Federated Deep Learning for Botnet Attack Detection in IoT Networks

S I POPOOLA

PhD 2022

Federated Deep Learning for Botnet Attack Detection in IoT Networks

SEGUN ISAIAH POPOOLA

A thesis submitted in partial fulfilment of the requirements of
Manchester Metropolitan University
for the degree of Doctor of Philosophy

Department of Engineering
Manchester Metropolitan University
in collaboration with Cyraatek Ltd

2022

Declaration of Authorship

I, Segun Isaiah POPOOLA, declare that this thesis titled, “Federated Deep Learning for Botnet Attack Detection in IoT Networks” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

Abstract

THE wide adoption of the Internet of Things (IoT) technology in various critical infrastructure sectors has attracted the attention of cyber attackers. They exploit the vulnerabilities in IoT to form a network of compromised devices, known as botnet, which is used to launch sophisticated cyber attacks against the connected critical infrastructure. Recently, researchers have widely explored the potentials of Machine Learning (ML) and Deep Learning (DL) to detect botnet attacks in IoT networks. However, there are still some challenges that need to be addressed in this area, which include the determination of optimal model hyperparameters, low classification performance due to imbalanced sample distribution in the training set, high memory space requirement for network traffic data storage, inability to detect zero-day attacks, and lack of data privacy. In order to address these problems, a Federated Deep Learning (FDL) method is developed for botnet attack detection in IoT-enabled critical infrastructure.

First, a hyperparameter optimisation method is developed for DL-based botnet attack detection in IoT networks to achieve high classification performance. The effectiveness of the method is evaluated using the Bot-IoT and N-BaIoT datasets, and the DL models achieved $99.99 \pm 0.02\%$ accuracy, $97.85 \pm 3.77\%$ precision, $98.72 \pm 2.77\%$ recall, and $97.72 \pm 4.51\%$ F1 score. Then, an oversampling algorithm is combined with DL models to improve the classification performance when the training data is highly imbalanced, without any significant increase in the overall computation time. This method improved the precision, recall, and F1 score of the DL models by $1.66 - 13.23\%$. Furthermore, a hybrid DL method is developed to reduce the amount of memory space required to store the network traffic data. This method reduced the memory space requirement for DL-based botnet attack detection by $86.45 - 98.26\%$. Finally, a FDL method, which also employed the hyperparameter optimisation, class balance, and memory space reduction methods, is developed to detect zero-day botnet attacks in IoT edge nodes, while preserving the data privacy of IoT users. The FDL models achieved high classification performance, and they had low communication overhead and low network latency.

Acknowledgements

First and foremost, all praise to God, the Father of my Lord Jesus Christ, for the privilege of formal education from primary school to the level of pursuing a doctoral degree. When natural circumstances were working against me and my back was against the wall, He made ways for me where there was no way. I am an embodiment of His mercies, grace, favour, provision, and faithfulness. His hands provided all that I needed for a successful completion of my studies. He gave me supernatural wisdom, knowledge, and understanding of deep things in my subject areas. To God alone be all the glory!

Special thanks to my supervisory team for their immense contributions to the successful completion of my study. My principal supervisor, Prof Bamidele Adebisi, connected me to the opportunities that funded my research as well as collaborations that improved the quality and international reputation of the work. My first supervisor, Prof Mohammad Hammoudeh, connected me to potential employability opportunities for my future career. The team provided an excellent academic and administrative leadership, mentorship, encouragement, inspiration, support, and insightful comments throughout the journey.

I want to express my sincere appreciation to the management of Cyraatek Ltd UK and the Faculty of Science & Engineering at the Manchester Metropolitan University for the full-time scholarship to carry out this study. The directors of Cyraatek Ltd UK, Mr Akinbami Ande and Dr Ruth Ande, provided the required technical and financial resources as well the conducive work environment for the research work. Also, I appreciate the technical support of Mr Fatai Batatunde Kassim; he setup and configured the high-performance computer and the software that was used for the research.

