上海交通大学硕士学位论文

Collaborative Anomaly Detection Based on Decentralized Federated Learning Framework for Internet of Things

硕士研究生: Seongwoo Kim

学　　号: 118036990004

导　　师: 化存卿

申请学位: 工程硕士

学　　科: 网络空间安全

所在单位: 电子信息与电气工程学院

答辩日期: 2021 年 1 月 11 日

授予学位单位: 上海交通大学

Dissertation Submitted to Shanghai Jiao Tong University
for the Degree of Master

COLLABORATIVE ANOMALY DETECTION BASED ON DECENTRALIZED FEDERATED LEARNING FRAMEWORK FOR INTERNET OF THINGS

Candidate:	Seongwoo Kim
Student ID:	118036990004
Supervisor:	Prof. Cunqing Hua
Academic Degree Applied for:	Master of Engineering
Speciality:	Network Security
Affiliation:	School of Electronic Information and Electrical Engineering
Date of Defence:	January 11, 2021
Degree-Conferring-Institution:	Shanghai Jiao Tong University

COLLABORATIVE ANOMALY DETECTION BASED ON DECENTRALIZED FEDERATED LEARNING FRAMEWORK FOR INTERNET OF THINGS

ABSTRACT

Since the growth of fifth-generation mobile communication (5G) and the extreme development of communication technologies, smart technology devices have become increasingly widespread. Recently, the Internet of Things (IoT) is recognized as one of the most leading technologies since 5G can manage the extensive data generated by the IoT network. In order to make the anomaly detection system (ADS) more feasible in IoT networks, various researches have been organized by several academic studies.

To overcome the challenges of traditional (centralized) ADS, We adopt the federated learning (FL) algorithm based on the notable FL virtues that can enhance personal privacy and decrease bandwidth demand and communication latency. However, the traditional FL algorithm has still been challenged by centralized topologies, such as communication failure rate, inflexibility, limited scalability. In order to intensify flexibility, scalability, and security protection, we develop the decentralized frameworks based on the synchronous and asynchronous parameter updating algorithms.

As the local training method on each client in the decentralized FL framework, two different ways of neural network anomaly classifier methods are operated; multi-layer perceptron (MLP) and stacked-autoencoder method. Both methods enable us to implement non-linear pattern classification based on the multiple hidden layers. In our adoption, the MLP is empowered by the triple hidden layers, and the Stacked-NDAE deploys six encoders (hidden layers).

After organizing local training, broadcaster clients spread the result parameter to the connected clients for the new model creation. In the synchronous model, every client is selected to the broadcaster, but in the asynchronous model, only the specified number of clients are activated as the broadcasters. Since every client broadcasts their parameter in the synchronous scheme, we can expect a quite comprehensive model aggregation. On the other hand, only a part of the clients becomes broadcasters in each round in the asynchronous model. The flexibility is expected to soar according to decreasing the number of broadcasters, but also the performance trade-off should occur since the generated model is less comprehensive than the case of more parameter broadcasters.

We implement several experiments to verify the proposed models' performance in the real-world demonstration. Initially, the performance evaluation shows that our proposed decentralized FL-ADS still has a competitive performance compared to the traditional FL-ADS. Subsequently, the convergence trend evaluation verifies that more broadcaster generates more effective parameter aggregation. Moreover, the convergence point is delayed in the multi-hop than in the full-mesh topology due to the parameter is not fully distributed in the multi-hop, unlike the case of the full-mesh. In addition, the Stacked-NDAE can perform better than the MLP method due to the deepness of the model. Consequently, our proposed models can contribute to most prominently scalability and flexibility while maintaining competitive performance by decentralized FL idea.

KEY WORDS: Anomaly Detection, Federated Learning, Decentralized Network, Auto-Encoder, Multi-layer Perceptron

Contents

Chapter 1 Introduction **1**
1.1 Internet of Things 1
1.2 Anomaly Detection Problem for IoT 3
1.3 Thesis Overview 4
1.3.1 Research Contents 4
1.3.2 Chapter Arrangement 5

Chapter 2 Related Works and Anomaly Detection **7**
2.1 Anomaly Detection Schemes 7
2.1.1 Anomaly Data Problems 7
2.1.2 Anomaly Detection Techniques 10
2.2 Collaborative Anomaly Detection Schemes 16
2.2.1 Federated Learning-based Schemes 17
2.2.2 Decentralized Collaborative Schemes 21
2.3 Chapter Summary 23

Chapter 3 Decentralized Federated Learning-based Anomaly Detection Schemes **25**
3.1 Overall Framework 25
3.2 Anomaly Detection Model Training 27
3.2.1 Activation Functions 28
3.2.2 Root Mean Square Propagation 30
3.2.3 Multi-layer Perceptron 30
3.2.4 Stacked-asymmetric Deep Autoencoder 32
3.3 Decentralized Federated Learning Design 32
3.3.1 Locally Parameter Processing Algorithm 33
3.3.2 Synchronous Parameter Broadcasting Scheme 34
3.3.3 Asynchronous Parameter Broadcasting Scheme 36
3.4 Chapter Summary 39

Chapter 4 Experimental Results **41**
4.1 Experimental Settings 41
4.1.1 Experimental Setup 43
4.1.2 Data set 43
4.1.3 Data Pre-Processing 44
4.1.4 Performance Evaluation Methods 50
4.2 Performance Evaluation 52
4.2.1 MLP-based Anomaly Detection Scheme 52
4.2.2 Stacked-NDAE-based Anomaly Detection Scheme 54
4.2.3 Section Summary 54
4.3 Convergence Evaluation 57
4.3.1 Comparison in Full-mesh Network 57
4.3.2 Comparison in Multi-hop Network 59
4.3.3 Section Summary 61
4.4 Chapter Summary 63

Chapter 5 Conclusion and Future Works **65**
5.1 Thesis Summary 65
5.2 Future Work 67

Bibliography **69**

Acknowledgements **75**

Publications **77**

List of Figures

Figure 1–1 Division of the IoT Applications in Real Life 2

Figure 2–1 An Example of Normal and Anomaly Data Distribution in a 2-Dimensional Data set [14] 8
Figure 2–2 The Anomaly Behavior Difference Based on the Value Detection [17] 9
Figure 2–3 Key components associated with the anomaly detection techniques [24] 9
Figure 2–4 Typical Multi-Layer Perceptron Framework 12
Figure 2–5 Development of Nonsymmetic Deep Auto-Encoder 13
Figure 2–6 Nonsymmetric Deep Auto-Encoder Anomaly Classifier on each edge 15
Figure 2–7 Architecture of the Typical Federated Learning Framework. 18

Figure 3–1 Decentralized Federated Learning Overview 26
Figure 3–2 Workflow at the edge node. 28
Figure 3–3 Applied Activation Functions in Our Scheme (Left: Sigmoid, Right: ELU) 29
Figure 3–4 multi-layer perceptron anomaly classifier on each edge 31
Figure 3–5 The Anomaly Detection System Based on Synchronous Decentralized Federated Learning Framework. 35
Figure 3–6 The Anomaly Detection System Based on Asynchronous Decentralized Federated Learning Framework. 37
Figure 3–7 Overview of Model Comprehensiveness Comparison Between the Cases of Fewer Broadcaster and More Broadcaster among Four Clients 38

Figure 4–1 Decentralized Inter-communication Framework in Each Network Topologies 42

Figure 4–2 MLP-based Train Loss Trend Comparison Between Synchronous&Asynchronous Decentralized FL models in the Full-mesh Network 57
Figure 4–3 MLP-based Train Accuracy Trend Comparison Between Synchronous&Asynchronous Decentralized FL models in the Full-mesh Network 58
Figure 4–4 Stacked-NDAE-based Train Loss Trend Comparison Between Synchronous&Asynchronous Decentralized FL models in the Full-mesh Network 58
Figure 4–5 Stacked-NDAE-based Train Accuracy Trend Comparison Between Synchronous&Asynchronous Decentralized FL models in the Full-mesh Network 58
Figure 4–6 MLP-based Train Loss Trend Comparison Between Synchronous&Asynchronous Decentralized FL models in the Multi-hop Network 59
Figure 4–7 MLP-based Train Accuracy Trend Comparison Between Synchronous&Asynchronous Decentralized FL models in the Multi-hop Network 60
Figure 4–8 Stacked-NDAE-based Train Loss Trend Comparison Between Synchronous&Asynchronous Decentralized FL models in the Multi-hop Network 60
Figure 4–9 Stacked-NDAE-based Train Accuracy Trend Comparison Between Synchronous&Asynchronous Decentralized FL models in the Multi-hop Network 61

List of Tables

Table 4–1 Hardware Environment 41
Table 4–2 Experimental Setup Parameters 43
Table 4–3 NSL-KDD Data set Feature list; 44
Table 4–4 An Example of Data Set before One-hot Encoding Procedure 46
Table 4–5 An Example of Data Set after One-hot Encoding Procedure 46
Table 4–6 Convergence Setting Based MLP FL-ADS Model Performance Comparison..... 53
Table 4–7 Convergence Setting Based Stacked-NDAE FL-ADS Model Performance Comparison 55
Table 4–8 Performance Experiment Result Comparison 56
Table 4–9 Time Consumption to Finish 50 Rounds..... 61

Chapter 1 Introduction

We developed the decentralized framework-based federated learning anomaly detection systems (ADS) for the Internet of Things (IoT) networks [1]. Before the interpretation of the suggested algorithm, we present the basis of this research.

1.1 Internet of Things

According to the radical growth of communication technologies, the fifth-generation mobile communication (5G) optimization to the real-world, and raising of the sixth-generation wireless communication (6G), the internet devices have become increasingly prevalent. Recently, IoT technology is recognized as a mainstream of cutting-edge technology trends. The representative IoT applications are classified as below categories;[2]

- *Consumer IoT*: light fixtures. home devices, voice assistance, and etc.
- *Commercial IoT*: healthcare system, and transportation system industries, such as smart pacemaker, traffic network monitoring system, and V2X communication (vehicle to vehicle/infra/etc).
- *Industrial IoT (IIoT)*: digital control systems, statistical evaluation, smart agriculture, and industrial big data.
- *Infrastructure IoT*: the smart cities' facility connection through the use of infrastructure sensors, management systems, and user apps.
- *Military IoT*: surveillance and detection robot, and wearable biometric arms for battle.

Based on the market trend's perception, diverse and massive research has been organized by heterogeneous academic studies. Since it is one of the mainstream, the research [3] shows that there were 560 billion euros (approx. 668 billion USD based on the exchange rate 1.19 USD/EUR, Nov 27th, 2020) of damage caused by attacks on the network security in the world in 2016, also the 8.4 billion of IoT devices were active globally in the year 2017. The number of active IoT devices is predicted to rise to 30 billion devices by 2020 in the study [4]. Moreover, the study about IoT statistics and trends [5] shows 127 new IoT devices join to the web every second, and the number of devices by 2025 is

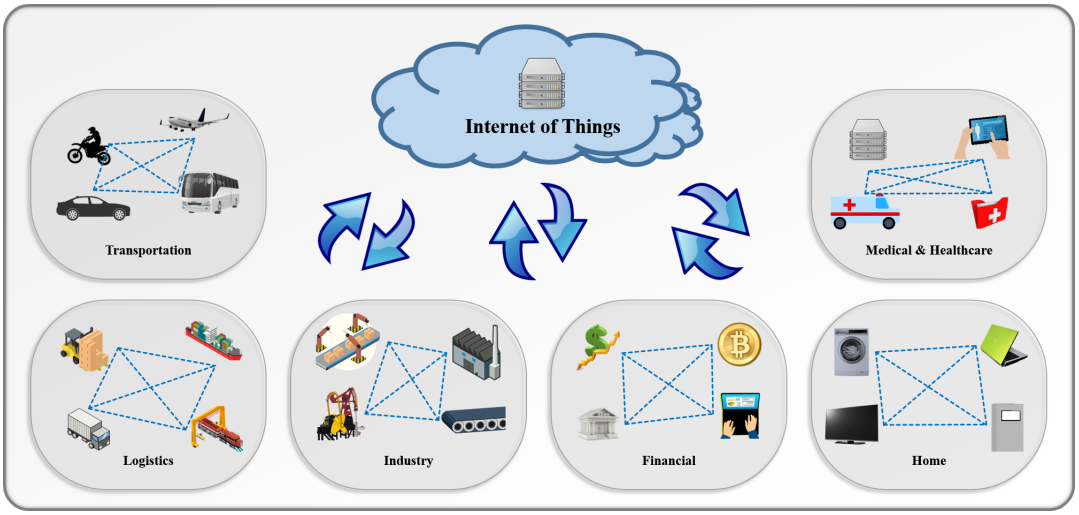


Figure 1–1 Division of the IoT Applications in Real Life

estimated to be up to 75 billion devices. Also, they estimated the IoT market size to reach 124 billion USD by 2021, and the cellular IoT connection number is predicted to be more than 3.5 billion connections by 2023. In accordance with the trend of the IoT market and the number of devices, we can clarify that both of them are remarkably expanding.

Since ubiquitous IoT devices are interconnected, two significant challenges are recognized as the typical problems; the massive data to be transmitted and various network security issues. (Figure 1–1)

While the IoT environment gives us enormously convenient life, on the other hand, certain risks and challenges are also expended and raised on the basis of the massive number of heterogeneous inter-communications. Based on the fact that the IoT network consists of massive inter-communications between various network systems and devices, the study [6] presents that the volume of global datasphere was predicted from 33 Zetabytes (ZB) in 2018 to 175 ZB by 2025; a ZB represents the volume of data that a trillion giga-bytes. They also interpret that it may need 1.8 billion years to download the 33 ZB of 2025 datasphere by the average download speed rate of 25MB/s.

Based on this radical extension of IoT devices and the market, we are motivated to design a system to protect IoT network security while not bothering their main functions.

1.2 Anomaly Detection Problem for IoT

According to the challenges from the radical development of IoT technology mentioned in the last section, supermassive inter-connection and ubiquitous devices must be a beautiful environment for eavesdroppers and cyber attackers to get obscene profits. They threaten people's privacy information, national and corporate national secret information, and also each device's regular operation. In practice, cyber-attacks, namely worm, botnet[7], malware, backdoor, and insider attack[8], infrequently transpire, and these abnormal data use quite a few system resources.

The IoT network ADSs watch misbehavior processings among the massive device transmission data. Since there are enormously complex inter-connections between IoT devices, the misbehavior detection tasks must be hard assignments. Consequently, the anomaly detection task becomes one of the most crucial challenges in the current network development status. Nevertheless, the above-mentioned cyber attacks are absolutely detectable assignment to anomaly data since they appear among the network intercommunication.

In a centralized ADS architecture, increasing the number of terminals may increase data transmission costs, enormously reduce the server unit's computation speed, and occur high transmission delays. In the traditional method, a large number of samples are transmitted, calculated, and analyzed on the server, and a large amount of raw data transmission between the edges and the cloud does not meet the confidentiality requirements of user data. In general, the ADSs are based on various technologies, namely statistics[9], data mining[10], information theory[11], and machine learning (ML)[12]. Among the several anomaly detection methods, the ML-based technology is rising as the leading fraction of ADS recently. Because it enables us to extract the referable features from the data set autonomously, It performs comparatively very precise classification tasks and predictions using different models. However, the standard ADS technologies based on ML are still suffering from challenges that have to be overcome to enhance ADS performance in super complex conditioned IoT networks. According to the motivation which needs to solve the existing challenges in the IoT ADS area, this study aims to propose several solutions that can well-alleviate the challenges above.

1.3 Thesis Overview

The key concept to achieve the goal is adopting federated learning[13] (FL) algorithm to ADS application. The FL is a famous machine learning method based on the fact that it reduces bandwidth demand, transmission weight, and enhances data leakage problems such as personal privacy and private data in the network. The significant virtues come from transmitting the local training result parameters to the cloud server instead of raw data, according to the basic idea that ML is on each edge node, not on the cloud server.

1.3.1 Research Contents

Due to the data communications in the IoT network consist of not only transmissions from devices to the server but also inter-communications between IoT devices, the traditional anomaly detection methods keep challenged to higher requirements for data distribution, and the detection results also have relatively large limitations. Thus, the centralized system has an extensive central processing system; the central processing system is a high-performance, expandable computing equipment; all data, computing, and processing tasks are organized on the central computer system. On the other hand, the multiple devices are connected to the central server work for input and output functions but do not have data processing tasks.

This research focuses on solving the massive data management and privacy protection challenges; we adopt the FL method. However, there is one more notable characteristic of IoT circumstance that each element device's in and out of the network is very often. To deal with the dynamicity of the IoT element status, we redesign the FL algorithm to be a decentralized framework. Owing to awareness of the decentralized inter-connections, the existing central cloud server's management is eliminated. Thus, the redundant tasks and cost can be alleviated, and tolerance to unpredictable element dynamicity can be much enhanced.

In order to test our novel models' performance in real-world networks, we demonstrate the full-mesh network, which is usually adopted to the backbone networks, and multi-hop network, which more represents IoT circumstance. In the experiments, we first implement the performance comparison between the traditional FL-ADS, Synchronous Decentralized FL-ADS, and Asynchronous Decentralized FL-ADS based on our specific convergence conditions. Subsequently, the convergence trend is visualized to compare

how the models are going to converge and how the performance gets diverged.

The contributions of our research are shown as follows.

- Based on FL utilization for ADS, the contents of transmission become the anomaly detection models instead of raw data. From this change, eavesdropping is prevented, the transmissions between communication mediums become much lighter, and the effectiveness of bandwidth usage is enhanced.
- We transfer the parameter aggregation task to edge nodes from the traditional FL to build the decentralized FL for detecting anomaly data in IoT networks. Therefore, the shortcomings from the centralized network are eliminated, such as prone to failures, users' security and privacy risks, and longer access times for far edges. And it enables us to manage the network to flexible, scale network area significantly, be fault-tolerant.
- By utilizing two different cutting-edge techniques, the usage of each application's pros and cons are analyzed. The MLP can perform desirable accuracy with the light-weight model, and the Stacked-NDAE enhanced the anomaly detection performance, but it takes more time to run the training since the model is more complicated.

1.3.2 Chapter Arrangement

As the first chapter of this paper, the introduction, motivation, overall research construction are presented in this chapter. Such as the importance of an accurate and efficient anomaly detection system in the context of the rapid expansion of the current Internet of Things area. Also, the advantages and disadvantages of the existing anomaly detection methods and system structures are briefly described. Finally, the main content and innovation points of the research project are explained.

Chapter 2 shows that the related technology trends and summaries are in accordance with such research. It mainly describes three detailed categories; firstly, the current anomaly detection methods and principles with interpreting the advantages and disadvantages of various studies at the emergence stage and the problems to be solved. Since our research is about optimizing the network processing in the IoT networks, the existing studies about collaborative networking-based anomaly detection architectures and the current Federated learning-based ADS are followed.

According to the background and motivations, we propose the synchronous and

asynchronous inter-communication-based FL-ADS framework in Chapter 3. The utilized classifier modules on each edge node are presented minutely, and the explanation of the synchronized result parameter transmission algorithm between each edge is followed. Subsequently, another novel decentralized mode, the asynchronous decentralized FL algorithm, is displayed. Afterward, our proposed schemes' evaluation methods and criteria are explained.

The experiment to testify our novel models' performance in real-world simulation is presented in Chapter 4. The experiment section is divided into two big categories; the performance comparison experiment based on our convergence condition settings and the convergence trend comparison within 50 rounds. The experiments are organized based on the comparison targets; the traditional FL-ADS, synchronous/asynchronous decentralized FL-ADS algorithms, and the two different neural network methods, the MLP and Stacked-NDAE, are utilized for each experiment, respectively.

In accordance with the proposed schemes, we draw the final conclusion and future work based on the result analysis in Chapter 5.

Chapter 2 Related Works and Anomaly Detection

In this chapter, we review existing studies related to task on DL-based anomaly detection algorithms. In general, ADS relies on a centralized management method to collect and process data generated from end-users or IoT devices.

