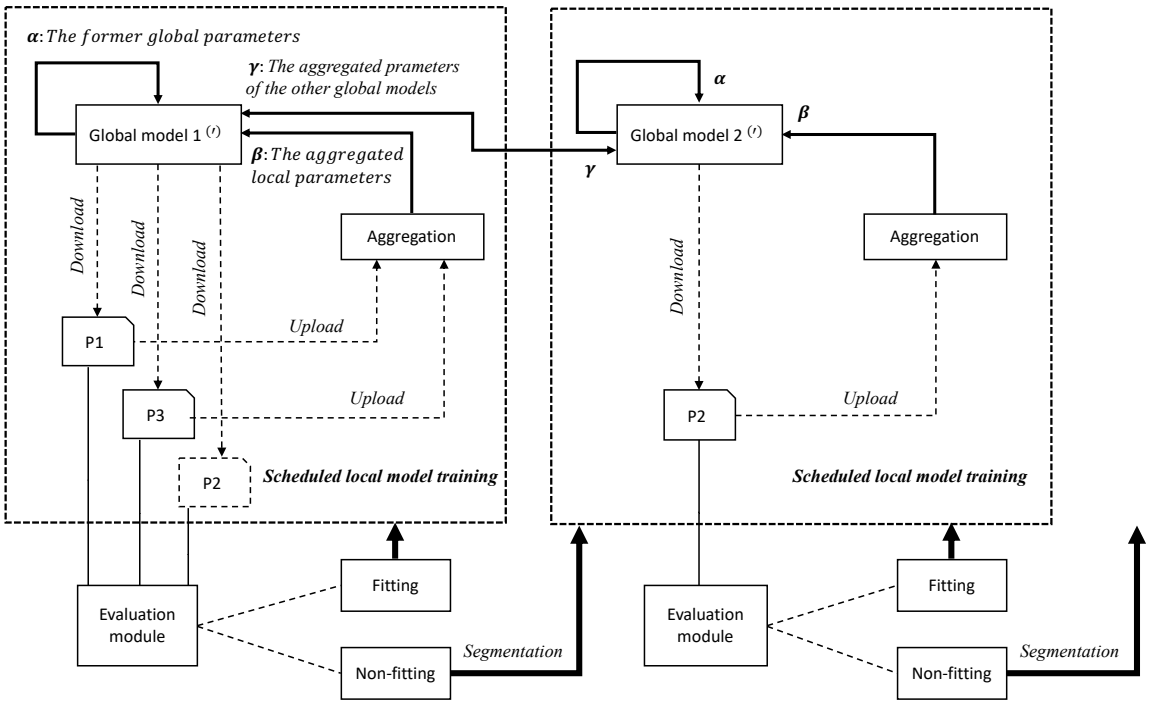


Fig. 6.4. Periodical evaluation and segmentation: if a participant's model fits into the current global model according to the result of evaluation module, they remain in the group, but if the participant does not fit into the current global model, we move it to another group.

Algorithm 6.1 *SEGMENTED-FL.* $p_{g=1}$ is the initialized global model. N_g is the number of participants related to global model g. N_t is the number of participants conducting local model training in a group. D_t is the local dataset of participant t. G represents all global models. h_t is the number of rounds for periodic local model evaluation. L_g is a list including segmentation information of participants. B is the batch size. E is the local training epoch. η is the learning rate. α, β and γ are the component ratios for aggregation. p_{new} is the newly initialized global model from the segmentation.

Server executes:
  initialize $p_{g=1}$
  for each round t = 1, 2, . . . **do**
    for each global model g = 1, 2, . . . **do**
      $B_t \leftarrow$ (split N_g into batches with a size N_t)
      $s_t \leftarrow$ (participants conducting model training from B_t)
      $s_r \leftarrow$ (the other participants not conducting model training)
      for each participant $t \in s_t$ **in parallel do**
        Execute(p_t, D_t)
      for each participant $r \in s_r$ **in parallel do**
        $p_r \leftarrow p_g$
      $p_g \leftarrow$ Aggregate($p_t, t \in s_t; p_g, g \in$ G)
      If t % h_t == 0 **do**
        for each participant $k \in s_g$ **do**
          $E_k \leftarrow$ avg(validation results of k in recent R_e rounds)
        $L_g \leftarrow$ Segment($E_k, k \in s_g; L_g$)
Execute(p_t, D_t)**:**
  $p_t \leftarrow p_g$
  $D_b \leftarrow$ (split D_t into batches of size B)
  for each local epoch i from 1 to E **do**
    for batch $b \in D_b$ **do**
      $p_t \leftarrow p_t - \eta \nabla / (p_t; b)$
  return p_t to server

Aggregate($p_t, t \in s_t; p_g, g \in$ G)**:**
  $G_{others} \leftarrow$ (the other global models except for the current one)
  $p_g \leftarrow \alpha \cdot p_g + \beta \cdot \text{avg}(p_t, t \in s_t) + \gamma \cdot \text{avg}(p_o, o \in G_{others})$
  return p_g to server

Segment($E_k, k \in s_g; L_g$)**:**
  $e_k \leftarrow \text{sigmoid}(E_k - \text{avg}(E_k, k \in s_g))$
  if e_k < threshold
    $L_g \leftarrow$ (remove participants k from the current global model)
    initialize p_{new}
    $L_g \leftarrow$ (attach k to the newly initialized global model p_{new})
  return L_g to server

In detail, the periodical performance evaluation is used for the quality verification of local model training in recent several rounds, based on the evaluation module defined in (6.4). By employing the average validation result as the metric, we periodically evaluate the performance of each participant's local model for training and sharing with the current global model. If a participant's model fits into the current global model according to the result of evaluation module, they remain in the group, but if the participant does not fit into the current global model, we move it to another group. If there is no group already created, a new group will be initialized based on the average-aggregated result of them. To decide whether a participant's model fits into the current global model, we apply a flexible threshold, and an evaluation result below the threshold represents it doesn't fit. Additionally, a relatively low threshold brings fewer participants for segmentation, whereas, a relatively high threshold brings more participants to a new global model. The participants moved into the new group conduct the next round's collaborative learning with the initialized global model. The maximum number of possibly existing global models can be set to five for the sake of simplicity.

$$d_i = E_i - \frac{\sum_{i=1}^{n} E_i}{n}$$

$$e_i = \frac{1}{1+e^{-d_i}} \tag{6.4}$$

$$threshold = 0.5 - h_f \times 0.01$$

Where n is the number of participants, E_i represents the average validation result of participant i in the recent several rounds, d_i is the computation result of the difference between participant i's performance and the average performance of all group members, e_i is the output of the evaluation module using the Sigmoid function to covert d_i into the interval of (0, 1), and h_f is the segmentation fineness to adaptively adjust the threshold.

For the parameter aggregation to update a global model, we applied the former global parameters, the average-aggregated local parameters, and the average-aggregated parameters of the other global models with varying component ratios, defined in (6.5).

$$p_t = \alpha \cdot p_{t-1} + \beta \cdot \frac{\sum_{i=1}^{n} p_i}{n} + \gamma \cdot \frac{\sum_{i=1}^{m} q_i}{m} \ (\alpha + \beta + \gamma = 1\ ) \tag{6.5}$$

Where p_t represents the updated global parameters, p_{t-1} represents the former global parameters, p_i represents the average-aggregated local parameters, q_i represents the average-aggregated parameters of the other global models, n is the number of participants who conducted local model training in this round, and m is the number of the other global models. α, β and γ represent the ratios of each component respectively, with a sum of 1.

Considering the balance of each component for the update and the stability of Segmented-FL, we employed 0.1 as β, 0.01 as γ, and consequently (0.9 - 0.01 · m) as α. The intact algorithm of Segmented-FL is shown as Algorithm 6.1.

6.4 Evaluation

6.4.1 Experiment setting

We adopted the experiment data of 20 participants from the LAN-Security Monitoring Project (Fig. 6.6). In this project, a device connected to a port of the switching hub or the router was used for collecting network traffic in the LAN (Fig. 6.7). Then the collected data was compressed, encrypted, and transported to the central server daily. The collected traffic data in a LAN includes broadcasts and direct traffic to the device. The Network Time Protocol (NTP) was employed for clock synchronization between a device and the server. We studied and applied a total of 60 days' network traffic data from 20 participants' LANs, from 1st October to 29th November in 2019.

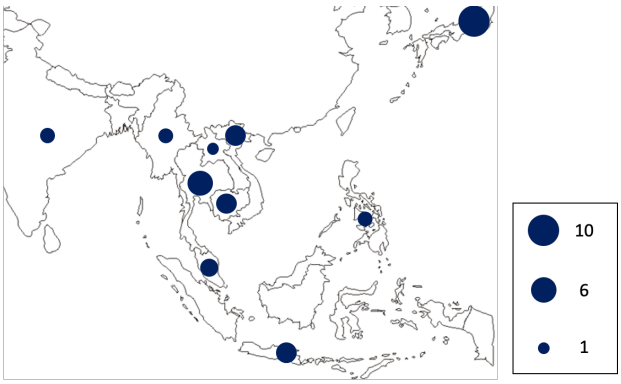


Fig. 6.6. The massively distributed LANs applied in the experiments.

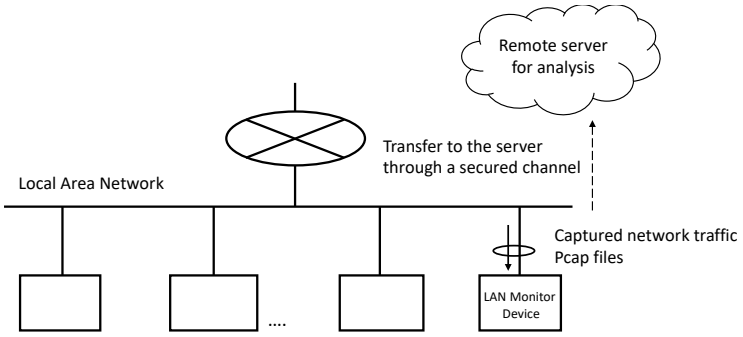


Fig. 6.7. The scheme of local network traffic collection: a monitor device was employed for collecting and transporting data in a LAN to the server.

By employing the aforementioned visual analytics, we generated the feature maps from collected network traffic data. (Fig. 6.8).

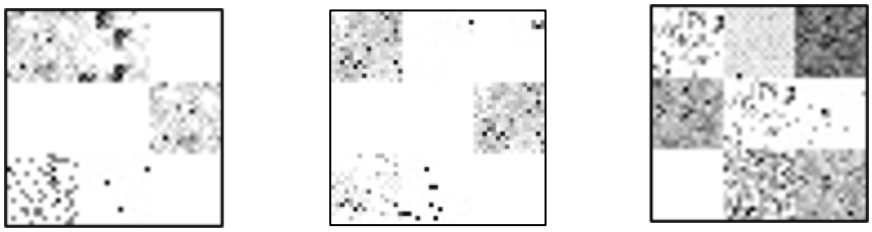


Fig. 6.8. Samples of the generated feature maps of network traffic.

To evaluate the adaptivity of Segmented-FL in typical applications, we have studied three types of virtual ground truths. This is because we cannot obtain the actual ground truth (i.e., manually labeled data) in the current phase. However, if Segmented-FL could adapt to many virtually generated ground truths, it says that it is possible to adapt to the actual ground truth when we obtain manually labeled data. We admit that a knowledge-based labeling method is a general approach to label possibly malicious data, doesn't certainly fit all participants' data. The employed three types of knowledge-based labeling methods are shown in Table 6.1.

Table. 6.1. Knowledge-Based Labeling

Methods	Knowledge Type	Definition
A	Server Message Block (SMB) attacks	Detection of any SYN445 to the monitor device.
B	TCP SYN flood attacks	TCP SYN from the same MAC address for more than three times within the time window of 128 seconds.
C	UDP unicast attacks	Detection of any UDP unicast to the monitor device (except NTP with a source port of 123 and DNS with a source port of 53).

First, for Method A, SYN445 (TCP port 445) is used to operate Server Message Block (SMB) directly over TCP/IP, where SMB is used for file sharing. However, this port is usually blocked or disabled. The TCP port 445 of a monitor device was opened for luring underlying attacks through SMB. Considering

a deployed monitor device in a LAN doesn't possess a purpose of file sharing, detected SYN445 to the monitor device would be malicious. Second, for Method B of a TCP SYN flood attack, an attacker usually sends repeated SYN packets to every port of the target. The target receives multiple requests to establish communication and responds to each request with an SYN-ACK packet from each open port, unaware of the attack. As the target's connection overflows, requests from other users would be denied, and the target could malfunction. As a result, we extracted MAC addresses of users in a LAN and considered the users sending three or more TCP SYN for requesting a connection with another user in the LAN during the time window of 128 seconds as malicious. Finally, for Method C, unicast is communication where information is sent from one point to another point directly in a network, and UDP is used to send datagrams. However, due to the monitor device not possessing a purpose of direct datagram sharing with other users, UDP unicast communications to the monitor device would be considered malicious, except for fundamental functions of DNS with a source port of 53 and NTP with a source port of 123. The ratios of results with each labeling method for 20 participants are shown above (Fig. 6.9). The graph shows the varying and imbalanced data compositions of participants.

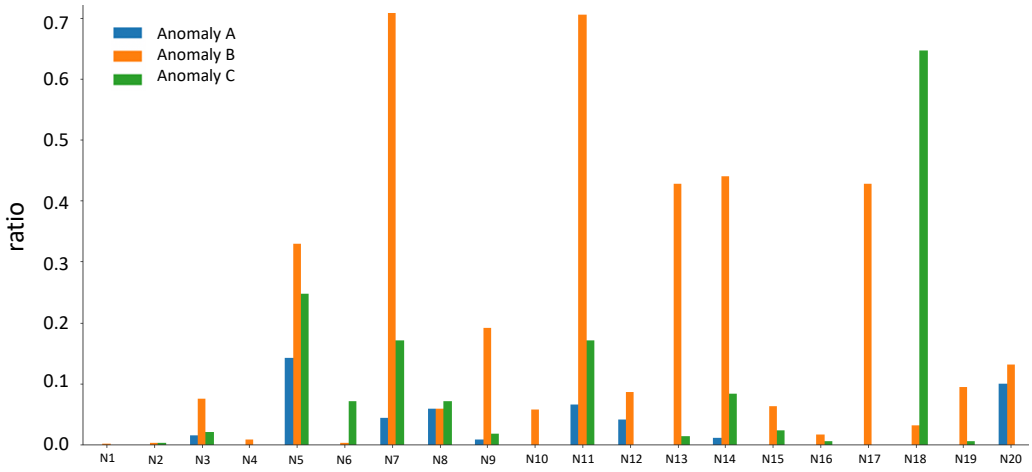


Fig. 6.9. The ratios of each type's anomaly in each node's dataset.

The constitution of the generated dataset for Segmented-FL is shown below (Table 6.2).

Table. 6.2. Dataset Profile

Participants	Method A		Method B		Method C		Participants	Method A		Method B		Method C	
	Benign	Malicious	Benign	Malicious	Benign	Malicious		Benign	Malicious	Benign	Malicious	Benign	Malicious
N001	46575	0	46497	78	46562	13	**N011**	43531	3044	13763	32812	38600	7975
N002	44512	0	44386	126	44381	131	**N012**	44688	1887	42568	4007	46569	6
N003	47138	706	44230	3614	46847	997	**N013**	46575	0	26644	19931	45950	625
N004	45589	0	45223	366	45585	4	**N014**	46073	502	26081	20494	42662	3913
N005	39140	6502	30639	15003	34364	11278	**N015**	45393	9	42521	2881	44363	1039
N006	46544	31	46433	142	43260	3315	**N016**	45949	3	45190	762	45689	263
N007	44541	2034	13639	32936	38613	7962	**N017**	46575	0	26652	19923	46575	0
N008	42248	2612	42228	2632	41654	3206	**N018**	46047	0	44604	1443	16303	29744
N009	46196	379	37671	8904	45747	828	**N019**	46113	2	41767	4348	45840	275
N010	46575	0	43913	2662	46575	0	**N020**	41925	4650	40489	6086	46567	8

Then, we employed Segmented-FL and the four-layer CNN as the basis of our scheme. The scheme conducted collaborative learning on a daily basis, which means for each round, participants conducted local model training based on the same day's data from the datasets. For each round, a specific number of participants conducting local model training were selected on a rolling basis. The reason for the rolling basis is to eliminate the imbalance between users' training times, thus bringing to more precise model evaluation and participant segmentation.

The batch learning method was applied, where the dataset was partitioned into small chunks for training purposes. Usually, a smaller batch size brings out a more satisfactory model at a slower speed, in contrast, a larger one can speed up the training process, albeit bringing out poorer training results.

A dataset separating from the datasets for collaborative learning was used to initialize the global model based on the four-layers CNN model with a learning rate of 0.00001, a batch size of 200, and an epoch of five. The dataset for initialization included 782 benign feature maps and 1186 malicious feature maps, which were generated from participant N001's network traffic in September 2019.

Besides, for local model training, we applied a learning rate of 0.00001, a batch size of 50, and an epoch of one. We employed the dataset consisting of sixty days' network traffic data to conduct experiments. Since for each round, participants applied data from one day, a total of sixty rounds' adaptive collaborative learning were conducted based on the dataset.

6.4.2 Varying hyperparameters of Segmented-FL

Precision, recall, and f1 score formed the basis of our metrics for evaluating the performance of the scheme, defined as (6.6). Precision shows the fraction of relevant feature maps successfully classified among all data in the validation set; recall shows the fraction of ones successfully classified among existing relevant feature maps. And the f1 score shows overall performance. Besides, these metrics were weighted by support, namely the number of true instances for each label, to deal with data imbalance.

$$Weight_P = \frac{TP+FN}{TP+FP+TN+FN}$$

$$Weight_N = \frac{TN+FP}{TP+FP+TN+FN}$$

$$Weighted - Precision = \frac{TP}{TP+FP} \times Weight_P + \frac{TN}{TN+FN} \times Weight_N \quad (6.6)$$

$$Weighted - Recall = \frac{TP}{TP+FN} \times Weight_P + \frac{TN}{TN+FP} \times Weight_N$$

$$Weighted - F1\ Score = \frac{2 \times Weighted-Precision \times Weighted-Recall}{Weighted-Precision + Weighted-Recall}$$

Where *TP* (True Positives) indicates the number of correct malicious network flows classified by the model in the data, *FP* (False Positives) indicates the number of incorrect malicious network flows classified in the data, *TN* (True Negatives) indicates the number of correct benign network flows classified by the model in the data, and *FN* (False Negatives) indicates the number of incorrect benign network flows classified in the data.

To optimize the scheme's architecture, we experimented with various hyperparameters to effect performance changes, including the factor of selected participants, evaluation frequency, and

segmentation fineness (Table III). Equation (6.7) was employed to compute the number of participants for local model training in a group from the factor parameter h_t.

Table. 6.3. Varying Hyperparameters of Segmented-FL

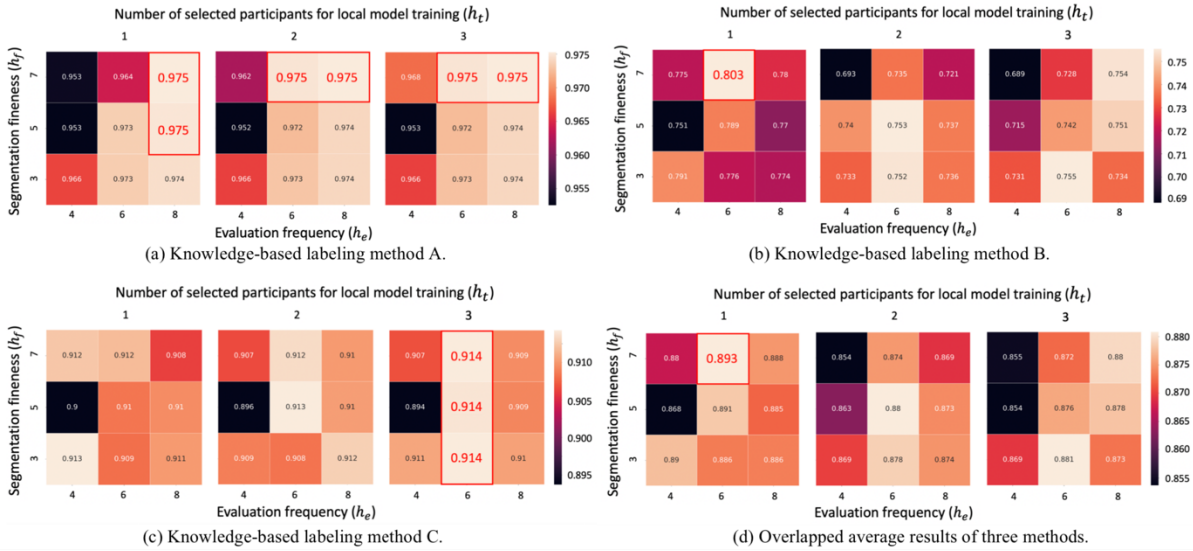
	Hyperparameter*	
Factor of selected participants for local model training (h_t)	**Evaluation frequency (h_e)**	**Segmentation fineness (h_f)**
1	4	3
2	6	5
3	8	7

*The hyperparameters are combined to find the best solution.

$$N_t = floor\left(\frac{max(N,h_t)}{h_t}\right) \tag{6.7}$$

Where N_t is the number of participants for local model training, N is the total number of participants in the group, h_t is the factor parameter, max is a function to compute the maximum between N and h_t, and floor is a function to obtain the greatest integer less than or equal to the input.