Finally, I would like to thank my parents, family, and friends for their understanding and support all the way. I appreciate all the support, contributions, and encouragement of my father, Mr Sunday Popoola, my mother, Mrs Olanike Popoola, my sister, Mrs Oluwatosin Nwaubani, and my brother, Oluwafunso Popoola. Also, I appreciate the family of Mr & Dr Olamide Jogunola for hosting me in their apartment on my arrival in the UK. I would like to also acknowledge the support of the Overseas Fellowship of Nigerian Christians (OFNC), especially the Manchester Branch. Special thanks to Dr Victor Nwegbu and his wife for their love and support for my family when we needed a bigger accommodation and when my wife was pregnant.

To my lovely family,

Inioluwa and Toluwanimi

I want to thank you so much for your love, prayers, encouragement, support, and understanding. You both sacrificed so much to enable me to reach this point and I could not have done it without you. I love you dearly You are the best.

Contents

List of Figures

List of Tables

List of Abbreviations

5G	Fifth Generation
6G	Sixth Generation
Adam	Adaptive Moment Estimation
ADSYN	Adaptive Synthetic
AE	Autoencoder
AI	Artificial Intelligence
ANN	Artificial Neural Network
AUC	Area Under Curve
BDS	Botnet Detection System
BLSTM	Bidirectional Long Short-Term Memory
BNN	Binarised Neural Network
BPTT	Back Propagation Through Time
C&C	Command and Control
CAE	Convolutional Autoencoder
CDL	Centralised Deep Learning
CNN	Convolutional Neural Network

CPBR	Consumer Privacy Bill of Rights
CPS	Cyber Physical System
DBM	Deep Boltzmann Machine
DBN	Deep Belief Network
DDoS	Distributed Denial of Service
DeAE	Denoising Autoencoder
DE	Data ex-filtration
DL	Deep Learning
DNN	Deep Neural Network
DoS	Denial of Service
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
DT	Decision Tree
ELU	Exponential Linear Unit
FDI	False Data Injection
FDL	Federated Deep Learning
FL	Federated Learning
FNN	Feedforward Neural Network
FN	False Negative
FP	False Positive
FTP	File Transfer Protocol

GBDT	Gradient Boosted Decision Tree
Gbps	Gigabit per second
GDPR	General Data Protection Regulation
GRU	Gated Recurrent Unit
GR	Gain Ratio
GWO	Grey Wolf Optimisation
HTTP	Hypertext Transfer Protocol
ICS	Industrial Control System
IEEE	Institute of Electrical and Electronics Engineers
IFG	Information Gain
IIoT	Industrial Internet of Things
IoHT	Internet of Healthcare Things
IoMT	Internet of Medical Things
IoT	Internet of Things
IRC	Internet Relay Chat
KL	Keylogging
kNN	k-Nearest Neighbour
LAE	Long Short-Term Memory Autoencoder
LAN	Local Area Network
LDL	Local Deep Learning
LR	Logistic Regression

LSTM	Long Short-Term Memory
MDPI	Multidisciplinary Digital Publishing Institute
MEC	Mobile Edge Computing
MLP	Multilayer Perceptron
ML	Machine Learning
NB	Naive Bayes
OCSVM	One-Class Support Vector Machine
OSELM	Online Sequential Extreme Learning Machine
OSF	Operating System Fingerprinting
P2P	Peer-to-Peer
PCA	Principal Component Analysis
PCC	Pearson Correlation Coefficient
PCE	Pearson Correlation and Entropy
PSO	Particle Swarm Optimisation
PV	Photovoltaic
RBM	Restricted Boltzmann Machine
ReLU	Rectified Linear Unit
RFMDA	Random Forest Mean Decrease Accuracy
RF	Random Forest
RMSprop	Root Mean Squared Propagation
RNN	Recurrent Neural Network

RO	Random Over-sampling
RT	Random Tree
RU	Random Under-sampling
SAE	Stacked Autoencoder
SGD	Stochastic Gradient Descent
SIoT	Satellite Internet of Things
SMOTE	Synthetic Minority Oversampling Technique
SNN	Self-normalising Neural Network
SSAE	Stacked Sparse Autoencoder
SS	Service Scanning
SVM	Support Vector Machine
t-SNE	t-distributed Stochastic Neighbour Embedding
Tbps	Terabit per second
TCP	Transmission Control Protocol
TN	True Negative
TOPSIS	Technique for Order of Preference by Similarity to Ideal Solution
TP	True Positive
UAV	Unmanned Aerial Vehicles
UDP	User Datagram Protocol
VAE	Variational Autoencoder
VIoT	Vehicular Internet of Things