Since our proposed scheme encourages the federated learning algorithm to be decentralized to overcome the existing challenges in recent centralized anomaly detection networking, the conventional FL-ADS operates the anomaly classification tasks on each edge device, which is also called client. We distribute this chapter into four major subsections: Anomaly Detection Schemes, Collaborative Anomaly Detection Schemes, and Decentralized Collaborative Schemes.

2.1 Anomaly Detection Schemes

This research's main purpose is developing the ADS optimized for IoT networks. In order to clarify the insight of anomaly detection tasking, we describe the anomaly data and several anomaly detection methods.

2.1.1 Anomaly Data Problems

Anomaly detection is usually carried out based on data, which refers to finding problems in the data that are different from a specific normal pattern. These unusual data indications are called anomaly data in general. Anomaly detection has been widely used in various applications, such as network security intrusion detection, anomaly detection, industrial attack protection, financial and credit card fraud, and military surveillance.

The theoretical basis that anomaly detection can rely on anomalous data is that the anomalies often represent abnormal operation information. For example, computer network traffic data's abnormality may indicate that the attacked host may send sensitive information to unauthorized targets [15]. Early 19th century, people have studied the anomaly detection algorithm based on the statistic approaches [16]. According to the continuous development of academic level, science, and technology, numerous anomaly detection technologies have been exploited from different fields. Many of these technologies are specifically developed for specific application areas, while others are developed

for more general methods.

Anomalies can be detected by the data pattern that does not correspond to a determined normal behavior concept. Figure 2–1 shows an example of anomalies in a two-dimensional data set. N_1 and N_2 are two normal regions based on the fact that most of the samples are in these two regions. On the other hand, the point o_1, point o_2, and O_3 are out of the normal area. Thus we define them as anomaly data. Outlier data can be divided into point anomaly (Global Outlier), conditional anomaly (Contextual Outlier), and collective anomaly (Collective Outlier). (Figure 2–2)

- Point Anomaly: Point anomaly, which is also called "Global Outlier," is defined based on a single sample different from other data such as o_1 and o_2. This type of data is the simplest and most common target of existing anomaly data detection studies.
- Conditional anomaly: Context anomaly, also called contextual outlier [18] refers to the abnormal behavior of data in the specific sequences. The abnormal data and trends appear in a particular normal data information sequence. Conditional anomalies are mostly occurred in time-series data [19] and spatial series data [20].
- Collective anomaly: Collective data anomaly (Collective Outlier) is detected in accordance with a set of data that behaves abnormally compared to other normal

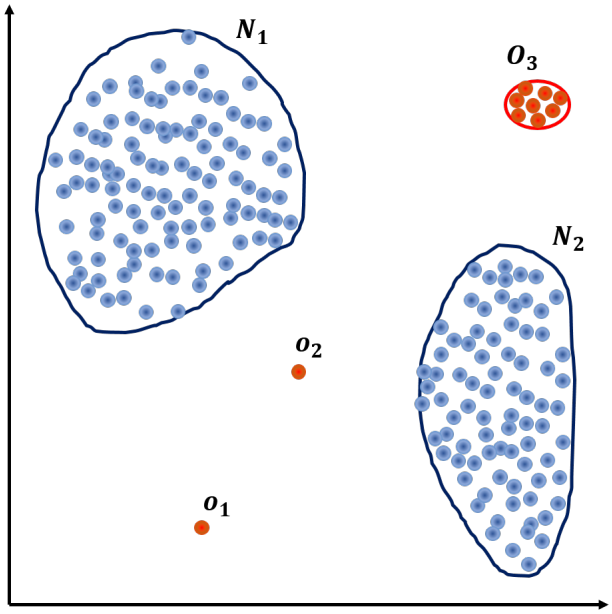
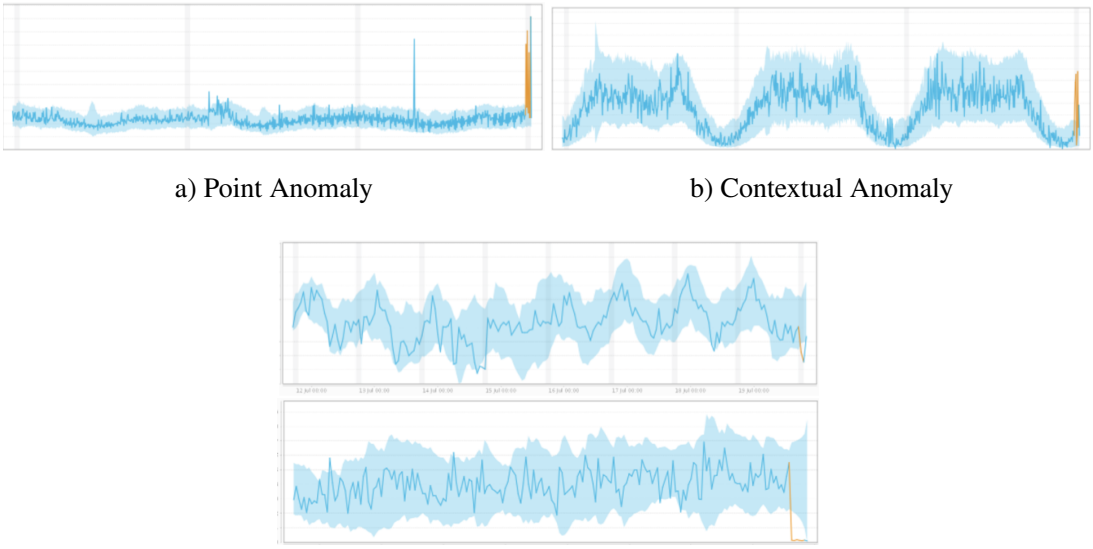


Figure 2–1 An Example of Normal and Anomaly Data Distribution in a 2-Dimensional Data set [14]



a) Point Anomaly

b) Contextual Anomaly

c) Collective Anomaly

Figure 2–2 The Anomaly Behavior Difference Based on the Value Detection [17]

data in the whole data set. A single sample in the abnormal data set could be considered to the normal range. However, the occurrence of this specific data set at a specific time and area can be the feature to be shown as abnormal behavior. Collection anomalies commonly occur in time series data [21], spatial series data [22], and image data [23].

Discovering the characteristics and performance patterns of abnormal data and analyzing and locating abnormalities through abnormal data performance is the key to abnormal detection technology. Figure 2–3 shows the key components in the anomaly detection task [24]. We use data features and data labels to order to analyze and learn data. In defining anomaly types, the sample's predicted output classifies the data between nor-

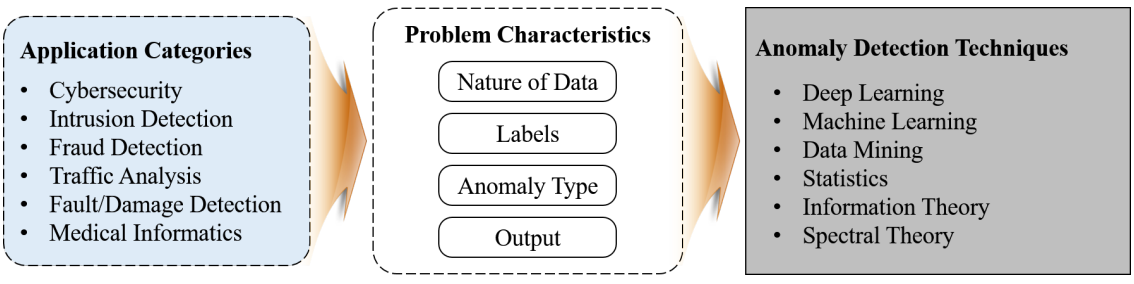


Figure 2–3 Key components associated with the anomaly detection techniques [24]

mal and abnormal in the application. At present, people have explored anomaly detection techniques in the fields of statistics, data mining, information theory, machine learning, and deep learning.

2.1.2 Anomaly Detection Techniques

In recent years, various anomaly detection methods are existing, such as statistics, data mining, information theory, and ML (and DL) as shown in Section 1.2. The ML methods can be categorized into three leading technologies, such as supervised learning, which needs the guides to conduct a training (e.g., KNN[25], decision trees[26]), unsupervised learning, which does not need to get addressed to learn (e.g., clustering[27]), and semi-supervised learning that the mixed concept from both of them (e.g., semi-supervised SVM[28], semi-supervised Deep Learning (DL)[29]). Since the DL-based ADS techniques are rising as the leading fraction, the ML and DL-based ADS studies are introduced in this section.

There is an attempt to design the flow-based anomaly detection algorithm [30]. To achieve the extraction of feature data sets from flow records, they utilize the K-means clustering algorithm. In this study, the training data set diverges into clusters of normal and abnormal traffic time intervals. This model allows us to optimize the detection method for scalable real-time detection. Moreover, in order to design an enhanced unsupervised network intrusion detection system, the authors in [31] propose a new high dimensional clustering algorithm, fpMAFIA, based on density and grid. This approach is verified that the proposed model can reduce computational complexity and is capable of the large data sets, whereby maintaining a reasonable anomaly detection rate and low positive rate. The research by Guan *et al.* [32] proposes a novel K-means-based clustering algorithm, which is called "Y-means." The designed Y-means scheme is independent of the parameter k value. In the proposed procedure, the k value is tuned while the algorithm is executed. This scheme gets over the existing K-means' two typical challenges; the number of clusters dependence and retrogression.

In [33], the authors utilize the Support Vector Machine (SVM) for anomaly classification since the SVM is one of the most influential classification techniques in the data mining realm. However, the SVM still has a considerable challenge that requires a long time to train. To overcome the challenge, they applied the hierarchical clustering analysis for the case of large data sets to train. Also, they utilize the Dynamically Growing

Self-Organizing Tree (DGSOT) method to get over the drawbacks of conventional hierarchical clustering methods. According to the comparison with the Rocchio Bundling and random selection, the proposed model is proved to vitally improve SVM's training process with significant generalization accuracy and outperform the Rocchio Bundling algorithm.

In accordance with the fact that the current mainstream of anomaly detection has developed from traditional ML to DL methods, more DL-based researches are surveyed.

In the existing research, Chalapathy *et al.* [34] presents eclectic state-of-art intrusion detection systems (IDS) on the basis of DL, namely, unsupervised, supervised, semi-supervised, one-class neural networks, and hybrid models. The authors introduced plenty of studies about ADS. Typically the approaches [35] and [36] are introduced. Mudassar *et al.* [35] suggests an unsupervised anomaly detection technology that utilizes a convolutional neural network (CNN) backbone and a restricted Boltzmann machine (RBM). The RBM discovers unusual samples refer to high energy values since it designates low energy rates to reliable samples.

Subsequently, there is an approach that uses auto-encoders (AE), artificial neural network technology, to enhance ADS performance in wireless sensor networks (WSNs) on the basis of the IoT network [36]. The proposed AE-based scheme consists of two big algorithms; the sensor part and the cloud part. In the sensor part, the outlier data can be identified at sensors in a fully distributed manner without any transmitting with other sensors and cloud. Afterward, the cloud node can manipulate the computationally complicated intensive learning. The authors confirm the proposed scheme's performance based on random setup and prioritized setup. Because the prioritized setup allows the AE model to concentrate more on fresh inputs than when working on the random setup, the model shows a lower false-positive rate (FPR) result 60% better than the random setup based experiment.

Among the heterogeneous approaches to detect anomaly data, we select two different cutting-edge techniques for our decentralized FL-ADS networks. The adopted methods are displayed on the next two subsections; the multi-layer perceptron and Stacked-Nonsymmetric Deep AutoEncoder, respectively.

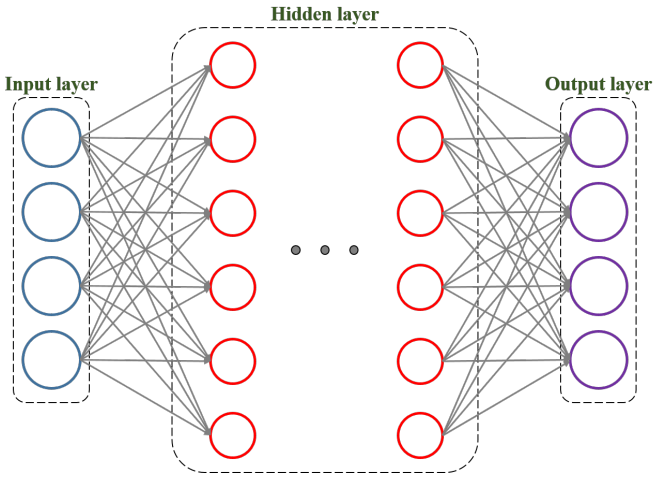


Figure 2–4 Typical Multi-Layer Perceptron Framework

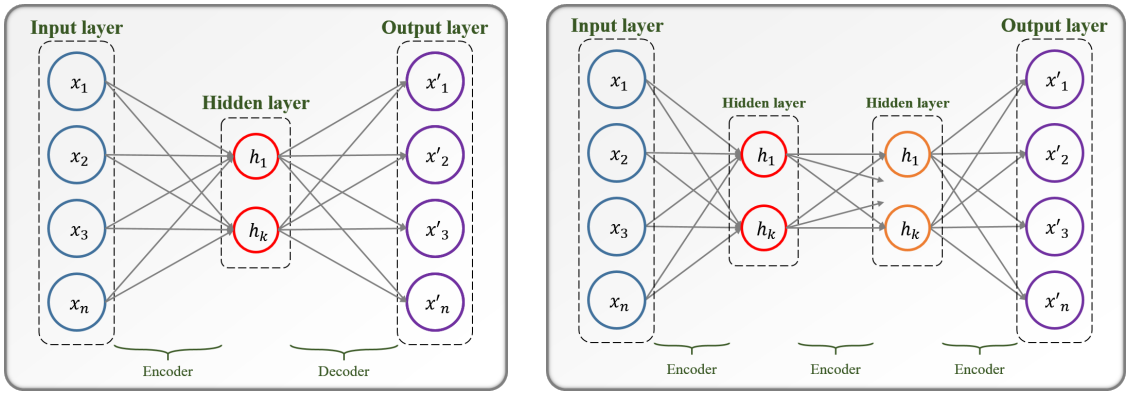
2.1.2.1 Multi-layer Perceptron

In our system, we use a multi-layer perceptron (MLP) mechanism as the anomaly classifier in the neural networks' particular term since it provides us non-linear classification function based on several hidden layers. The notable characteristic is that MLP has more than one hidden layer, unlike Traditional ML models. Thus, It includes an input layer, an output layer, and multiple hidden layers.

Firstly, the input layer injects corresponding samples ($x_1, x_2, .., x_n$ in X). Each hidden layer is linked with each fully connected layer to ensure significant feature extraction and classification. $f(WX + b)$ is plotted as each hidden layer's output, where W is the corresponding weight, b is the offset, and f denotes the allocated activation function for each layer.

At the beginning of our proposed schemes, each client chooses the basic pre-settings, such as the local model's learning rate and hyper-parameter. Based on the pre-setting, the MLP model on each client is operated to learn the anomalies' misbehavior based on the reconstructed data.

A flow-based IDS technology utilizing the J48 decision tree and multi-layer perceptron (MLP) is developed [37]. Their experimental result verifies the outstanding performances of J48-based and MLP-based methods in both Winter and UNSW-NB15 data sets. Their accuracy achievements are 1.000 and 0.998 based on the Winter data set and 0.985 and 0.910 based on the UNSW-NB15, respectively. According to the overall result, we can conclude that the J48-based scheme offers slightly better performance than the MLP-



a) Traditional Auto-Encoder b) Nonsymmetric Deep Auto-Encoder

Figure 2–5 Development of Nonsymmetic Deep Auto-Encoder

based algorithm. The reason is shown that the J48 algorithm has higher complexity than MLP.

2.1.2.2 Stacked Nonsymmetric Deep AutoEncoder

Another classifier method we choose is motivated by Shone Nathan *et al.* [38] that Stacked-Asymmetric Deep Auto-Encoder (Stacked-NDAE), which contains double asymmetric AE paradigms. The non-symmetric AE (NDAE) algorithm is developed for unsupervised learning, which only contains encoder parts, unlike typical AE in the study, which contains encoder and decoder realms. First of all, they swift the decoder parts from the traditional AE to encoder in order to generate Non-symmetric Auto-Encoder (NDAE) as a single novel AE. Subsequently, they attach two NDAEs to propose Stacked-NDAE, making more robust performance than the single NDAE [39]. The reasoning behind making non-symmetric frameworks enables to alleviate both computational and time costs, with minimal performance trade-offs on accuracy and efficiency. The detailed introduction of Stacked-NDAE follows below;

First of all, Figure 2–5a represents Auto-encoder, also known as one of the practical neural network algorithms as unsupervised learning. This AE algorithm generally consists of two big parts; the encoder and decoder parts. It extracts features based on the procedure that learning the best parameters required to reconstruct output values as close to its input values as possible through encoder and decoder realms. It is known as a more non-linear and robust generalization than the PCA algorithm.

The virtues are achieved by organizing backpropagation and equalizing the target

values to the inputs. In the traditional AE, the input layer and output layer dimensions must be the same, and the hidden layer has to include a smaller number of dimensionality than that of the input layer. Therefore, dimensional input data transformed into a lower-dimensional layer based on the encoding procedure and expanded to reconstruct the original data by the decoder. On the basis of reducing the dimensionality in the encoder part, the AE can extract the most representative features of the data distribution; it is the motivation to apply the novel AE mechanism for our study. Based on the fact that the AE's goal is generating the same output values as much as it can by encoding and decoding procedures; the overall goal function shown in function (2–1)

$$h_{W,b}(x) \approx x \tag{2–1}$$

Where, h is a non-linear hypothesis function to reconstruct given data (x) based on the weighting parameters W and bias b. Furthermore, the reconstruction error minimization function as the learning process is shown as the function (2–2).

$$L(x, d(f(x))) \tag{2–2}$$

Where, L is a loss function punishing $d(f(x))$ for discrepancy index from original x, f and d are the encoding and decoding functions, respectively.

Subsequently, Figure 2–5b represents the NDAE framework. In order to facilitate enhanced classification results, the system utilizes the devised AE paradigm that is shifted from the symmetric (encoder-decoder) paradigm to the nonsymmetric encoder paradigm, which is called NDAE. According to the fact that only encoders are utilized, the NDAE algorithm can alleviate the computational burden and time overheads with minimized accuracy and efficiency trade-offs. This model can extract prominent features as a hierarchical unsupervised stance that facilitates to accommodate high-dimensional input data.

We assume that the NDAE takes an input vector $x \in R^d$ and maps it to the next hidden layer $h_i \in R^{d_i}$ step by step, where d is the vector's dimension, applying a deterministic function shown as the function (2–3).

$$h_i = \sigma(W_i \cdot h_{i-1} + b_i);\ i = \overline{1, n} \tag{2–3}$$

Here, $h_0 = x$, σ represents an activation function, which is the sigmoid function in our study, and n means the number of hidden layers.

Since the NDAE algorithm does not contain any decoder procedure, unlike the traditional AE, the output vector is computed by the function (2–4).

$$y = \sigma(W_{n+1} \cdot h_n + b_{n+1}) \tag{2–4}$$

The model estimator $\theta = (W_i, b_i)$ can be achieved by the square reconstruction error minimization over m of training samples $(x^{(i)}, y^{(i)})_{i=1}^{m}$, as follow;

$$E(\theta) = \sum_{i=1}^{m} (x^{(i)}, y^{(i)})^2 \tag{2–5}$$

Consequently, Figure 2–6 shows the Stanked-NDAE neural network. Based on the NDAE as explained above, Staking the NDAE mechanism is used in our proposed scheme as the local anomaly classifier. This method allows us to learn the complicated correlations between different features. Based on the fact that it offers excellent feature extraction abilities, this model progresses the performance by prioritizing the most prominent features. Therefore the twin NDAEs are tacked in the Stacked-NDAE. Each NDAE contains three hidden layers, respectively, and the structure of the neuron number allocated to each hidden layer is the same as each other, as shown in figure 2–6. In other words, the structures of NDAE 1 and NDAE 2 are identical. Therefore, the Stacked-NDAE, which we use, allows us to conduct performance evaluations without overfitting jeopardy.