We tuned a total of 27 combinations of the various hyperparameters for each labeling method. By comparing the average validation weighted F1 score within the last several rounds (according to the evaluation frequency, e.g. an evaluation frequency of four brings the last four rounds' average), we evaluated the performance of Segmented-FL. The evaluation results for the three types of labeling methods are shown above (Fig. 6.10). In addition, we applied the F1 score as the metric for the periodic local model evaluation.



(a) Knowledge-based labeling method A.

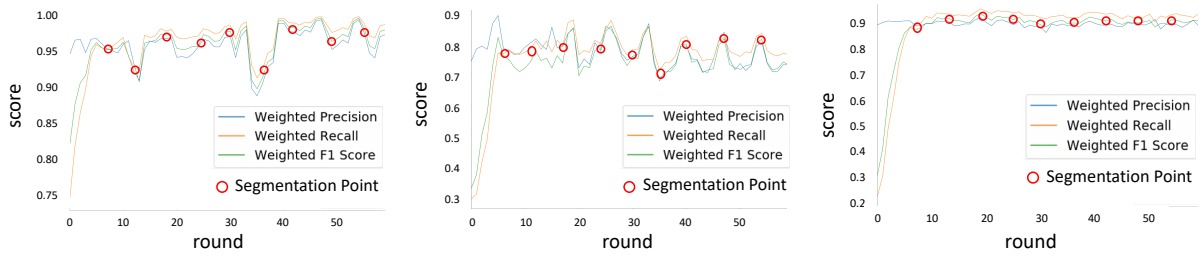
(b) Knowledge-based labeling method B.

(c) Knowledge-based labeling method C.

(d) Overlapped average results of three methods.

Fig. 6.10. Heatmaps of validation weighted F1 scores with various combinations of the hyperparameters.

We employed the average evaluation results of 20 participants in the last h_e rounds (the last evaluation cycle) as the final performance. Based on the study on the performance of Segmented-FL for various combinations of the hyperparameters, we selected an evaluation frequency of six, a segmentation fineness of seven, and a factor of selected participants for local model training of one. The averaged validation results of 20 participants at each round of Segmented-FL with the selected hyperparameters are shown below (Fig. 6.11).

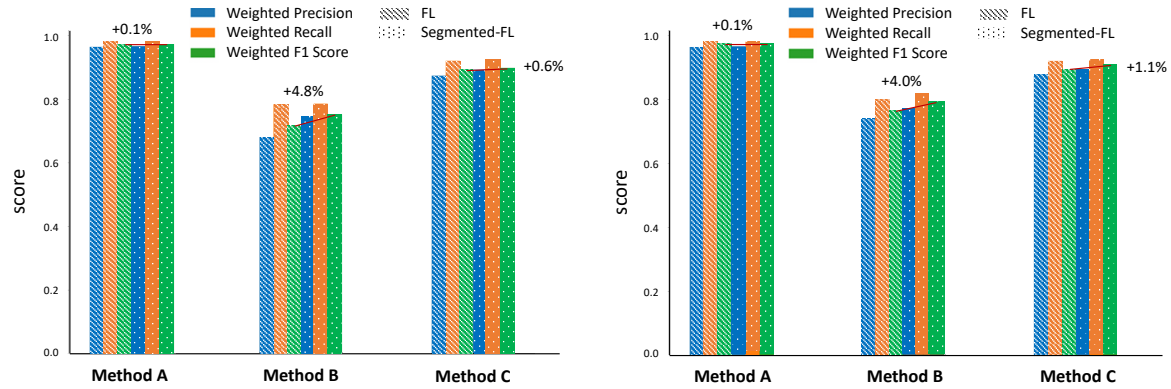


(a) Method A. (b) Method B. (c) Method C.

Fig. 6.11. The averaged validation results of 20 participants at each round of Segmented-FL with the optimized hyperparameters.

6.4.3 A comparison with Federated Learning for intrusion detection in massively distributed networks

We employed FL on the dataset with a factor of selected participants for local model training of one, which is the same as the selected hyperparameter of Segmented-FL. Then, the metrics of weighted precision, recall, and F1 score were used to evaluate the performance with the three knowledge-based labeling methods. The average of the last six rounds' evaluation results was used as the final performance of FL. Besides, considering the situation that for each round not all of the participants would conduct local model training ($h_t \neq 1$), we also studied the performance of FL and Segmented-FL when the factor of selected participants was two. A comparison was conducted between the performance of FL and the proposed method, with the two different settings of hyperparameters (Fig. 6.12).



(a) When the factor of selected participants is one. (b) When the factor of selected participants is two.

Fig. 6.12. A comparison between the performance of FL and Segmented-FL against a range of metrics including weighted precision, recall, and F1 score, when applying the two different hyperparameter settings.

As shown in the graphs, for each type of method, the average weighted F1 score based on validation sets had an increase. For Method B which showed relatively disadvantaged performance, the proposed method with the optimized hyperparameters contributed an increase up to 4.0% in the average weighted F1 score based on validation sets. When the factor parameter h_t was two, the gain rose to 4.8%. That's because a larger h_t brought to fewer participants for local model training at each round, contributing to a considerable decrease in the performance of FL. For Method A and Method C, though the obtained gains were not so large, the increases in performance were still sufficient since they showed relatively high validation scores. Secondly, data constitution imbalance in the networks let the weighted metrics put more emphasis on the validation results from benign data with these two methods. The comparison result shows Segmented-FL has better performance for intrusion detection in large-scale massively distributed networks.

6.5 Discussion

Segmented-FL has a feature of automatic architecture transformation based on the periodic evaluation, thus adapting to diverse data from massively distributed networks. Compared with FL, it showed more adaptability to collaborative learning with imbalanced and diverse data. The labeling methods employed were based on network traffic patterns within a small time slot of 128 seconds, which allowed us to detect malicious behavior precisely. Besides, the visualization of network traffic by feature maps provided more explainability for the detection of malicious behavior.

Several considerations of future improvement of the scheme include: a feature map was generated based on frequency information of several network protocols, however, other information such as packet length and payload size could also include hidden features of malicious behaviors; the experiment traffic data was observed by a normal host connected to a LAN (broadcasts and direct packets), hence, a combination with upstream captured network traffic would provide more insights into underlying malicious traffic features; $\alpha,\ \beta,$ and γ were employed to represent the ratio of each component for aggregation, further discussion on these parameters could be considered.

6.6 Conclusion

We proposed Segmented-FL to solve the problem of a network's traffic data not always fitting into the single global model of FL situation. This research is focusing on the study of how Segmented-FL works for intrusion detection in real-world massively distributed LANs. By studying the optimized hyperparameters of Segmented-FL and employing three evaluation methods, the validation result shows that Segmented-FL has better performance in all three types of intrusion detection tasks, achieving validation weighted F1 scores of 0.964, 0.803, and 0.912 with Method A, Method B, and Method C respectively. For each method, this scheme shows a gain of 0.1%, 4.0% and 1.1% in performance compared with FL.

Part III Security and Privacy in Federated Learning

Chapter 7 Defense Data Poisoning Attacks with Blockchain

7.1 Introduction

The current digitalization happening in various industries as well as in people's daily life is bringing us to a new era of information society. Federated learning has been used to collaboratively train a global model of machine learning, with distributed datasets, protecting users' data privacy. However, within a decentralized scheme like this, end-point threat caused by adversarial users can seriously affect the overall performance of the system, even revealing users' private data.

In this research, to address this issue, we consider a scenario where the participants are collaborating with each other for a multi-cluster classification with federated learning. This scheme consists of three users who train their local convolutional neural networks to classify the binary images of handwritten digits, within the range of (0, 9). Moreover, for the aggregation function of federated learning, the average is used to compute global parameters from uploaded local ones. Here, through uploading local parameters after training on their local datasets, users share information about the classification with each other. Besides, the parameter server conducts the aggregation of these local parameters thus generating a global model. After dozens of rounds' training, the scheme achieves a stable state for the classification task.

We consider one of the three users is an adversary, who performs an attack on the parameter server by uploading corrupted local parameters, thus affecting the performance of the other users. To represent these corrupted local parameters, instead of generating corrupted parameters directly, for each round, a certain degree of noise is added to the local training data of the adversary. Then the aggregation process at the parameter server will include the corrupted parameters into the global ones. After the other users download these latest global parameters, the performance of their local model for classification will be impacted due to the adversary's upload.

Moreover, we generate corrupted local training data based on two methods, salt and pepper noise, and circle occlusion. Then the adversary trains the local model based on the training datasets with various degrees of noise and occlusion individually, and federated learning with the three users for a total of 20 rounds is conducted. At last, we analyze the classification performance of these users and compare the results when adopting various degrees of noise.

In this research, we adopt a blockchain architecture, where uploaded local parameters with other related information are stored in the form of a block in the blockchain, which is located in the parameter server. Furthermore, a committee will perform a consensus to verify a qualified upload of local parameters, with a qualification score higher than the threshold. Then the block of the qualified update will be added to the blockchain along with a specific hash.

As a result, we evaluate the overall performance when adopting the blockchain architecture, and compare it with the one of federated learning without blockchain implemented. At last, It shows that the blockchain-based federated learning overperforms the other, with robust performance when it comes to various degrees of corrupted local training data.

This chapter is organized as follows. Section 2 discusses related works about the implementation of blockchain in federated learning. Section 3 provides an overview of the blockchain-based federated learning including end-point threat of corrupted local training data, and the implementation of the blockchain architecture against the threat. Section 4 presents the performance evaluation of the scheme with validation accuracy compared with the traditional federated learning. Section 5 discusses the evaluation result. In section 6, we conclude the chapter and give out the future work.

7.2 Related Work

In recent years, federated learning (FL) has emerged as a promising solution to challenges of addressing private and sensitive data collected and owned by distributed entities in a centralized server [55]. On the other hand, the blockchain technology has been showing great ability in privacy protection and security management. There have been many researches on blockchain's implementation in federated learning. For example, Li et al. [56] presented a decentralized federated learning framework based on blockchain to address the security issues inside the scheme. They use the blockchain for the global model storage and the local model update exchange. Moreover, Kim et al. [57] demonstrated an end-to-end latency model of a chained federated learning architecture, as well as characterizing the optimal block generation rate decided by communication, computation, and consensus delays. Furthermore, Weng et al. [58] presented the implementation of DeepChain, which is a distributed, secure, and fair deep learning framework. They conducted experiments on a real dataset for different settings to evaluate the scheme. It shows that the scheme can guarantee data privacy and provide auditability for the learning process. Preuveneers et al. [59] illustrated a use case of MultiChain in federated learning, where the autoencoder is adopted as the machine learning model for chained anomaly detection.

Different from the purposes of these former researches, we propose a method of combining blockchain with federated learning for combating end-point threats caused by various degrees of adversarial training data. Through the process of the consensus, the quality of local parameters is verified, thus alleviating the influence of the adversarial local parameters.

7.3 Blockchain-Based Federated Learning

7.3.1 End-point threat with corrupted local training data

A federated learning scheme allows users to collaborate with each other to train a global model using the local datasets, obtaining a more adaptive model for a specific task. In this research, we build a collaborative learning scheme for a multi-cluster classification of the binary images of handwritten digits within a range of (0, 9). We use the dataset of MNIST [60], and divide the training set, with a total of 60000 images, into three datasets, each of which is used as the local dataset of a user. Besides, all images are chosen randomly to ensure every user owns images covering the all ten digits from 0 to 9. We further divide the dataset of a user into 20 batches for each round's training, which means a user will conduct local training phases based on various datasets for the federated learning scheme. Then, for each round, every user trains the local artificial neural network with a standard architecture using these datasets. Then all newly updated parameters are uploaded to the parameter server for aggregation after a round's training phrases. For the next round, all users will download the aggregated global parameters from the server to update their local model. In total, we conduct 20 rounds' training phases for the multi-cluster classification task in this research (Fig. 7.1).

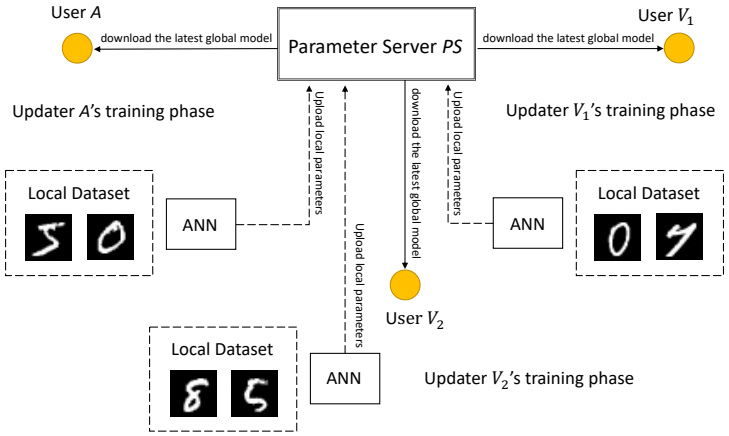
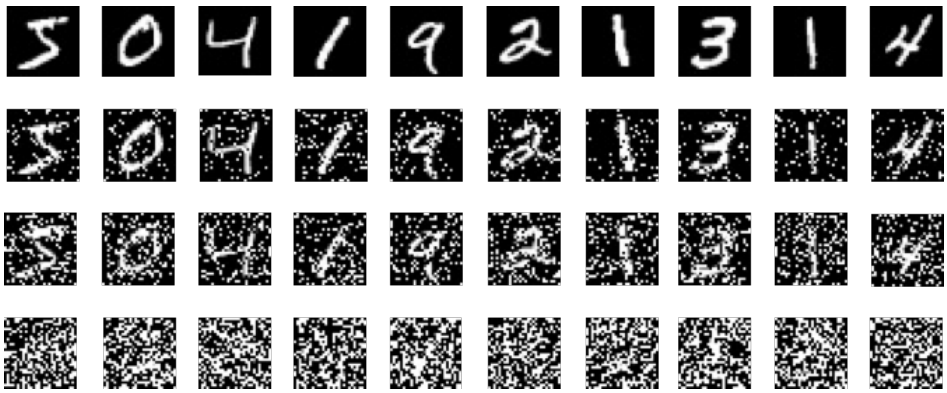


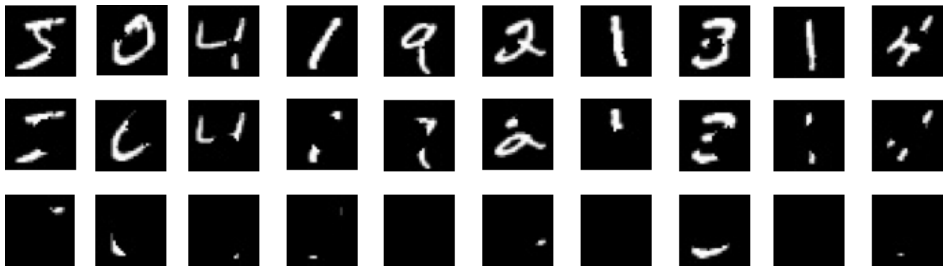
Fig. 7.1. The architecture of the federated learning for the multi-cluster classification task.

Moreover, we consider a scenario where an adversarial user in the scheme trains the local model with corrupted training data purposely. Then, through uploading the corrupted local parameters to the parameter server, the adversary can affect the other users' classification performance due to the parameter aggregation.

Here, to analyze and understand how an end-point threat will affect the overall performance of federated learning, we adopt two methods to generate the corrupted data of the adversary, who is the User A in the scheme. First, a method called the salt and pepper noise for adding random noise to the local data is utilized. We generate the corrupted images with various degrees of noise individually to conduct federated learning, thus analyzing the relationship between the noise degree and the overall performance of the scheme. The noise degree means the proportion of noise to an image, covering 0, 0.1, 0.2, 0.4, 0.8, and 1.0. Second, a method of occlusion using a random positioned circle with various diameters is adopted. Similarly, we conduct federated learning using these various degrees of occlusion to explore the relationship between the occlusion degree and the overall performance of the scheme (Fig. 2). Besides, to ensure a circle successfully covers a certain part of an image, the random position of it is limited within the center square of the image, with a side length of 14 pixels (half of an image's side length). The diameters of a circle cover 0, 6, 14, 20, 24, and 28 pixels.



(a) Samples generated with various degrees of noise (from top to bottom: when the degree is 0, 0.1, 0.2, and 0.4 separately).



(b) Samples generated with various degrees of occlusion (from top to bottom: when the diameter of the circle is 6, 14, and 28 pixels separately).

Fig. 7.2. Samples of the corrupted training data of the User A.

In addition, a four-layer convolutional neural network (CNN) with the following architecture is used as the standard artificial neural network in the scheme for federated learning (Table 7.1). This CNN model consists of two convolution layers and two fully-connected layers at the end. The kernel size of the first convolution layer is 3×3, with a stride of 1, and the one of the second convolution layer is 1×1, with a stride of 1. The activation function of ReLU is used between layers to define the outputs of layers. And the activation function of Softmax is used to convert the computation result into a range of (0, 1) as the output, representing the prediction confidence of each cluster, with a sum of 1.

Table. 7.1. The architecture of the convolutional neural network

Layer	Neurons	Kernel Size	Stride	Padding	Activation
Convolutional	784	3×3	1	1	ReLU
Convolutional	900	1×1	1	1	ReLU
Fully-connected	200	-	-	-	ReLU
Fully-connected	10	-	-	-	Softmax

Furthermore, we adopt a learning function called RMSProp with a discounting factor of 0.9, a learning rate of 0.001, an epoch of five, and a batch size of 64 for the training phase. We conduct federated learning using the four-layer convolutional neural network discussed above for the multi-cluster classification task individually with various degrees of noise and occlusion, each of which for a total of 20 rounds' training phases. Besides, for the parameter aggregation in federated learning, we adopt a method of the average, which means global parameters are defined by the average of all uploaded local parameters. Then, for each round, we compute the classification accuracy of a user based on the newly-downloaded global model form the parameter server and the local dataset of the user, as the validation accuracy at the current round. For example, the variance of each user's validation accuracy when there is no corrupted data added is shown below (Fig. 7.3).

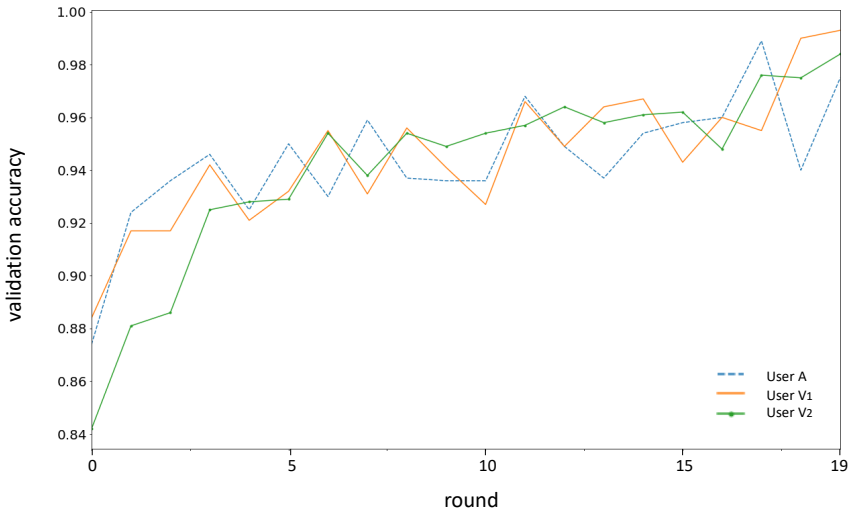


Fig. 7.3. The variance of each user's validation accuracy when the noise degree is 0 in the federated learning scheme.

5.3.2 Blockchain-based federated learning

In this research, we adopt the blockchain to manage uploaded local parameters and various versions of the global model. For each update, we store the information of the updater's identity, the timestamp, and local parameters, which can be used to identify the user related to a specific update and further refer to the content of it.

In detail, for each round of federated learning, the update from a user will first be added to a buffer in the parameter server, with other related information about the update process covering the identity of the updater, and the time of the update. Besides, for each new update, a committee consisting of several

members who is responsible for verifying the update will conduct a process called consensus. And if more than half of the members confirm an update meets all the criteria, the update will be verified and further added to the blockchain.

For the consensus, first, the information from the buffer will be converted to the form of a block in the blockchain, which is assigned with a specific hash based on the content of the block, using SHA-256. SHA-256 generates a unique 256-bit (32-byte) signature for the block of data. Consequently, the hash will change greatly even when the content of a block has a few differences from another one. In addition, the difficulty level of a hash in this research is three.