XGBoost	Extreme Gradient Boosting
XMP	X-Mean clustering with Particle Swarm Optimisation

Nomenclature

Φ	Trainable model parameters
ψ	Optimiser
σ_h	Hidden layer activation function
σ_y	Output layer activation function
θ	Cross-entropy loss function
$\tilde{y}$	Predicted class label vector
b	Bias vector
c	Number of unique class labels
c	number of class labels in a data set
d	Feature dimensionality
e	Number of epochs
h	Hidden state vector
L	Cross-entropy loss
l_h	Hidden layer of a neural network
l_i	Input layer of a neural network
l_o	Output layer of a neural network
n	Total number of samples in a data set
n_b	Batch size

r	Learning rate
u	Hidden units in a neural network
W	Weight matrix
X	Network traffic feature matrix
x	Network traffic sample
X_{ma}	Network traffic feature matrix of a majority class
X_{mi}	Network traffic feature matrix of a minority class
y	True class label vector

Chapter 1

Introduction

The Internet of Things (IoT) paradigm enables physical objects in critical infrastructure[1] to connect and exchange useful information via the Internet for intelligent decision making with little or no human intervention. The IoT technology enables smart operations in critical national infrastructure sectors shown in Figure 1.1 [1]–[3]. For instance, in modern healthcare systems, IoT facilitates activity recognition, fitness assistance, vital sign monitoring, daily dietary tracking, and sleep monitoring [4]. Also, IoT-based solutions have been developed to combat the current COVID-19 pandemic [5], [6]. In smart grids, IoT facilitates a bidirectional flow of

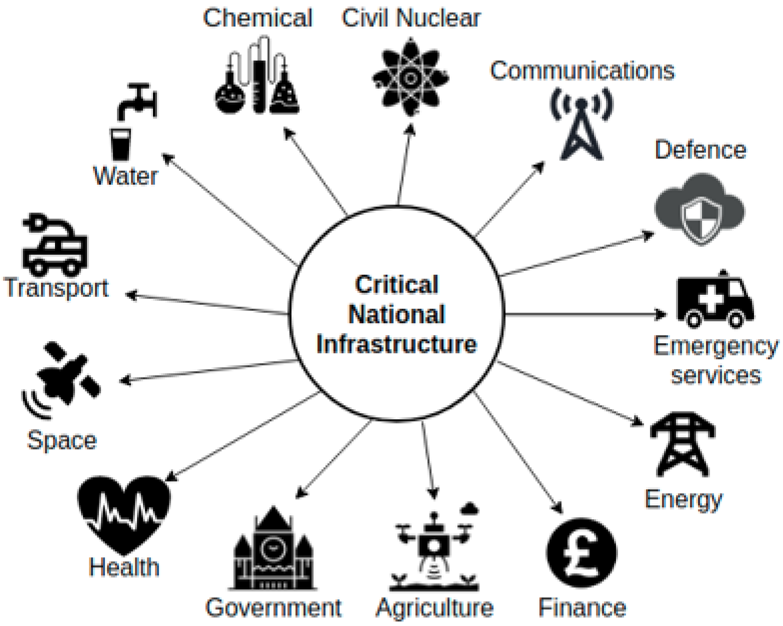


FIGURE 1.1: Critical national infrastructure sectors in the United Kingdom

[1] Critical infrastructure are the national infrastructure whose loss or compromise could affect the integrity or availability of essential services and national security [1].

electricity and data for efficient power delivery and energy management [7]. Electricity distribution companies install smart meters in buildings to measure energy consumption and the data is automatically transmitted to a cloud server. On the other hand, electricity consumers can manage their energy usage more efficiently. In agriculture, IoT enables livestock farmers to: monitor diseases, stress, feed intake, rumination, and weather condition; track and control of wildlife and whole herd; and automate milking as well as food and water supply [8]. In logistics, IoT facilitates the transportation, warehousing, loading, unloading, carrying, packaging, processing, and distribution [9]. In near future, the sixth generation (6G) wireless communication technology will expand the real-life deployment of emerging IoT applications such as Internet of Healthcare Things (IoHT), Vehicular IoT (VIoT) and autonomous driving, Unmanned Aerial Vehicles (UAV), Satellite IoT (SIoT), and Industrial IoT (IIoT) [10], [11].