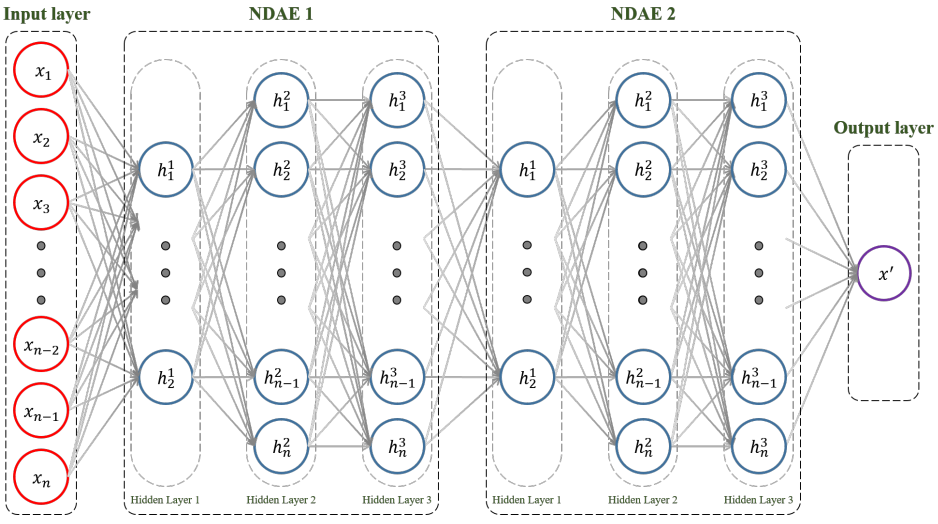


Figure 2–6 Nonsymmetric Deep Auto-Encoder Anomaly Classifier on each edge

2.2 Collaborative Anomaly Detection Schemes

Our research focuses on designing the novel decentralized anomaly detection networks based on the federated learning framework. The studies designing the anomaly detection networks are presented in this section to recognize the current ADS network development trend.

In order to soar the reliability of time-sensitive IoT applications, the Fog-Empowered anomaly detection is proposed by authors in [40]. This novel scheme is empowered by employing the processing ability of the Fog computing platform and utilizing an effective hyperellipsoidal clustering algorithm. The DL-based cyber-attack detection model for edge networking is introduced [41]. The authors say that fog-to-things computing is one of the best techniques for cyber-attack detection since an extensive volume of the IoT devices' data can help DL models to make better performance than shallow models, namely training accuracy, false alarm rate, and detection rate.

According to [42], the authors suggest a collaborative anomaly detection system on the basis of the permissioned-blockchain. They focus on detecting the assets' outlier behaviors by sensor data; in other words, the condition-based management (CbM) anomaly detection algorithm is proposed. Moreover, authors in the work [43] present a collaborative ADS on the basis of the active learning technique, whereby the edge node only allows the most beneficial data to transmit to the server utilizing uncertainty sampling approaches, which can alleviate the bandwidth efficiency and transmission latency.

The work by Moustafa *et al.* [44] proposes a collaborative intrusion detection algorithm based on the modification of the Gaussian Mixture Model for identifying cyber-attacks from cloud computing systems. This system deployed on each node in the cloud network and each node in the training sequence generates a statistical model of standard data by approximating probability distribution function based on the Gaussian Mixture Model. The proposed system is demonstrated on the UNSW-NB15 dataset to verify its reliability while deploying it in cloud computing networks. The authors accomplish a performance of 96% in detection rate and accuracy and 4% to 9% in false-positive rate. The experiment presents that the framework can manage large-scale systems well, as its implementation needs only computing the statistical models from network observations.

2.2.1 Federated Learning-based Schemes

In the previous section, we overview the existing ADS network designs. On the other hand, our proposed decentralized FL-ADS concludes every IoT components' inter-communication in the specific network area. Thus, there is a need to understand FL and existing studies.

2.2.1.1 Federated Learning

Because the ADS we suggest is built on the basis of the decentralized FL algorithm backbone, we interpret the FL algorithm to get the core insight of the FL algorithm.

FL technology is one of the mainstream machine learning methods in the heterogeneous academic fields due to the listed virtues above. The virtues come from the fact that the data inter-communications are based on delivering the training result parameter instead of the raw data. The beneficial key points are presented as shown below: [45]

- Privacy Security Protection: Since plenty of inter-connections exist and frequently occur in IoT network, the evasion of privacy information leakage must be one of the most important issues to be solved. The FL is more beneficial for dodging personal/industrial/national information leakage because each client's parameters are transmitted instead of local raw data. In the case that the parameters are leaked to somewhere or someone, It does not matter because there is no clue to tract visible information.
- Network Bandwidth Efficiency: Also, because of the massive interactions between the IoT elements, how to manage and optimize the network bandwidth is one of the biggest challenges to maintain stable network conditions. Since the delivering model parameters have much more light-weight than when we transmit the original raw data, the demand for network bandwidth is significantly decreased. In other words, we can contain more devices and serve higher-quality applications. According to the super complex and many connections are in the IoT network, decreasing bandwidth demand leverages significantly alleviated work burden.
- Low Latency: There are many time-critical applications in IoT networks such as cutting-edge industrial facilities, intelligent transportation systems (ITS), theft detection systems. If the network is busy, the possibility of the time-critical applications' transmission failure may soar, and it becomes crucial damage for the

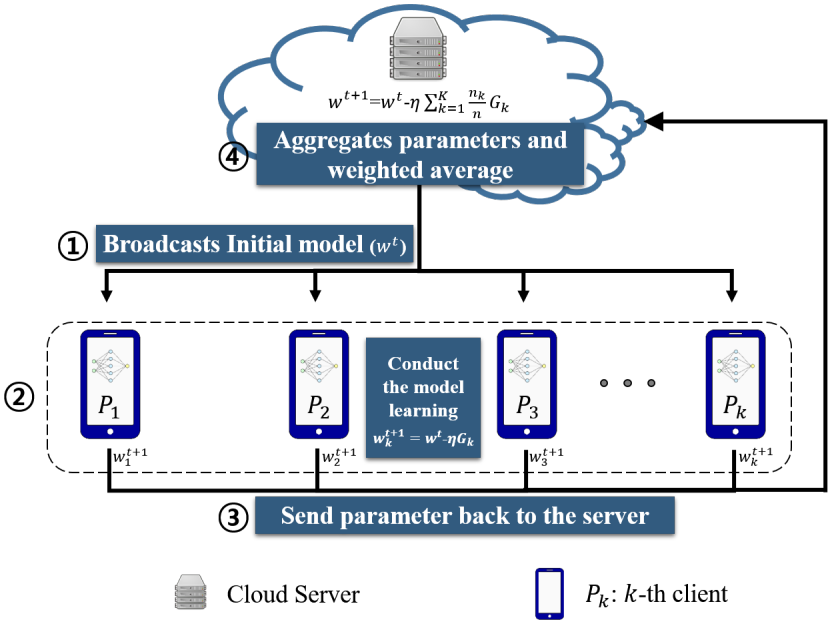


Figure 2–7 Architecture of the Typical Federated Learning Framework.

overall network and our real-life. Due to light-weight transmissions in FL, the communication latency becomes extremely diminished compared to that of systems where raw data is updated to the cloud server. Whereby emergency devices and any other time-sensitive applications work in IoT, the low latency is recognized as unquestionably a desirable feature.[46]

In accordance with Figure 2–7, the comprehensive FL framework is divided into two big categories; the server node and client nodes.

The server node is in charge of supervising the overall FL mechanism, namely broadcasting a global model to clients for each round and producing a new global model for the subsequent round on the basis of aggregating the received local model parameters every round.

On the other hand, clients conduct local training, which is operated on the cloud server in conventional ML networking frameworks. Firstly, they receive the pre-setting from the cloud server and conducts local data pre-processing for the local machine learning. Afterward, they operate local training based on the received global model from the cloud server, generate local training the parameter, and send the parameter to the cloud back.

To explain the FL procedure from the beginning, each edge node accumulates the local data and operating local training based on the opted DL model. Afterward, each client transmits the training result parameters to the cloud server, whereby the cloud generates a new model for the next round through aggregating the arrived local parameters. When the new global model is generated on the cloud, the cloud decides between model convergence or not based on the generated new model. If the new model is not converged, the server broadcasts the new global model to every client, and each client continues the next round $(t + 1)$. Otherwise, If the model is converged, the server broadcasts the new global model as final, and each local edge accepts the global model as the permanent model. This round procedure keeps iterated until satisfactory accuracy is accomplished, or the aggregated result on the server is converged.

The algorithm of the FL procedure is sequentially presented as follows;

- *(Step 1) Task Initialization:* The basic settings for the local DL task are set on the server, namely the global model's hyper-parameters, learning rate, and target applications. Thereafter, the cloud server transmits the initialized global model $w^t (= w_G^t)$ and the pre-settings mentioned above to randomly picked K number of clients at the present round t, the picked clients are included into the subset S_t.
- *(Step 2) Local model training:* The clients in the subset S_t organizes the local ML using the collected local raw data and the received global model w^t. As a result of the local training in edge k, the parameter w_k^t is printed. Afterward, the parameters $w_k^t (k = 1, .., K)$ are transmitted to the cloud server as the sources of new global model formation. As shown the function (2–6), the optimal local parameter $w_k^{t^*}$ is chased by the minimized loss function $L(w_k^t)$.

$$w_k^{t^*} = \underset{w_k^t}{argmin} L(w_k^t) \tag{2–6}$$

- *(Step 3) Model Aggregation:* The parameters from the clients in the subset S_t arrive at the cloud server as the sign and report of finishing local training in client i. The server aggregates the parameters until parameters from all participants in S_t arrive and create a new global model for the next round. The server makes a decision that it is converged or not on the basis of the printed new model and the previous global model records. If the new global model is cognized that it is not converged yet, the server sends a trigger to run the next round to each client in the

network with the newly generated global model. This procedure is arranged to a round process. The round is keep iterated until it is converged. Consequently, when it converges or reaches the maximum number of rounds, the server declares stopping the training, and the newly generated global model is broadcasted to every client as the final model.

The cloud server is encouraged to minimize the aggregated global loss function $L(w_G^t)$, i.e.,

$$L(w_G^t) = \frac{1}{K}\sum_{k=1}^{K} L(w_k^t) \tag{2–7}$$

The above-listed explanation from (Step 1) to (Step 3) describes each round's entire framework, which is iterated until the global loss function 2–7 converges or the target training accuracy is obtained.

The FedAvg is the process that is aggregating the arrived local parameters from each client by averaging method on the server. The parameter averaging method is described below function (2–8).

$$w_G^t = \frac{1}{\sum_{k\in K} D_k w_k^t}\sum_{k=1}^{K} D_k w_k^t \tag{2–8}$$

Here, D_k is the regional dataset in the client k.

The above-described algorithm so far is the typical conventional FL framework. We utilize the method for the anomaly detection networking algorithm in the IoT circumstance. To heighten the insight of FL algorithm's development trend, the related researches are introduced as follows.

2.2.1.2 Federated Learning-based Anomaly Detection Techniques

According to the enormous benefit from the FL technique, it is adapted to the heterogeneous area to boost their performance and alleviate the transmission weight.

The authors in [47] suggest a privacy-preserving data sharing method on the basis of the federated learning combined with the permissioned blockchain consensus process for the industrial IoT (IIoT). The secure data sharing design is motivated by blockchain, and the privacy-preserved FL ensures data privacy reliabilities. They prove the proposed

model that improves the data sharing productivity and computing resource utilization and also it allows reliable data sharing.

Furthermore, Khan *et al.* [48] presents that resource optimization and incentive mechanism scheme of FL network edges. First of all, the fundamental design viewpoints for the edge of FL networks are presented, for example, learning algorithms, hardware-software co-designing, resource optimization, and incentive mechanism designing. Subsequently, the Stackelberg-game-based FL incentive mechanism is introduced. It allows the FL operators to fix the local repetition number to maximize their performance benefit.

In [49], the authors display an enhanced FL technology that federated averaging mechanism (FedAvg) utilized the distributed Adam optimization and compression of uploaded models. The proposed novel FedAvg model, which is called communication-efficient FedAvg (CE-FedAvg), is confirmed that it can get over the standard FedAvg model's general challenges; the sizeable round number to get converged in non-IDD local data set, and expensive transmission costs for each round. According to the experimental result, the CE-FedAVG converged by more than six times fewer rounds and updated more than three times fewer data than the conventional FedAvg experiment results.

Kim *et al.* [50] proposes collaborative ADS on the basis of FL technology, which is also one of our study's motivations. (refer to Figure 2–7) The proposed scheme utilizes the FL method based on the fact that it allows them to alleviate high bandwidth demand and relieve security threats in the IoT network. In the experiment, they clarify the proposed model FL-ADS's performance compared to the non-FL model, which does not aggregate the local result parameters and just learn respectively on each client. Consequently, the FL model performed better accuracy and slightly worse loss value than the comparison target model, due to the FL model comprises overall data in the network from each local client, but clients in non-FL trains only their local data.

2.2.2 Decentralized Collaborative Schemes

On the other hand, there are plentiful FL approaches to utilize in the decentralized network.

The RoboChain robot networking method [51], which ensures a decentralized, secure, and computationally powerful framework that comprises robot units in multiple fields is proposed. The proposed novel model adopts the blockchain and FL technology to intensify privacy security protection and moderate computational task burdens.

The approach by Weng *et al.* [52] develops the DeepChain system, restricting security problems in the FL system. The proposed scheme implements the blockchain-based value-driven incentive mechanism in order to guide participants to do correct behavior. Consequently, several virtues that data confidentiality, incentivized participant management, and computational auditability in FL are achieved.

Moreover, a reputation-based client selection algorithm applying a multi-weight subjective logic method [53] is proposed to obtain reliable FL in mobile networks. On the other hand, a consortium blockchain is run for a secure reputation administration based on the decentralized framework. The developed models are confirmed that they enhance the learning accuracy for secure FL protection significantly. The authors in [54] apply local differential privacy to counter adversaries that invade to snoop on sensitive private data in FL procedure. Besides, the blockchain-based scheme prevents malicious model updates by guaranteeing that general model updates are held responsible.

The BlockFL [55] is leveraged by the blockchain consensus mechanism. It allows us to operate on-device ML without the central server-based coordination and orchestration. The blocks in the system represent each training round, and the transactions between each block role the model transmissions from each client in that round. This novel technique allocates incentives to the devices containing more local training samples and spending more computational energy consumption, which means more devices are coordinated. Additionally, the local training results are manually verified, thereby enlarging the federation area to unreliable devices in the public networks.

From another approach, the BAFFLE [56] is proposed, which is the blockchain-based decentralized FL (aggregator free). This novel model utilizes blockchain technology to store and broadcast the global model and the Smart Contracts (SC) to perform the parameter aggregation procedure. Since the disappearance of the central cloud server, the existing challenges in the traditional FL that stability, fairness, and server security protection could become out of risk realm. On the other hand, the new challenge raises that the significant computational overhead in each round in case of too many clients because the computation and network transmissions are concentrated on each edge node.

In accordance with the above current studies' display, our research is motivated to build ADS to keep each device's characteristics and tolerance from IoT components' dynamicity from frequent in and out of the network. Simultaneously, we attempt to alleviate the data transmission volume and defend the device user's privacy information. Con-

sequently, we design two different federated learning-based anomaly detection systems (FL-ADS) on the basis of the decentralized network.

2.3 Chapter Summary

We overview the related works in the field of anomaly detection at the current stage. It explains and introduces various anomaly detection techniques and collaborative anomaly detection systems. In general, classification-based anomaly detection methods have excellent performance and good results. Many machine learning-based anomaly detection methods belong to classification anomaly detection methods. Furthermore, the various method-based collaborated ADS are also displayed. From the existing collaborative ADS approaches, we can realize FL has very crucial benefits, and the decentralized network designing is keep researched.

Since our goal is building anomaly detection systems in decentralized networking circumstance, we refer to the studies to enhance performance with our novel ideas. Therefore, two different parameter updating-based decentralized FL models are designed in this paper.

Chapter 3 Decentralized Federated Learning-based Anomaly Detection Schemes

According to the rapid growth of IoT devices and development, the number of transmissions and the volume of data is also exponentially increased. Based on the IoT spreads, security risk protection is also raised as a topic to develop to maintain a stable network condition. Many existing collaborative anomaly detection technologies basically from a centralized framework. However, the centralized management manner keep faces crucial challenges since the number of transmissions and the volume of data increase, and technology is getting advanced geometrically. The central server can get overall network organization's risk caused by a single point of failure, has limited scalability unless giving up cost-effectiveness, could become a lack of bandwidth by the inrush of simultaneous user requests. In order to overcome the challenges from the centralized networking method and reinforce the network condition, we redesign the conventional FL framework to the decentralized FL. Since we utilize FL-based decentralized networking to ADS, it can offer several significant advantages. The network gets management flexibility, advanced scalability, fault tolerance, minimized failure rate. Based on the crucial advantages for IoT networks, we develop two different decentralized FL algorithms; the synchronous parameter updating-based FL algorithm and asynchronous parameter updating-based FL algorithm.

3.1 Overall Framework

The decentralized FL's clients operate local anomaly detection training, the result parameter broadcasting, aggregating received parameters, and generating a new local model because the responsibilities from the server in traditional FL are transferred to edges due to the elimination of the server. In our synchronous model, each edge conducts the entire above mentioned main procedures simultaneously. On the other hand, only a set number of clients framework participate in broadcasting their local parameter in the asynchronous model. Figure 3–1 represents our proposed decentralized FL-ADS algorithm. The overall decentralized FL-ADS procedure is categorized step by step below;

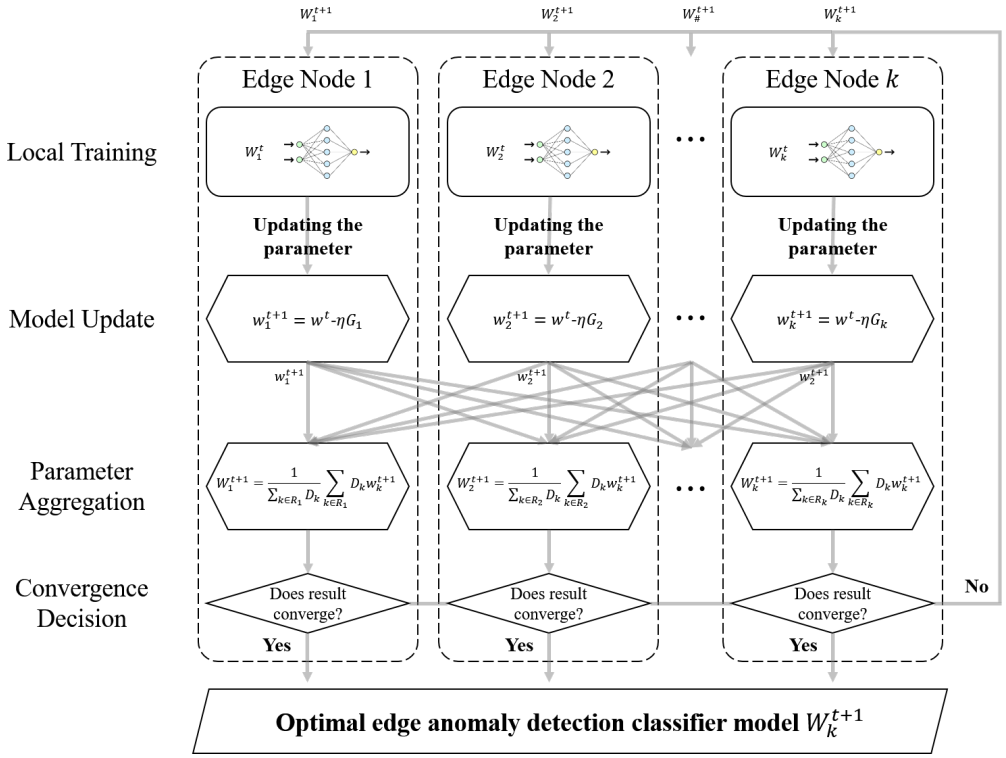


Figure 3–1 Decentralized Federated Learning Overview

- *Local Model Training*: At the beginning of the training, each edge gets pre-set for the parameters for the local neural network, such as hyper-parameters, learning rate, batch size, epoch, and so on. Furthermore, they conduct pre-processing for raw data optimization for the local DL step. Afterward, the local training is implemented, namely the MLP, or Stacked-NDAE. As the result of local training, the anomaly detection model is updated to w_k^{t+1}, which means weight w on the client k at the round t as the local updated model.
- *Parameter Broadcasting*: The two different models become diverged from the broadcasting step. In the synchronous model, every edge sends its local updated parameter to other connected clients when the local model update is done. On the flip side, a portion of clients distributes the locally weighted parameter to other connected clients in the asynchronous model.