Then a committee member will verify if an update is qualified and if the hash is valid as well as meeting the difficulty criteria, also called the proof of work. And a nonce is the abbreviation for "number only used once," which is a number added to the encrypted block, thus meeting the difficulty level of three when rehashed. As a result, during the proof of work, these items are verified before adding the block to the blockchain:

- If the update of local parameters is qualified for aggregation
- If the generated hash of the block is valid and satisfies the difficulty criteria
- If the previous hash stored in the block matches with the hash of the latest block in the blockchain

At last, if a block meets the criteria listed above, it will be added to the blockchain. Besides, since there are a total of three users in this scenario, for every three updates, the parameter server will conduct an average aggregation process to generate a new global model based on verified update(s) in the blockchain. In addition, for evaluating the quality of an update, the local parameters of it will be used to conduct an inference using a committee member's local dataset, with the accuracy rate as the score of the update. Then an adaptive threshold based on the latest global model is adopted, which is defined in (7.1). If the score of an update is higher than the threshold, it means the update is qualified, otherwise, it is unqualified and will be excluded.

$$Threshold = \quad A_G(committee\ member's\ local\ dataset) - b \tag{7.1}$$

Where A_G represents the validation accuracy of the latest global model with the committee member's local dataset, and b is a bias to adjust the threshold thus managing sensitivity of the judgment. A larger bias allows more updates to be judged as qualified, whereas a smaller one makes it more difficult to be judged as qualified. Here, we adopt a value of 0.03 as the bias, which shows relatively stable and accurate judgment.

Moreover, the architecture and related explanation of the block in the blockchain is shown below (Fig. 7.4) (Table 7.2).

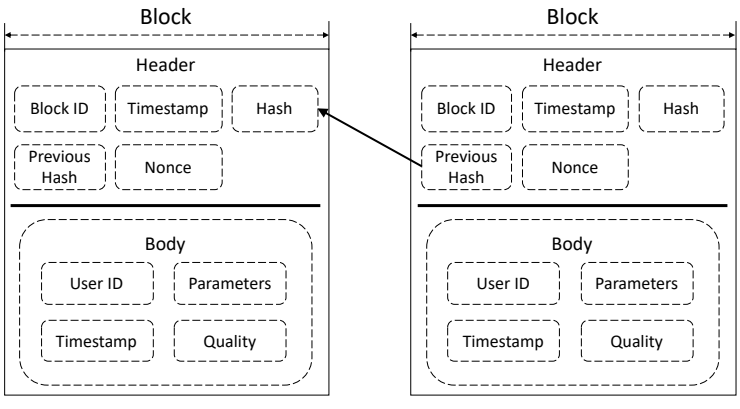


Fig. 7.4. The architecture of the block in the blockchain.

Table. 7.2. Explanation of the block

Type		Explanation
Header	*Block ID*	The height of the block
	Hash	Hash value of the block using a hash function
	Previous Hash	Hash value of the previous block using a hash function
	Timestamp	The time of the block being created
	Nonce	Nonce (number only used once) is a number added to a hashed block that, when rehashed, meets the difficulty level restrictions
Body	*User ID*	Identity of the parameter updater
	Parameters	The parameters of a local trained neural network
	Timestamp	The time of the body being added to the block
	Quality	The evaluation result of the update based on a committee member's local dataset

7.4 Evaluation

In this research, for evaluating the proposed scheme, we add various degrees of corruption to Adversary A's local training data with two methods of noise and occlusion. Then we conduct the blockchain-based federated learning discussed above. By adopting this method, we are supposed to exclude the adversarial updates of Adversary A. Besides, the other users in the scheme are considered as victims for evaluating the influence of the adversarial local data on the performance of these users.

Furthermore, we adopt accuracy defined as (7.2) to evaluate the performance of a model. We compute the average validation accuracy of Victim V1 and Victim V2 after 20 rounds' learning phases as the final result. Then, we compared the results of it with the ones of using the traditional federated learning for solving the multi-cluster classification task.

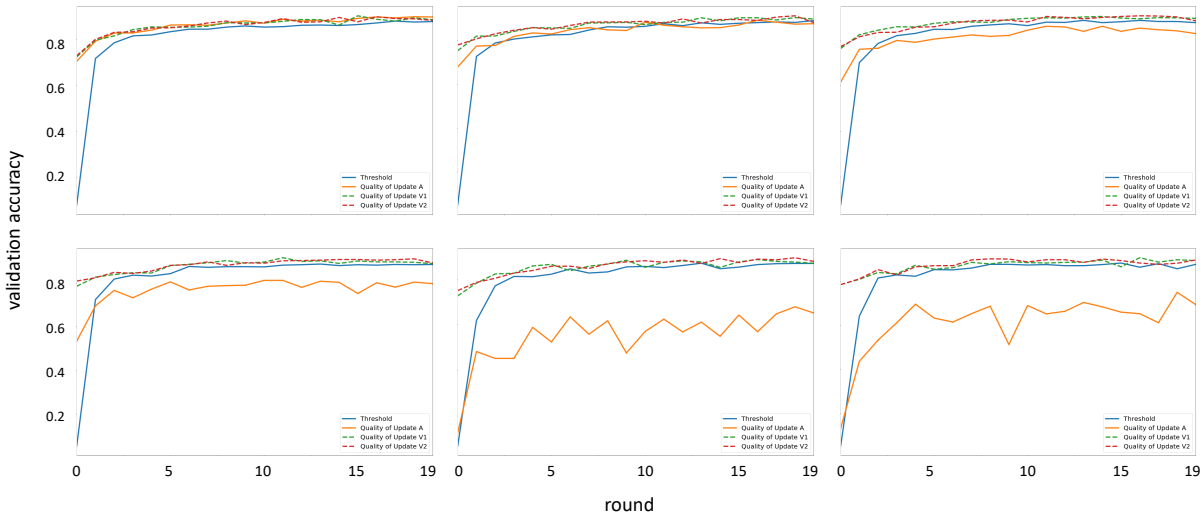
Besides, several degrees of noise were added to Adversary A's local data thus evaluating the ability of the scheme for combating end-point threat, including 0, 0.1, 0.2, 0.4, 0.8, and 1.0. Here, 0 means there is no noise added to data of Adversary A while 1.0 means instead of original data images generated by random noise are used for Adversary A's local training. Similarly, several degrees of circle occlusion were also adopted for the performance evaluation of the scheme, covering the diameters of 0, 6, 14, 20, 24, and 28 pixels.

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions\ made} \tag{7.2}$$

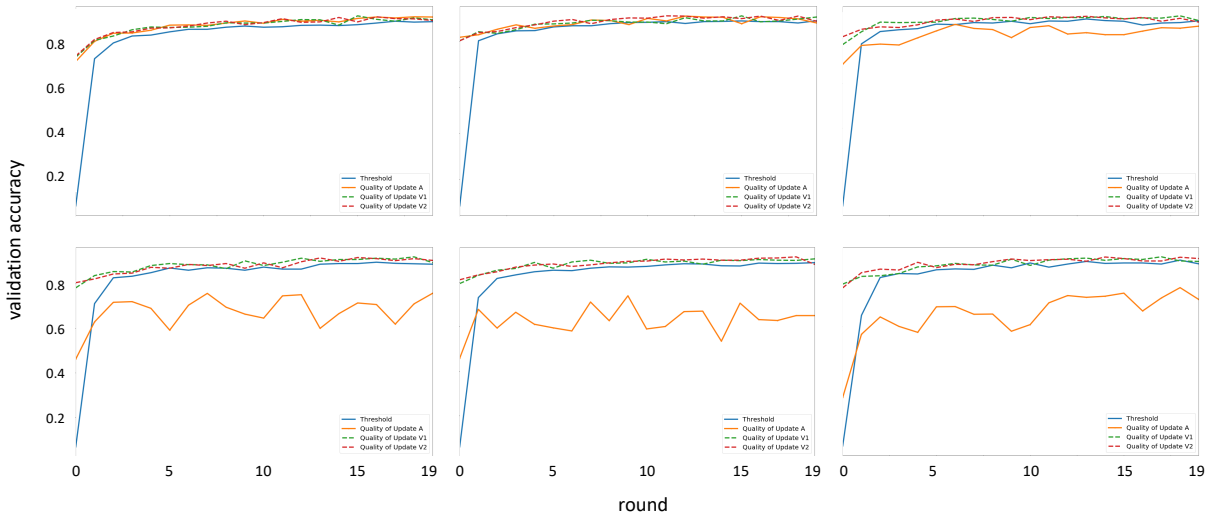
Moreover, since the scheme is evaluated for an intact federated learning process, we consider that the constitution of the committee is unchangeable during the 20 rounds' learning. Besides, in this research, we adopt a committee consisting of only one member. As a result, the judgment result based on the member's local dataset is considered as the result of the committee. In addition, a total of 10000 images from the test set of MNIST are used as the local dataset of the member, with 500 images for each round of learning.

Then, for each update, the local parameters of it are used to compute the accuracy for classifying the local dataset of the committee member, which shows the quality of the update. Furthermore, an adaptive

threshold is computed from the latest global model and the member's local dataset. If the quality is below the adaptive threshold, it will be considered as malicious to the scheme, whereas it will be considered as a qualified update. The judging processes of the committee when adopting various degrees of corruption to Adversary A are shown above (Fig. 7.5).



(a) When adopting various degrees of noise to Adversary A's data (top: from left to right are 0, 0.1, and 0.2; bottom: from left to right are 0.4, 0.8, and 1.0).



(b) When adopting various degrees of occlusion (circles with various diameters) to Adversary A's data (top: from left to right are 0, 6, and 14 pixels; bottom: from left to right are 20, 24, and 28 pixels)

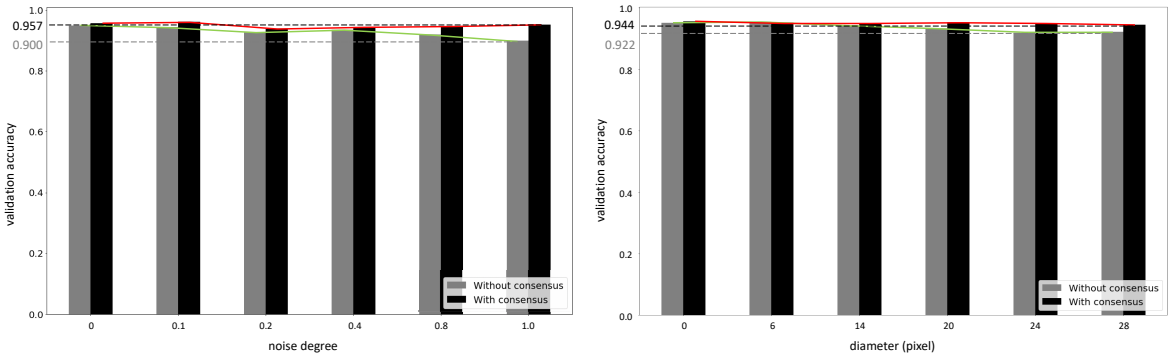
Fig. 7.5. The adaptive threshold for evaluating a new update's quality for the multi-cluster classification.

As a result, we excluded almost all adversarial updates by using this method and obtained a blockchain at the central server including information about all the updates and the global models of each round, which we can refer to quickly and restore a previous version of a global model when necessary (Table 7.3).

Table. 7.3. The information stored in a block of the blockchain

Body	{**'userID'** : 'A', **'update'**: "{'W1': array([-0.0379207, ..., -0.01780896])}", **'timestamp'**: 1594177035.9936042, **'quality'**: 0.831}
Head	{**'index'**: 5, **'body'**: Body, **'timestamp'**: 1594177674.3792052, **'previous hash'**: '3ffaa12094765d43b43c834ba6b0c58cc36d6753f9b6dea5044bd5c87cb6c76d', **'nonce'**: 13861, **'hash'**: '0004a9acdd2415eda84b9a889e5927c4e64db403eeb13215527a23f118ccca35'}

Furthermore, we conducted the blockchain-based federated learning for the handwritten digit binary image classification and computed the validation accuracy of each user's local model at each round. The accuracy is computed immediately after global parameters are downloaded and copied to a user's local model. Besides, the average accuracy of Victim V1 and Victim V2 at the last round is utilized as the overall performance of the scheme. Then, we evaluated the scheme when adding various degrees of noise and occlusion to Adversary A's local data separately. Then, we computed the overall validation accuracy results of the traditional federated learning, thus showing the comparison between these two methods as below (Fig. 7.6).



(a) When adopting various degrees of noise. (b) When adopting various degrees of occlusion.

Fig. 7.6. The average evaluation result of User V1 and User V2 for the multi-cluster classification based on the proposed method (with consensus) and the traditional federated learning (without consensus).

As we can see, with the increase of data corruption's degree, the proposed blockchain-based federated learning with consensus keeps great performance, showing an overall validation accuracy of 0.957 for classifying the handwritten digit binary images when noise with a degree of 1.0 is added to the scheme, and one of 0.944 when circle occlusion with a diameter of 28 pixels added. On the other hand, the traditional federated learning without consensus shows a decreasing validation accuracy result while the degree of noise and occlusion keeps increasing. At last, it decreases to 0.900 when the degree of noise rises to 1.0, and one of 0.922 when the diameter of circle occlusion rises to 28 pixels.

7.5 Discussion

In this research, we proposed the blockchain-based federated learning for combating adversarial end-point user. A process called the consensus allows the overall evaluation of an update before being added to the blockchain and further aggregated into the global model. By this method, the end-point threat's influence on federated learning can be alleviated, the scheme achieving robust performance with various degrees and types of adversarial data. Compared with the traditional federated learning, the proposed method shows a more robust inference process for the classification.

7.6 Conclusion

The blockchain architecture is used in federated learning, where a block is used to store information related to the update, and the consensus is used for judging the quality of a new update. We evaluated the scheme by using the average validation accuracy of the two victims in the scheme, as well as comparing the results with the ones of the traditional federated learning method. Several degrees of noise and occlusion are adopted to evaluate the robustness of the scheme. At last, it shows a resilient and stable classification process while adopting various degrees and types of corruption, with an overall validation accuracy rate of 0.957 when the degree of noise arrives at 1.0, and one of 0.944 when the diameter of circle occlusion rises to 28 pixels. For future work, a scheme with more users participating can be considered. Besides, the method of selecting a committee, and the threat from inside the committee will be further discussed.

Chapter 8 Mount Adversarial Attacks with Generative Adversarial Networks

8.1 Introduction

Privacy is a dominant concern in different aspects of the digital world, so is the case in deep learning (DL) systems. Though the current digitalization and knowledge acquisition happening in various industries is bringing us to a new era of smart society, the two essential questions have been how to protect personal data while enabling compliance with training model sharing regulations.

Federated learning (FL) has been used as a collaborative learning scheme for users with partial and imbalanced data to train a DL model for a shared goal. The privacy-oriented nature of FL allows users to collaborate through sharing a local model, as a result, without disclosing their private data.

Unfortunately, even within such a decentralized learning scheme, an adversarial attack launched by an end-point attacker could effectively exacerbate the privacy and security of such a learning environment, revealing the private data of a user.

An adversarial attack on FL involves several aspects such as manipulating endpoint training data and falsifying a local training model. We focused on the manipulation of endpoint training data to affect the learning process. We launched adversarial attacks based on generative adversarial networks (GANs) targeting at a specified data class (Fig. 8.1). In our settings, the adversary had access to the latest global model and knew the cluster distributions of the victims' datasets.

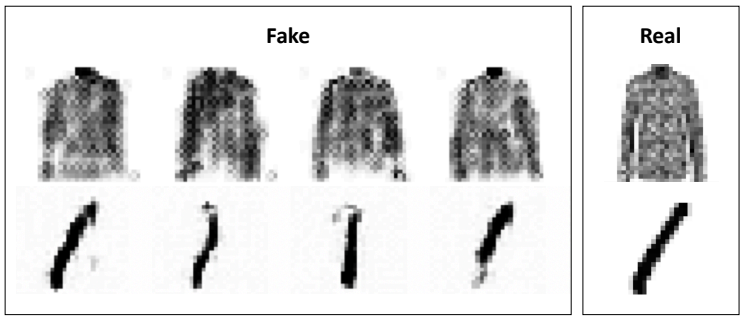


Fig. 8.1. Samples of generated adversarial data based on the target-oriented GANs.

We measured the quality of reconstructed adversarial samples by employing the Euclidean distance, showing that such attacks could successfully steal a victim's private local data while keeping the performance of main training tasks mostly unchangeable thus being difficult to detect.

This chapter is organized as follows. Section 2 discusses related work on adversarial attacks on DL and FL systems. Section 3 presents the mechanism of the GANs-based threat model in FL. Section 4 presents the performance evaluation of the attacking scheme based on a range of metrics with two image datasets of MNIST [60] and Fashion-MNIST [61]. Section 5 discusses the pros and cons of the threat model. Finally, in section 6, we conclude the chapter and give out the future work of this study.

8.2 Related Work

Despite the extensive applications and utilization of DL, many researchers have shown an adversary can manipulate the decision result of the trained model, and even steal original data from the model.

Unlike centralized learning schemes, FL is a distributed learning approach that allows multiple actors to collaboratively build a global model without disclosing private data. Unfortunately, recent research has shown that FL systems are often vulnerable to adversarial attacks such as data poisoning, backdoor attacks, etc. An adversary can perturb an FL system with sufficient shared gradients [62][63]. For instance, Zhang et al. [64] proposed a backdoor poisoning attack on FL to produce a global model that performs high accuracy on its main task while also exhibiting good attacking performance on targeted inputs. Cao et al. [65] presented a distributed poisoning attack on FL based on label-flipping, with 10 users involved, and researched two key factors affecting the attack success rate, the numbers of poisoned samples and attackers.

On the other hand, the implementation of GANs has been increasingly used to defeat machine learning algorithms [66][67]. For instance, Fredrikson et al. [68] designed a threat model against a facial recognition system, reconstructing the facial images of a participant through accessing the application programming interface (API). Additionally, Zhang et al. [69] presented an adversarial approach to violate the health data of patients in cloud-assisted e-health systems.

Whereas former research has focused on the success rate of an attack, our efforts focus on measuring reconstruction quality. We conducted target-oriented adversarial attacks based on GANs using two datasets of MNIST and Fashion-MNIST, with different difficulties for perturbing the system, to evaluate the efficiency of the threat model. Additionally, we studied how the attack would affect the performance of the other users and the global model, which is also different from former research, where weights were put on the adversary's performance.

8.3 Adversarial Attacks on Federated Learning with Generative Adversarial Networks

8.3.1 Dataset

We employed two datasets for conducting the experiments, including MNIST and Fashion-MNIST. These datasets pose different degrees of difficulty for perturbing the FL system. MNIST [60] is a handwritten digit image dataset containing 50,000 training samples labeled as 0-9, and 10,000 test samples. The size of the images in it is 28×28. Fashion-MNIST [61] is an image collection of 10 types of clothing containing 50,000 training samples labeled as shoes, t-shirts, dresses and so on and 10,000 test samples. The size of the images in it is 28×28. Each data class of MNIST and Fashion-MNIST includes 5,000 training samples and 1,000 test samples.

8.3.2 federated learning

Federated learning (FL) is a collaborative learning scheme that shares intelligence on local model parameters across the participating users to generate a global model with private data never leaving the nodes. In detail, first users download the latest global model from the central server, thus updating their local models. Then, a subset of the participants would conduct the model training based on local datasets. The trained models are uploaded to the central server for aggregation where the global model is stored and maintained. There are several methods used for the aggregation, and the commonest one is the averaging aggregating, i.e., computing the average values of all the uploaded local parameters.

In our study, we considered a scenario where several participants worked together to solve a multi-cluster classification problem. Each participant only had access to limited data from a single cluster. For instance, with MNIST, each participant had the images of a specific digit and the common goal of the users was to train a global model with the ability to identify all the ten classes of images. By sharing trained local model parameters, they achieved a global model with satisfying performance in the multi-cluster classification task. Additionally, participants agreed on the same training model architecture and hyper-parameters such as the learning rate, the number of update epochs at each iteration, etc.

8.3.3 Threat model: adversarial attacks using GANs

The generative adversarial networks (GANs) involve both a generator network and a discriminator network. GANs are usually employed to generate highly imitated images and data distributions through a race between the discriminator and the generator. The discriminator is trained to detect fake data drawn from a real data set and a fake data set created continuously by the generator (Fig. 8.2). Each party learns to improve its performance as much as possible by minimizing the prediction loss. The loss of the generator is computed from the prediction results of the generated image and the original one. And the loss of the discriminator is computed from the predicted results and the ground truths. The tradeoff for updating the model can be summarized with Equation (8.1).

$$\min_{G}\max_{D} V(D,G) = \mathbb{E}_{x\in p_{target}(x)}[\log \mathrm{D}(x)] + \mathbb{E}_{z\in p_{noise}(z)}\left[\log\left(1-\mathrm{D}\left(G(z)\right)\right)\right] \tag{8.1}$$

Where G represents the generator, D represents the discriminator, x is the original input image, z is the input gaussian noise of the generator, and $G(z)$ is the generated adversarial sample.