Unfortunately, the increased inter-connectivity and the use of Industrial Control Systems (ICS) renders smart critical infrastructures vulnerable to cyber attacks. Most of the IoT devices that are in use today were developed with little or no consideration for cyber security [12]. Meanwhile, security breaches in IoT networks may lead to wider network congestion, loss of confidential information, corruption of sensitive data, financial loss, application downtime or even loss of lives in critical use cases [13]–[15]. Nowadays, hackers form a network of compromised IoT devices, known as botnet, to launch different types of cyber-attack against Internet-enabled infrastructures [16]–[21]. ICS and IoT devices have been proven to be easily hackable and remotely controllable to form IoT-based botnets [22], [23]. Successful exploitation of a single vulnerable IoT device can lead to leakage of sensitive information and serious security breaches in the wider IoT-enabled system [24]. This makes them an attractive target to diverse botnet attacks, especially when they are deployed in critical environments. In September 2016, *Mirai* botnet compromised several IoT devices to launch Distributed Denial of Service (DDoS) attack traffic of up to 620 Gigabit per second (Gbps) against a web server [25]. Also, in October 2016, a web-host and cloud service provider, *Dyn*, was hit with DDoS attack traffic of about 1.1 Terabit per second (Tbps). *Mozi* botnet was first discovered in October 2019. It accounted for close to 90% of the total IoT network traffic monitored by IBM Security X-Force from that time through June 2020; this incidence has increased IoT attack volume by 400%, compared to the total IoT attack cases in the last two years [26]. Cybercriminals can manipulate the energy market for a significant payoff of up to $24 million if they have access to 50,000 high-wattage IoT devices for just three hours a day, 100 days a year [27], [28]. With the on-going COVID-19 pandemic, corporate and IoT networks have become more vulnerable to botnet attacks because these networks are now being accessed remotely more often than before [29], [30]. Most recently, a new IoT Peer-to-Peer (P2P) botnet, named HEH, exploited insecure Telnet services on ports 23 and 232 to wipe out all the data in IoT devices using the brute force method [31]. Therefore, IoT-enabled critical infrastructure must be properly monitored and protected to detect and prevent botnet attacks.

Information and network protection mechanisms such as encryption, authentication, and access control may not be strong enough to protect IoT-enabled critical infrastructure against botnet attacks [32], [33]. Therefore, an efficient Botnet Detection System (BDS) is needed to complement existing security mechanisms. BDS will scan and monitor all the network traffic traces generated within IoT networks to detect botnet attacks. This system can be developed and installed at the gateways of IoT networks to alert network administrators and prevent botnet attacks. Machine Learning (ML) has become a popular method for intrusion detection in IoT networks [34]–[38]. Popular ML methods include Random Forest (RF), Support Vector Machine (SVM), Decision Tree (DT), k-Nearest Neighbour (kNN), Random Tree (RT), and Naive Bayes (NB) [39]. However, shallow neural networks cannot process big data effectively because they have a limited number of trainable parameters. Deep Learning (DL) is an advanced ML method that has more than one hidden layer in its neural network, and it learns the feature representation of training data using multiple levels of abstraction [40]. Common DL architectures include Deep Neural Network (DNN), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Deep Belief Network (DBN), Autoencoder (AE), and Restricted Boltzmann Machine (RBM) [41]. However, traditional DL-based BDS employ a centralised approach, where all the IoT edge nodes transmit their network traffic data to a central cloud server for model training. This approach does not preserve the privacy of the network device owners because it involves the aggregation of private network data from multiple sources. On the other hand, the classification performance of a DL model that is trained locally is limited to the network traffic data within a specific application.