  Thereafter, the received parameter is recorded on the receiver client k, and the sender client identifier is listed on the R_k. Thus, if it was the synchronous framework, all the connected clients' identifiers are included in the R_k. And only the connected broadcaster clients' identifiers and their own are contained to the R_k,

in the case of the asynchronous model.

- *Parameter Aggregation*: Based on the listed clients in the R_k, the client k organizes parameter aggregation by the averaging method to get the comprehensive anomaly detection model. As the output of the aggregation task, W_k^{t+1} is plotted, and the client k distinguish it is converged or not compared to the previous aggregated model W_k^t. The above procedure is denoted as one round, and it is keep iterated until the final decision is made to "converged."

Since every edge is fully-activated for the parameter broadcasting task, the performance tends to be the same as the conventional FL's performance while getting the above-mentioned advantages of decentralized frameworks. Nevertheless, it can suffer by transmissions over the limitation in the network and computational overhead on each client by the influx of tacks on the client, and there is more need to be flexible. In this idea, we are motivated to build another novel decentralized federated learning that asynchronous tasking-based FL, which is called "Asynchronous Decentralized Federated Learning." Since some clients do not participate in the parameter distribution, the network flexibility is greatly enhanced, and the network busyness is mitigated.

Before the detailed introduction of each decentralized framework, we present the local anomaly detection model.

3.2 Anomaly Detection Model Training

First of all, we introduce the local classifier models in our proposed algorithms. Figure 3–2 displays the overall local training framework from collecting raw data to generating new models. The local anomaly detection training models are interpreted as follows.

In recent years, the neural network model is regarded that it has excellent feature extraction capabilities and classification effects and makes better performance than other traditional anomaly detection methods [57]. In addition, the weight and bias parameters in the neural network can directly characterize the network structure and model values, and different models can be integrated through the interaction and fusion of model parameters. Therefore, this paper chooses two different neural network models for the edge anomaly classification model; MPL and Stacked-NDAE.

In an anomaly detection system based on model parameter transmission, each edge model's parameters need to be uploaded to the parameter aggregation nodes. Therefore,

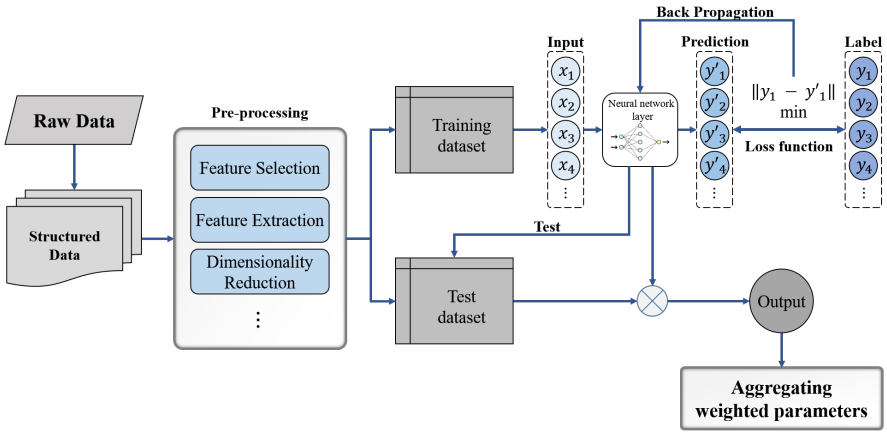


Figure 3–2 Workflow at the edge node.

when choosing a neural network classification model, not only the classification effect of the model must be considered, but also the structure and size of the model. The data format in this system is mainly traffic flow. Because the amount of a single piece of stream data is small, and the feature extraction of stream data is more straightforward than that of pictures, the multi-layer perceptron can accurately complete the anomaly detection of stream data [37]. It is suitable for the anomaly detection scenarios in this research, and the network structure of the multi-layer perceptron is more lightweight than other models such as CNN, DBN, and GAN networks. Therefore, this paper tries to apply a multi-layer perceptron to design the anomaly classification module of the collaborative anomaly detection system based on model parameter interaction in this chapter.

On the other hand, the stacked-nonsymmetric deep auto-encoder (Stacked-NDAE) is adopted since it is ensured that it makes a good performance in the classification task as the intrusion detection [38]. The NDAE has a different tasking structure with the ordinary (symmetric) AE that consists of only encoders instead of the decoder(s). This framework includes two NDAEs to amplify the classification effect from a single NDAE. It allows us to mitigate both computational overheads and time latency, with minimized performance trade-offs.

3.2.1 Activation Functions

Activation functions are utilized to determine the output signal from each neuron in neural networks. In other words, activation functions hand over the output of the hidden layer's each neuron to each neuron in the next hidden layer to tune the importance of the

node connections. In our utilized DL models, namely, the MLP and Stacked-NDAE, use activation functions to distinguish the output is related to model prediction or not. As the most popular activation functions, we introduce Sigmoid [58], ReLU (Rectified Linear Unit) [59], and ELU [60] functions (Figure 3–3);

- Sigmoid Function: Since this function's output range is only from 0 to 1, even though the massive data is input to the sigmoid function, it prints the value from 0 to 1. Thus it has the benefit to manage big sized data to handle-able. However, it also has a significant drawback, the possibility of the gradient vanishing problem.

$$f(z) = \frac{1}{1 + e^{-z}} \tag{3–1}$$

- ReLU: This is one of the most commonly used activation function. This function has a straightforward structure that outputs 0 for the values not greater than 0 and outputs the same as the input value for the values greater than 0. Therefore, it prevents the gradient vanishing problem, and it has a significantly light computational burden.

$$f(x) = max(0, x) \tag{3–2}$$

- ELU Function: In the ReLU function could occur the dying ReLU problem because it prints only 0 for the negative and 0 inputs. In order to avoid the dying problem, the ELU function is developed. This function contains ReLU's most benefits but takes the task cost of the exp function.

$$f(x) = \begin{cases} x, & if \quad x > 0 \\ a(e^x - 1), & otherwise \end{cases} \tag{3–3}$$

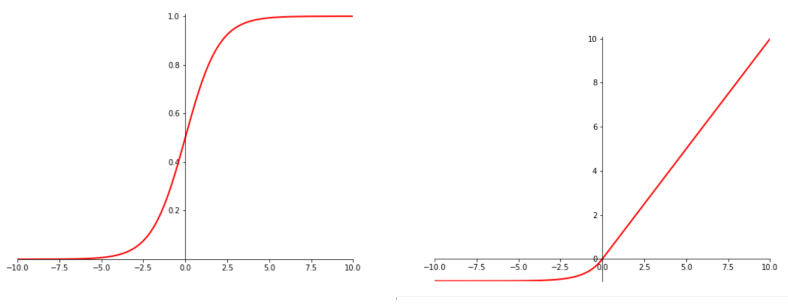


Figure 3–3 Applied Activation Functions in Our Scheme (Left: Sigmoid, Right: ELU)

3.2.2 Root Mean Square Propagation

After our anomaly classification neural network, we utilize the Root Mean Square Propagation (RMSProp)[61] as the optimization system after the neural network module. This algorithm mainly solves the problem that the parameter variation range is too extensive after the loss function is updated during the optimization process and accelerates the model's convergence speed. The RMSProp algorithm uses a differential square weighted average for the gradients of weights W and offsets b.

$$s_{dw} = \beta s_{dw} + (1-\beta) dW^2 \tag{3–4}$$

$$s_{db} = \beta s_{db} + (1-\beta) db^2 \tag{3–5}$$

$$W = W - \alpha \frac{dW}{\sqrt{s_{dw}} + \varepsilon} \tag{3–6}$$

$$b = b - \alpha \frac{db}{\sqrt{s_{db}} + \varepsilon} \tag{3–7}$$

Where, s_{dw} and s_{db} are respectively, the gradient momentum accumulated by the loss function during the last iteration. β is the gradient accumulation index. The RMSProp optimization algorithm calculates the differential square weighted average of the gradients, which is helpful to eliminate the update volume in the direction of the large width of the wave during the optimization process and can correct the wave so that the width of the wave in all dimensions are comparatively small. This way can raise the speed of the function convergence. In order to avoid the denominator being zero, we adopt the minimum value ε for smoothing the function.

3.2.3 Multi-layer Perceptron

Figure 3–4 displays an overview of our MLP application. Our designed MLP module includes three hidden layers with an input layer and an output layer. An activation function follows each hidden layer, the sigmoid function tags along with the first hidden layer, and the ELU function follows the second and last hidden layers. Subsequently, the initial couple of hidden layers are followed by a dropout layer, respectively, to omit a fixed number of neurons to mitigate the overfitting problem.

The sigmoid function is allocated in the first hidden layer because of the fact that it allows us to prevent the gradient explosion problem, and the ELU function is activated in

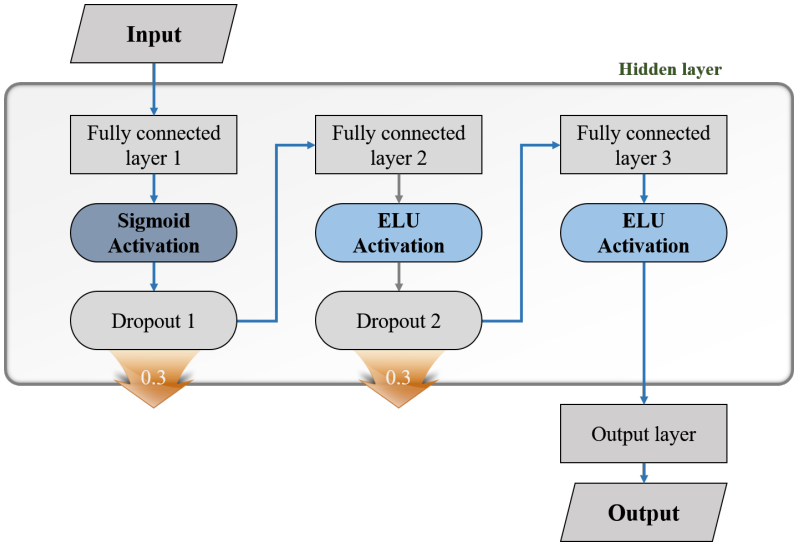


Figure 3–4 multi-layer perceptron anomaly classifier on each edge

the last two hidden layers due to it is profitable for protecting the critical data features more perfectly. Each activation function handovers the results to the next layer. Ultimately, the prediction of the labels between normal data and outliers is organized by the multiple hidden layers' procedures in the output layer, under the activation function in the output layer is selected to the sigmoid function to generate the result to the binary classification. Furthermore, sixteen neurons are deployed on the first hidden layer, and eight neurons are set to the second and third hidden layers.

The Sigmoid function maps the input to the range of [0, 1]. Since the input sample label is binary label 0, 1 during the training process of this system, the output of the hidden layer will also be projected as a binary classification result of 0,1 after sigmoid mapping. Based on the results of this classification, this scheme conducts the subsequent model evaluation.

To finalize the neural network procedure, the RMSProp method is deployed as an optimization system. It can eliminate the update volume in the direction of the large width of the wave during the optimization process and correct the wave so that the wave width in all dimensions becomes comparatively small. Consequently, it can raise the speed of the function convergence.

3.2.4 Stacked-asymmetric Deep Autoencoder

As another anomaly classifier method, we adopt the Stacked-NDAE state-of-art neural network technique. This framework structure consists of twin NDAEs that is devised from traditional symmetric AE to the nonsymmetric procedure. The traditional AE contains both encoder and decoder parts to extract features and backpropagate them. On the other hand, the NDAE includes only encoder parts instead of the decoder procedure to enhance the classification performance. Since the Stacked-NDAE contains a double NDAE, the classification ability became enlarged. The detailed interpretation of the Stacked-NDAE is displayed in Subsection 2.1.2.2.

According to the Stacked-NDAE includes two same structured NDAEs, the number of layers and the contained number of neurons are identical to each other (Figure 2–6). In our utilization of Stacked-NDAE in the decentralized FL-ADS, the first hidden layer contains 14 number of neurons, and the second and third hidden layers include 28 neurons, and only sigmoid function is applied on each hidden layer based on the setting in the paper [38].

After the Stacked-NDAE neural network procedure, the optimization follows. From the output layer in Stacked-NDAE, the output data is inserted into the optimizer operator. The RMSProp method is adopted the same as MLP (Subsection 3.2.3).

3.3 Decentralized Federated Learning Design

Figure 2–7 in Subsection 2.2.1.1 represents the traditional FL, which is on the centralized framework basis. However, devices in the IoT networks conduct intercommunication not only with the cloud server but also with other devices directly. Moreover, IoT networks have heterogeneous condition status, such as monotonous environment (e.g., office, warehouse; occurs less in and out of the network), and dynamic environment (e.g., airport, highway service station; occurs often in and out of the network). The decentralized FL structure is the server eliminated format. We design two different decentralized formats; basically, the synchronous decentralized processing-based FL-ADS, which should be more beneficial in the monotonous IoT conditions, and the asynchronous decentralized processing-based FL-ADS, which more should be suited for the dynamic IoT conditions.

Based on the explained classifier technologies above, the MLP and Stacked-NDAE, we design the anomaly detection networking system using the decentralized FL algo-

rithms synchronous and asynchronous. The reason why we use the novel FL framework is that the FL method offers very significant virtues for IoT networks, such as privacy protection, network bandwidth effectiveness, and low latency.

Our proposed schemes can lighten power consumption, computational processing overheads, and server management cost. In the decentralized FL process, each client operates the local training and distributes the local training result parameter to other clients for generating each local's new model. When the parameter aggregation is done in a local, the created new model becomes the next round's local parameter. The mentioned processes above represent the basic idea of a round in the decentralized FL.

3.3.1 Locally Parameter Processing Algorithm

In this subsection, we describe the plotted parameters by local training. In our proposed model, the specific deep neural network is organized to classify the anomalies in the network by processed data. In our case, we utilize the MLP and Stacked-NDAE. Each client prints the result of the local DL as a weighted parameter and performance evaluation results. Basically, the loss function and weighted model is printed from the local training. Thus, we focus on them to explain below.

The loss function is a very important thing to evaluate the model's precision and be the elements for deciding system convergence. In the local client, based on the assumption that there are the number n of total samples, (x_i, y_i) denotes the ith samples, and w is the model weighted parameter. Each sample's loss function is represented to $f_i(w) = l(x_i; y_i; w)$. Therefore the summarized loss function for all samples becomes as shown in function (3–8).

$$f(w) = \frac{1}{n} \sum_{i=1}^{n} f_i(w). \tag{3–8}$$

Suppose the edge k's sample set represents L_k, and n_k is the number of samples in the kth client. Hence the kth client's average loss function can be shown as the function (3–9). At the end of every round's local training, the clients compute their own loss value $h_k(w)$ to evaluate the model's level of completeness. As the round goes, the loss function converges to a particular point.

$$h_k(w) = \frac{1}{n_k} \sum_{i \in L_k} f_i(w) \tag{3–9}$$

The weighted parameter is updated as per each after the local training. The local parameter represents the local client's optimized ML model. The algorithm to update the weighted parameter is presented below.

In round t, each edge initially updates its local ADS weighted model and plots the average gradient as below function (3–10):

$$G_k^t = \bigtriangledown h_k(w_t). \tag{3–10}$$

Consequently, the new weighted parameter of the kth edge is reorganized by the gradient algorithm as below:

$$w_k^{t+1} = w_k^t - \eta G_k^t, \tag{3–11}$$

Here, η denotes the learning rate of the anomaly detection modeling. The newly updated local parameter is transmitted to each other to aggregates the other clients' parameters.

After printing the local loss function and updating the locally weighted parameter, each client distributes the results to other clients. When the client receives the parameters from other connected clients, it aggregates the received values by the averaging method. The detailed decentralized synchronous networking algorithm is explained in the next subsection.

3.3.2 Synchronous Parameter Broadcasting Scheme

The contrast between synchronous and asynchronous FL algorithms is mainly this parameter networking mechanism, which broadcasts and aggregates training result parameters.

The synchronous inter-communication model is the scheme that every client learns the anomaly classification model, broadcasts parameters, and creates a local anomaly detection model simultaneously. From Figure 3–5, we can see every client participate in the parameter broadcasting, according to the fact that the red colored point on each client's vertical line denotes that it is an active participant.

Due to every client receives the same number of and the same value of weighted parameters, the generated global model (Local models from each client is the same so far) calculation function becomes very similar to the traditional FL's global model function. Therefore we can assume the synchronous FL-ADS can perform the same as traditional

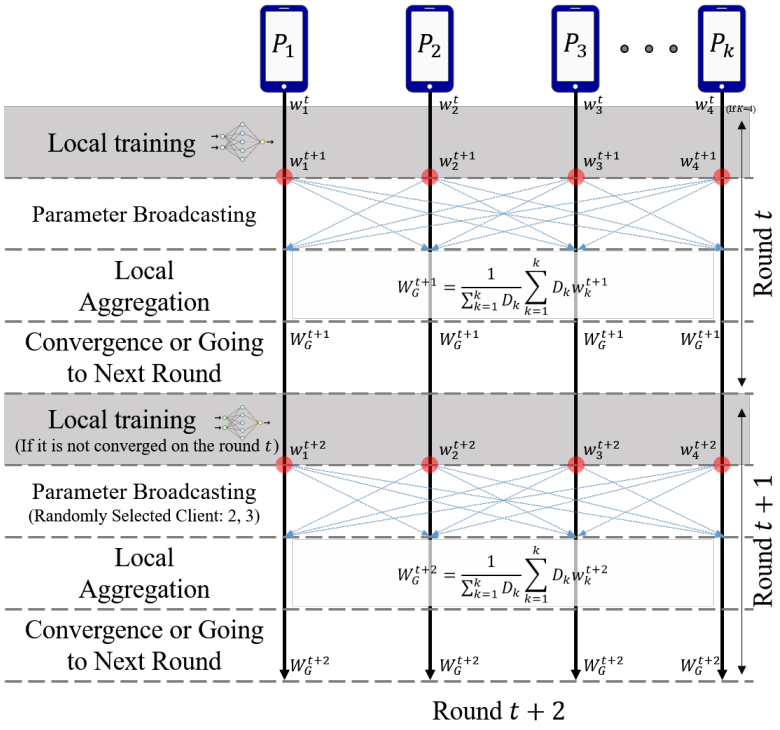


Figure 3–5 The Anomaly Detection System Based on Synchronous Decentralized Federated Learning Framework.

FL-ADS, which also aggregates every edge's weight on the cloud server. The detailed mechanism of the synchronous model is presented as follows.

From the beginning of each round, each client in the network operates local training on the basis of the local data using a specific deep learning model, such as MLP, or Stacked-NDAE in our study. Subsequently, based on the presented method in previous subsection 3.3.1, each client updates a weighted parameter w_k^t to w_k^{t+1}, and distributes that to other clients. Thereafter, each client averages the arrived local parameters, and it's own by the FedAvg process as follows;

$$W_G^{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} w_k^{t+1} \tag{3–12}$$

Based on the above function 3–12, each client creates a new local aggregated model W_G^{t+1}, which can also be the global model by parameter aggregation.

Consequently, the generated new model W_G^{t+1} is adopted for the next round's training model. Due to the newly generated local models being the same each other, the G is marked as the subscript of W_G^t that globally comprehended in the network. On the basis

of the decision that the entire model is converged, they stop running to the next round, and W_G^t is declared as the final semi-permanent abnormal data classifier model. Finally, each client tests the final model W_G^{t+1} with their local data if the aggregated loss function is converged. Otherwise, if the aggregated result is not converged yet, they keep stepping to the next round to make the concrete anomaly detection model until pre-designed model convergence conditions are satisfied. The convergence satisfaction criteria are introduced in Section 4.1.4.