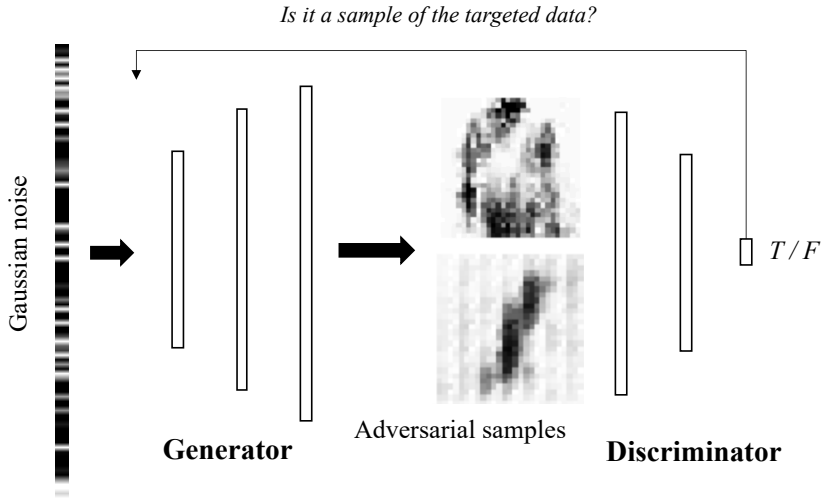


Fig. 8.2. Adversarial samples generation using GANs.

By employing GANs, an adversary in a collaborative learning scheme such as FL can steal private data from other users by just downloading the latest global model, which includes the aggregated parameter information of users' trained local models. To implement GANs in an adversary's training scheme, first, the adversary downloads the latest global model from the central server and uses it to replace the discriminator of implemented GANs. Then the generator generates adversarial samples from Gaussian noise and updates itself based on the computing loss between the inference result of the adversarial samples from the discriminator and the targeted classes, as defined in (8.2).

$$L_G(\theta_g) = \mathbb{E}_{z\in p_{noise}(z)}\left[\log\left(\mathrm{D}\left(G(z)\right)\right)\right] \tag{8.2}$$

Where L_G is the loss of the generator, θ_g is the model of the generator, z is the input gaussian noise of the generator, and $G(z)$ is the generated adversarial sample.

We designed and employed a four-layer deep convolutional neural network as the agreed model architecture to train on local datasets (Fig. 8.3). The first convolutional layer has a convolution kernel of size 3×3 with a stride of 2 and it takes one input plane and it produces 32 output planes, followed by a ReLU activation function. Whereas the second convolutional layer takes 32 input planes and produces 64 output planes and it has a convolution kernel of size 3×3 with a stride of 2, followed by ReLU. Then the output is flattened to a size of, where a linear transformation is applied which takes as input the tensor and outputs a tensor of size 200. A ReLU activation function is applied to the output, which is then followed by another linear transformation which takes as input the tensor of size 200 and outputs a tensor of size 11, where the 11th output is associated with the generated results by the adversary.

Another four-layer deep convolutional neural network was employed as the architecture of the generator of GANs (Fig. 8.3). Generator takes as input a 100-dimensional uniform distribution followed by a linear transformation with an output of size 1568, and a ReLU activation function. Then the output is reshaped to 32 planes with a size of 7×7. A transposed convolution kernel of size 3×3 with a stride of 2 is applied, followed by ReLU. Then another transposed convolution kernel of size 3×3 with a stride of 2 is applied, taking 32 input planes and producing 16 output planes, followed by ReLU. Finally, a transposed convolution kernel of size 3×3 with a stride of 1 followed by a Tanh activation function converts the input to a 28x28 image as output.

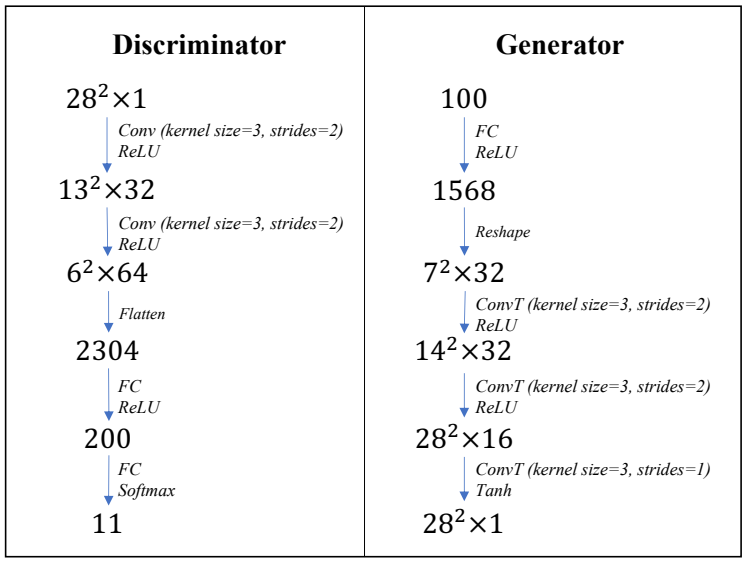


Fig. 8.3. The architecture of the discriminator and the generator.

Furthermore, we employed the Adam learning function to update the model parameters as defined in (8.3). We found through experiments that applying a relatively larger learning rate of the discriminator with respect to the learning rate of the generator could bring better attack performance. We applied a learning rate of 0.005 and 0.001 for the discriminator and generator, respectively. Despite the difference in the learning rate, we applied the same learning function to update the models. A batch size of 16 and an epoch of one was employed to update a local model at each round, allowing a small batch size to contribute to more delicate reconstruction results.

$$m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * \frac{\partial L}{\partial W_t}$$

$$V_t = \beta_2 * v_{t-1} + (1 - \beta_2) * \frac{\partial L}{\partial W_t} \odot \frac{\partial L}{\partial W_t}$$

$$\widehat{m_t} = \frac{m_t}{1-\beta_1^t} \tag{8.3}$$

$$\widehat{v_t} = \frac{v_t}{1-\beta_2^t}$$

$$W_{t+1} = W_t - \eta * \frac{1}{\sqrt{\widehat{v_t}}+\epsilon} \odot \widehat{m_t}$$

Where L is the training loss, W is weights of the node, η is the learning rate, β_1 and β_2 are the exponential decay rates with values of 0.5 and 0.999 respectively, and ϵ is used to prevent a zero-valued denominator, with a value of 1e-7.

To successfully reconstruct a victim's local data, we mounted the adversarial attack on the global model achieving a validation accuracy better than 0.90 with the test set of the applied dataset (Fig. 8.4).

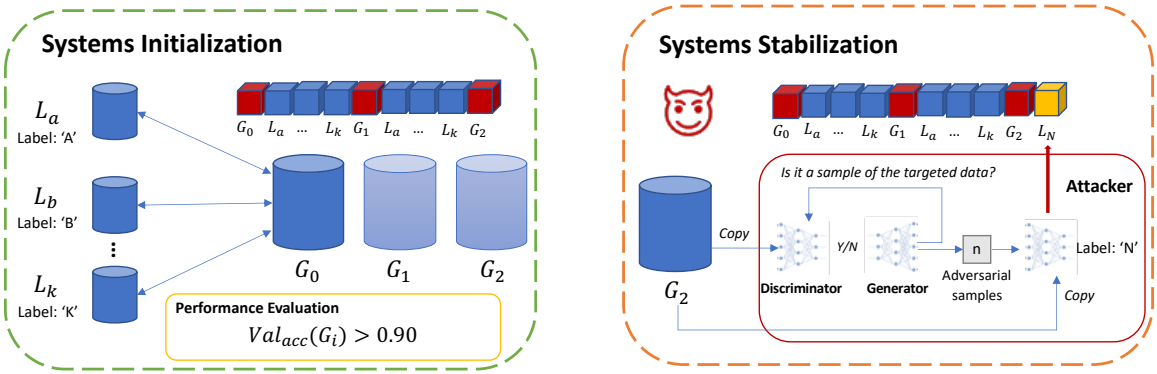


Fig. 8.4. The threat model in FL: the adversary mounted a target-oriented adversarial attack on FL after the global model of the systems achieved a validation accuracy better than 0.90.

For the adversarial training of the GAN at each round, the adversary conducted 500 epochs' updates with a batch size of 16. The adversarial samples generated from the updated generator were used to train the local model with the agreed architecture identical to other users' local models, using the 11th class as labels. Upon all users completing the training, the complete updated local parameters were uploaded and further aggregated to update the global model by computing the average value of each parameter. Finally, the updated global model was broadcast to the participants for a new round of learning.

The adversary updates the model of its discriminator network by downloading the latest global model from the central server, as a result, generating more qualified adversarial samples to train the adversary's local model with the misguiding labels. The injection of the malicious model parameters is aggregated to the global model and downloaded by the victim. As such, the victim would need more effort on training a model that could correctly classify the local data due to the misguiding labeled data, thus exposing more features of the local data class to the adversary. Whereas the generator performed better and better, generating highly imitated images, revealing the private data of the victim.

8.4 Evaluation

8.4.1 Experiment Setup

We considered a scenario where 10 users were training with FL, each of which had 5,000 images from one data class in the training set. On the other hand, the adversary didn't possess any data itself, thus

training purely on the generated adversarial samples from GANs using the same 5,000 images. Then, for each round, a user randomly selected 50 images from the local dataset to conduct the local model training.

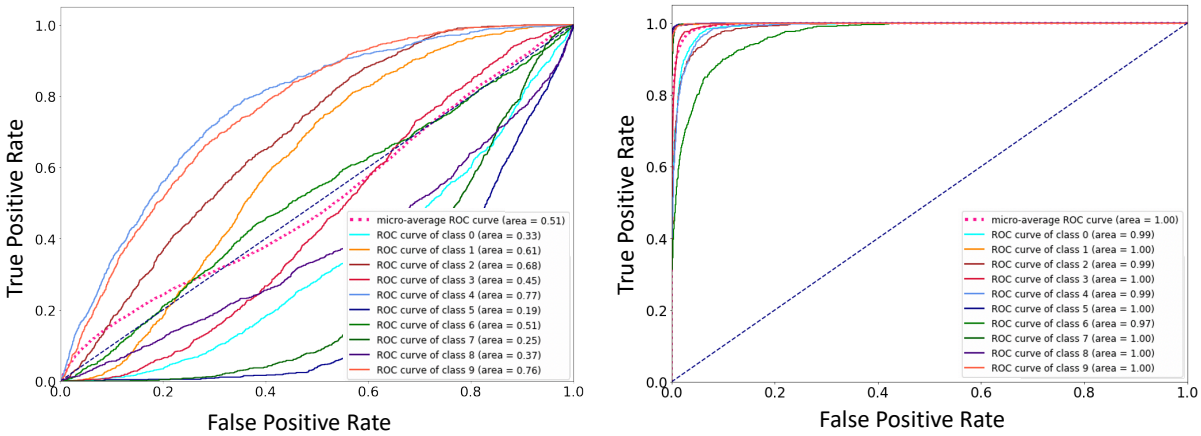
We employed the two datasets respectively to conduct the experiments. For each attack attempt, we assigned the adversary a specified target, i.e., a specific class in the dataset available to a user in the scheme.

8.4.2 Numerical Results

We first mounted the adversarial attacks on the FL systems based on MNIST with 11 users including the adversary. The attack was initialized after the performance of the global model arrived at 0.90. To evaluate the systems' performance when initializing and stabilizing respectively, we employed the ROC curve (receiver operating characteristic curve), which is a graph showing the performance of a classification model through the relationship between the true positive rate (TPR) and the false positive rate (FPR), as defined in (8.4). Additionally, AUC (Area under the ROC Curve) measures the entire two-dimensional area underneath the entire ROC curve, providing a measurement approach to model performance across all possible classification thresholds, where a larger AUC represents the better performance. The ROC curves and AUCs of the global model concerning each class when systems initializing and achieving the desired performance are shown below (Fig. 8.5).

$$TPR = \frac{TP}{TP+FN}$$
$$FPR = \frac{FP}{FP+TN} \tag{8.4}$$

Where TP (True Positives) and FP (False Positives) indicate the number of relevant images correctly and incorrectly classified by the model, respectively. TN (True Negatives) and FN (False Negatives) indicate the number of irrelevant images correctly and incorrectly classified, respectively.



(a) Systems initializing. (b) Systems achieving the desired performance of 0.90.

Fig. 8.5. ROC curves and AUCs of the global model at different phases.

Moreover, to evaluate the system's performance before and after mounting the adversarial attacks, we employed a range of metrics covering Accuracy, Precision, Recall, and F1-Score defined in (5). Precision represents the fraction of relevant images successfully classified among all images in the validation set; Recall represents the fraction of relevant images successfully classified among all existing relevant images. F1-score represents the overall performance by combining Precision and

Recall. All metrics are macro values computed by each class and taken the average of all classes.

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

$$Recall = \frac{TP}{TP+FN} \tag{8.5}$$

$$F1\text{-}score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

We mounted the adversarial attacks and strived to reconstruct the private data of a user in the scheme, targeting the class of digit one in MNIST. From the initialization of the systems, we conducted a total of 100 rounds' learning. We evaluated the performance of the global model at each round based on the aforementioned metrics, with the test set of MNIST consisting of 10,000 images in 10 classes as the validation dataset (Fig. 8.6).

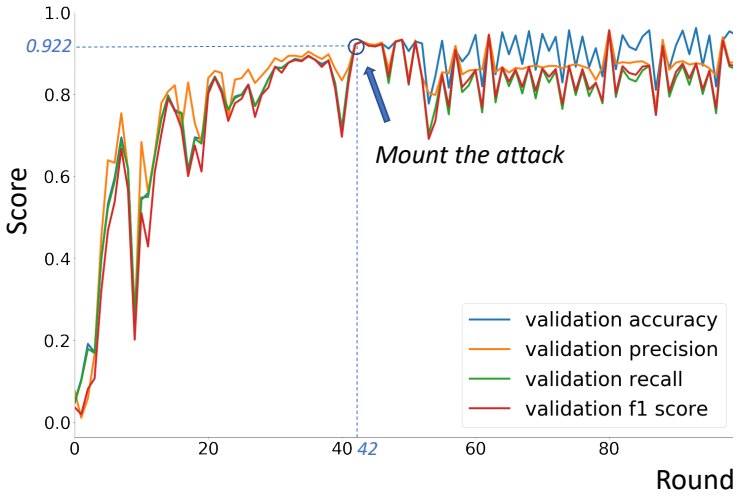


Fig. 8.6. Measurement of global model performance. We mounted the attack after the validation accuracy of the global model reached 0.922 at Round 42.

As we can see, after launching the attack, the validation accuracy stayed stable at a high score, while the macro precision, recall, and F1-score showed a slight decrease, which was due to the targeted class one was affected by the malicious model training of the adversary based on the adversarial samples generated by GANs.

To evaluate the quality of a reconstructed image at each round, we employed the Euclidean distance defined in (6), which is usually used to measure distance in high dimensional space. For each round's adversarial training, we computed the Euclidean distance between the reconstructed images generated by GANs and the targeted original images, representing the reconstruction quality (Fig. 8), where a smaller Euclidean distance showed better quality.

$$Dis(I_{fake}, I_{real}) = \sqrt{\sum (I_{fake} - I_{real})^2} \tag{6}$$

Where I_{fake} represents the adversary's reconstructed image, I_{real} represents the targeted victim's original image.

As shown in the graph, the distance kept decreasing with the update of GANs through adversarial

model training. After 49 rounds of adversarial training - represented as Round 91 in Figure. 8.6, the model achieved a distance value of 0.00173, which was significantly smaller than the distance at the initialization stage of the attack, 0.221.

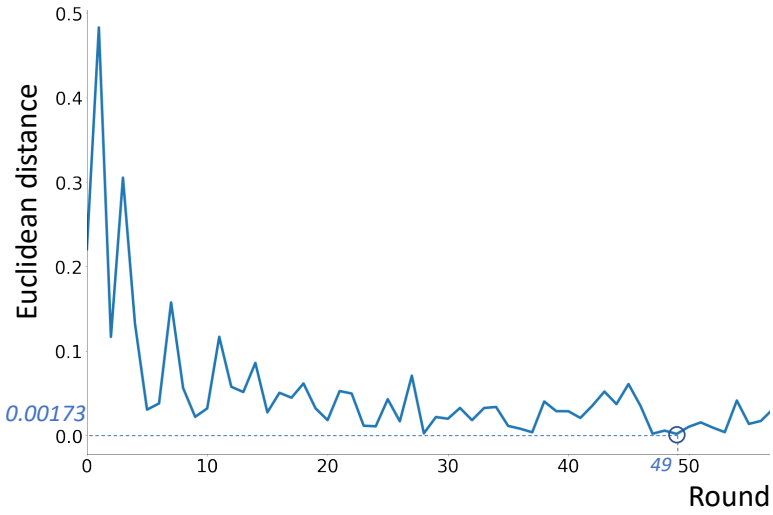


Fig. 8.7. Euclidean distances between the generated adversarial samples distribution and the targeted original images distribution of the victim at each round of the adversarial training.

Similarly, we launched adversarial attacks targeted at the data class of "shirt" based on Fashion-MNIST. A total of 1250 rounds were conducted to complete the attack. The measurement results of global model performance and the Euclidean distances are shown below (Fig. 8.8).

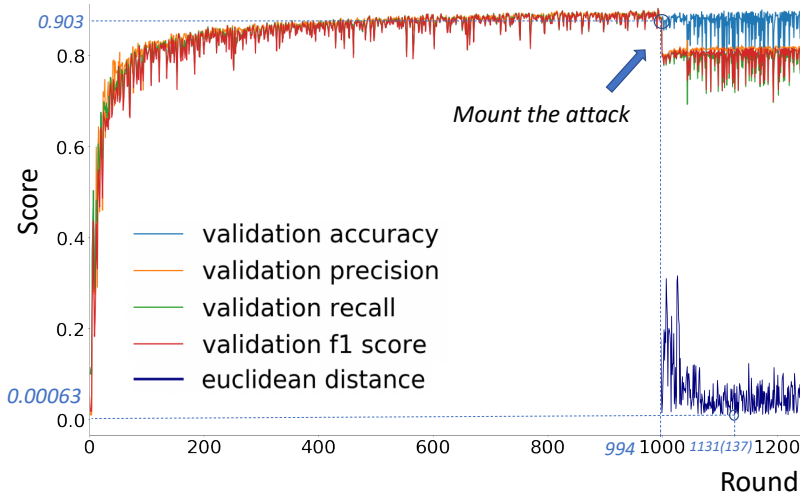
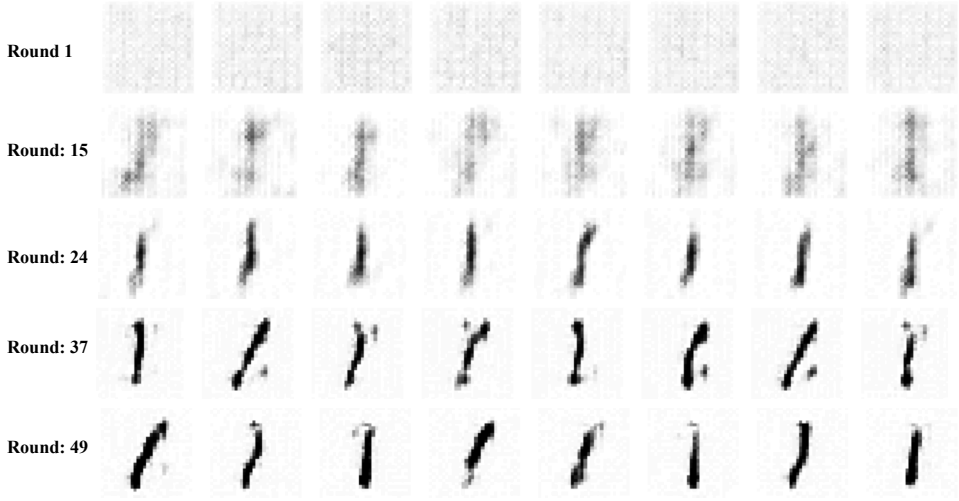
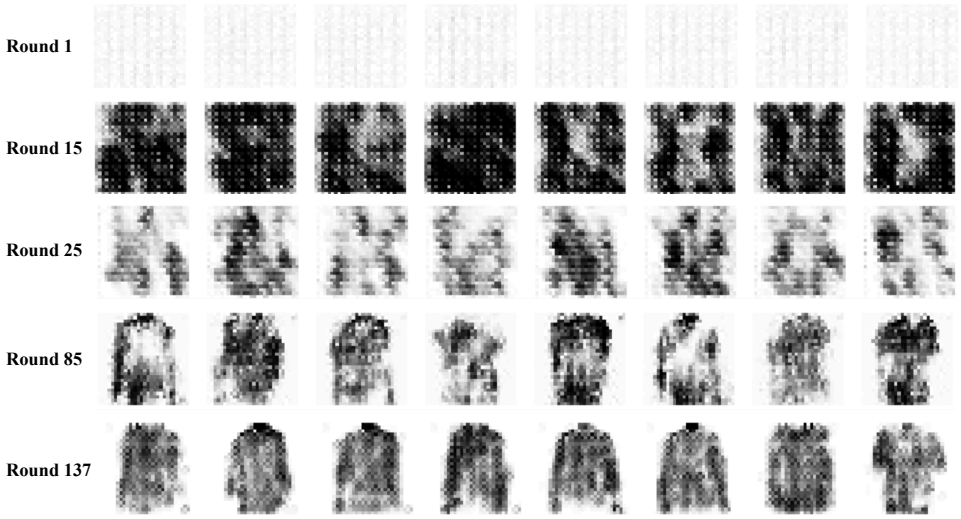


Fig. 8.8. Measurement results of global model performance and the Euclidean distances between the generated adversarial samples distribution and the targeted original images distribution when applying Fashion-MNIST as the dataset.