Recently, strict laws such as the General Data Protection Regulation (GDPR)[2] and the Consumer Privacy Bill of Rights (CPBR)[3] were enacted to address data privacy concerns. Unfortunately, Centralised DL (CDL) method does not guarantee the privacy and security of IoT devices because it involves the transmission of network traffic features from all participating IoT devices to a central cloud server. Specifically, the use of a third-party cloud server for CDL will introduce a high risk of privacy leakage in IoT systems because the network traffic features may contain sensitive information about the owners of the IoT devices [42]–[45]. The violation of data privacy protection regulations can lead to a serious penalty. In the case of GDPR, the fine of data privacy breach could be as high as €10 million[4]. Federated Learning (FL) is an advanced Artificial Intelligence (AI) technique which seeks to protect the privacy of participating nodes without a significant compromise in the classification performance and the generalisation ability of the DL models [46]–[49]. Basically, this concept involves the training of Local DL (LDL) models with private data sets, and the aggregation of their parameters in a central cloud server. Therefore, this research seeks to explore the potentials of Federated Deep Learning (FDL) for privacy-preserving botnet attack detection in IoT-enabled critical infrastructure.

1.1 Research Questions

This thesis contributes to the current research efforts that seek to develop efficient methods for botnet attack detection in IoT-enabled critical infrastructure by answering the following questions:

(a) What are the state-of-the-art ML, DL, and FL methods that have been proposed for botnet attack detection in IoT networks? What are the current research gaps in the ML, DL, and FL-based botnet attack detection methods?

(b) How can the right set of hyperparameters (the number of hidden layers, the number of hidden units, the learning rate, the optimiser, the activation function, the batch size, and the number of epochs) be determined for efficient DL-based botnet attack detection in

[2] https://gdpr.eu/data-privacy/

[3] https://www.congress.gov/bill/116th-congress/senate-bill/2968/text

[4] https://gdpr-info.eu/issues/fines-penalties/

IoT-enabled critical infrastructure? Considering both binary and multi-class classification scenarios, what are the optimal sets of hyperparameters when the Bot-IoT and N-BaIoT data sets are used for DL model development? What is the classification performance of the optimal DL models based on accuracy, precision, recall, and F1 score? What is the computational efficiency of the optimal DL models in terms of training time and testing time?

(c) How can the classification performance of the DL-based botnet detection method be improved when the IoT network traffic data that is available for training is highly imbalanced?

(d) How can the memory space required for network traffic data storage in IoT central cloud server be reduced without compromising the classification performance of the DL-based botnet detection models?

(e) How can zero-day botnet attacks be detected with high classification performance, low communication overhead, low memory requirement, and low latency without compromising the data privacy of IoT network users? How can such an efficient intrusion detection be achieved in an heterogeneous network environment?

1.2 Research Aim

The aim of this research is to develop FDL method for efficient botnet attack detection in IoT-enabled critical infrastructure.

1.3 Research Objectives

The specific objectives of this research are to:

(a) conduct a comprehensive literature review to identify and understand the recent classes of botnet attacks, the most relevant network traffic data sets, and the state-of-the-art ML and DL methods that researchers have proposed for botnet attack detection in IoT networks;

(b) develop a hyperparameter optimisation algorithm for DL-based botnet attack detection models to improve their classification performance i.e. accuracy, precision, recall, and F1 score;

(c) develop an oversampling algorithm to improve the classification performance of DL-based botnet attack detection models when the training data is highly imbalanced, without any significant increase in the overall computation time;

(d) develop a hybrid DL algorithm to reduce the feature dimensionality of network traffic data, consequently reducing the size of network bandwidth required to transmit the data from the IoT edge nodes to the central cloud server, and also reducing the amount of memory space required to store the data on the central cloud server;

(e) develop a FDL algorithm to detect zero-day botnet attacks in IoT-enabled critical infrastructure with high classification performance, low communication cost, low network latency, and data privacy preservation guarantee.

1.4 Main Contributions

The main contributions of this thesis are summarised as follows:

(a) a hyperparameter optimisation method is proposed to determine the most appropriate combination of the numbers of hidden layers and hidden units, the learning rate, the optimiser, the activation function, the batch size, and the number of epochs for DL-based botnet attack detection models to achieve an optimal classification performance. The proposed method helps in choosing the best sets of hyperparameters that are most suitable to train efficient DNN-based botnet attack detection models to perform binary, 5-class, 10-class, and 11-class classification of network traffic data using the Bot-IoT and N-BaIoT data sets. In all the classification scenarios, the models achieved 99.99 ± 0.02% accuracy, 97.85 ± 3.77% precision, 98.72 ± 2.77% recall, and 97.72 ± 4.51% F1 score. This contribution was published as chapter in Springer's book [50] and another research paper is currently under peer-review for publication in IEEE Internet of Things Journal;