3.3.3 Asynchronous Parameter Broadcasting Scheme

In this section, we introduce the most significant difference from the previously proposed synchronous networking algorithm. The weighted parameter inter-communication is operated simultaneously from every client in the synchronous method. On the other hand, only an arranged number of clients broadcasts the local training parameter to others by randomly selecting broadcaster clients each round. Since the number of broadcaster number is lessened, we can expect the effect that the degree of communication congestion is significantly diminished, and the flexibility is remarkably enhanced. Because of the fact that the number of IoT devices are very soaring, we have to arrange the method to manage in the case of very crowded device condition. The detailed algorithm description follows below.

Figure 3–6 shows an example of the asynchronous model on the basis that only two clients are selected as the round's broadcasters in four clients.

At the beginning of the asynchronous decentralized FL-ADS process, each client prepares local training by conducting data optimization and pre-processing procedures. Based on the well-prepared bases, such as optimized local raw data, hyper-parameters, learning rate, and target applications, each local edge can run local training focused on anomaly classification.

As the output of the local training, the weight parameter w_k^t is updated to w_k^{t+1}. Thereafter, randomly selected only a part of clients distributes the result parameter w_k^{t+1} to every client; only two clients are broadcasting on each round as shown that only two red points are allocated to P_1 and P_4 ($K = 4$) at the Round t, and P_2 and P_3 at the Round $t + 1$.

We denote R_k^t as a list of clients that contains the origin clients of the received parameters at the client k. According to the case of Figure 3–6, the R_1^t contains P_1 and

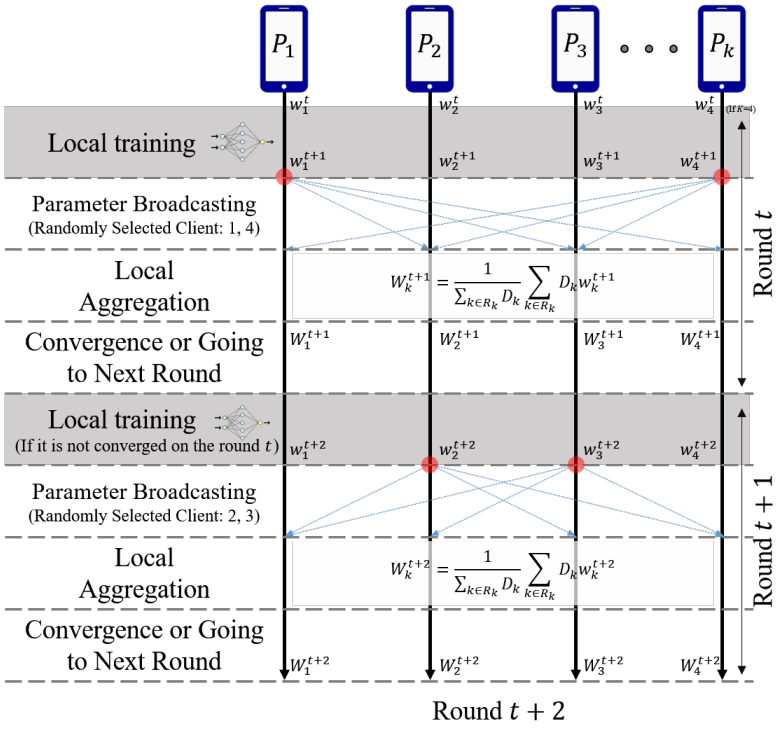


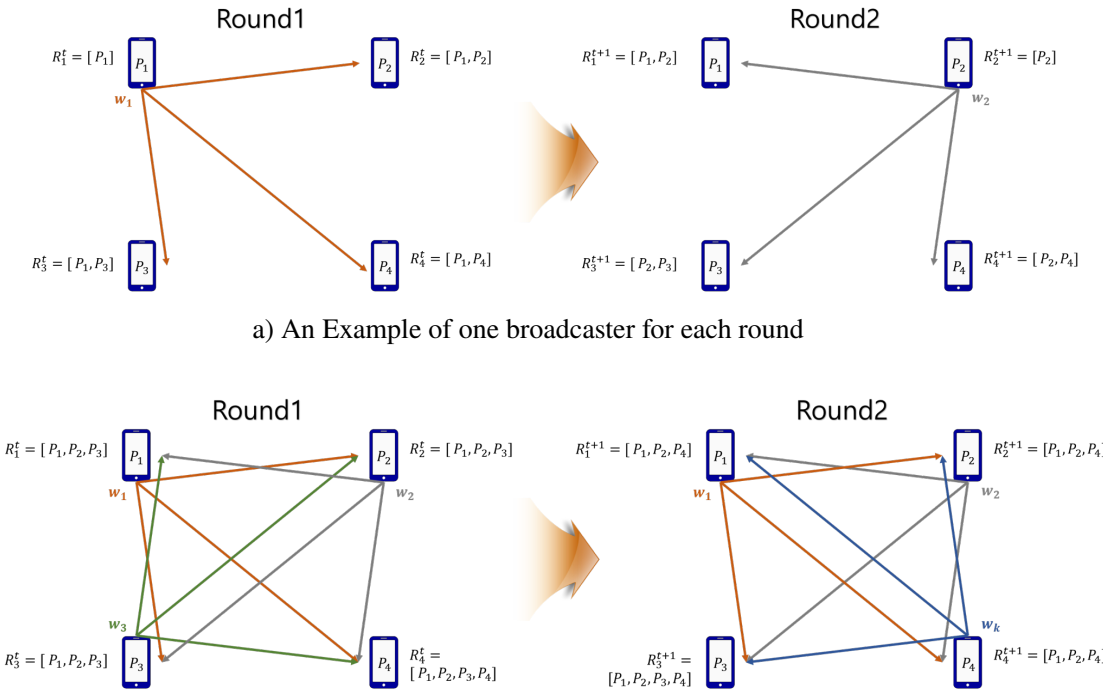
Figure 3–6 The Anomaly Detection System Based on Asynchronous Decentralized Federated Learning Framework.

P_4 since the client P_1 got the w_1^t by it's local training itself, and w_4^t by P_4's parameter transmission at the round t. On the other hand, the R_2^t contains P_1, P_2, and P_4 because the client P_2 got the w_2^t by it's local training itself, and w_1^t and w_4^t by the broadcasters' parameter transmission at the round t. In other words, when the client k is a broadcaster client, the R_k^t includes the broadcaster clients' weight parameters. If the client k is not a broadcaster, the R_k^t holds one more client than when it is a broadcaster client because it receives the parameters by broadcaster's transmission and its own local training.

The diverged $R_k^t(k = 1, ..., K)$ among the participant clients cause divergence of the local aggregated model, unlike the equalized local models on each client in the synchronous system. The local aggregated weight parameter on client k at round t is shown as the Function (3–13).

$$W_k^{t+1} = \sum_{k \in R_k^t} \frac{n_k}{n} w_k^{t+1} \tag{3–13}$$

Compared to equation (3–12), equation (3–13) has different summarization method. (3–12) summarizes the weight parameters from every participant, but (3–13) aggregates



a) An Example of one broadcaster for each round

b) An Example of three broadcasters for each round

Figure 3–7 Overview of Model Comprehensiveness Comparison Between the Cases of Fewer Broadcaster and More Broadcaster among Four Clients

the weight parameters from a part of participants which are contained weight's generator clients as explained above ($= R_k^t$). Even though locally aggregated models diverge during the asynchronous FL procedure, the models become close to each other as the round iteration goes on.

From the perspective of the number of selected broadcasters in each round, we can assume fewer broadcasters should worsen the effect by parameter aggregation, and it may occur more critical performance reduction. However, It should significantly decrease the communication load, such as the number of transmissions, bandwidth demand, communication interference, bottle-neck. On the other hand, more broadcasters can enhance the effect of the parameter aggregation, and it affects generating high performance. However, as the number of broadcasters increases, the transmission load in networking is also very raised.

Figure 3–7 shows the model comprehensiveness difference between the above two different cases; one broadcaster within four clients and three broadcasters within four clients. Even though the above full-mesh topology has less majority than multi-hop topol-

ogy, we display it to make it easier to get the idea.

From comparing the networking status, we can clarify how transmissions are massive depending on the broadcaster number setting. In Figure 3–7a, only three transmissions are operated in one round. So we can realize possible problems by the massive network can be prevented by the low number of broadcaster settings. However, the number of elements in R_k^t is very low; we can assume that the model enhancement by model aggregation is not desirable.

On the other hand, the network 3–7b has many parameter inter-transmissions compared to the network 3–7a. Thus each R_k^t in each client contains more elements by receiving more parameters. Therefore, we can expect significant performance enhancement through comprehensive networking. However, some trade-offs by the massive network can follow at the same time. Therefore, the number of broadcaster setting should be decided by the professional network designer's environmental, technical analysis.

3.4 Chapter Summary

In this chapter, we display a synchronous and asynchronous parameter updating-based federated learning mechanism for anomaly detection tasks in IoT networks.

As the anomaly classifier models, we apply the MLP and Stacked-NDAE deep neural network methods in our system. They enable us to train the complex classification tasks enforced by multiple hidden layers.

From the reformation of the traditional FL framework by decentralization, the server is eliminated, and each client inter-communicates with each other for parameter delivering. Server elimination occurs several benefits that network area scalability, management flexibility, fault tolerance, and the loss failure rate. These virtues should be very crucial for the adoption of IoT networks.

In the synchronous decentralized model, every client transmits the local training result parameter in the same round. Therefore, the result parameters are fully aggregated. It causes the generating of the most comprehensive anomaly detection model.

On the other hand, only a portion of the client broadcast their local parameters to their connected clients. As the number of activated clients lessen, the aggregated model's comprehensiveness becomes less effective. However, it enhances the network's flexibility.

Chapter 4 Experimental Results

In this chapter, we implement our proposed novel decentralized FL-ADS, namely the synchronous decentralized and asynchronous decentralized FL-ADS. In order to compare the influence of the neural network models, we adopt two cutting-edge neural network technologies that multi-layer perceptron (MLP) and Stacked-Nonsymmetric Deep AutoEncoder (Stacked-NDAE) to each suggested decentralized FL model as the anomaly classifier. Firstly the straightforward performance comparison is organized by ten times experiments. Subsequently, the FL network convergence trend investigation is implemented in the environments of full-mesh and multi-hop topologies.

4.1 Experimental Settings

First of all, we clarify our experimental environment in this section. Table 4–1 represents the hardware environment in which our experiments are implemented.

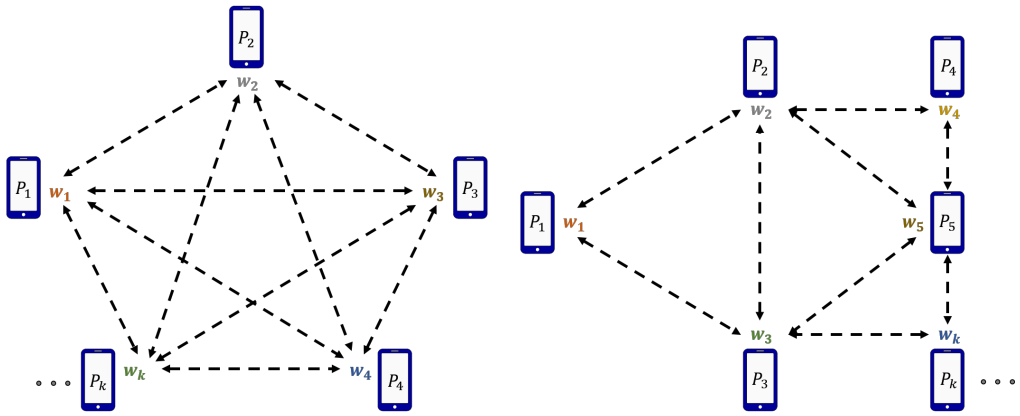
Table 4–1 Hardware Environment

Standard	**Hardware**
OS	Windows 10 Home 64bit
CPU	Intel Core i5-4210U 1.70 GHz
RAM	8.00 GB
Hard Disk	912 GB

In order to incarnate edge to edge transmissions in the decentralized networks, we utilize *Socket*, which allows us to conduct communication between programs. The *Socket* function is organized by IP address and port number of each edge to recognize and direct at the communication target edges. During the communication procedure, the data sender updates the messages to the host's Socket, and the Socket transmits to the destination edge by peer-to-peer inter-communication (P2P) interface. After the receiver gets the data by the Socket procedure, it can conduct the next step for the local procedure. Furthermore, *Keras* library is used as a software supporter for the neural network anomaly detection classifiers at the edge, namely MLP and Stacked-NDAE. *Keras* is a deep learning neural network library based on *Python* programming language. This platform is known as

having very advantageous characteristics, which are consistent with our research experiments' requirements;

- The *Keras* platform is highly modular. In the neural network anomaly detection model in our proposed models, the neural network layer, optimizer, initialization method, activation function, and other modules can be incarnated through Keras to debug and obtain the optimal model.
- The *Keras* platform is easy to expand. In the debugging of the neural network classifier, the network model structure can be easily adjusted, and the number of network layers and the number of neurons in the layer can be modified and optimized.



a) Full-Mesh Topology b) Multi-Hop Topology

Figure 4–1 Decentralized Inter-communication Framework in Each Network Topologies

Furthermore, our research focuses on utilizing the proposed systems in real-world networks. Thus, we are aware of two different network topologies that the fully connected mesh (Full-Mesh; Figure 4–1a) and multi-hop (Figure 4–1b) networks. The full-mesh is the network topology that every edge is connected to every other. This structured topology is very costly to execute and occurs too much excess redundancy burdens. Thus, it is generally employed in the backbone network. On the flip side, in the wireless multi-hop topology, each node communicates directly only to the nodes in the arranged transmission range. Since it succeeds in interference mitigation, spectrum reuse optimization, load balancing, radio range expansion, and requested energy minimization, this topology

is very popular to design the wireless network in the real-world. The suggested model utilizations in these networks are described in the below Section 4.2 and Section 4.3.

4.1.1 Experimental Setup

We verify the performance and the characteristics of the proposed schemes in this section by organizing demonstration experiments based on the python programming language, as explained above. First of all, to compare the performance between traditional FL-ADS and proposed two decentralized FL-ADSs, we organize the performance comparison demonstration based on our specific convergence condition (Function 4–7). Subsequently, the training comparison experiment until reaching to 50th round is organized to analyze the target models' convergence trends between full meshed network and multi-hop network backgrounds.

Table 4–2 Experimental Setup Parameters

Parameter	**Value**
Number of Local Epochs	60, 75, 110
Batch Size	1,000, 2,000
Number of Clients in Full-Mesh	4
Number of Clients in Multi-Hop	6
Total data set Size	10,000
Convergence Threshold (δ)	0.02
Max Round Limit	50
Data Set Setting	6 : 1 : 3 (training : validation : test)

The particular settings are summarized in Table 4–2. The numbers of local epochs and the batch size are specified based on the uncountable implements of simulation to find the proper values. Finally, we found out the best three sets, namely 110 and 2,000, 75 and 1,000, and 60 and 1,000, respectively. Moreover, the applied number of clients are four in the full-mesh network and six clients in multi-hop network experiments.

4.1.2 Data set

In this project, the main experiment goal is anomaly detection based on network transmission flow data. In machine learning tasks, network traffic data sets with low

redundancy, and less repeated data are selected as much as possible to simulate actual application scenarios. To demonstrate a real-world network, we apply NSL-KDD open packet data set[62] for the experiments. This NSL-KDD is an improved data set from KDD-CUP 99 data set to make it more suitable for anomaly detection tasks in machine learning. This NSL-KDD data set consists of a set of internet packet records, which comprises 22 anomaly attacks and 41 attributes. This data set contains 125,973 total records divided into 67,343 normal and 58,630 abnormal records (It contains 45,927 pieces of DDOS attack data). Table 4–3 displays overall features and their categories in the NSL-KDD data set. The features in NSL-KDD are categorized into four parts [63]; from the feature number (F#) 1 to 9 are "*basic features*", 10 to 22 are "*content features*", 23 to 31 are "*time-based network traffic statistics features*", and 32 to 41 are "*host-based network traffic statistics features*."

Table 4–3 NSL-KDD Data set Feature list;

F#	Feature Name	F#	Feature Name	F#	Feature Name
1	duration	15	Su_attempted	29	Same_srv_rate
2	Protocol_type	16	Num_root	30	Diff_srv_rate
3	Service	17	Num_file_creations	31	Srv_diff_host_rate
4	Flag	18	Num_roNum_shellsot	32	Dst_host_count
5	Src_bytes	19	Num_access_files	33	Dst_host_srv_count
6	Dst_bytes	20	Num_outbound_cmds	34	Dst_host_same_srv_rate
7	Land	21	Is_hot_login	35	Dst_host_diff_srv_rate
8	Wrong_fragment	22	Is_guest_login	36	Dst_host_same_src_port_rate
9	Urgent	23	Count	37	Dst_host_srv_diff_host_rate
10	Hot	24	Srv_count	38	Dst_host_serror_rate
11	Num_failed_logins	25	Serror_rate	39	Dst_host_srv_serror_rate
12	Logged_in	26	Srv_serror_rate	40	Dst_host_rerror_rate
13	Num_compromised	27	Rerror_rate	41	Dst_host_srv_rerror_rate
14	Root_shell	28	Srv_rerror_rate		

4.1.3 Data Pre-Processing

Different manufacturers and data formats have caused serious isomerization of data sources due to the very heterogeneous IoT devices and services. It is not easy to obtain a set of comprehensive pre-processing methods to generate a blended and standardized data set. Data pre-processing is a necessary task since the collected raw data cannot be used

directly in a DL model. Each pre-processing must be independently designed according to different application scenarios. In the anomaly detection system based on data interaction, the data is formatted and transformed into the same standard text. The text data and text sequence are then converted into multi-dimensional word vectors based on bytes, and feature extraction is organized. The pre-processed structured data set is expressed as a determinant elements' matrix. Each row represents an individual data in the data set with summarizing the information in the data, and each column of the matrix corresponds to a particular feature of the data symbols (characteristics). Therefore, a data set summarizes the operations of specific data generation mechanisms. Also, a data set represents the distribution of the generated data samples based on a statistical perspective's insight. Then a machine learning method is used to model the data set. By performing operations such as feature selection, feature extraction, and dimensionality reduction, the data set is redesigned to a form that is easier for the machine learning system to understand and utilize the data. How to expose features in data is a crucial step to improve machine learning performance.

In our proposed scheme, we operate the DL procedure, but the harvested raw data is not able to be input to DL directly. Therefore, the raw data needs specific and intricate procedures to be optimized for DL. Since many techniques are adopted in our system's pre-processing, we briefly preview the methods in this subsection.

4.1.3.1 Vectorization: One-hot Encoding

A variety of machine learning models have pattern learning capabilities, such as the SVM model, n-gram model, neural network model. Most ML models still need to convert data samples into vectors to make them easy to learn. Therefore, word vectorization is still needed for the text data.

The one-hot code conversion uses the idea of enumerating variables. The dimension of the vector is determined by the number of features in the data. After that, the byte index position has a value of 1, and others have a value of 0. Through this type of conversion, the text sequence can be converted into a fixed-digit vector sequence. An example of one-hot encoding follows below;

The data from Table 4–4 can be converted to Table 4–5 by one-hot encoding method. The '**protocol_type**' and '**flag**' features are diverged to '**protocol_type_tcp**' and '**protocol_type_udp**', and '**flag_S0**' and '**flag_SF**'. And the samples get a digit 1 in

Table 4–4 An Example of Data Set before One-hot Encoding Procedure

Duration	**protocol_type**	**flag**
0	tcp	SF
0	udp	SF
0	tcp	S0

valid column otherwise 0.

Table 4–5 An Example of Data Set after One-hot Encoding Procedure

Duration	**protocol_type_tcp**	**protocol_type_udp**	**flag_S0**	**flag_SF**
0	1	0	0	1
0	0	1	0	1
0	1	0	1	0

It can be shown as the above example that one-hot encoding can vectorize text word data. Moreover, the properties of the data set remain unchanged after one-hot encoding, but the number of features increases.