Consequently, by implementing GANs into the local model training, the adversary successfully reconstructed the private data of a targeted user through model sharing when applying both datasets, the results of which are shown below (Fig. 8.9).



(a) Reconstructions targeting at the data class “digit one” in MNIST.



(b) Reconstructions targeting at the data class “shirt” in Fashion-MNIST.

Fig. 8.9. Reconstruction results of the adversarial attacks on the FL systems.

8.5 Discussion

We studied an approach based on GANs for mounting target-oriented attacks on FL systems. We showcased the evolution of generated adversarial samples by the adversary in the two datasets using the Euclidean distance. There are still several challenges remaining. We experimented using the image datasets, whereas the adaptability to other types of data such as network traffic data for intrusion detection needs more validation. A system with more users involved using more imbalanced training data will be focused on in the future.

Furthermore, for defending such an adversarial attack on FL systems, approaches have been proposed to marginate these threats. Shokri et al. [63] proposed the privacy-preserving collaborative learning with selectively shared partial gradients. Lu et al. [70] presented a decentralized federated learning framework based on blockchain to address the security issues. In future work, we consider implementing a blockchain-based defense model against such adversarial attacks on FL systems.

8.6 Conclusion

By manipulating local training data based on insights from shared global parameters, we constructed an adversarial attack using a GAN architecture. To evaluate the reconstruction results, we introduced the Euclidean distance, and finally, we succeeded in attacking an FL system with 10 users and reconstructed the targeted data class with a minimum Euclidean distance of 0.00173 and 0.00063, in MNIST and Fashion-MNIST, respectively.

Part IV Modeling and Optimization

Chapter 9 Aircraft Detection in Remote Sensing Images

9.1 Introduction

Aircraft detection is a hot topic nowadays. Tasks such as illegal aircraft detection from satellite images and the control of communication in the airport attract many attentions. For example, if we consider the task of controlling communication in the airport, usually the control tower can monitor the airplanes which is going to take off and landing. However, when it comes to the monitor of the inactive airplanes parking on the field, it is usually considered difficult to monitor and control the communication of these airplanes.

The difficulties of aircraft detection include the relatively small aircraft compared to the large sensing image. In detail, images taken far from the ground have a problem that the pixels compounding aircraft are relatively small compared to the background with various color intensities and complex features. Furthermore, another problem is that the shadow from light in the environment will have a great influence on the detection of aircraft. The aircraft with a strong shadow is considered difficult to be detected. What is more the different angles, colors of the wing and body of the aircraft will obviously contribute to the difficulty of this task.

In previous researches, detecting aircraft in images have great achievements. Many methods have been proposed to solve Aircraft detection in remote sensing images. Some researches use methods named proposal regions to generate thousands of regions which are used to detect aircraft. However it usually can be difficult to solve the problem like the influence of light, different angles of wings and so on.

In our research, we use an approach of saliency map including series of preprocessing which inferences the edge and other features of an object in an image. Researches of using saliency map to solve aircraft detection problems are still not so many. Through extracting the objects from the generated saliency map, we can get several intact objects instead of several separated parts of objects. To extract the objects, we use an algorithm named minAreaRect to find the smallest circumscribed rectangles of objects as proposal regions in an image. Then, we crop and resize these regions, and save the transformed regions.

At the same time, we build a CNN model using normalization for each layer and a learning function named RMSProp to adjust the performance. We collect the remote sensing images and build a dataset covering both aircraft and backgrounds, which is used to train the model. After training, the model can tell if a region is an airplane or a part of the background. We use the trained CNN model to excite the inference, through which we can extract the aircraft from all the detected objects in one image.

In the part of generating a saliency map (Fig. 9.1), we use a pipeline of several processes to extract features of objects from the original image. First, we convert the image into a gray scale image and use the blur to delete noises in the image.

Then we use the method of threshold to advert images into binary images based on the similarity among the neighboring pixels. After that, we use a method of combining dilation with erosion to connect separated parts into an intact object and also narrow the boundary of the airplane at the same time. Lastly, we use flood-fill to separate the object from the surrounding background. Through this step, we can get several alternate regions for further detecting.

Considering the backgrounds of images are usually three types: grassland, land, and cement. In this research, we collect images of these three different types of background as a negative dataset. On the other hand, we collect images of different types of airplanes and also parts of the airplane to build a positive dataset. At the same time, we use the argumentor to build a robust dataset, which improves the representation of the objects.

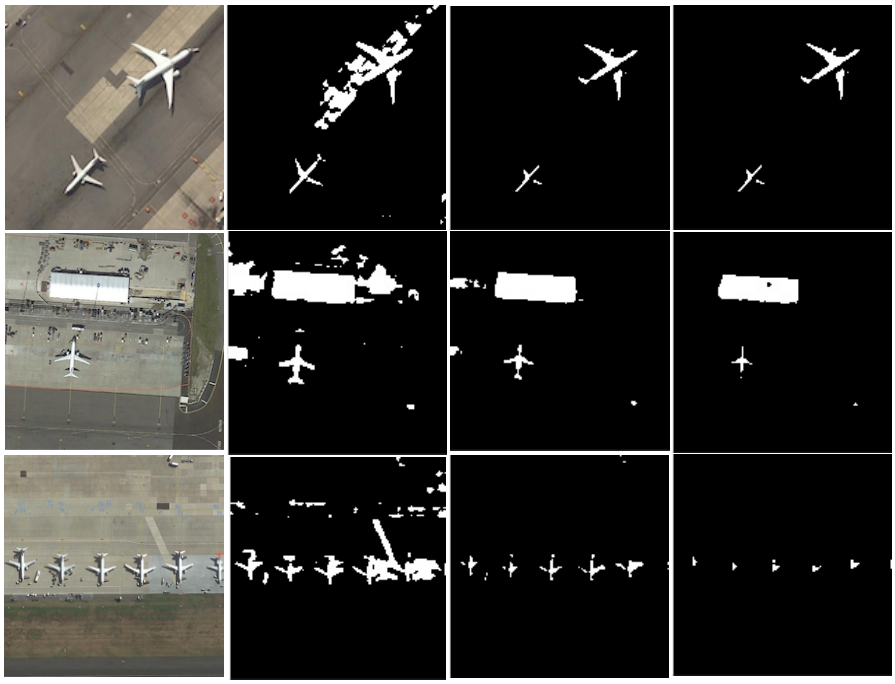


Fig. 9.1. A saliency map of an original remote sensing image.

At last, we use a CNN model to learn the features of aircraft and background. After training, the model can do inferring to judge if a region is an airplane, and the selected regions are the result of aircraft detection.

This chapter is organized as follows. Section 2 discusses related works about aircraft detection. Section 3 provides an overview of our algorithm including the generating of a saliency map and training of the CNN model using a prepared dataset. Section 4 presents the performance evaluation of the algorithm for aircraft detection based on the recall rate. Section 5, we discuss the result of the experiment. Section 6, we conclude the chapter and give out some possibilities of broader practical use of the algorithm.

9.2 Related Work

The task of aircraft detection in remote sensing images with a complicated background attracts many attentions. In order to solve this problem, there are many methods already being proposed. Ammour et al. [71] used a two-stage method to solve the car detection problem. They first use mean-shift algorithm to segment the image, and for the second phase, they use VGG16 [72] model to extract region feature, followed by a Support Vector Machine (SVM) classifier.

As a powerful generic feature extractor, CNNs has been used in a wide field in the computer vision area [73,74,75]. A breakthrough object detection method in the paper of R. Girshick et al. is Region based CNN (R-CNN) [73], using a method named selective search to extract the region proposal from the original image, and then using the convolutional neural network to represent the features of objects in the region proposals. Another widely used approach is YOLO v1, which is proposed in the paper of J.Redmon et al. [76]. YOLO contains 24 convolutional layers followed by 2 fully connected layers.YOLO divides the input image into an S × S grid, each of which is associated with one object.

However, the problems caused by the strong shadow of the airplane and its relatively small body compared to the large scale background, are still difficult to solve. In a recent research of Guoxiong Huet al., they proposed an approach to replace the selective search with saliency [77], using several preprocessing which greatly reduces the number of proposal areas, thus improving the efficiency and accuracy of subsequent feature extraction and classification.

In our research, we use the saliency to reinforce the feature like color, intensity, and texture of the objects in images, which contributes to a result of fewer proposal areas, reducing the computing cost greatly. After that, we use a CNN model to judge the region whether a region is an aircraft. Compared with the previous research, we evaluate our scheme in a larger scale remote sensing images, aimed to detect multiple aircraft in relatively large remote sensing images efficiently.

9.3 Aircraft Detection using Saliency and CNN

9.3.1 Saliency Map

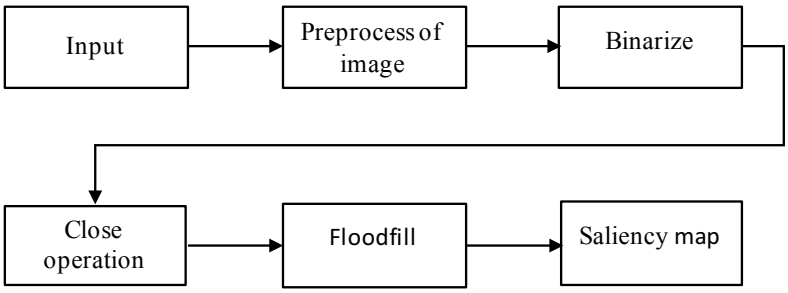


Fig. 9.2. Preprocessing flowchart.

In this research, we propose an approach which uses saliency to reinforce the contrast between objects and background in an image (Fig. 9.2). As a preprocessing of the image, first, we convert the original image into a grayscale image, and use a filter with a size of 3×3 to delete the noise in the image such as labels on the road, cars and the shadow of buildings at the same time.

Second, we adopt a method called threshold to advert an image into a binary image. Threshold is used to extract the objects from an image by setting all the pixels with intensity values above a threshold to a value which can be considered as the foreground. At the same time, all the remaining pixels will be adjusted to a background value. The threshold is adjusted to 180 in this research. The threshold mentioned above has a great influence on the detection of aircraft in images. We adjusted the value of the threshold and verify the result of detecting, aimed to find an appropriate value as a threshold. We investigate three different values of the threshold from an interval of (0,255), like 160, 180, 200, and use three different images to compare the results of them. As we can see, when we use the threshold of 160, it shows many noises, even after we adopted the method of a blur to delete the noise in the image with a kernel of 3×3 in advance. On the other hand, if we choose a threshold of 200, it seems that the small airplane may be difficult to be detected but possible to be considered as noise in the image. As a result of the comparison, we use a threshold of 180 as a suitable value.

However, we found that when the shadow of an airplane is very strong, the result of the saliency map will not be so ideal, instead of an intact airplane the result shows several detected separated parts of the airplane (Fig. 9.3). So, the regions of interest detected will be only parts of an airplane such as wings, tails, or bodies, adding to the difficulty of the classification between the airplane and background.

Fig. 9.3. The influence of the shadow when imposing saliency on the original image: left: a saliency map when the shadow is not so strong; right: a saliency map when the shadow is very strong.

R-CNN is one of the earliest algorithms used in the task of object detection. In R-CNN, we attain the proposal regions by using a selective search algorithm [78], which randomly generates around 2k regions for the next step of processing of the CNN. We compared the result of the saliency map and the selective search algorithm as below (Fig. 9.4). From the comparison, we can see through the method of saliency map, fewer regions are detected as targets with less noise, contributing to both the reduction of computation and accuracy of CNN in the next step.

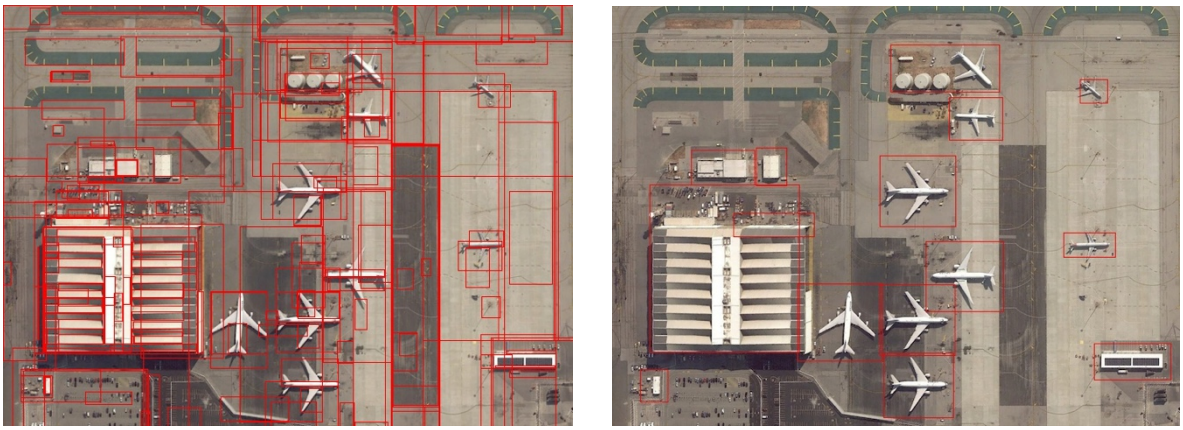


Fig. 9.4. The comparison between the selective search algorithm and the saliency map method: top: the result when using the selective search; bottom: the result when using the saliency map.

9.3.2 Dataset

In order to find only aircraft from all the detected objects through the generating of the saliency map, we train a convolutional neural network in order to classify all the regions. For the training, we prepare datasets that includes a positive one, namely the aircraft, and a negative one consisting of different kinds of backgrounds of the remote sensing images.

When it comes to the backgrounds, usually three different types can be considered. They are the grass, land, and cement like buildings and tracks. We extract the three types of backgrounds from the raw images, twenty images for each type and adopt an approach named Augmentor to add more variational data into the negative dataset through some methods like the elastic distortion and perspective transform.

At the same time, considering the possibility of detecting only some separated parts of the aircraft when we use a saliency map, we adopt the approach of Augmentor to build more multifarious data and add them into the positive dataset too, contributing to the detection of aircraft, only the wings or body of which is detected. Through the Augmentor, we are expecting to get some training data of only wings, the tailor or the body of aircraft.

Erosion is used to compute a local minimum over the area of the kernel which makes the object in white smaller. On the other hand, dilation is used to compute the maximum inside the kernel and replace the image pixel in the anchor point position with the maximum.

Here, we use a combination of the erosion and dilation to solve the problem mentioned above by putting these separated parts into an intact airplane. At the same time, we can also narrow the bounding box of the detected objects through the erosion function. At last, we get an intact saliency result of the airplane by using a combination of the erosion and dilation.

The regions of objects can be difficult to extract from the background because many detected points of bounding boxes are overlapped with the regions of the background. In order to extract the objects from the background, we use the flood-fill, an algorithm that integrates many related small areas from the same object based on the target color.

Then, we use an approach named contours to connect all the continuous points with similar color or intensity. There are many different contour approximation methods to be used, and we use the CHAIN APPROX TC89 L1 here. Through the generation of the saliency map and contours, we find the contours of objects in the image (Fig. 9.5).

Fig. 9.5. Top: Objects detection result of one airplane from the saliency map; bottom: Objects detection result of multiple planes from the saliency map.

Totally, we generate 400 multifarious data through the Augmentor and 400 images of aircraft from the remote sensing images, totally 800 images as the positive dataset labeled as the airplane. On the other hand, 800 images of backgrounds as a negative dataset which is also generated by the Augmentor are prepared. The negative dataset is labeled as the background (Fig. 9.6)

Fig. 9.6. Top: samples of different kinds of backgrounds; bottom: generated samples of aircraft from Augmentor.

9.3.3 CNN

A convolutional neural network is one kind of deep learning, consisting of convolution layers, pooling layers and sometimes fully connected layers, which has a movement invariance with respect to the input data and exhibit a goodperformance. In this research, we adopt an adjusted CNN model [78] (Fig. 9.7)

with 2 convolution layers combined with the pooling layer after each one, and 2 fully connected layers in the end. For each convolution layer, we use an approach of normalization function to transfer the input data into a normal distribution before computing it in the next layer. And we use the RMSProp as the learning function to improve the performance of the model and also the stability of training.

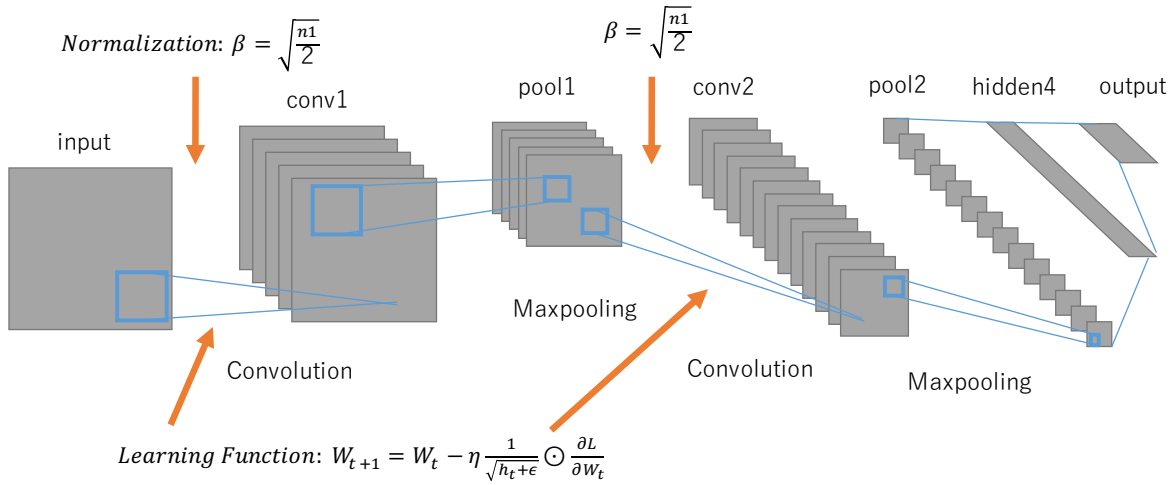


Fig. 9.7. The CNN model for classifying the dataset between backgrounds and aircraft.

We use the batch learning, which divides the dataset into small batches when training. The normalization is used in the model, which is used to transform the input data in each layer in order to improve the invariance to the input data at each layer. What is more, in order to get an output of classification, at the last layer, we use a fully-connected layer with an output of two classes as aircraft and no-aircraft, using an activation function named softmax (8.1), applying a standard exponential function to each element x_i of the input vector x and normalize these values by dividing by the sum of all these exponentials. After this, each component including negative, greater than one, or might not sum to 1, will be in the interval (0, 1), with a sum of 1.

$$\boldsymbol{y_i} = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x^j}} \tag{1}$$

Furthermore, we use the RMSProp function as a learning function with the following characteristics, an emphasis is placed on the latest gradient information more than the past gradient information, and gradually forget the past gradient information so that the new gradient information is greatly reflected. In (8.2), L is the loss function defined in advance, W is the weight of the network, and ρ is the decay rate, which is 0.9 here. In (8.3), η is the learning rate, and ϵ is a factor to prevent division by zero.

$$h_t = \rho * h_{t-1} + (1-\rho) * \frac{\partial L}{\partial W_t} \odot \frac{\partial L}{\partial W_t} \tag{2}$$

$$W_{t+1} = W_t - \eta \frac{1}{\sqrt{h_t+\epsilon}} \odot \frac{\partial L}{\partial W_t} \tag{3}$$

We train the model using the dataset prepared in advanceincluding two datasets of aircraft and background in order to attain a CNN model which can classify between aircraft and background. After that, we use this trained model to extract aircraft from all the detected objects in the saliency map.

9.3.4 Detection of airplanes

As mentioned above, we use a CNN model to judge whether a detected object is an aircraft. After the step of saliency, we transform all the regions of interest, the regions of objects detected in the saliency map from the original sensing image into the same size, as an input of the model to do the inference. Then the model will identify whether it is an aircraft or other objects existing in the image (Fig. 9.8). At last, we can get the bounding boxes of objects detected as aircraft. And the information of the bounding box will be used to label the position of aircraft.

Fig. 9.8. Results of the detection and classification using the trained CNN model. The detected objects in images are labeled by green lines, and the objects classified as airplanes are labeled by red rectangles. In the third image from the left, we can see that only the airplane is labeled in a red rectangle but not the roof of the building, which are both detected as objects in images in the step of saliency.

As the results shown as below, the scheme can detect objects from the image, classifying between aircraft and other objects in the background at the same time. However, when there are many relatively small airplanes existing in the same region of an image, the result shows not so ideal, because the airplanes are so small compared with the full image that they are considered as connected with each other after the step of flood-fill which is mentioned above.

9.4 Evaluation of The Model

Fig. 9.9. The result of the method using the remote sensing image.

To evaluate our algorithm, we use the Recall which is defined as below:

- $\text{Recall} = \text{Sensitivity} = \frac{TP}{TP+FN}$

Where TP (True Positives) indicates the number of aircraft successfully detected by the algorithm. FN (False Negative) indicates the number of aircraft that the algorithm did not recognize them as aircraft [75]. We use the Recall to show how many aircraft are successfully detected in all aircraft for each image. Through the Recall, we can evaluate the performance of the scheme in the task of aircraft detection.