(b) a framework, which combines Synthetic Minority Oversampling Technique (SMOTE) with a DL architecture, is proposed to improve the classification performance of DL-based botnet attack detection models when the network traffic data in the training set is highly imbalanced. Considering the highly imbalanced Bot-IoT data set, the proposed oversampling method helps in generating a total of 52139 synthetic minority samples in less than 950 milliseconds. This improved the precision, recall, and F1 score of DNN, RNN, Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU) models by $2.07 - 13.23\%$, $1.66 - 10.56\%$, and $2.66 - 12.43\%$, respectively. This contribution was published in Elsevier's Computers & Electrical Engineering journal [51] and MDPI's Sensors journal [52];

(c) a hybrid DL method, which employs LSTM Autoencoder (LAE) and a Bidirectional LSTM (BLSTM) architectures, is proposed to reduce the feature dimensionality of the network traffic data without any significant adverse effect on the classification performance. Consequently, the amount of memory space required to store the data on the central cloud server is reduced. The hybrid DL model reduced the feature dimensionality of the Bot-IoT data set by $72.97 - 78.38\%$, and the memory space requirement reduced by $86.45 - 89.19\%$. For the binary, 5-class, and 11-class classification scenarios, the hybrid DL model achieved $99.90 \pm 0.16\%$ accuracy, $99.07 \pm 0.62\%$ precision, $98.25 \pm 1.11\%$ recall, and $98.63 \pm 0.63\%$ F1 score. The hybrid DL model reduced the feature dimensionality of the N-BaIoT data set by $93.04 - 96.52\%$, and the memory space requirement reduced by $96.52 - 98.26$. For the binary and 10-class classification scenarios, the hybrid DL model achieved $99.94 \pm 0.05\%$ accuracy, 99.71 ± 0.29 precision, $99.69 \pm 0.34\%$ recall, and $99.70 \pm 0.32\%$ F1 score. This contribution was published in IEEE Internet of Things journal [53] and MDPI's Electronics journal [54];

(d) a FDL method, which combines FL and DL algorithms, is proposed for zero-day botnet attack detection in IoT edge devices to reduce data transmission cost, reduce network latency, and preserve the privacy of IoT network users. FDL models are developed with the 11-class BoT-IoT and 10-class N-BaIoT datasets to evaluate the effectiveness of the proposed method. The FDL model achieves high classification performance, preserves the data privacy of critical infrastructure users, has low communication overhead, requires low memory space for the storage of network traffic data in the edge nodes, and has low network latency. The FDL model outperformed the CDL and LDL models. This contribution was published in IEEE Internet of Things journal [55] and

IEEE Vehicular Technology conference proceeding [56].

1.5 Thesis Organisation

The remaining parts of this thesis are organised as follows: In Chapter 2, the overview of the concepts of IoT and botnet is presented. Then, the state-of-the-art ML, DL, and FL methods that were proposed for botnet attack detection in IoT networks were reviewed. Furthermore, the network traffic data sets that were used to develop ML-based, DL-based, and FL-based botnet attack detection in IoT networks are reviewed. Finally, the current research gaps in this area of research are identified and discussed. In Chapter 3, a model hyperparameter optimisation algorithm is proposed for DL-based botnet attack detection in IoT-enabled critical infrastructure. Optimised DNN models are developed with the Bot-IoT and N-BaIoT dataset to evaluate the effectiveness of the optimisation method in binary and multi-class classification scenarios. In Chapter 4, a framework named SMOTE-DL is proposed to improve the classification performance of DL-based botnet attack detection in IoT-enabled critical infrastructure whenever the network traffic data in the training set is highly imbalanced. SMOTE-DNN, SMOTE-RNN, SMOTE-LSTM, and SMOTE-GRU models are developed with the highly imbalanced 11-class network traffic data in the Bot-IoT dataset to evaluate the effectiveness of the proposed framework. The performance of these models is compared with that of the DNN, RNN, LSTM, and GRU models. In Chapter 5, a hybrid DL algorithm named LAE-BLSTM is proposed for memory-efficient botnet attack detection in IoT-enabled critical infrastructure. LAE method reduces the feature dimensionality of network traffic data in a central cloud server or an IoT edge node, while BLSTM method learns the feature representation of the low-dimensional network traffic data to classify benign network traffic and botnet attack traffic correctly. LAE-BLSTM models are developed with the Bot-IoT and N-BaIoT datasets to evaluate the effectiveness of the proposed algorithm. In Chapter 6, a FDL method is proposed for zero-day botnet attack detection in IoT-enabled critical infrastructure. Zero-day botnet attack scenarios are modelled with the Bot-IoT and N-BaIoT datasets. The LAE-BLSTM method is used to train local models with private network traffic data in multiple IoT edge nodes. In a model parameter cloud server, Federated Averaging (FedAvg) algorithm aggregates the local model updates from all the IoT edge nodes to form a global FDL model that preserves the data privacy of IoT-enabled critical infrastructure users. In Chapter 7, the major findings of the research is summarised, and recommendations are proposed for future work.