In order to adopt the raw (text) data to our DL, it needs to be vectorized. Our project adopts the one-hot encoding conversion method for the vectorization task from text data to vector data. The one-hot encoding makes the properties from the data set to remain unchanged but makes more features. For example, the NSL-KDD data set's feature dimension becomes 41 to 118 by the one-hot encoding in our experiments.

4.1.3.2 Label Binarization

The data label binarization is conducted in a similar way to the one-hot encoding. The normal data is marked as number 1, and any types of anomalies are labeled to the number 0. On this basis, the follow-up anomaly detection and classification task is carried out. Based on the Subsection, the labels are binarized to 0 or 1; the normal data is labeled to the number 0, and the anomaly data is labeled to the number 1.

4.1.3.3 Standardization: Z-score

The Z-score algorithm is one of the most commonly used standardization methods. Before entering the data into the neural network, the data needs to be standardized. This is because:

(1) The numerical range of the input data will affect the initialization effect of the neural network model. The standardization of data can ensure to a greater extent that the initialization of each neuron is within the effective range.

(2) The range of the gradient can be regulated. If the data is not standardized, it may cause the gradient of the propagation calculation to be too large or too small, thereby affecting the model effect and detection accuracy.

(3) It can make the learning rate more stable. After the data is standardized, the gradient size is relatively stable, and the learning rate changes according to the gradient so that the standardization can obtain a more stable learning rate.

(4) It can avoid the problem of model bias caused by too much model weight discrepancy. In the neural network, the interval between different weights is too large, which leads to the model's training bias to update larger weights, thereby affecting the final training effect. The standardized data can avoid the situation of excessive weight difference.

The data standardization is organized by using the Z-score algorithm. Before entering the data into the neural network, the data needs to be standardized because it helps the system to train faster and avoid the possibility of occurring local minima problems. Standardization procedure is a method of data processing based on statistics. Data standardization aims to eliminate the influence of different data ranges on the classifier by adjusting the data values of different columns to a fixed range in the same data set. After the data set is standardized, the numerical difference between different features is still proportional to the original data set. Therefore, in theory, the data set standardization process can maintain the original data set's feature relevance. In machine learning tasks, the data set values need to be adjusted so that the influence of outliers on the classifier's training results can be eliminated to a certain extent. The standardized processing method of Z-score is shown in formula 4–1;

$$Z_x = \frac{X_i - \bar{X}}{\bar{S}_x}, \tag{4–1}$$

Where, $\bar{X}$ and $\bar{S}_x$ are the mean value and standard deviation of the original data set, respectively.

4.1.3.4 Feature Selection and Extraction

The data type in this system is stream data, and its features contain network transmission protocol, transmission time correlation, number of stream data transmitted per hour, number of packets of each stream data, the average number of bytes contained in each packet, and the average number of bytes transferred per a second. According to different attack models, we need to operate extraction and process with different features to detect different types of attacks' anomalies. For example, in detecting Distributed Denial of Service (DDoS) attacks, the number and duration of request packets sent by the terminal per unit time are used as essential data features to determine the type of attack. For another example, in the detection of botnet attacks, the features of the number of stream data transmitted per hour, the number of packets of each stream data, the average number of bytes contained in each packet, and the average number of bytes transmitted per second are regarded as essential data. Features are extracted and analyzed, and a large number of terminals with similar behaviors are found through clustering and correlation analysis. Therefore, the selection and extraction of features have a crucial influence on the detection results. This project obtains a formatted data set by eliminating the original data's redundancy and normalizing the text. It is organized by analyzing the correlation between every two features. On this basis, the statistical modeling of sensitive data within the threshold time limit is carried out to obtain its operating features and formats and classify and locate abnormal information.

4.1.3.5 Dimensional Reduction: Principal Component Analysis [64]

This project uses the principal component analysis (PCA) algorithm to reduce the data's dimensionality. The original data may contain many redundant data and useless features, causing much waste of computing resources. Therefore, the data set should be reduced in dimensionality to remove redundant information in the data processing. The PCA dimensionality reduction method addresses n-dimensional features to k-dimensional ($k < n$) and construct a new k-dimensional feature containing the most crucial information. This new k-dimensional feature is called the principal component. Based on the PCA dimensionality reduction procedure in pre-processing, it is possible to exclude redundant information and lessen the amount of data while preserving vital data in the data set. Thereby we can save the computational cost of machine learning.

Assuming a given data input matrix is $M \in \mathbb{R}_{m \times n}$, then M can be written as:

$$M = U \sum V^T \tag{4–2}$$

Where $U \in \mathbb{R}_{m\times n}$ is a unitary matrix, $\sum \in \mathbb{R}_{m\times n}$ is a positive diagonal matrix,the value on the diagonal is the singular value of the M matrix, and $V^T \in \mathbb{R}_{m\times n}$ is also a unitary matrix. According to the singular value of M, the M matrix in the lowdimensional space can be reflected to $\widehat{M}$

$$\widehat{M} = \widehat{\sum} V^T, \tag{4–3}$$

Based on the $\widehat{M}$ calculation, it is assumed that a k-dimensional matrix is obtained after dimensionality reduction, the $m-k$ number of smallest singular values in $\sum$ becomes to 0, then the k number of largest eigenvalues can be retained, and the most critical features of k dimensions can be obtained. Finally, the matrix $\widehat{M}$ retains some of the most critical information from the original matrix. In this way, it is possible to exclude redundant information and reduce the amount of data while retaining essential data in the data set.

Before the experiment starts, the original data set must be pre-processed first, and the pre-processing of the data set is explained in detail in Subsection 4.1.3. The pre-processing is divided into data optimization (data vectorization, binarization, data standardization), feature selection, feature extraction, and dimensionality reduction. Through the procedures, the clients can get ready to organize the deep learning step, as shown in Figure 3–2. The comparison targets are accuracy and loss value by the final test set evaluation and the number of rounds to get converged. Moreover, to generalize the experiment result, we iterate the experiment ten times for each epoch/batch size setting. Thus, we implement 30 times for each different FL framework; a total of 90 times experiments are organized as final.

4.1.3.6 Data set Division

According to the awareness of underfitting and overfitting evasions, the total data set is divided into three subsets; 0.6 of the training set, 0.3 of the validation set, and 0.1 of the test set over the total data set 1.0. To demonstrate the realistic scenario, we try to allocate the data as diverse as possible. Thus, we adopt random sampling to obtain the non-independent and identically distributed (non-IDD) data. Each round starts with the local training with the training set. Through the weights by the training set learning, the

validation test is organized to verify the model's effectiveness in another data set basis. When the training loss function gets converged, the generated model is operated on each edge's test set to evaluate the model performance as final.

4.1.4 Performance Evaluation Methods

In our decentralized FL-ADS model, we categorize the entire data set into three subcategories; training data set, validation data set, and test data set. At the end of each round, each client evaluates the aggregated model's performance in the training set and validation set.

At each end of the round, the client aggregates the transmitted other client's local model, the loss value, accuracy value, and others. The loss values from the categorized data set give us the clues to decide the model converges and determine the model's effectiveness. And the model's training accuracy, verification accuracy, and test accuracy can straightforwardly reflect the performance of the model.

Our research chooses Binary Cross Entropy as the loss function of the model. This is a particular case of Softmax Cross Entropy. The expression of binary cross entropy is:

$$loss = -\sum_{i=1}^{n} \hat{y}_i \ \log y_i + (1 - \hat{y}_i) \ \log(1 - \hat{y}_i) \tag{4–4}$$

$$\frac{\partial \ loss}{\partial \ y} = -\sum_{i=1}^{n} \frac{\hat{y}_i}{y_i} - \frac{1 - \hat{y}_i}{1 - y_i} \tag{4–5}$$

Only when y_i and $\hat{y}_i$ are equal, the loss value is 0, and the more significant the interval between the value of y_i and $\hat{y}_i$ the greater the loss value. Therefore, binary cross-entropy measures the proportion of the difference between the model's classification result and the correct result. Therefore, by calculating the size of the loss, the performance of the model can also be evaluated. When the model training loss or test loss meets the upload threshold, each client broadcasts the edge model parameters for the local parameter aggregation.

The aggregated loss function can be explained as the function below (4–6).

$$aggr\_loss^t = \sum_{k=1}^{K} \frac{n_k}{n} G_k(t) \tag{4–6}$$

Through the above function, the model can print aggregated train loss ($aggr\_train\_loss^t$), validation loss ($aggr\_valid\_loss^t$), and test loss ($aggr\_test\_loss^t$) at tth round. Based on the comparison between previous round's and current round's $aggr\_train\_loss^t$, the model convergence decision is made as shown in the function (4–7).

$$\frac{\left|aggr\_train\_loss^{t+1} - aggr\_train\_loss^t\right|}{aggr\_train\_loss^t} < \delta \tag{4–7}$$

Here, δ is the threshold value that distinguishes the model from converged or not.

The evaluations of the training set and validation set are operated at the end of each round. In accordance with the above-explained convergence criterion (4–7), when the model converges, the final model is evaluated on the test data set to prove the finalized model's performance.

Consequently, the results of evaluations on each diverged dataset are divided into "training accuracy" and "training loss" by the training data set, "validation accuracy" and "validation loss" by the validation data set, and "test accuracy" and "test loss" by the test data set as final. These accuracy evaluation criteria can be expressed as below;

$$training\_accuracy = \frac{n\_correct_{trn}}{n_{trn}} \tag{4–8}$$

Here, $n\_correct_{trn}$ denotes the number of correct detection in training set anomaly classification, and n_{trn} is the total number of training set samples.

$$validation\_accuracy = \frac{n\_correct_{val}}{n_{val}} \tag{4–9}$$

where, $n\_correct_{val}$ means the number of correct detection in validation set anomaly classification, and n_{val} is total number of validation set samples.

$$test\_accuracy = \frac{n\_correct_{tst}}{n_{tst}} \tag{4–10}$$

here, $n\_correct_{tst}$ is the number of correct detection in test set anomaly classification, and n_{tst} is total number of test set samples. By averaging each accuracy value, they can determine the performance of the target data set in each round. For instance, by averaging *training accuracy* and *validation accuracy* each round, the model's performance in the overall network can be experimented with by each training data set and validation data set. As the final, when the system decides the model is converged, each client conducts a

local model evaluation with test data set and broadcast the result to others and make the final averaged *test accuracy* value.

The training accuracy, verification accuracy, and test accuracy of the model can intuitively reflect the performance of the model. Training loss, verification loss, and test loss can reflect the effect of model training and testing.

4.2 Performance Evaluation

In this section, we test the effectiveness of the novel collaborative FL-ADS. For comparing the accuracy and loss performances between the traditional FL-ADS and two proposed decentralized FL-ADS, we set specific rules to define convergence through function (4–7) and setting 50 as a maximum number of rounds. Thus, if $\frac{\left|aggr\_train\_loss^{t+1} - aggr\_train\_loss^{t}\right|}{aggr\_train\_loss^{t}}$ becomes smaller than 0.02 ($=\delta$), the model is determined that it is converged on the current round ($t+1$). In this experiment we set that only two clients participate for each round's parameter broadcasting in the asynchronous model. Moreover, this experiment diverges into two different neural network utilizations, the afore-mentioned MLP and Stacked-NDAE based comparisons.

4.2.1 MLP-based Anomaly Detection Scheme

The MLP is known as a cutting-edge DL technique for non-linear classification tasks. Therefore, we adopt this technology to identify outlier data in the IoT network. We employ three hidden layers in the MLP deep neural network. The sigmoid activation function is applied to the first hidden layer, and the ELU activation function is used on the last two hidden layers. Moreover, to avoid the over-fitting problem, the Dropout layer is utilized on the first two hidden layers after the activation function (Figure 3–4).

In Table 4–6, we compare the performance between the conventional (centralized network) FL-ADS, synchronous and asynchronous decentralized FL-ADSs based on MLP neural network.

According to the experiment result, we can see the performance interval is not significant between the epoch/batch size settings since we select the best three setting to generate the best performance. The core objective of this experiment is to compare the performance of the different FL framework-based ADS. In accordance with the model total averaged performance, the traditional model's average accuracy, loss, and the round result are 98.02%, 6.80%, and 10.27 rounds (to be determined the model is converged), the

Table 4–6 Convergence Setting Based MLP FL-ADS Model Performance Comparison

Epoch per Round	Batch Size	Traditional Federated Learning			Synchronous Decentralized			Asynchronous Decentralized		
		TestAcc	TestLoss	Rounds	TestAcc	TestLoss	Rounds	TestAcc	TestLoss	Rounds
110	2,000	97.35	8.36	6	97.57	8.31	11	96.41	13.46	8
		97.78	8.09	7	97.94	6.71	11	97.47	7.22	11
		97.80	8.86	7	98.04	6.54	10	97.66	8.34	11
		97.82	6.07	16	98.04	6.69	12	97.68	6.07	9
		97.86	8.02	14	98.10	4.67	8	97.75	6.84	17
		98.17	7.95	12	98.13	8.31	16	97.96	7.56	12
		98.26	7.17	8	98.19	7.09	7	98.04	6.93	7
		98.47	5.60	7	98.32	8.72	10	98.14	5.98	24
		98.73	5.19	16	98.35	6.72	7	98.24	7.53	8
		99.14	3.74	10	98.41	6.21	8	98.53	5.28	16
Avg		**98.14**	**6.91**	**10.30**	**98.11**	**7.00**	**10.00**	**97.79**	**7.52**	**12.30**
75	1,000	97.40	9.85	9	97.51	7.57	9	95.14	17.82	15
		97.63	5.83	6	97.56	7.59	9	95.84	25.69	8
		97.66	6.82	9	97.65	6.70	5	97.62	7.53	9
		97.98	6.32	16	97.81	6.74	4	97.72	7.21	9
		98.04	6.37	8	97.81	9.01	6	97.83	7.38	8
		98.09	5.56	9	97.88	8.50	6	97.91	6.19	11
		98.15	7.53	6	97.96	6.13	19	97.92	6.74	10
		98.19	6.55	15	98.30	7.37	7	98.07	8.09	13
		98.24	6.77	9	98.33	6.87	7	98.13	7.15	29
		98.41	5.40	16	98.48	7.42	8	98.29	6.33	10
Avg		**97.98**	**6.70**	**10.30**	**97.93**	**7.39**	**10.20**	**97.43**	**10.17**	**12.50**
60	1,000	97.39	5.35	12	97.87	6.94	7	97.39	7.68	19
		97.50	8.08	8	97.87	7.50	16	97.46	7.80	8
		97.78	7.48	9	97.88	7.19	6	97.74	7.47	23
		97.90	7.30	7	97.97	6.54	17	97.76	7.32	10
		97.99	6.89	7	97.98	8.54	8	97.79	5.65	8
		98.14	6.61	16	98.00	6.21	18	97.89	5.19	16
		98.14	7.07	9	98.07	7.94	7	97.89	6.44	11
		98.16	7.56	15	98.47	7.45	10	97.89	9.12	15
		98.18	6.41	10	98.51	6.74	7	97.96	7.84	11
		98.21	5.25	9	98.75	7.45	9	98.00	6.75	14
Avg		**97.94**	**6.80**	**10.20**	**98.14**	**7.25**	**10.50**	**97.78**	**7.13**	**13.50**
Model Total Avg		**98.02**	**6.80**	**10.27**	**98.06**	**7.21**	**10.23**	**97.66**	**8.27**	**12.77**

synchronous model's results are 98.06%, 7.21%, and 10.23 rounds, and the asynchronous model's outputs are 97.66%, 8.27%, and 12.77 rounds, respectively.

The summarized result proves that the proposed synchronous decentralized FL-ADS algorithm performs very similar results to the conventional FL-ADS in the comparison targets that test set-based the accuracy value, loss value, and the number of rounds to get converged. On the other hand, the asynchronous decentralized model has worse accuracy ($\approx$ 0.40%) and loss ($\approx$ 1.20%) performances and takes more rounds ($\approx$ 2.5 *rounds*) to get converged in general. We consider only experiment with MLP is not enough to prove the presented decentralized FL-ADS's usability. Therefore, we investigate the same experiment with the Stacked-NDAE anomaly classifier.

4.2.2 Stacked-NDAE-based Anomaly Detection Scheme

Table 4–7 shows the comparison of the performance between the traditional FL-ADS, synchronous, and asynchronous decentralized FL-ADS frameworks based on Stacked-NDAE cutting-edge deep learning method.

According to the results on the table, we also can assume the effect from the epoch/-batch size settings on the classification performance is not significant as well. However, the performance difference between the different models is conspicuous. In terms of the model toral average performance, the traditional model's average accuracy, loss, and the round result are 98.24%, 1.67%, and 8.07 rounds, the synchronous model's are 98.24%, 1.66%, and 8.13 rounds, and the asynchronous model's performance is 97.89%, 1.97%, and 9.37 rounds, respectively.

Similar to the MLP-based experiment, the performance of the traditional FL-ADS algorithm and the synchronous FL-ADS algorithm since the parameter aggregation methods are very similar. On the contrary, the asynchronous decentralized model shows worse performance, namely accuracy ($\approx$ 0.35%) less and loss ($\approx$ 0.31%) more and needs more rounds ($\approx$ 1.3 *rounds*) to get converged.

4.2.3 Section Summary

We implement the performance evaluation of our proposed models and the traditional FL with two different DL methods. From a total of 180 times of the stacked performance experiments, we can clarify how the networking algorithms and neural network methods affect the anomaly detection performance. Table 4–8 summarizes the above

Table 4–7 Convergence Setting Based Stacked-NDAE FL-ADS Model Performance Comparison

Epoch per Round	Batch Size	Traditional Federated Learning			Synchronous Decentralized			Asynchronous Decentralized		
		TestAcc	TestLoss	Rounds	TestAcc	TestLoss	Rounds	TestAcc	TestLoss	Rounds
110	2,000	97.71	1.89	10	97.95	1.93	6	97.44	2.09	14
		97.92	1.98	6	97.96	1.93	8	97.59	2.33	6
		98.04	1.89	8	97.98	1.20	6	97.64	1.89	14
		98.13	1.77	8	98.07	1.87	6	97.84	2.05	6
		98.19	1.35	12	98.08	1.85	7	97.93	1.94	6
		98.22	1.78	7	98.16	1.77	10	98.01	1.88	6
		98.37	1.55	9	98.25	1.69	7	98.16	1.76	7
		98.43	1.49	8	98.34	1.61	9	98.17	1.76	9
		98.73	1.20	6	98.71	1.22	9	98.18	1.73	7
		98.75	1.14	7	99.03	0.94	14	98.70	1.29	15
Avg		**98.25**	**1.60**	**8.10**	**98.25**	**1.60**	**8.20**	**97.97**	**1.87**	**9.00**
75	1,000	97.80	2.30	6	97.59	2.27	7	97.53	2.26	9
		97.90	2.00	7	97.86	2.24	6	97.61	2.53	15
		97.97	1.93	7	98.02	1.89	8	97.63	2.26	8
		98.03	1.89	7	98.20	1.75	14	97.74	2.15	9
		98.08	2.07	8	98.24	1.71	7	97.75	2.03	8
		98.15	1.75	6	98.26	1.74	7	97.85	2.12	7
		98.23	1.65	11	98.33	1.69	7	97.98	2.01	14
		98.28	1.67	11	98.44	1.60	9	98.34	1.64	8
		98.51	1.45	11	98.44	1.20	8	98.40	1.55	7
		98.71	1.25	9	98.56	1.38	11	98.54	1.33	12
Avg		**98.17**	**1.80**	**8.30**	**98.19**	**1.75**	**8.40**	**97.94**	**1.99**	**9.70**
60	1,000	97.69	2.16	7	97.88	2.24	7	97.00	2.82	7
		97.80	2.08	7	97.94	1.59	13	97.24	1.47	12
		97.82	2.10	8	98.02	1.91	6	97.43	2.67	8
		98.19	1.70	7	98.06	1.86	7	97.71	2.17	11
		98.25	1.70	10	98.16	1.79	6	97.74	2.28	8
		98.43	1.51	11	98.30	1.63	8	97.75	2.17	9
		98.43	1.53	7	98.30	1.64	7	97.80	1.76	9
		98.61	1.32	7	98.52	1.43	7	97.81	2.02	7
		98.93	1.05	8	98.57	1.28	7	98.32	1.61	12
		99.00	1.01	6	99.02	0.91	10	98.78	1.61	11
Avg		**98.32**	**1.62**	**7.80**	**98.28**	**1.63**	**7.80**	**97.76**	**2.06**	**9.40**
Model Total Avg		**98.24**	**1.67**	**8.07**	**98.24**	**1.66**	**8.13**	**97.89**	**1.97**	**9.37**

Table 4–8 Performance Experiment Result Comparison

Epoch per Round	Batch Size	Neural Network Model	Traditional Federated Learning			Synchronous Decentralized			Asynchronous Decentralized		
			TestAcc	TestLoss	Rounds	TestAcc	TestLoss	Rounds	TestAcc	TestLoss	Rounds
110	2,000	MLP	98.14	6.91	10.30	98.11	7.00	10.00	97.79	7.52	12.30
		Stacked-NDAE	98.25	1.60	8.10	98.25	1.60	8.20	97.97	1.87	9.00
75	1,000	MLP	97.98	6.70	10.30	97.93	7.39	10.20	97.43	10.17	12.50
		Stacked-NDAE	98.17	1.80	8.30	98.19	1.75	8.40	97.94	1.99	9.70
60	1,000	MLP	97.94	6.80	10.20	98.14	7.25	10.50	97.78	7.13	13.50
		Stacked-NDAE	98.32	1.62	7.80	98.28	1.63	7.80	97.76	2.06	9.40
MLP Total Avg			98.02	6.80	10.27	98.06	7.21	10.23	97.66	8.27	12.77
Stacked-NDAE Total Avg			98.24	1.67	8.07	98.24	1.66	8.13	97.89	1.97	9.37

experiments' final results (from Subsection 4.2.1 and 4.2.2).