As a test dataset, we prepared 40 remote sensing images which are large scale images collected from Google Earth, and all have multiple aircraft existing with a complicated background at the same time (Fig. 9.9). We compute the value of TP and FN according to the equation we mentioned above. Then we use the Recall as the evaluation of the performance of the scheme. We attained a Recall rate of 0.7217 on average for the all 40 images in the task of aircraft detection from the remote sensing images.

9.5 Discussion

We proposed an approach by combining the saliency map and CNN to solve the problem of aircraft detection. Firstly, we use several methods of the processing followed as transferring to gray-scale images, deleting the noise, transferring to the black-white images using threshold, erosion and dilation and flood-fill to extract the features of objects in the image. After the step of generating a saliency map, we use the algorithm of CHAIN APPROX TC89 L1 to find contours of detected objects. Secondly, we extract the regions including the all possible targets covering aircraft and other objects from the original images. Then we build a CNN model based on the LeNet-5 and use a dataset consisting of both positive and negative data to train the model which can classify classes between airplanes and backgrounds after the training. Then, we use the trained model to excite the inference about the possibility of the extracted regions appearing to be an airplane.

From the result of the evaluation, we can see that the scheme can detect multiple airplanes in a relatively large image which has a complicated background. The CNN model can tell the detected object is an airplane or other objects like buildings in the background. Considering we only process limited regions, the scheme can contribute to reduction of the computing cost compared with previous researches.

9.6 Conclusion

We use an approach called saliency to build a saliency map through preprocessing to extract the objects from the complex background which can be used to detect the possible targets. Different from the previous research which use the method of R-CNN, generating around 2000 region proposals. We build a CNN model and prepare a dataset covering different types of backgrounds which can be considered in the remote sensing images and images of aircraft and some parts of aircraft. We train the model and do a performance evaluation of the scheme based on recall, the result of which shows the scheme has robust adaptability to complicated backgrounds in remote sensing images.

For further research, the scheme can not only detect aircraft but also classify the different types of aircraft. What is more, the different types of objects existing in the background such as buildings, tracks and landing spaces can also be detected and classified.

Chapter 10 Trajectory Optimization for Driving Across Stochastic Traffic Flows

10.1 Introduction

In recent years, researchers have been putting considerable effort into the manipulation of autonomous vehicle driving. And the trajectory planning would allow it to find an optimized path to the desired goal with other constraints such as avoiding obstacles on the road and consuming as little fuel as possible. There is a lot of research on the vehicle driving along the road and avoiding obstacles based on trajectory optimization. However, research on an autonomous vehicle driving across a road with stochastic traffic flows is still insufficient. To improve the robustness of manipulation, more scenarios need considered.

In this research, the scenario of an autonomous vehicle driving across a road in two ways and four lanes is given, where different flow rates of traffic are adopted. And the goal of the autonomous vehicle is to drive across the road to the desired destination without collisions with other moving vehicles on the road, which are driving at a constant speed. Here, all vehicles are simplified to the shape of a circle, with the same radius. Besides, the optimized solution is considered to minimize the total consumption of fuel from the start point to the desired goal (Fig. 10.1).

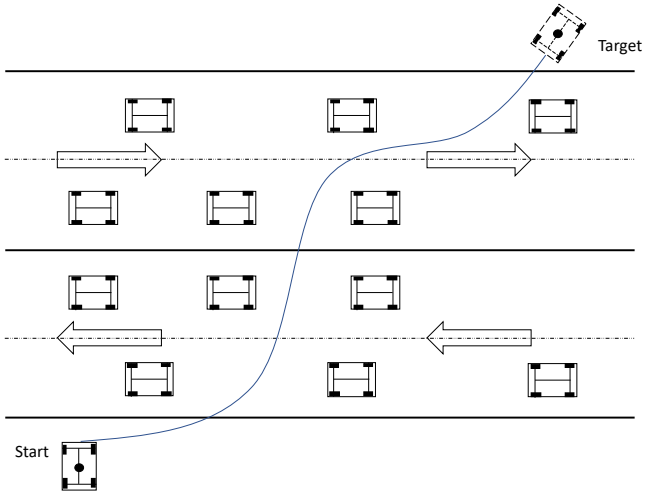


Fig. 10.1 The scenario of an autonomous vehicle driving across a traffic flow without a collision with other moving vehicles on the road, using minimum consumption of fuel, where the blue line represents the desired path for achieving the goal.

First, we analyze and clarify the dynamics of the autonomous vehicle and others. For moving vehicles on the road, we set the speed of them as constant, and initialize the locations of them randomly in specific lanes. Moreover, we keep a deterministic interval between every two vehicles driving on the same lane, as well as a deterministic interval between each lane of the road.

Then, we adopt the direct collocation method for achieving optimized trajectories based on the dynamics and constraints of the system such as no collisions. Furthermore, in direct collocation, both the input trajectory and the state trajectory are represented explicitly as piecewise polynomial functions, and the gradients of the objective and constraints are computed, thus finding the optimized solution.

At last, we evaluate the scheme using various traffic flow rates which are decided by the traffic density and the traffic speed, showing that by adopting this method, the optimized trajectory for an autonomous vehicle crossing the road could be achieved under different situations of traffic flow with great robustness.

10.2 Related Work

The research on trajectory optimization has a long history of robotic manipulation. Especially, in recent years, there are increasing research on the autonomous vehicle driving on a road and the quadrotor flying through a random terrain. And the trajectory optimization and path planning is playing a significant role in manipulating these systems. For instance, Zhou et al. [79] presented a kinodynamic path searching method for a quadrotor to find a safe, kinodynamic feasible, and minimum time initial trajectory in the discretized control space with various obstacles. Geisert et al. [80] discussed applications of the direct multiple shooting approach for solving a trajectory optimization problem. Moreover, Oleynikova et al. [81] proposed a motion planning method that uses trajectory optimization in continuous time to find collision-free paths between obstacles. Deits et al. [82] demonstrated an optimal trajectory planning method that pre-computes convex regions of safe space thus achieving faster pathfinding progress as well as ensuring the entire piecewise polynomial trajectory is free of collisions.

Furthermore, especially for trajectory optimization problems of autonomous vehicles, lots of methods have been proposed. For example, Bryan et al. [83] presented a real-time numerical method to compute the curvature polynomials of cubic order, which are ideal primitive trajectories for car-like robots. Xu et al. [84] proposed a method of discretizing the plan space and searching for the best trajectory based on a set of cost functions and an iterative optimization which accelerates the progress of optimization and improves the trajectory quality. They evaluated this method in the scenario of lane driving and circumventing a slower car. In addition, Dolgov [85] et al. demonstrated a practical path-planning algorithm that generates smooth paths for an autonomous vehicle operating in an unknown environment, using an adjusted A* search algorithm and numeric non-linear optimization.

10.3 Trajectory Optimization with Direct Collocation

10.3.1 The Scenario: An autonomous vehicle driving across stochatic traffic flows

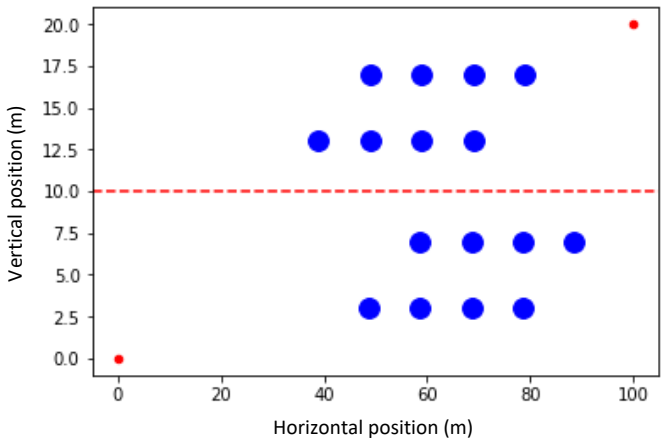


Fig. 10.2. A simulation of the optimization problem. The blue points show the initial positions of moving vehicles on the road. The red dash line shows a separation between lanes in different directions.

First, we demonstrate a scenario where all vehicles driving on a two-way road with a constant speed. And the vehicles in the outer lanes move faster than the ones in the inner lanes. The distance between every two vehicles in the same lane is deterministic, with a value of ten meters. The distance between each lane of the road is four meters, which are twice the length of vehicles. All vehicles in this research are simplified to circles with a radius of two meters. Moreover, the positions of them are randomly initialized, with a constant speed for the whole driving time. The autonomous vehicle has the same size as other vehicles on the road (Fig. 10.2).

Moreover, the start point of the autonomous vehicle is (0, 0), and the target point is (100, 20). The goal is to find an optimized path for an autonomous vehicle driving across the road without collisions with other vehicles, however with limited speed and torque. Furthermore, we are aimed to minimize the total fuel consumption during this progress, where we adopt the equation of torque as the cost.

10.3.2 The dynamics of the autonomous vehicle and moving vehicles on the road

We analyze the dynamics of the autonomous vehicle, where we consider the steering torque of the vehicle as vertical torque and horizontal torque separately. Then we adopt a second-order control differential equation described by the equations (10.1) (10.2).

$$\mathbf{x} = \begin{bmatrix} q_1 \\ q_2 \\ \dot{q_1} \\ \dot{q_2} \end{bmatrix} \tag{10.1}$$

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} \dot{q_1} \\ \dot{q_2} \\ \ddot{q_1} \\ \ddot{q_2} \end{bmatrix} \tag{10.2}$$

$$\text{subject to } \ddot{q_1} = \frac{u_1}{m} \quad \ddot{q_2} = \frac{u_2}{m}$$

Where q_1 is the horizontal position of the autonomous vehicle. q_2 is the vertical position of the autonomous vehicle. $\dot{q_1}$ represents the horizontal velocity and $\dot{q_2}$ represents the vertical velocity. $\ddot{q_1}$ shows the horizontal acceleration and $\ddot{q_2}$ shows the vertical acceleration.

Then, for the other moving vehicles on the road with a constant speed in this research, the dynamics of them can be represented by the following equations (10.3).

$$\boldsymbol{x_i} = \begin{bmatrix} q_{i1} \\ q_{i2} \\ \dot{q_{i1}} \\ \dot{q_{i2}} \end{bmatrix}$$

$$\dot{q_{i1}} = 1 + j \tag{10.3}$$

$$\dot{q_{i2}} = 0$$

$$q_{i1}(t) = (q_{i1}(0) + \dot{q_{i1}} \cdot t) \ \%\ loop$$

$$\text{subject to } j \in (1, 2)$$

Where j represents the j^{th} lane of the road for each direction. $\boldsymbol{x_i}$ represents the states of the i^{th} moving vehicle on the road. And *Loop* is used to control the trace of moving vehicles thus constructing a continuous traffic flow within the scope of the road simulated.

10.3.3 Nonconvex trajectory optimization based on direct collocation

We use the piecewise polynomial functions to represent explicitly both the input trajectory and the state trajectory. The piecewise polynomial function is a function defined by multiple sub-functions, each of which is applying to a certain interval of the main function's domain. Moreover, for the values of u(t) during a period, we adopt a first-order polynomial and for x(t), we adopt a cubic polynomial.

Furthermore, since we add several constraints such as no collisions to the system, it becomes a nonconvex trajectory optimization problem. Hence, in this research, we adopt the direct collocation method, where the necessary decision variables for optimization are those on the break points of a spline (Fig. 10.3). Considering the first order interpolation on u(t), if we can obtain u(tk) and u(tk+1), then any value between tk and tk+1 could be computed. And for x(t), from the dynamics of the autonomous vehicle $\dot{x} = f(x, u)$, we could compute $\dot{x_k} = f(x_k, u_k)$. As a result, all values of x(t) over the interval x(tk) and x(tk+1) can be completely defined by x(tk), $\dot{x_k}$, x(tk+1) and $\dot{x}_{K+1}$.

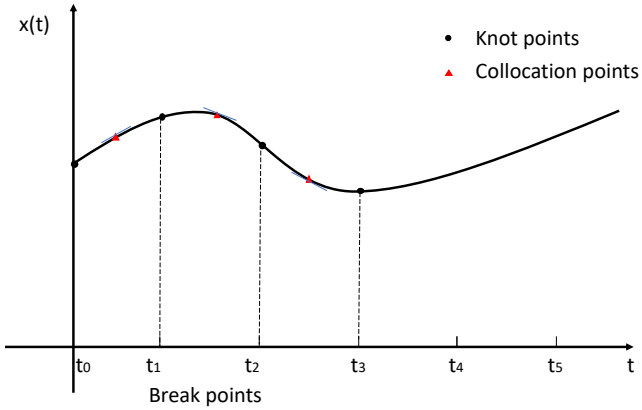


Fig. 10.3. The break points, knot points and collocation points in the direct collocation method.

Moreover, for this optimization problem, we add several constraints to the system thus preventing the autonomous vehicle driving along an unreasonable trace. These constraints are as follows:

- No collisions with other vehicles
- Complete the task with minimum torque consumption
- Maximum speed limit
- Maximum torque limit

In direct collocation, we add the constraints above to the derivative of the spline at the collocation points for optimization. In particular, we choose the midpoints of the spline as the collocation points. Moreover, we could obtain the dynamics of the autonomous vehicle at these collocation points by the following equations (10.4).

$$u(t_{ck}) = \frac{1}{2} \cdot (u(t_k) + u(t_{k+1}))$$

$$x(t_{ck}) = \tfrac{1}{2} \cdot \big(x(t_k) + x(t_{k+1})\big) + \tfrac{h}{8} \cdot \big(\dot{x}(t_k) - \dot{x}(t_{k+1})\big) \tag{10.4}$$

$$\dot{x}(t_{ck}) = -\frac{3}{2h} \cdot \big(x(t_k) - x(t_{k+1})\big) - \frac{1}{4} \cdot \big(\dot{x}(t_k) + \dot{x}(t_{k+1})\big)$$

$$\text{subject to } h = t_{k+1} - t_k$$

Where $u(t_k), x(t_k)$ represent the torques and the states of the autonomous vehicle at the break points separately. $u(t_{ck}), x(t_{ck})$ represent the torques and the states of the autonomous vehicle at the collocation points. h represents the time interval. Besides, $x(t_{ck})$ and $\dot{x}(t_{ck})$ are derived from the Taylor series about $x(t_k)$.

Then for the constraints at these collocation points, we could represent them using the following equations (10.5), which are further used for solving the optimized problem with direct collocation.

$$\begin{aligned} &\min_{u(t_c),x(t_c)} \textstyle\sum_{n=0}^{N-1} h_n \cdot u(n) \\ &\text{subject to } \dot{\boldsymbol{x}}(t_{c,n}) = f\big(\boldsymbol{x}(t_{c,n}),\, \boldsymbol{u}(t_{c,n})\big)\,, \\ &x\big(t_{k,0}\big) = x(0), x\big(t_{k,N}\big) = x(N), \\ &-s_{max} \le \sqrt{\dot{q_1}^2 + \dot{q_2}^2} \le s_{max}, \\ &-u_{max} \le \sqrt{{u_1}^2 + {u_2}^2} \le u_{max}, \\ &\frac{1}{2} \cdot l_{auto} + \frac{1}{2} \cdot l_{vehicle} \le d_{center} \end{aligned} \tag{10.5}$$

Where h_n represents the time interval, $x(0)$ is the start point, $x(N)$ is the target point, s_{max} is the maximum speed limit of the autonomous vehicle. u_{max} is the maximum torque limit of it, l_{auto} shows the diameter of the autonomous vehicle, $l_{vehicle}$ shows the diameter of other moving vehicles on the road, and d_{center} represents the distance between the center of a moving vehicle and the one of the autonomous vehicle.

After defining and clarifying the dynamics of the system as well as the constraints, we adopted a solver called Snopt provided by Drake [86] to compute the result of pathfinding based on direct collocation, using the optimization method and constraints demonstrated in (10.5). Here, the Snopt solver is based on the sparse SQP (sequential-quadratic programming) algorithm with limited memory quasi-Newton approximations to the Hessian of Lagrangian [87], considered to be effective when it comes to solving a nonlinear problem.

To find an optimized trajectory faster, we chose an initial guess of the start point and the end point near to the real ones, thus providing some intuitions for solving the problem. Then we just drew a direct line connecting the start point and the end point as the initial guess of states and set the initial guess of all torques as zero, defined by (10.6). Moreover, we adopt a time interval of 0.5 seconds for generating break points in the direct collocation method.

$$\begin{aligned} x_{guess}(t) &= \frac{t}{T} \cdot \begin{bmatrix} 100 \\ 20 \\ 0 \\ 0 \end{bmatrix} \\ u_{guess}(t) &= 0 \end{aligned} \tag{10.6}$$

Where T is the total time interval for autonomous vehicle driving from the start point to the target point.

10.4 Evaluation

For evaluating the scheme to solve the optimization problem in this scenario, we verified the performance of it under different environments with various traffic flow rates, which are controlled by the traffic density and the traffic speed. We evaluated the robustness of the scheme by finding a path to the goal with traffic flows of various traffic densities and traffic speeds.

10.4.1 Robustness to traffic flows with various traffic densities

In this research, we use the number of all vehicles on the same lane to represent the traffic density. For instance, have a traffic density of one means there is only one moving vehicle on each lane thus four vehicles on the road in total. Then, we adopted the Snopt solver above to compute and visualize the path for the autonomous vehicle driving across the traffic flow to the target point, with a comparison with the initial guess of the trajectory. Moreover, the change in the velocity and the torque of the autonomous vehicle during the total time interval is shown as well, using a normalized representation with respect to the maximum limit of them. In this research, we evaluated the performance of the scheme under traffic flows with a traffic density of one, two, four, and six. The corresponding results of these situations are shown above (Fig. 10.4).

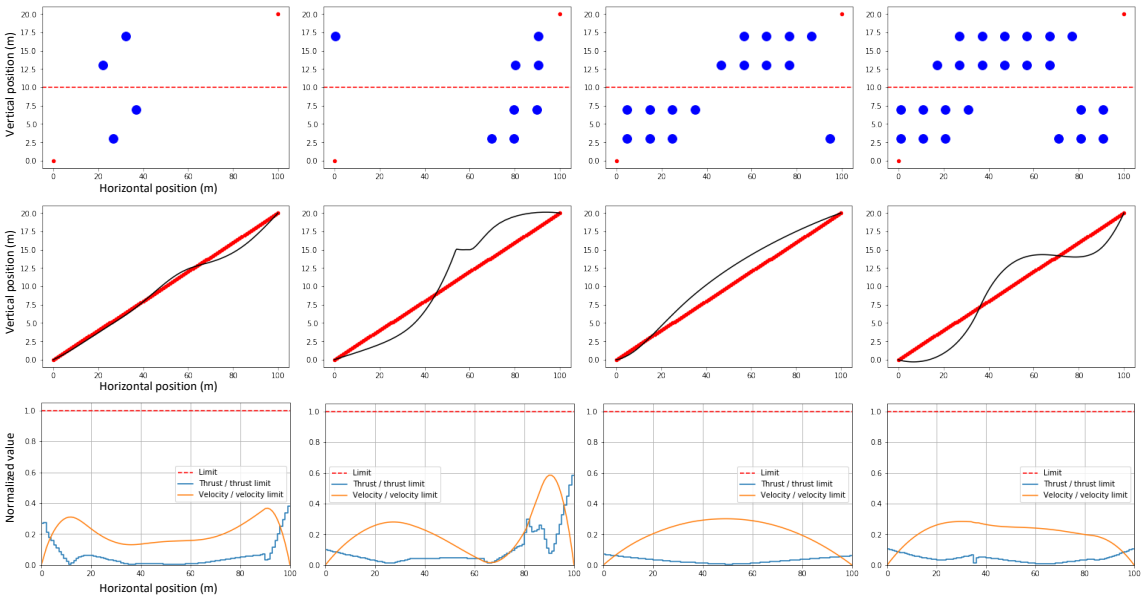


Fig. 10.4. The evaluation results of robustness to traffic flows with various traffic densities. From left to right: a traffic flow with a traffic density of one, two, four, and six, with corresponding total numbers of moving vehicles on the road as four, eight, 16, and 24. The constant driving speed of all vehicles on the outer lanes is set as 2 m/s and the speed of vehicles on the inner lanes is set as 1 m/s. The initial positions of all vehicles on the road are randomly generated.