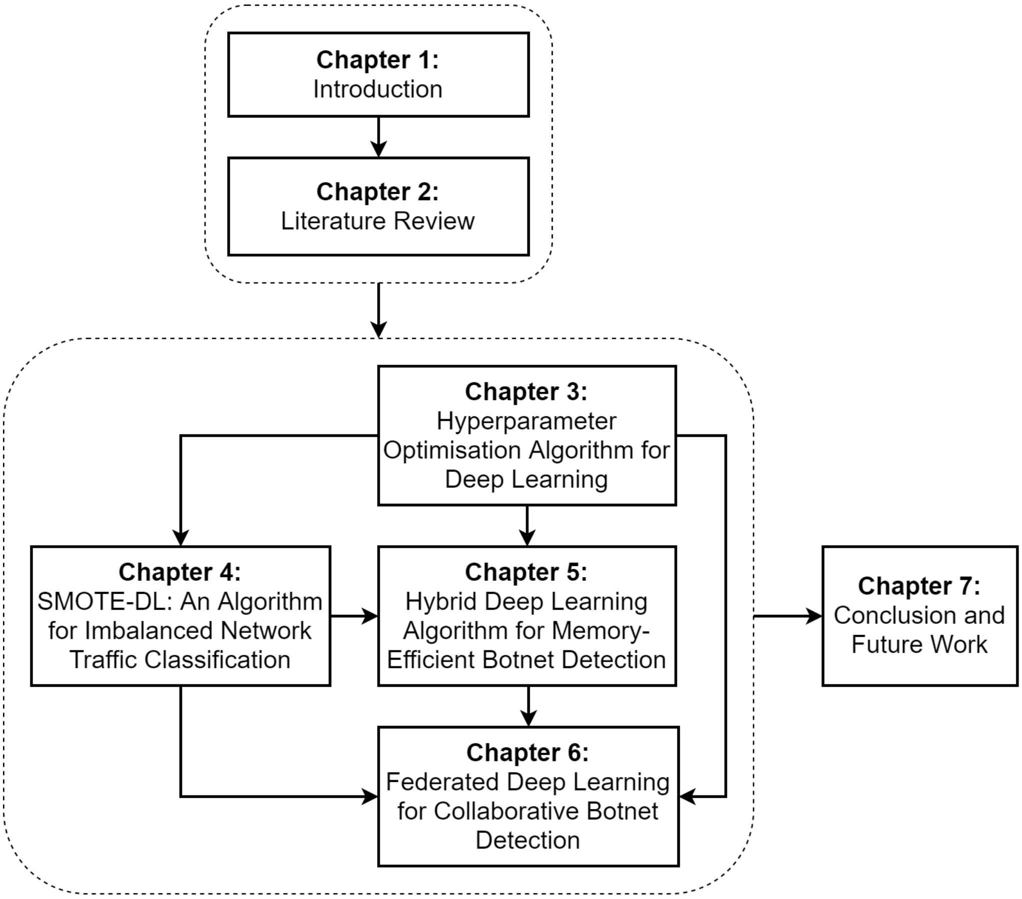

Chapter 1:
Introduction
Chapter 2:
Literature Review
Chapter 3:
Hyperparameter Optimisation Algorithm for Deep Learning
Chapter 4:
SMOTE-DL: An Algorithm for Imbalanced Network Traffic Classification
Chapter 5:
Hybrid Deep Learning Algorithm for Memory-Efficient Botnet Detection
Chapter 6:
Federated Deep Learning for Collaborative Botnet Detection
Chapter 7:
Conclusion and Future Work