Both neural network methods based experiments show the traditional FL-ADS and the synchronous decentralized FL-ADS perform obviously better to detect anomaly data than the asynchronous decentralized FL-ADS. The reason for the worsening performance in the asynchronous model is assumed that comes from the parameter aggregation procedures' distinction with others. In other words, only a part of clients, in our case, two clients over four clients, is participating in the parameter distribution. Therefore, the number of received parameters becomes unbalanced between the clients, so the generated new model is less comprehensive than the case of synchronous and traditional FL algorithm. However, since a portion of clients broadcasts the parameters, the number of transmissions in the network significantly decreased. Therefore, the network designing needs awareness of the pros and cons of each model.

On the view of the performance divergence between the neural network models, Stacked-NDAE shows significantly better than the case of the MLP method. Approximately 0.20% of accuracy, 5.13-6.30%, and 2.10-3.40 rounds better performance is enhanced compared to the MLP-based experiment. The reason for the performance difference is considered that because of more complexity and hidden layers in Stacked-NDAE.

We consider that each model's difference comes from the contrast of the convergence trend. However, the organized performance comparison experiments do not present the convergence status clearly. Therefore, we investigate each algorithm's convergence trend in the next section.

4.3 Convergence Evaluation

Since our proposed models are based on the FL technique, which is the collaboration of local client's DL tasks, how stably converge to the proper point is one of the most core perspectives to assess the usability of the FL techniques. In our proposed schemes also aggregates the local training results to improve current classifier models on each client. Therefore, we implement each comparison target models without convergence limit (Function 4–7) until they finish the 50th round based on two different network topologies; full-mesh network, multi-hop network.

4.3.1 Comparison in Full-mesh Network

In this section, we analyze the convergence trends between the experiment target FL models, namely the traditional FL-ADS, synchronous decentralized FL-ADS, and asynchronous decentralized FL-ADS based on the different number of broadcaster settings (1 to 3) in the full-mesh network topology.

4.3.1.1 MLP-based Anomaly Detection Scheme

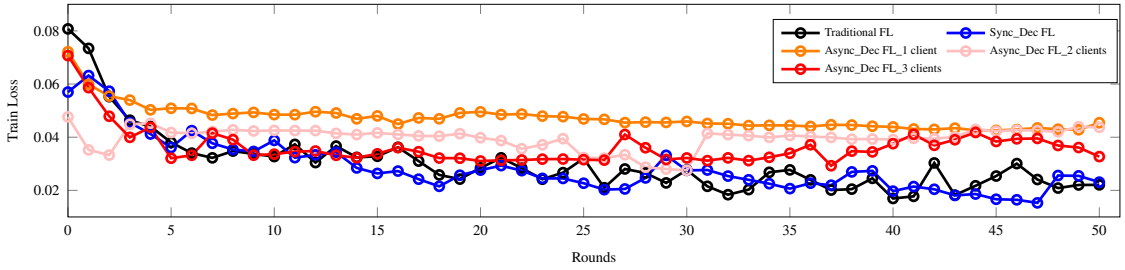


Figure 4–2 MLP-based Train Loss Trend Comparison Between Synchronous&Asynchronous Decentralized FL models in the Full-mesh Network

We display each target models' convergence trends within 50 rounds based on the MLP neural network classifier. Figure 4–2 and Figure 4–3 present our convergence trend experiment results, the loss value and accuracy performance in the full-mesh network, respectively. The above convergence trend graphs also prove that the synchronous FL (*blue*) algorithm and conventional FL (*black*) deep learning have similar trends. Both models' results converge to 2.00% loss value and 99.50%, approximately. Moreover, the asynchronous model shows the worse performance when fewer clients broadcast on each round. Therefore the orange-colored trend shows the worst performance since only one

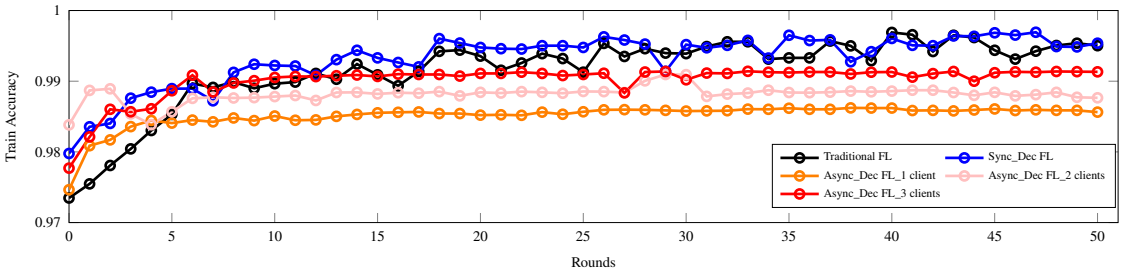


Figure 4–3 MLP-based Train Accuracy Trend Comparison Between Synchronous&Asynchronous Decentralized FL models in the Full-mesh Network

client is activated on each round, but the red colored data is very close to the synchronous data. Consequently, the models are converged to 3.5% loss value and 99.1% in three broadcasters setting (*red*), 4.0% loss value and 98.8% in two broadcasters setting (*pink*), 4.5% loss value and 98.6% in one broadcaster setting (*orange*), approximately.

4.3.1.2 Stacked-NDAE-based Anomaly Detection Scheme

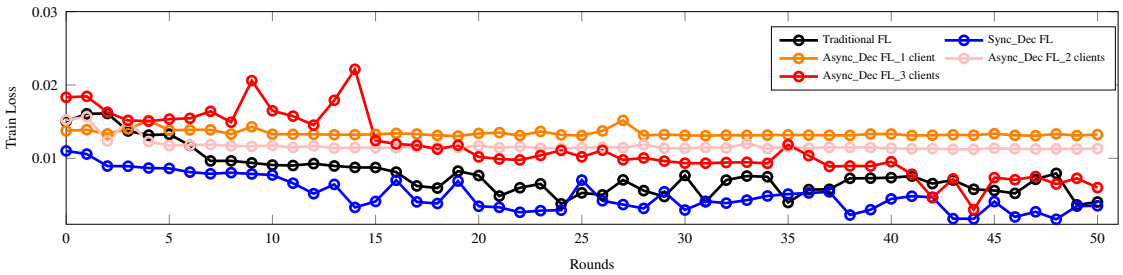


Figure 4–4 Stacked-NDAE-based Train Loss Trend Comparison Between Synchronous&Asynchronous Decentralized FL models in the Full-mesh Network

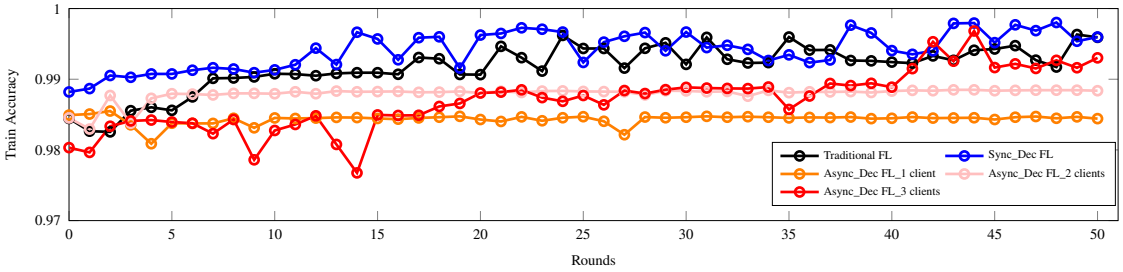


Figure 4–5 Stacked-NDAE-based Train Accuracy Trend Comparison Between Synchronous&Asynchronous Decentralized FL models in the Full-mesh Network

Figure 4–4 and 4–5 show the convergence trends of the proposed anomaly detection model with the Stacked-NDAE based on the above shown full-mesh network. The fundamental test method is the same as the previous MLP-based convergence comparison experiment. The training set test result is remarkably enhanced than the same case of the MLP-based experiment above. The traditional and synchronous FL-ADS shows 0.4% loss value and 99.6%, and asynchronous FL-ADS shows 0.6% loss and 99.3% accuracy in three broadcaster setting, 1.1% loss and 98.8% accuracy in two broadcaster setting, and 1.3% loss and 98.4% accuracy in three broadcaster setting, approximately. Notably, the starting point of the training data set test performance, performance in the 0th round, is significantly better than the MLP-based model.

4.3.2 Comparison in Multi-hop Network

The real-world IoT networks mostly have the multi-hop network that the devices communicate with the devices in their specific coverage area. Therefore, we implement the convergence trend investigation of the target models in the multi-hop network topology. We set six clients and the communication connection network in the multi-hop network, as shown in Figure 4–1b.

4.3.2.1 MLP-based Anomaly Detection Scheme

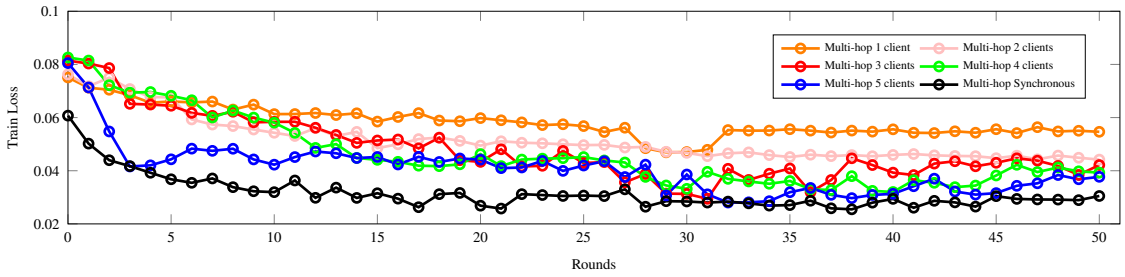


Figure 4–6 MLP-based Train Loss Trend Comparison Between Synchronous&Asynchronous Decentralized FL models in the Multi-hop Network

In order to demonstrate close to the real-world network, we simulate the proposed models in the wireless multi-hop network. The loss and accuracy of convergence trends are plotted in Figure 4–6 and Figure 4–7, respectively. Our experiment result shows that the asynchronous model with client one to five broadcasters per each round and synchronous model (6 broadcasters) converge to approximately 5.5%, 4.6%, 3.8%, 3.6%,

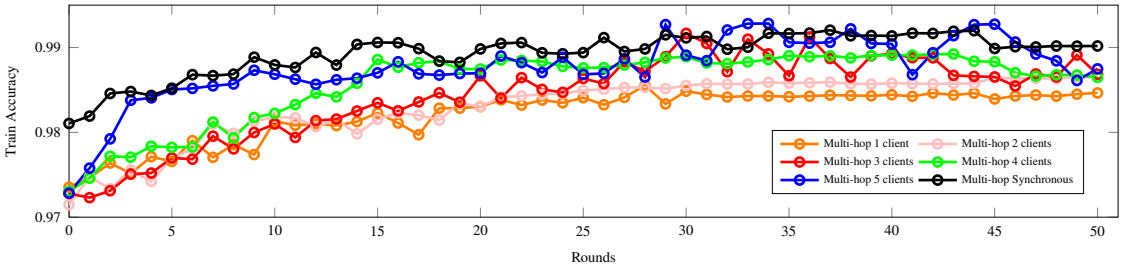


Figure 4–7 MLP-based Train Accuracy Trend Comparison Between Synchronous&Asynchronous Decentralized FL models in the Multi-hop Network

3.1%, 2.8% loss values, and 98.4%, 98.5%, 98.6%, 98.8%, 98.9%, 99.0%, respectively (*black*: the synchronous model - six active clients, *blue*: the asynchronous model - five active clients, *green*: the asynchronous model - four active clients, *red*: the asynchronous model - three active clients, *pink*: the asynchronous model - two active clients, *orange*: the asynchronous model - one active clients). Since each client broadcasts the parameter to only adjacent clients, when a few broadcasters are active, the parameter aggregation effectiveness is unsatisfactory.

4.3.2.2 Stacked-NDAE-based Anomaly Detection Scheme

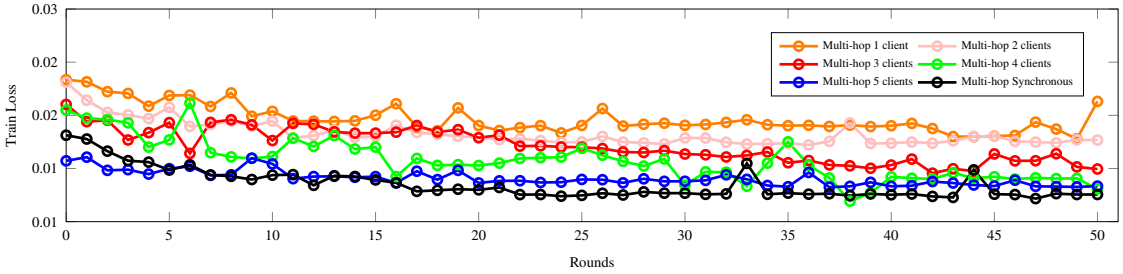


Figure 4–8 Stacked-NDAE-based Train Loss Trend Comparison Between Synchronous&Asynchronous Decentralized FL models in the Multi-hop Network

The Stacked-NDAE-based decentralized FL-ADS models are implemented as shown the above Figure 4–8 and Figure 4–9. As discussed above experiments, the performance of Stacked-NDAE is more beneficial than the MLP-based implementation. The Stacked-NDAE-based synchronous model converges to 0.7% loss value and 99.2% accuracy, approximately. Moreover, the asynchronous model converges to approximately 0.8% loss value and 99.0% accuracy in five broadcasters setting, 0.9% loss value and 98.9% accuracy in four broadcasters setting, 1.0% loss value and 98.7% accuracy in three

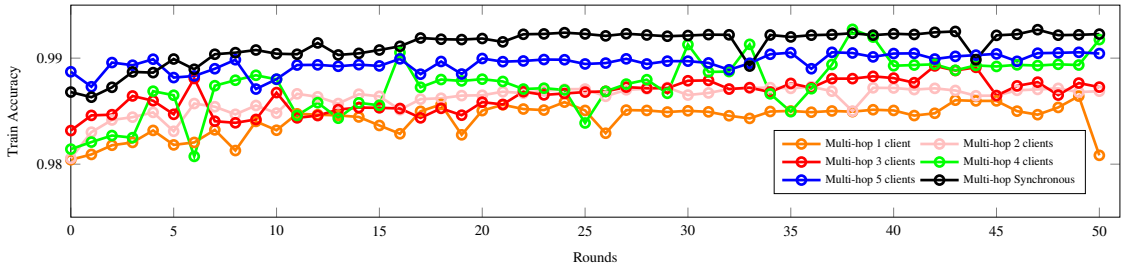


Figure 4–9 Stacked-NDAE-based Train Accuracy Trend Comparison Between Synchronous&Asynchronous Decentralized FL models in the Multi-hop Network

broadcasters setting, 1.3% loss value and 98.6% accuracy in two broadcasters setting, 1.4% loss value and 98.5% accuracy in one broadcaster setting, respectively.

4.3.3 Section Summary

We implement the model running until it completes the 50th round to monitor the convergence trends. The organized experiments show us the pronounced differences between different FL-ADS frameworks and different neural network classification methods.

Based on our comparisons of the anomaly detection classifier model, we can recognize the Stacked-NDAE is more potent than the MLP method. Even though the performance is an essential metric to evaluate the model, but also the required time is inevitable to consider. Table 4–9 displays the spent time from the above convergence trend experiments.

Table 4–9 Time Consumption to Finish 50 Rounds

Classifier Model	FL Model	MLP	StackedNDAE
Full-Mesh	Traditional FL	859	1,070
	Sync FL (C4)	1,000	1,239
	Async FL (C3)	1,077	1,210
	Async FL (C2)	1,089	1,273
	Async FL (C1)	1,129	1,203
Multi-Hop	Sync FL (C6)	1,640	1,965
	Async FL (C5)	1,588	1,825
	Async FL (C4)	1,706	1,761
	Async FL (C3)	1,635	1,785
	Async FL (C2)	1,533	1,874
	Async FL (C1)	1,659	1,741

According to the time records from the table, we can realize two observations.

- First, the traditional FL is faster than decentralized frameworks. It is assumed that because the traditional model is more well organized, due to the server orchestrates the overall network. However, the clients are less organized in the decentralized framework caused by each clients' training time and aggregation time is not synchronized, so it could get some delays to step to the next round to wait for the late-coming clients.
- Secondly, Stacked-NDAE takes 13.62% (in total average) more time than MLP-based FL-ADS results. It is because of the complex difference between them. Since Stacked-NDAE has more hidden layers (6) than MLP (3) and more neurons in hidden layers as shown in Subsection 3.2, it has more than doubled parameters of MLP. The Stacked-NDAE contains 4,537 total parameters, and the MLP includes 2,089 total parameters in their neural networks. Therefore, we estimate the required time interval gap comes from the complexity of the utilized models.

Furthermore, it is verified that the synchronous decentralized FL model can perform similarly to the traditional FL. If the less active clients (broadcasters) are set, the model aggregation effectiveness becomes significantly weaker than the case of more active clients. The horizontality of one or two active clients' cases on each trend experiment represents the worsen aggregation power.

Since the core idea of FL is the interaction of local weighted parameters, a few numbers of broadcaster setting should be meaningless. However, if too many devices are active for parameter distribution in the IoT networks, the massive interconnections and transmissions might cause some critical network problems.

In the full-mesh network experiments, all models start to converge from the 18th round with the MLP classifier, but it converges from the 10th round with the Stacked-NDAE in general. On the other hand, the MLP-based models converge from the 25th round, and the Stacked-NDAE-based models get converged at the 18th round in the multi-hop network simulations. We estimate because the broadcasting in the multi-hop topology is not fully distributed to other clients, it takes more time to get converged than in the case of the full-mesh topology.

4.4 Chapter Summary

Based on the above-organized experiments, we can clarify the insight into the decentralized FL-ADS characteristics. Overall, our experiments to test the robustness of our proposed model is categorized into two different tests that convergence condition-based performance comparison and convergence trend comparison within 50 rounds. Based on the categorized experiments, the two state-of-art DL classification models are analyzed simultaneously.

From the implemented experiments, we can figure out the Stacked-NDAE has a more powerful ability to classify anomaly data than the MLP algorithm. In view of the fact that comprehensive empirical training-based FL can make an excellent performance, well-aggregated traditional FL-ADS and synchronous decentralized FL-ADS is able to perform more salutary than the asynchronous model.