10.4.1Robustness to traffic flows with various traffic speeds

We adopted various driving speeds for moving vehicles on the road to evaluate the performance of the scheme for finding an optimized path from the start point to the target point without collisions with them as well as using minimum fuel consumption where we consider the torque as the only source of the fuel consumption. And the speed unit here is meter per second. We consider a scenario where two vehicles drive on each lane of the road (a traffic density of two). Then, we adopted a driving speed of 3 m/s (4m/s for the outer), 4m/s (5m/s for the outer), 6m/s (7m/s for the outer), and 8m/s (9m/s for the

outer) separately for these moving vehicles and computed the corresponding optimized trajectories for achieving the goal. The results of pathfinding with various traffic speeds are shown above. (Fig. 10.5)

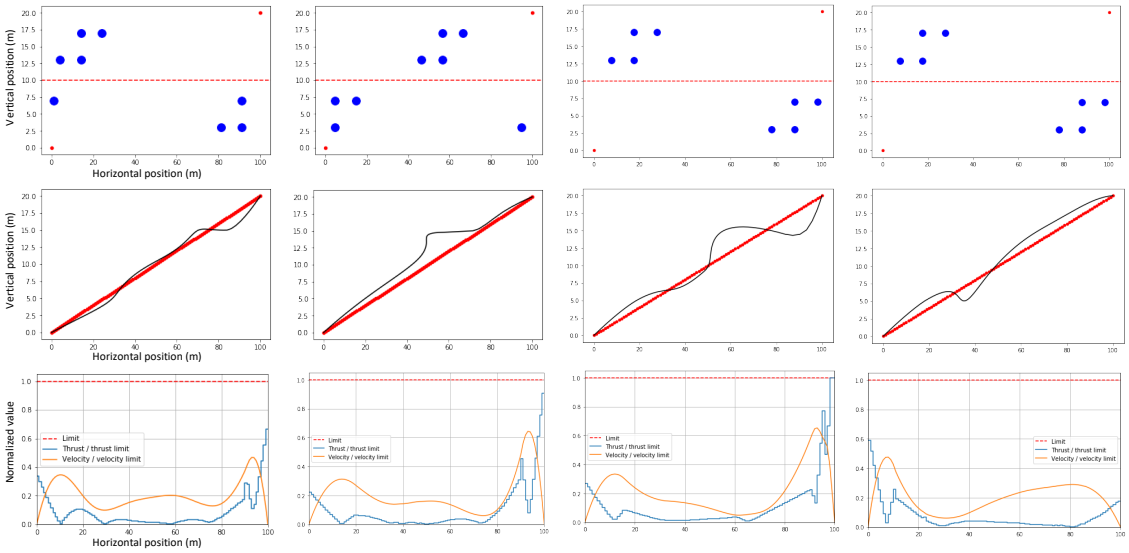


Fig. 10.5. The evaluation results of robustness to traffic flows with various traffic speeds. From left to right: a traffic flow with a traffic speed of 3 m/s, 4m/s, 6m/s, and 8m/s (for vehicles on the outer lanes) and 4 m/s, 5m/s, 7m/s, and 9m/s (for vehicles on the inner lanes). Here, for each lane, there are two moving vehicles (a traffic density of two). The initial positions of all vehicles on the road are randomly generated.

10.4.2 Discussion on the dynamics of the autonomous vehicle

We visualized the change in the autonomous vehicle's dynamics, covering the horizontal position with respect to the horizontal velocity and the vertical position with respect to the vertical velocity, when different traffic flow rates adopted (Fig. 10.6). From the graphs, we can see with the increase of the traffic density and the traffic speed, it shows more likely for the autonomous vehicle to take sharp steering. Moreover, when the increase in these two factors arrives at a great extent, it has been showing no feasible solution for the autonomous vehicle driving across the road without collisions. Regarding this, one explanation could be the interval between every two moving vehicles on the road, which is set as 10 meters in this research, is too small for the autonomous vehicle to drive through compared with the traffic speed and the size of the vehicle itself.

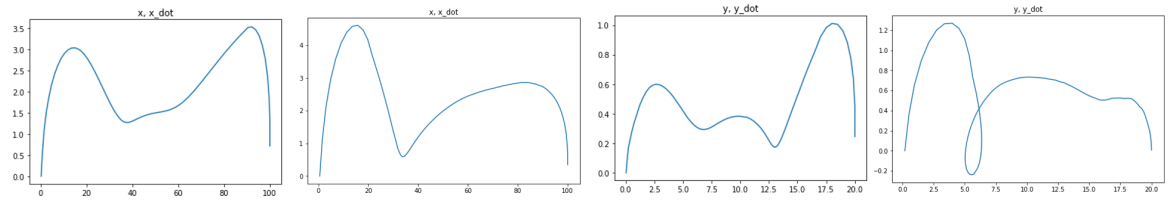


Fig. 10.6. The dynamics of the autonomous vehicle with different traffic flows. From left to right: the horizontal dynamics when the traffic density is one and the traffic speed is 1m/s (2m/s), the horizontal dynamics when the traffic density is two and the traffic speed is 8m/s (9m/s), the vertical dynamics when the traffic density is one and the traffic speed is 1m/s (2m/s), and the vertical dynamics when the traffic density is two and the traffic speed is 8m/s (9m/s).

10.5 Conclusion

In this research, we present a scenario where vehicles drive on a two-way road consisting of four lanes, with a constant speed. Then we analyze and clarify the optimization problem of an autonomous vehicle driving across traffic flows with various traffic densities and traffic speeds, from the start point at the one side of the road to the target point at the other side of the road. We add several constraints to the systems including avoiding collisions, maximum speed limit, and maximum torque limit. We are supposed to achieve an optimized path for the autonomous vehicle arriving at the goal with minimum fuel consumption.

We analyze the dynamics of the systems and give out a solution based on the direct collocation method, where all the constraints are satisfied at the collocation points. We compute the optimized trajectory using the Snopt solver provided by Drake. At last, for evaluating the scheme, we adopt various traffic flow rates to compute an optimized trajectory for this problem, by adjusting the traffic density and the traffic speed separately. And the result of it shows that we could solve the demonstrated trajectory optimization problem under various situations of the traffic flow rate, showing great robustness of the scheme to the stochastic traffic flow. For future work, other approaches such as RRT* (Rapidly-exploring random trees star) [88] and Hybrid A* [89] are considered to be adopted for solving the trajectory optimization problem presented in this research. A comparison between the performance of these methods could give more insights into trajectory optimization for autonomous vehicles.

Part V Conclusion

Chapter 10 Conclusion

In this thesis, we presented DL-based intrusion detection approaches, especially focusing on the extension of adaptability and liability of such systems. First, several visual analytics were discussed for analyzing and representing features hidden in network traffic. The position-based method and the channel-based method were designed to combine with the CNNs model while maintaining comprehensive protocol-wise information. In contrast, the raw data-based method provides a solution to the time-consuming feature design and feature map generation existing in the other two methods. The types of intrusion and malware are diverse that we conducted research on detection for various anomalous network events. For that reason, we also explored the classification and detection of malicious domain names based on lexical analysis combined with VAE and network flow-based analysis combined with web crawling respectively.

To improve the adaptability of DL-based IDS, an efficient approach proves to be Federated Learning, which makes full use of edge computing where DL models are trained individually offline based on local datasets. After local training, only the training results are shared with the community to help improve each other's performance. The privacy concerns once existing in a single DL-based system, especially the DL-based IDSs, is alleviated due to the distributed architecture of the systems. The Segmented-Federated Learning proposed during our research is the first DL-based approach to address the adaptability-wise issues of IDSs. Despite the extensive application of FL on NIDS, a network's traffic data does not always fit into the single global model of FL; some networks have similarities with each other but other networks do not. Edge agents collaborate with each other through model segmentation and periodical evaluation.

Though the aforementioned methods show great performance in intrusion detection and anomaly detection. It is essential to ensure the security and liability of DL-based detection systems themselves, where misguided decision-making of an agent could bring about false alarms of a perspective intrusion and be compromised to allow communications with a malicious user. As a result, we studied the security and privacy of the FL framework through mounting adversarial attacks and data poisoning attacks, discussing the influence of such attacks on the systems. These attacks would either exacerbate the detection performance or disclosing personal information from shared model parameters. Moreover, we introduced blockchain-based defense strategies against such attacks, detecting and removing malicious updates from edge agents, thus stabilizing the detection performance and safeguarding users' personal local training data from disclosing. Finally, the additional research on object detection and trajectory optimization contribute to building the theoretical and mathematical fundaments of solving problems of optimization and modeling.

References

[1] J P. Anderson, Computer Security Threat Monitoring and Surveillance, Tech- nical Report, Computer Security Division of the Information Technology Laboratory, National Institute of Standards and Technology, 1980.

[2] U. Fiore, et al., Network anomaly detection with the restricted Boltzmann machine, Neurocomputing, 2013.

[3] Hsu, C.-W., Lin, C.-J., A comparison of methods for multiclass support vector machines. IEEE Trans. Neural Netw. 13 (2), 415–425., 2002.

[4] Mehdi Mohammadi, Ala Al-Fuqaha, Sameh Sorour, Mohsen Guizani, Deep Learning for IoT Big Data and Streaming Analytics: A Survey, IEEE COMMUNICATIONS SURVEYS & TUTORIALS, 2018.

[5] Asmaa Elsaeidy, Kumudu S. Munasinghe, Dharmendra Sharma, Abbas Jamalipour, Intrusion detection in smart cities using Restricted Boltzmann Machines, Journal of Network and Computer Applications 135 76–83, 2019.

[6] Kazumasa Y amauchi, Junpei Kawamoto, Y oshiaki Hori, Kouichi Sakurai, Evaluation of Machine Learning Techniques for C&C Traffic Classification, Information Processing Society of Japan Vol.56 No.9 1745–1753, 2015.

[7] Yuji Tokozume, Tatsuya Harada, LEARNING ENVIRONMENTAL SOUNDS WITH END-TO-END CONVOLUTIONAL NEURAL NETWORK, IEEE International Conference on Acoustics, Speech and Signal Processing, 2017.

[8] Yann LECun, Patrick Haffner, Leon Bottou, and Yoshua Bengio, Object Recognition with Gradient-Based Learning, Shape, Contour and Grouping in Computer Vision, p.319, 1999.

[9] F. Palmieri, U. Fiore, Network anomaly detection through nonlinear analysis, Comput. Secur. 29 (7), 737–755, 2010.

[10] Hsu Chih-Wei, Lin Chih-Jen, A comparison of methods for multiclass support vector machines, IEEE Trans. Neural Netw. 13 (2), 415–425, 2002.

[11] Mehdi Mohammadi, Ala Al-Fuqaha, Sameh Sorour, Mohsen Guizani, Deep Learning for IoT Big Data and Streaming Analytics: A Survey, IEEE COMMUNICATIONS SURVEYS & TUTORIALS, 2018.

[12] Eric Krokos, Alexander Rowden, Kirsten Whitley, and Amitabh Warshney, “Visual Analytics for Root DNS Data”, IEEE, 2018.

[13] Simonyan, K., Zisserman, A., Very deep convolutional networks for large-scale image recognition, ICLR 2015.

[14] Upeka K. Premaratne, Jagath Samarabandu, Tarlochan S. Sidhu, Robert Beresh, and Jian-Cheng Tan. 2010. An intrusion detection system for IEC61850 automated substations. IEEE Transactions on Power Delivery 25, 4 (October 2010), 2376–2383.

[15] Michele Di Santo, Alfredo Vaccaro, Domenico Villacci, and Eugenio Zimeo. 2004. A distributed architecture for online power systems security analysis. IEEE Transactions on Industrial Electronics 51, 6 (December 2004), 1238–1248.

[16] Gonzalo Marín, Pedro Casas, Germán Capdehourat. DeepMAL - Deep Learning Models for Malware Traffic Detection and Classification. arXiv:2003.04079. 2020.

[17] Félix Iglesias and Tanja Zseby. Analysis of network traffic features for anomaly detection. Learn 101, 59–84. https://doi.org/10.1007/s10994-014-5473-9. 2015.

[18] M. H. Bhuyan, D. K. Bhattacharyya and J. K. Kalita. Network Anomaly Detection: Methods, Systems and Tools. IEEE Communications Surveys & Tutorials, vol. 16, no. 1, pp. 303-336, doi: 10.1109/SURV.2013.052213.00046. 2014.

[19] Kateryna Chumachenko. Machine Learning Methods for Malware Detection and Classification. 2017.

[20] "tcpdump and libpcap latest release". The Tcpdump Group. Retrieved 2020-08-10.

[21] J. Nazario and T. Holz. As the net churns: Fast-flux botnet observations, International Conference on Malicious and Unwanted Software, pp. 24–31, 2008.

[22] R. Villamarn-Salomn and J.C. Brustoloni. Identifying botnets using anomaly detection techniques applied to DNS traffic, Consumer Communications and Networking Conference, pp. 476–481, 2008.

[23] N. Chatzis and R. Popescu-Zeletin, Flow Level Data Mining of DNS Query Streams for Email Worm Detection. CISIS, pp. 186—194, 2008.

[24] J. Erman, M. Arlitt, and A. Mahanti, Traffic classification using clustering algorithms. MineNet ’06, pp. 281–286, 2006.

[25] Eric Krokos, Alexander Rowden, Kirsten Whitley, and Amitabh Warshney, “Visual Analytics for Root DNS Data” IEEE, 2018.

[26] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas. Ddos in the iot: Mirai and other botnets. Computer, vol. 50, no. 7, pp. 80–84. 2017.
[27] D. Atienza, Á. Herrero, and E. Corchado. Neural analysis of http traffic for web attack detection. Computational Intelligence in Security for Information Systems Conference. Springer, pp. 201–212. 2015.
[28] Ikinci, Ali & Holz, Thorsten & Freiling, Felix. Monkey-Spider: Detecting Malicious Websites with Low-Interaction Honeyclients.. 407-421. 2008.
[29] N. Eric, D. Ahmad, G. Andrew, M. Ericsson, M. Vineet, P. Vivin, R. John, S. Jana, T. Amanda, and S. Paulo. Darknet and deepnet mining for proactive cybersecurity threat intelligence. 7-12. 10.1109/ISI.2016. 7745435. 2016.
[30] A. Idan, M. John, H. William, X. Zhi, M. Yinnon, and W. Yigal. Machine Learning in Cyber-Security - Problems, Challenges and Data Sets. 2019.
[31] M. Mohammad, R. Muhammad, H. L. Arash, S. Natalia, and G. Ali. Detecting Malicious URLs Using Lexical Analysis. 9955. 467-482. 10.1007/978-3-319-46298-1_30. 2016.
[32] H. B. McMahan, E. Moore, Daniel Ramage, and B. A. Arcas, Federated Learning of Deep Networks using Model Averaging, arXiv preprint, arXiv:1602.05629v1. 2016.
[33] Abeshu and N. Chilamkurti, Deep learning: the frontier for distributed attack detection in fog-to-things computing, IEEE Communications Magazine, vol. 56, no. 2, pp. 169–175, 2018
[34] S. Wang et al., "Adaptive Federated Learning in Resource Constrained Edge Computing Systems," in IEEE Journal on Selected Areas in Communications, vol. 37, no. 6, pp. 1205-1221, June 2019, doi: 10.1109/JSAC.2019.2904348.
[35] "LAN-Security Monitoring Project", www.lan-security.net, Accessed 5th Oct. 2020
[36] Upeka K. Premaratne, Jagath Samarabandu, Tarlochan S. Sidhu, Robert Beresh, and Jian-Cheng Tan. 2010. An intrusion detection system for IEC61850 automated substations. IEEE Transactions on Power Delivery 25, 4 (October 2010), 2376–2383.
[37] Michele Di Santo, Alfredo Vaccaro, Domenico Villacci, and Eugenio Zimeo. 2004. A distributed architecture for online power systems security analysis. IEEE Transactions on Industrial Electronics 51, 6 (December 2004), 1238–1248.
[38] Andrea Carcano, Alessio Coletta, Michele Guglielmi, Marcelo Masera, Igor Nai Fovino, and Alberto Trombetta. 2011. A multidimensional critical state analysis for detecting intrusions in SCADA sys- tems. IEEE Transactions on Industrial Informatics 7, 2 (May 2011), 179–186.
[39] Sooyeon Shin, Taekyoung Kwon, Gil-Yong Jo, Youngman Park, and H. Rhy. 2010. An experimental study of hierarchical intrusion detection for wireless industrial sensor networks. IEEE Transactions on Indus- trial Informatics 6, 4 (November 2010), 744–757.
[40] Chi-Ho Tsang and Sam Kwong. 2005. Multi-agent intrusion detection system in industrial network using ant colony clustering approach and unsupervised feature extraction. In *Proceedings of the International Conference on Industrial Technology*. Hong Kong, China, 51–56.
[41] Nilamadhab Mishra, Sarojananda Mishra, Support Vector Machine Used in Network Intrusion Detection, National Workshop on Internet of Things (IoT), 2018.
[42] T. Omrani, A. Dallali, B. C. Rhaimi and J. Fattahi, Fusion of ANN and SVM classifiers for network attack detection, 18th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering, 2017
[43] M.A. Salama, H.F. Eid, R.A. Ramadan, A. Darwish, A.E. Hassanien, Hybrid intelligent intrusion detection scheme, Soft Computing in Industrial Applications, pp.293–303, 2011.
[44] J. Yang, J. Deng, S. Li, Y. Hao, Improved traffic detection with support vector machine based on restricted boltzmann machine, Soft Computing 21(11) (2017) 3101–3112, 2017
[45] P.H. Duy, N.N. Diep, Intrusion detection using deep neural network, Southeast Asian Journal of Sciences 5 (2) (2017) 111–125, 2017
[46] J. Saxe, K. Berlin, Deep neural network based malware detection using two dimensional binary program features, Proceedings of the 10th International Conference on Malicious and Unwanted Software (MALWARE), 2015
[47] M. Yousefi-Azar, V. Varadharajan, L. Hamey, U. Tupakula, Autoencoder-based feature learning for cyber security applications, Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN), 2017
[48] Briland Hitaj, Giuseppe Ateniese, Fernando Perez-Cruz, Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning, Session C3: Machine Learning Privacy, CCS'17, 2017
[49] J. Schneible and A. Lu, "Anomaly detection on the edge," MILCOM 2017 - 2017 IEEE Military Communications Conference (MILCOM), Baltimore, MD, 2017, pp. 678-682.
[50] Ying Zhao, Junjun Chen, Di Wu, Jian Teng, and Shui Yu. 2019. Multi-Task Network Anomaly Detection using Federated Learning. In Proceedings of the Tenth International Symposium on Information and

Communication Technology (SoICT 2019). Association for Computing Machinery, New York, NY, USA, 273–279. DOI:https://doi.org/10.1145/3368926.3369705
[51] Harshit Daga, Patrick K. Nicholson, Ada Gavrilovska, and Diego Lugones. 2019. Cartel: A System for Collaborative Transfer Learning at the Edge. In Proceedings of the ACM Symposium on Cloud Computing (SoCC '19). Association for Computing Machinery, New York, NY, USA, 25–37.
[52] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In Proceedings of the 4th International Conference on Information Systems Security and Privacy, pages 108–116, 2018
[53] Z. Yu, J. Hu, G. Min, Z. Zhao, W. Miao and M. S. Hossain, "Mobility-Aware Proactive Edge Caching for Connected Vehicles Using Federated Learning," in IEEE Transactions on Intelligent Transportation Systems, doi: 10.1109/TITS.2020.3017474. 2020
[54] Yuwei Sun, Hideya Ochiai, Hiroshi Esaki. Visual Analytics for Anomaly Classification in LAN Based on Deep Convolutional Neural Network. IEEE International Conference on Informatics, Electronics and Vision, 2020
[55] Y. Qiang & L. Yang & C. Yong & K. Yan & C. Tianjian & Y. Han. Federated Learning. Synthesis Lectures on Artificial Intelligence and Machine Learning. 2019.
[56] Li, Yuzheng & Chen, Chuan & Liu, Nan & Huang, Huawei & Zheng, Zibin & Yan, Qiang. A Blockchain-based Decentralized Federated Learning Framework with Committee Consensus. 2020.
[57] Kim, Hyesung & Park, Jihong & Bennis, Mehdi & Kim, Seong-Lyun. Blockchained On-Device Federated Learning. IEEE Communications Letters. 10.1109/LCOMM.2019.2921755. 2019.
[58] Weng, Jiasi & Weng, Jian & Zhang, Jilian & Li, Ming & Zhang, Yue & Luo, Weiqi. DeepChain: Auditable and Privacy-Preserving Deep Learning with Blockchain-based Incentive. 2019.
[59] P. Davy & R. Vera & T. Ilias & S. Jan & J. Wouter & I. Elisabeth. Chained Anomaly Detection Models for Federated Learning: An Intrusion Detection Case Study. Applied Sciences. 8. 2663. 10.3390/app8122663. 2018.
[60] LeCun, Yann and Cortes, Corinna. "MNIST handwritten digit database". 2010.
[61] Han Xiao, Kashif Rasul, Roland Vollgraf, “Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms,” 2017, arXiv:1708.07747.
[62] B. Hitaj, G. Ateniese, and F. Perez-Cruz, “Deep Models under the GAN: Information leakage from collaborative deep learning,” 2017, doi: 10.1145/ 3133956.3134012.
[63] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," 2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton), Monticello, IL, 2015, pp. 909-910, doi: 10.1109/ALLERTON.2015.7447103.
[64] J. Zhang, B. Chen, X. Cheng, H. T. T. Binh, and S. Yu, “PoisonGAN: Generative Poisoning Attacks against Federated Learning in Edge Computing Systems,” IEEE Internet Things Journal, 2020, doi: 10.1109/jiot.2020.3023126.
[65] D. Cao, S. Chang, Z. Lin, G. Liu and D. Sun, "Understanding Distributed Poisoning Attack in Federated Learning," 2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS), Tianjin, China, 2019, pp. 233-239, doi: 10.1109/ICPADS47876.2019.00042.
[66] A. Piplai, S. S. L. Chukkapalli, and A. Joshi, “NAttack! Adversarial Attacks to bypass a GAN based classifier trained to detect Network intrusion,” 2020.
[67] H. Xu et al., “Adversarial Attacks and Defenses in Images, Graphs and Text: A Review,” International Journal of Automation and Computing. 2020, doi: 10.1007/s11633-019-1211-x.
[68] M. Fredrikson, S. Jha, and T. Ristenpart, “Model inversion attacks that exploit confidence information and basic countermeasures,” in Proc. ACM Comput. Commun. Secur., 2015, pp. 1322–1333.
[69] Y. Zhang, C. Xu, H. Li, K. Yang, J. Zhou, and X. Lin, “HealthDep: An efficient and secure deduplication scheme for cloud-assisted ehealth systems,” IEEE Trans. Ind. Informat., vol. 14, no. 9, pp. 4101–4112, Sep. 2018.
[70] Y. Lu, X. Huang, Y. Dai, S. Maharjan and Y. Zhang, "Blockchain and Federated Learning for Privacy-Preserved Data Sharing in Industrial IoT," in IEEE Transactions on Industrial Informatics, vol. 16, no. 6, pp. 4177-4186, June 2020.
[71] N. Ammour, H. Alhichri, Y. Bazi, B. Benjdira, N. Alajlan, and M. Zuair, “Deep Learning Approach for Car Detection in UAV Imagery,” Remote Sensing, vol. 9, p. 312, mar 2017.
[72] S.Ren,K.He,R.Girshick,andJ.Sun,“FasterR-CNN:TowardsReal- Time Object Detection with,” IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, 2017.
[73] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich fea- ture hierarchies for accurate object detection and semantic segmentation. arXiv preprint arXiv:1311.2524, 2013.

[74] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.
[75] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. arXiv:1310.1531, 2013.
[76] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, pp. 779–788, 2016.
[77] lan Goodfellow, Yoshua Bengio, Aaron Courvile, Deep Learning(Adaptive Computation and Machine Learning), Francis Bach, The MIT Press, 2016.
[78] Uijlings, J.R.R., van de Sande, K.E.A., Gevers, T. et al. Int J Comput Vis (2013) 104: 154. https://doi.org/10.1007/s11263-013-0620-5
[79] Boyu Zhou, Fei Gao, Luqi Wang, Chuhao Liu and Shaojie Shen. Robust and Efficient Quadrotor Trajectory Generation for Fast Autonomous Flight. IEEE ROBOTICS AND AUTOMATION LETTERS, 2019.
[80] Mathieu Geisert, Nicolas Mansard. Trajectory Generation for Quadrotor Based Systems using Numerical Optimal Control. International Conference on Robotics and Automation (ICRA 2016), IEEE, May 2016, Stockholm, Sweden. pp. 2958-2964. ffhal-01213392v2f.
[81] Oleynikova, Helen & Burri, Michael & Taylor, Zachary & Nieto, Juan & Siegwart, Roland & Galceran, Enric. (2016). Continuous-Time Trajectory Optimization for Online UAV Replanning. 10.1109/IROS.2016.7759784.
[82] Deits, Robin & Tedrake, Russell. (2015). Efficient mixed-integer planning for UAVs in cluttered environments. 2015. 42-49. 10.1109/ICRA.2015.7138978.
[83] Nagy, Bryan & Kelly, Alonzo. (2001). TRAJECTORY GENERATION FOR CAR-LIKE ROBOTS USING CUBIC CURVATURE POLYNOMIALS.
[84] Xu, Wenda & Wei, Junqing & Dolan, John & Zhao, Huijing & Zha, Hongbin. (2012). A Real-Time Motion Planner with Trajectory Optimization for Autonomous Vehicles. Proceedings - IEEE International Conference on Robotics and Automation. 10.1109/ICRA.2012.6225063.
[85] Dolgov, Dmitri & Thrun, Sebastian & Montemerlo, Michael & Diebel, James. (2008). Practical Search Techniques in Path Planning for Autonomous Driving. AAAI Workshop - Technical Report.
[86] Russ Tedrake and the Drake Development Team. Drake: Model-based design and verification for robotics. 2019. https://drake.mit.edu.
[87] P. E. Gill, W. Murray and M. A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. SIAM Review 47 (2005), 99-131.
[88] Jeon, Jeong hwan & Cowlagi, Raghvendra & Peters, Steven & Karaman, Sertac & Frazzoli, Emilio & Tsiotras, Panagiotis & Iagnemma, Karl. (2013). Optimal motion planning with the half-car dynamical model for autonomous high-speed driving. Proceedings of the American Control Conference. 188-193. 10.1109/ACC.2013.6579835.
[89] Montemerlo, Michael & Becker, Jan & Bhat, Suhrid & Dahlkamp, Hendrik & Dolgov, Dmitri & Ettinger, Scott & Haehnel, Dirk & Hilden, Tim & Hoffmann, Gabriel & Huhnke, Burkhard & Johnston, Doug & Klumpp, Stefan & Langer, Dirk & Levandowski, Anthony & Levinson, Jesse & Marcil, Julien & Orenstein, David & Paefgen, Johannes & Penny, Isaac & Thrun, Sebastian. (2008). Junior: The Stanford Entry in the Urban Challenge. Journal of Field Robotics. 25. 569 - 597. 10.1002/rob.20258.
[90] lan Goodfellow, Yoshua Bengio, Aaron Courvile, Deep Learning (Adaptive Computation and Machine Learning), Francis Bach, The MIT Press, 2016.
[91] Eric Krokos, Alexander Rowden, Kirsten Whitley, and Amitabh Warshney, "Visual Analytics for Root DNS Data" IEEE, 2018.
[92] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NIPS'12 Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1, Pages 1097-1105, 2012.
[93] Y.LeCun,K.Kavukcuoglu,andC.Farabet.Convolutionalnetworksand applications in vision. In Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on. IEEE, 2010.
[94] Y.LeCun, B.Boser, J.S.Denker, D.Henderson, R.E.Howard, W.Hubba rd and L.D.Jackel.Backpropagation Applied to Handwritten Zip Code Recognition, Neural Computation, p.541-551,1989.
[95] Anna L. Buczak, Member, IEEE, and Erhan Guven, A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection, IEEE COMMUNICATIONS SURVEYS & TUTORIALS, VOL. 18, NO. 2, SECOND QUARTER 2016.
[96] Yann LECun, Patrick Haffner, Leon Bottou, and Yoshua Bengio, Object Recognition with Gradient-Based Learning, Shape, Contour and Grouping in Computer Vision, p.319, 1999.

[97] Kazumasa Y amauchi, Junpei Kawamoto, Y oshiaki Hori, Kouichi Sakurai, Evaluation of Machine Learning Techniques for C&C Traffic Classification, Information Processing Society of Japan Vol.56 No.9 1745–1753, 2015.
[98] F. Palmieri, U. Fiore, A nonlinear, recurrence-based approach to traffic classification, Comput. Networks 53 (6), 761–773, 2009.
[99] S. Lau, The spinning cube of potential doom, Communications of the ACM, 47(6), 25–26, 2004.
[100] J. Erman, M. Arlitt, and A. Mahanti, Traffic classification using clustering algorithms, MineNet'06, pp. 281–286, 2006.
[101] Samaneh Mahdavifar, Ali A. Ghorbani, Application of deep learning to cybersecurity: A survey, Neurocomputing 347, 149–176, 2019.
[102] Eric Ke Wang,Yunming Ye, Xiaofei Xu, S.M.Yiu, L.C.K.Hui, K.P.Chow, Security Issues and Challenges for Cyber Physical System, IEEE/ACM International Conference on Green Computing and Communications & IEEE/ACM International Conference on Cyber, Physical and Social Computing, 2010.
[103] Sommer, Robin, Paxson, Vern, Outside the Closed World: On Using Machine Learning for Network Intrusion Detection, IEEE Symposium on Security and Privacy, 305-316. 10.1109/SP.2010.25., 2010.
[104] Yisroel Mirsky, Tomer Doitshman, Yuval Elovici, Asaf Shabtai, Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection, Network and Distributed Systems Security Symposium (NDSS), 2018.
[105] P.H. Duy, N.N. Diep, Intrusion detection using deep neural network, Southeast Asian Journal of Sciences 5 (2) (2017) 111–125, 2017.
[106] Roni Mateless and Michael Segal. Approximate String Matching for DNS Anomaly Detection, arXiv:1905.09455, 2019.
[107] Renée Burton, Characterizing Certain DNS DDoS Attacks, arXiv:1905.09958v1, 2019.
[108] Diederik P Kingma, Max Welling, Auto-Encoding Variational Bayes, arXiv:1312.6114, 2014.
[109] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A re- view and new perspectives. 2013.
[110] Karol Gregor, Andriy Mnih, and Daan Wierstra. Deep autoregressive networks. arXiv preprint arXiv:1310.8499, 2013.
[111] Yehuda Afek, Anat Bremler-Barr, Edith Cohen, Shir Landau Feibish, and Michal Shagam. Efficient Distinct Heavy Hitters for DNS DDoS Attack Detection, arXiv:1612.02636v1, 2016.
[112] Vojtech Havlicek et al., Supervised learning with quantum enhanced feature spaces. arXiv:1804.11326v2, 2018.
[113] Chi-Ho Tsang and Sam Kwong. 2005. Multi-agent intrusion detection system in industrial network using ant colony clustering approach and unsupervised feature extraction. In Proceedings of the International Conference on Industrial Technology. Hong Kong, China, 51–56.
[114] Nilamadhab Mishra, Sarojananda Mishra, Support Vector Machine Used in Network Intrusion Detection, National Workshop on Internet of Things (IoT), 2018
[115] T. Omrani, A. Dallali, B. C. Rhaimi and J. Fattahi, Fusion of ANN and SVM classifiers for network attack detection, 18th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering, 2017
[116] Samaneh Mahdavifar, Ali A. Ghorbani, Application of deep learning to cybersecurity: A survey, Neurocomputing 347, 149–176. 2019.
[117] L. Deri, M. Martinelli, T. Bujlow and A. Cardigliano. nDPI: Open-source high-speed deep packet inspection. International Wireless Communications and Mobile Computing Conference. 2014
[118] A. Elsaeidy, K. S. Munasinghe, D. Sharma, and Abbas Jamalipour, Intrusion detection in smart cities using Restricted Boltzmann Machines, Journal of Network and Computer Applications 135 76–83, 2019
[119] H. Briland & A. Giuseppe & P. Fernando. Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning. 2017.
[120] H. Muhammad & S. Khaled & D. Ernesto & S. Davor. Towards Blockchain-Based Reputation-Aware Federated Learning. 2020.
[121] Hu, Weiwei & Tan, Ying. Generating Adversarial Malware Examples for Black-Box Attacks Based on GAN. 2017
[122] M. Abadi et al., "Deep learning with differential privacy," Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (ACM CCS), pp. 308-318, 2016.
[123] Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," IEEE Trans. Inf. Forensics Sec., vol. 13, no. 5, pp. 1333–1345, May 2018.
[124] Yann LECun, Patrick Haffner, Leon Bottou, and Yoshua Bengio, Object Recognition with Gradient-Based Learning, Shape, Contour and Grouping in Computer Vision, p.319, 1999.

[125] Guoxiong Hu, Zhong Yang, Jiaming Han, Li Huang, Jun Gong and Naxixue Xiong, Aircraft detection in remote sensing images based on saliency and convolution neural network. EURASIP Journal on Wireless Communications and Networking, 2018.
[126] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard and L. D. Jackel. Backpropagation Applied to Handwritten Zip Code Recognition. Neural Computation, p.541-551,1989.
[127] Jiawei Zuo , Guangluan Xu, Kun Fu, Xian Sun, and Hao Sun, Aircraft Type Recognition Based on Segmentation With Deep Convolutional Neural Networks, IEEE GEOSCIENCE AND REMOTE SENSING LETTERS, VOL. 15, NO. 2, FEBRUARY 2018.
[128] Yuhang Zhang, Hao Sun, Jiawei Zuo, Hongqi Wang, Guangluan Xu and Xian Sun, Aircraft Type Recognition in Remote Sensing Images Based on Feature Learning with Conditional Generative Adversarial Networks, Remote Sens. 2018.
[129] Yann LeCnn et al. Gradient-Based Learning Applied to Document Recognition, PROC. OF THE IEEE, 1998
[130] Gonzalez Bautista, David & Pérez, Joshué & Milanes, Vicente & Nashashibi, Fawzi. (2015). A Review of Motion Planning Techniques for Automated Vehicles. IEEE Transactions on Intelligent Transportation Systems. 1-11. 10.1109/TITS.2015.2498841.
[131] Xu, Wenda & Pan, Jia & Wei, Junqing & Dolan, John. (2014). Motion planning under uncertainty for on-road autonomous driving. Proceedings - IEEE International Conference on Robotics and Automation. 2507-2512. 10.1109/ICRA.2014.6907209.
[132] Petrov, Plamen & Nashashibi, Fawzi. (2014). Modeling and Nonlinear Adaptive Control for Autonomous Vehicle Overtaking. Intelligent Transportation Systems, IEEE Transactions on. 15. 1643-1656. 10.1109/TITS.2014.2303995.
[133] Mcnaughton, Matthew & Urmson, Chris & Dolan, John & Lee, Jin-Woo. (2011). Motion Planning for Autonomous Driving with a Conformal Spatiotemporal Lattice. 4889 - 4895. 10.1109/ ICRA.2011.5980223.

Acknowledgment

Throughout the writing of this thesis, I have received a great deal of support and assistance. I would first like to thank my supervisor, Professor Hideya Ochiai, who has been continuously supporting me with my research and always providing invaluable expertise and feedbacks on methodology, experiments, and academic article writing. Your insightful suggestions and guidance throughout our discussion pushed forward me to understand better my research objectives, challenges, and room for improvement. I am greatly grateful for you providing me lots of precious opportunities to attend international conferences and the activities of the LAN-Security Monitoring Project. During this period, I was privileged to travel to various places in Thailand and attend different programs at other universities.

I would like to acknowledge my supervisor from my internship at UN Univ., Professor Ng Chong, for always supporting my work and providing inspirational thoughts to challenge me, allowing me to learn and grow. Thank you for your patient guidance and all the opportunities I have been given to further my research. I would like to thank the supervisor during my visiting program at Chulalongkorn Univ. in Thailand, Professor Nagul Cooharojananone for his valuable thoughts and guidance to my research project.

I would also like to thank the laboratory secretaries, Fumi Takahashi and Aiko Iwai, for their continual assistance with my administrative tasks. Finally, I would like to thank my parents in China. You are always there for me and support my every decision. I also want to give my gratitude to my friends at the laboratory of the university and friends from other countries during the journey for making my research life more substantial and enriching.

Publication

- **Journal/Transaction**

[1] Yuwei Sun, Hideya Ochiai, Hiroshi Esaki. Adaptive Intrusion Detection in The Networking of Large-Scale LANs with Segmented Federated Learning. IEEE Open Journal of the Communications Society. December, 2020.

- **International Conferences**

[2] Yuwei Sun, Hideya Ochiai, Hiroshi Esaki. Multi-Type Malware Detection in A Real-World Network Using Raw Network Traffic. IEEE Consumer Communications & Networking Conference (CCNC). Las Vegas, Nevada, USA. January 2021.

[3] Yuwei Sun, Hideya Ochiai, Hiroshi Esaki. Blockchain-Based Federated Learning Against End-Point Adversarial Data Corruption. 19th IEEE International Conference on Machine Learning and Applications (ICMLA). Miami, Florida, United States. December 2020.

[4] Yuwei Sun, Hideya Ochiai. Trajectory Optimization for An Autonomous Vehicle Driving across Stochastic Traffic Flows based on Direct Collocation. 4th IEEE International Conference on Control, Automation and Diagnosis (ICCAD). Paris, France. October 2020.

[5] Yuwei Sun, Hideya Ochiai, Hiroshi Esaki. Visual Analytics for Anomaly Classification in LAN Based on Deep Convolutional Neural Network. 9th IEEE International Conference on Informatics, Electronics and Vision (ICIEV). Kitakyushu, Fukuoka, Japan. August 2020.

[6] Yuwei Sun, Hideya Ochiai, Hiroshi Esaki. Intrusion Detection with Segmented Federated Learning for Large-Scale Multiple LANs. IEEE International Joint Conference on Neural Networks (IJCNN). Glasgow, United Kingdom. July 2020.

[7] Yuwei Sun, Ng S.T. Chong (UN Univ.), Hideya Ochiai. Text-based Malicious Domain Names Detection Based on Variational Autoencoder And Supervised Learning. 54th IEEE International Conference on Information Sciences and Systems (CISS). Princeton, NJ, United States. March 2020.

[8] Yuwei Sun, Hideya Ochiai, Hiroshi Esaki. Detection and Classification of Network Events in LAN Using CNN. 4th IEEE International Conference on International Conference on Information Technology (InCIT). Bangkok, Thailand. October 2019.

[9] Yuwei Sun, Nagul Cooharojananone (Chulalongkorn Univ.), Hideya Ochiai. Aircraft Detection Based on Saliency Map and Convolution Neural Network. 23rd IEEE International Conference on International Computer Science and Engineering Conference (ICSEC). Phuket, Thailand. October 2019.

- **Domestic Conferences**

[10]孫 昱偉, 落合 秀也, 江崎 浩. Anomaly Classification in LAN using Hilbert Curve and Deep Convolutional Neural Network. 超知性ネットワーキングに関する分野横断型研究会, IEICE Tech. Rep., ISSN 0913-5685. Tokyo, Japan. November, 2019.

[11]孫 昱偉, Ng S. T. Chong, 落合 秀也. TF-IDF と変分オートエンコーダーに基づく悪意のあるドメイン名の検出：量子サポートベクターマシンによる分類. サービスコンピューティング研究会 (SC). IEICE Tech. Rep., vol. 119, no. 275, SC2019-23, pp. 19-23. Shinshu, Japan. November, 2019.

[12]孫 昱偉, 落合 秀也, 江崎 浩. ディープ畳み込みニューラルネットワークに基づく LAN の異常分類のためのビジュアル解析ツール. ネットワークシステム研究会 (NS). IEICE Tech. Rep., vol. 119, no. 297, NS2019-121, pp. 7-12. Kobe, Japan. November, 2019.

[13]孫 昱偉, 落合 秀也, 江崎 浩. 深層学習に基づく LAN 内活動の検出システム. ネットワークシステム研究会 (NS). IEICE Tech. Rep., vol. 118, no. 465, NS2018-269, pp. 443-448. Okinawa, Japan. March, 2019.

Research Internships and Workshops

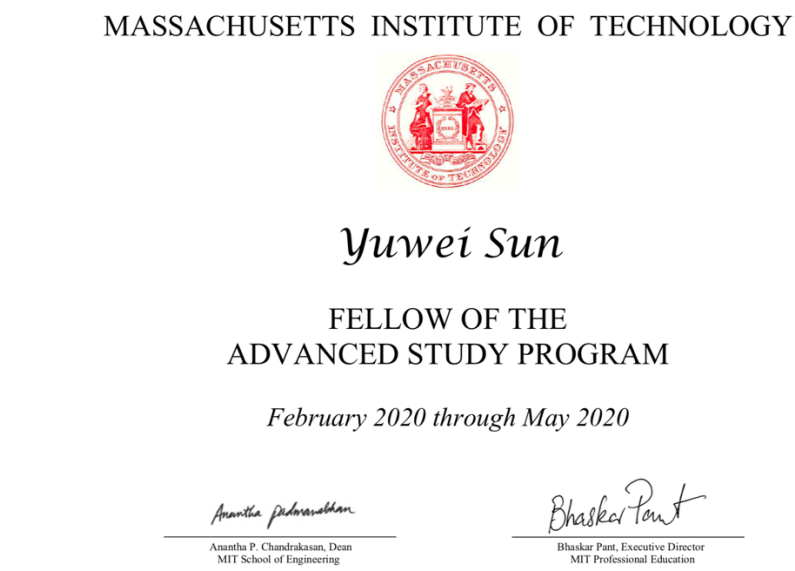
[1] Visiting student at the Faculty of Science, Chulalongkorn University. February 2019 – March 2019.

[2] Systems Engineer Intern (Research) at the Campus Computing Centra (C3), United Nations University Centra. May 2019 – Present.

[3] The fellow of the English Language Program at the University of Pennsylvania. August 2019 – October 2019.

[4] Workshop co-organizer at the UniNet, Thailand Ministry of Higher Education. January 2020.

[5] The fellow of the Advanced Study Program at the School of Engineering, Massachusetts Institute of Technology. February 2020 – May 2020 (Credit Program).



MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Yuwei Sun

FELLOW OF THE
ADVANCED STUDY PROGRAM

February 2020 through May 2020

Anantha P. Chandrakasan, Dean
MIT School of Engineering

Bhaskar Pant, Executive Director
MIT Professional Education

[6] AUA Entrepreneurship Initiative Online Program, Asian Universities Alliance, International Innovation Center of Tsinghua University, Shanghai. November 2020 – December 2020.

CERTIFICATE

This certifies that

Yuwei Sun

has successfully completed the courses and joined

the entrepreneurship competition in

AUA Entrepreneurship Initiative Online Program

December 11th, 2020

WANG YOUQIANG
Executive Director
International Innovation Center of Tsinghua University
Shanghai