Subsequently, the convergence trend experiment shows that the asynchronous model with fewer broadcasters has more unsatisfactory performance than the model with more broadcasters each round. Moreover, we analyze the neural network's structural difference to get a deep insight into the anomaly data classification method. Consequently, we determine the reason why the Stacked-NDAE-based model takes more time than the MLP model is that the complexity of the deep learning neural network difference.

Chapter 5 Conclusion and Future Works

We propose two different synchronous and asynchronous decentralized federated learning and verify their performance and modality in the real-world network. In this chapter, we make a conclusion based on the insight of our research results and post research next work plan.

5.1 Thesis Summary

In awareness of the rapid development of the Internet of Things, we focus on the in-depth analysis and research on the anomaly detection methods under the big data environment. First of all, the traditional anomaly detection methods and collaborative ADS structures are summarized, and two different parameter updating methods-based collaborative DL ADS are proposed for the problems and weaknesses of the traditional anomaly detection methods and centralized ADS structures.

In the traditional anomaly detection network structure in the big data cases, the centralized network has extremely high requirements for the amount of data upload, so the demand for bandwidth is huge, and at the same time, it will cause a high delay in the transmission process.

As our suggestion for the anomaly detection system, we design two different decentralized federated learning-based networking methods; the synchronous decentralized FL-ADS and the asynchronous decentralized FL-ADS. We reconstruct the conventional centralized federated learning due to the FL has several virtues. The mechanism makes the transmission much lighter, makes bandwidth effective, and security more robust. Furthermore, it overcomes many limitations from centralized networking and becomes more IoT-friendly by decentralized networking ideas. The proposed decentralized scheme utilizes the multi-layer perceptron and Stacked-NDAE as the local anomaly classifier model. The MLP enables non-linear deep learning empowered by the triple hidden layers. And the Stacked-NDAE framework contains six encoders (hidden layers).

The decentralized FL-ADS algorithm is divided into two different ways based on parameter broadcasting and aggregation methods. The synchronous decentralized FL-ADS is the method that every client participates in the local training result parameter transmission after the local training. Afterward, each client aggregates the received pa-

rameters and its own to generate the new anomaly detection model. Since the edges in the synchronous model are fully-activated, the aggregation effect is maximized. On the other hand, only a portion of clients operates the training result parameter broadcasting, unlike the synchronous framework. Thus, it can occur less comprehensive aggregated model than that.

We implement two experiments to test the performance of the proposed schemes and compare the difference between them.

Initially, we compare the test set performance evaluation based on our preset convergence condition. Based on the experimental results, we get some observations. First of all, the performance of the synchronous model is almost the same as the traditional FL model's result. But the asynchronous model prints worse performance with more rounds to get converged, same as our estimation. Secondly, the Stacked-NDAE generates more robust performance than the MLP adoption.

Subsequently, we implement the convergence trend investigation to monitor the drift of the performance as the round goes on. The experiment is organized based on two topologies that the full-mesh and multi-hop, to simulate the real-world network cases. From the experiments under the full-mesh network, we can observe the performances of the traditional FL-ADS and the synchronous FL-ADS correspond the same as the observation of the performance comparison experiment. Furthermore, it is verified that the asynchronous model with a more broadcaster setting is more robust than it with a less broadcaster setting. Moreover, we analyze the classifier models since the performance superiority of the Stacked-NDAE is very obvious compared to the MLP model in the anomaly detection task. As the trade-off to utilize Stacked-NDAE instead of the MLP, the training latency is raised due to the complexity difference of the models.

The two systems proposed in this paper, the synchronous and asynchronous decentralized FL-ADS, are verified that can make very significant performance in the IoT networks. Not only the benefits from the FL idea but also being IoT friendly and virtues from decentralized topology are achieved. The decentralization allows us to get more secured networking, reduction of server management cost, and flexibility with the orchestration of devices. The asynchronous model with less number of broadcasters is much more flexible and less busy with networking and local computation. However, it also has to sacrifice performance.

5.2 Future Work

Since our proposed models are based on decentralized interactions, there are challenges to overcome. The decentralized edge to edge communications can raise critical overhead by high computational work burdens on the basis of the responsibilities of local training and concentrated parameter aggregation. Moreover, there is a security risk of the cases that the case of an attacker can corrupt the edge model by injecting false data into the local database, thereby affecting the preventing the entire network security. In our future work, we will analyze the above problems to optimize and enhance the proposed system model. In addition, we will further carry out the proposed decentralized collaborative anomaly detection system experiments deployed in the big data environment to further evaluate and improve the system performance in the big data environment.

Bibliography

[1] Xia Feng, Yang Laurence T, Wang Lizhe, et al. Internet of things[J]. International journal of communication systems, 2012, 25(9): 1101.

[2] Maayan GD. The iot rundown for 2020: Stats, risks, and solutions[J]. Security Today, 2020, 13.

[3] Köhn Rüdiger. Konzerne verbünden sich gegen Hacker[EB/OL]. 2018. https://www.faz.net/aktuell/wirtschaft/digitec/grosse-internationale-allianz-gegen-cyber-attacken-15451953.html.

[4] Nordrum Amy. Popular Internet of Things Forecast of 50 Billion Devices by 2020 Is Outdated, August 2016[J]. URL https://spectrum. ieee. org/tech-talk/telecom/internet/popular-internet-of-things-forecastof-50-billion-devices-by-2020-is-outdated,

[5] Lynkova Darina. Iot statistics and trends to know in 2020[EB/OL]. 2019. https://leftronic.com/internet-of-things-statistics/.

[6] Rydning David Reinsel–John Gantz–John. The digitization of the world from edge to core[J]. Framingham: International Data Corporation, 2018.

[7] Bertino Elisa, Islam Nayeem. Botnets and internet of things security[J]. Computer, 2017, 50(2): 76-79.

[8] Khan Ahmed Yar, Latif Rabia, Latif Seemab, et al. Malicious insider attack detection in IoTs using data analytics[J]. IEEE Access, 2019, 8: 11743-11753.

[9] Bhuyan Monowar H, Bhattacharyya Dhruba Kumar, Kalita Jugal K. Network anomaly detection: methods, systems and tools[J]. Ieee communications surveys & tutorials, 2013, 16(1): 303-336.

[10] Phua Clifton, Lee Vincent, Smith Kate, et al. A comprehensive survey of data mining-based fraud detection research[J]. ArXiv preprint arXiv:1009.6119, 2010.

[11] Lee Wenke, Xiang Dong. Information-theoretic measures for anomaly detection[C]. in: Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001. 2000: 130-143.

[12] Lane Terran, Brodley Carla E. An application of machine learning to anomaly detection[C]. in: Proceedings of the 20th National Information Systems Security Conference: vol. 377. 1997: 366-380.

[13] McMahan Brendan, Moore Eider, Ramage Daniel, et al. Communication-efficient learning of deep networks from decentralized data[C]. in: Artificial Intelligence and Statistics. 2017: 1273-1282.

[14] Chandola Varun, Banerjee Arindam, Kumar Vipin. Anomaly detection: A survey[J]. ACM computing surveys (CSUR), 2009, 41(3): 1-58.

[15] Kumar Vipin. Parallel and distributed computing for cybersecurity[J]. IEEE Distributed Systems Online, 2005, 6(10).

[16] Edgeworth Francis Ysidro. Xli. on discordant observations[J]. The london, edinburgh, and dublin philosophical magazine and journal of science, 1887, 23(143): 364-375.

[17] Cohen Ira. A Quick Guide to the Different Types of Outliers[EB/OL]. Anodot. 2020. https://www.anodot.com/blog/quick-guide-different-types-outliers/.

[18] Song Xiuyao, Wu Mingxi, Jermaine Christopher, et al. Conditional anomaly detection[J]. IEEE Transactions on knowledge and Data Engineering, 2007, 19(5): 631-645.

[19] Salvador Stan, Chan Philip, Brodie John. Learning States and Rules for Time Series Anomaly Detection.[C]. in: FLAIRS conference. 2004: 306-311.

[20] Kou Yufeng, Lu Chang-Tien, Chen Dechang. Spatial weighted outlier detection[C]. in: Proceedings of the 2006 SIAM international conference on data mining. 2006: 614-618.

[21] Sun Pei, Chawla Sanjay, Arunasalam Bavani. Mining for outliers in sequential databases[C]. in: Proceedings of the 2006 SIAM international conference on data mining. 2006: 94-105.

[22] Shekhar Shashi, Lu Chang-Tien, Zhang Pusheng. Detecting graph-based spatial outliers: algorithms and applications (a summary of results)[C]. in: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining. 2001: 371-376.

[23] Noble Caleb C, Cook Diane J. Graph-based anomaly detection[C]. in: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining. 2003: 631-636.

[24] Jose Shijoe, Malathi D, Reddy Bharath, et al. A survey on anomaly based host intrusion detection system[C]. in: Journal of Physics: Conference Series: vol. 1000: 1. 2018: 012049.

[25] Zhang Min-Ling, Zhou Zhi-Hua. ML-KNN: A lazy learning approach to multi-label learning[J]. Pattern recognition, 2007, 40(7): 2038-2048.

[26] Rokach Lior, Maimon Oded. Top-down induction of decision trees classifiers-a survey[J]. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 2005, 35(4): 476-487.

[27] Syarif Iwan, Prugel-Bennett Adam, Wills Gary. Unsupervised clustering approach for network anomaly detection[C]. in: International conference on networked digital technologies. 2012: 135-145.

[28] Astorino A, Gorgone ENRICO, Gaudioso M, et al. Data preprocessing in semi-supervised SVM classification[J]. Optimization, 2011, 60(1-2): 143-151.

[29] Baur Christoph, Albarqouni Shadi, Navab Nassir. Semi-supervised deep learning for fully convolutional networks[C]. in: International Conference on Medical Image Computing and Computer-Assisted Intervention. 2017: 311-319.

[30] Münz Gerhard, Li Sa, Carle Georg. Traffic anomaly detection using k-means clustering[C]. in: GI/ITG Workshop MMBnet. 2007: 13-14.

[31] Leung Kingsly, Leckie Christopher. Unsupervised anomaly detection in network intrusion detection using clusters[C]. in: Proceedings of the Twenty-eighth Australasian conference on Computer Science-Volume 38. 2005: 333-342.

[32] Guan Yu, Ghorbani Ali A, Belacel Nabil. Y-means: A clustering method for intrusion detection[C]. in: CCECE 2003-Canadian Conference on Electrical and Computer Engineering. Toward a Caring and Humane Technology (Cat. No. 03CH37436): vol. 2. 2003: 1083-1086.

[33] Khan Latifur, Awad Mamoun, Thuraisingham Bhavani. A new intrusion detection system using support vector machines and hierarchical clustering[J]. The VLDB journal, 2007, 16(4): 507-521.

[34] Chalapathy Raghavendra, Chawla Sanjay. Deep learning for anomaly detection: A survey[J]. ArXiv preprint arXiv:1901.03407, 2019.

[35] Mudassar Burhan A, Ko Jong Hwan, Mukhopadhyay Saibal. An unsupervised anomalous event detection framework with class aware source separation[C]. in: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2018: 2671-2675.

[36] Luo Tie, Nagarajan Sai G. Distributed anomaly detection using autoencoder neural networks in wsn for iot[C]. in: 2018 IEEE International Conference on Communications (ICC). 2018: 1-6.

[37] Van Efferen Lennart, Ali-Eldin Amr MT. A multi-layer perceptron approach for flow-based anomaly detection[C]. in: 2017 International Symposium on Networks, Computers and Communications (ISNCC). 2017: 1-6.

[38] Shone Nathan, Ngoc Tran Nguyen, Phai Vu Dinh, et al. A deep learning approach to network intrusion detection[J]. IEEE transactions on emerging topics in computational intelligence, 2018, 2(1): 41-50.

[39] Bengio Yoshua. Learning deep architectures for AI[M]. Now Publishers Inc, 2009.

[40] Lyu Lingjuan, Jin Jiong, Rajasegarar Sutharshan, et al. Fog-empowered anomaly detection in IoT using hyperellipsoidal clustering[J]. IEEE Internet of Things Journal, 2017, 4(5): 1174-1184.

[41] Abeshu Abebe, Chilamkurti Naveen. Deep learning: The frontier for distributed attack detection in fog-to-things computing[J]. IEEE Communications Magazine, 2018, 56(2): 169-175.

[42] Idé Tsuyoshi. Collaborative anomaly detection on blockchain from noisy sensor data[C]. in: 2018 IEEE International Conference on Data Mining Workshops (ICDMW). 2018: 120-127.

[43] Cai He, Hua Cunqing, Xu Wenchao. Design of Active Learning Framework for Collaborative Anomaly Detection[C]. in: 2019 11th International Conference on Wireless Communications and Signal Processing (WCSP). 2019: 1-7.

[44] Moustafa Nour, Creech Gideon, Sitnikova Elena, et al. Collaborative anomaly detection framework for handling big data of cloud computing[C]. in: 2017 Military Communications and Information Systems Conference (MilCIS). 2017: 1-6.

[45] Lim Wei Yang Bryan, Luong Nguyen Cong, Hoang Dinh Thai, et al. Federated learning in mobile edge networks: A comprehensive survey[J]. IEEE Communications Surveys & Tutorials, 2020.

[46] Ananthanarayanan Ganesh, Bahl Paramvir, Bodďik Peter, et al. Real-time video analytics: The killer app for edge computing[J]. Computer, 2017, 50(10): 58-67.

[47] Lu Yunlong, Huang Xiaohong, Dai Yueyue, et al. Blockchain and federated learning for privacy-preserved data sharing in industrial IoT[J]. IEEE Transactions on Industrial Informatics, 2019, 16(6): 4177-4186.

[48] Khan Latif U, Pandey Shashi Raj, Tran Nguyen H, et al. Federated learning for edge networks: Resource optimization and incentive mechanism[J]. IEEE Communications Magazine, 2020, 58(10): 88-93.

[49] Mills Jed, Hu Jia, Min Geyong. Communication-efficient federated learning for wireless edge intelligence in iot[J]. IEEE Internet of Things Journal, 2019.

[50] Kim Seongwoo, Cai He, Hua Cunqing, et al. Collaborative Anomaly Detection for Internet of Things based on Federated Learning[C]. in: 2020 IEEE/CIC International Conference on Communications in China (ICCC). 2020: 623-628.

[51] Ferrer Eduardo Castelló, Rudovic Ognjen, Hardjono Thomas, et al. Robochain: A secure data-sharing framework for human-robot interaction[J]. ArXiv preprint arXiv:1802.04480, 2018.

[52] Weng Jiasi, Weng Jian, Zhang Jilian, et al. Deepchain: Auditable and privacy-preserving deep learning with blockchain-based incentive[J]. IEEE Transactions on Dependable and Secure Computing, 2019.

[53] Kang Jiawen, Xiong Zehui, Niyato Dusit, et al. Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory[J]. IEEE Internet of Things Journal, 2019, 6(6): 10700-10714.

[54] Zhao Yang, Zhao Jun, Jiang Linshan, et al. Mobile edge computing, blockchain and reputation-based crowdsourcing iot federated learning: A secure, decentralized and privacy-preserving system[J]. ArXiv preprint arXiv:1906.10893, 2019.

[55] Kim Hyesung, Park Jihong, Bennis Mehdi, et al. Blockchained on-device federated learning[J]. IEEE Communications Letters, 2019, 24(6): 1279-1283.

[56] Ramanan Paritosh, Nakayama Kiyoshi. Baffle: Blockchain based aggregator free federated learning[C]. in: 2020 IEEE International Conference on Blockchain (Blockchain). 2020: 72-81.

[57] Jadidi Zahra, Muthukkumarasamy Vallipuram, Sithirasenan Elankayer, et al. Flow-based anomaly detection using neural network optimized with GSA algorithm[C]. in: 2013 IEEE 33rd international conference on distributed computing systems workshops. 2013: 76-81.

[58] Glorot Xavier, Bordes Antoine, Bengio Yoshua. Deep sparse rectifier neural networks[C]. in: Proceedings of the fourteenth international conference on artificial intelligence and statistics. 2011: 315-323.

[59] Li Yuanzhi, Yuan Yang. Convergence analysis of two-layer neural networks with relu activation[C]. in: Advances in neural information processing systems. 2017: 597-607.

[60] Li Yang, Fan Chunxiao, Li Yong, et al. Improving deep neural network with multiple parametric exponential linear units[J]. Neurocomputing, 2018, 301: 11-24.

[61] Kurbiel Thomas, Khaleghian Shahrzad. Training of deep neural networks based on distance measures using RMSProp[J]. ArXiv preprint arXiv:1708.01911, 2017.

[62] Revathi S, Malathi A. A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection[J]. International Journal of Engineering Research & Technology (IJERT), 2013, 2(12): 1848-1853.

[63] Mishra Preeti, Varadharajan Vijay, Tupakula Uday, et al. A detailed investigation and analysis of using machine learning techniques for intrusion detection[J]. IEEE Communications Surveys & Tutorials, 2018, 21(1): 686-728.

[64] Wold Svante, Esbensen Kim, Geladi Paul. Principal component analysis[J]. Chemometrics and intelligent laboratory systems, 1987, 2(1-3): 37-52.

Acknowledgements

Since the master's course is going to finish, I look back on time in Shanghai Jiao Tong University. I would say I definitely could not make it if there were not help and inspiration from many people, such as my parents' support, my supervisor's guidance, and many friends' advice and cooperation. Their supports and assistance gave me to get insight into my research area and to be more internationalized.

First of all, I am very appreciative of Shanghai Jiao Tong University. I think I had a very lack of basic knowledge about the network security realm. Still, they recognized and believed in my future potential as a researcher and offered a very helpful scholarship. Afterward, they provided very competent professors and teachers and composed such an excellent environment to step forward. Since I do extremely appreciate my academic period at Shanghai Jiao Tong Univ, my gratefulness for their support is enormously big also.

Subsequently, I do feel very grateful to my supervisor Cunqing Hua (化存卿) Professor. His professionalism kept inspiring me to work hard and make excellent outputs as much as possible. His analytical perspective made me find and realize my many parts of inadequacy and made me step forward. His teaching affected my core perspectives and core values; I am sure they will change me to make better decisions and to select the more advanced direction in the future. 老师真的，非常感谢您的详细的照顾。因为您的教导，我可能培养专业知识，顺利毕业了。

To seniors, Pegnwenlong Gu, Hao Dong, He Cai, Yue Bi, Rui Li helped me to adapt quickly to life in China and at the school. Ke Zhu and Zhutian Feng encouraged and guided me to step the procedures in our school as my classmates at the same grade. And other labmates, headed by Jihang Jian and Yiqing Zhu, propped up me to keep studying. 谢谢大家。

To my classmates from all parts of the world, Changyo and Jeonghyeok gave me strong motivation for my research and taught me a lot of things about computer techniques. Kevin, Giulia, Hafiz, Ayan, Annachiara, Tshegofatso, Stefan, Kenneth, Sarah, Daud, Renaldy, Tanvir, Alireza, Amin, Michiel, Aisha, Jorge, Luis, Zino, Claire, Remy, Liz, Jaehyeong, Jose, Iga, Irhidian, Adria, Ariful, Egemen, and Yasamin helped me to understand the divergence of every part of the world's culture, religion, and so on. And

we leaned on each other to get over the faced various challenges. Furthermore, the ISU family, headed by Hwapaow and Keong, enriched and made my meaningful campus life.

Last but not least, I could keep studying due to my parents' material and spiritual supports. When I face big challenges, the fact they are on my unconditional side encouraged me to overcome any hurdles. In spite of the limited wealth, they tried their best to give me more opportunities to experience and progress from the beginning of my life. I very much appreciate the destiny that I was born as their son. Thank you so much Younggon Kim, Mijung Kim.

Publications

[1] S. Kim, H. Cai, C. Hua, P. Gu, W. Xu and J. Park, ”Collaborative Anomaly Detection for Internet of Things based on Federated Learning,” 2020 IEEE/CIC International Conference on Communications in China (ICCC), Chongqing, China, 2020, pp. 623-628, doi: 10.1109/ICCC49849.2020.9238913.