FIGURE 1.2: Thematic organisation of the thesis

Figure 1.2 shows the connection and the flow of the chapters in the thesis. The method proposed in Chapter 3 is used for model hyperparameter optimisation in Chapters 4, 5, and 6. The framework proposed in Chapter 4 is used to improve the classification performance of DL models in Chapters 5 and 6 when the network traffic data in the training set is highly imbalanced. The hybrid DL algorithm in Chapter 5 is used to achieve memory-efficient botnet attack detection in Chapter 6. All the main findings in Chapters 3, 4, 5, and 6 are summarised in Chapter 7.

1.6 List of Publications

1.6.1 Journal Papers

1. **Popoola, S. I.**, Ande, R., Adebisi, B., Gui, G., Hammoudeh, M., Jogunola, O. "Federated Deep Learning for Zero-Day Botnet Attack Detection in IoT Edge Devices", *IEEE Internet of Things Journal*, 9(5), 3930-3944, 2022.

2. **Popoola, S. I.**, Adebisi, B., Hammoudeh, M., Gui, G., Gacanin, H. "Hybrid Deep Learning for Botnet Attack Detection in the Internet-of-Things Networks", *IEEE Internet of Things Journal*, 8(6), 4944-4956, 2021.

3. **Popoola, S. I.**, Adebisi, B., Ande, R., Hammoudeh, M., Anoh, K., Atayero, A. A. "SMOTE-DRNN: A Deep Learning Algorithm for Botnet Detection in the Internet-of-Things Networks", *Sensors*, 21(9), 2985, 2021.

4. **Popoola, S. I.**, Adebisi, B., Ande, R., Hammoudeh, M., Atayero, A. A. "Memory-Efficient Deep Learning for Botnet Attack Detection in IoT Networks", *Electronics*, 10(9), 1104, 2021.

5. **Popoola, S. I.**, Adebisi, B., Hammoudeh, M., Gacanin, H., Gui, G. "Stacked recurrent neural network for botnet detection in smart homes", *Computers & Electrical Engineering*, 92, 107039, 2021.

1.6.2 Conference Paper

1. **Popoola, S. I.**, Gui, G., Adebisi, B., Hammoudeh, M., Gacanin, H. "Federated Deep Learning for Collaborative Intrusion Detection in Heterogeneous Networks", *2021 IEEE 94th Vehicular Technology Conference: VTC 2021-Fall*, 2021, pp. 1-6, doi: 10.1109/VTC2021-Fall52928.2021.9625505.

1.6.3 Book Chapter

1. **Popoola, S. I.**, Ande R., Kassim, B. F., Adebisi, B. "Deep Bidirectional Gated Recurrent Unit for IoT Botnet Detection in Smart Homes”, *Machine Learning and Data Mining for Emerging Trend in Cyber Dynamics: Theories and Applications*; Springer: Cham, Switzerland, 2021; p. 29.

1.6.4 Submitted Journal Papers (Under review)

1. **Popoola, S. I.**, Adebisi, B., Hammoudeh, M. "Optimizing Deep Learning Hyperparameters for Botnet Attack Detection in IoT Network", *IEEE Internet of Things Journal*.
2. **Popoola, S. I.**, Ande, R., Atayero, A. A., Badejo, J. A., Hammoudeh, M., Gui, G., Adebisi, B., Gui, G. “Optimized Lightweight Federated Learning for Botnet Detection in Smart Critical Infrastructure", *IEEE Transactions on Industrial Informatics*.

Chapter 2

Concepts and Literature Review

2.1 Internet of Things and Botnet Attacks

IoT is one of the main technologies in the fourth industrial revolution (Industry 4.0) [57]–[59]. It connects homes, industries, government, and businesses to a large-scale of computers, smart devices, sensors, vehicles, and industrial machines in smart cities. Figure 2.1 shows the basic architecture of IoT, which comprised the perception layer, the network layer, and the application layer. Smart sensors and actuators are deployed at the perception layer to collect relevant information. The network layer transmits the data from the perception layer to the application layer, where it is processed for intelligent decision making. A comprehensive survey on the enabling technologies, protocols, and applications of IoT can be found in [60].

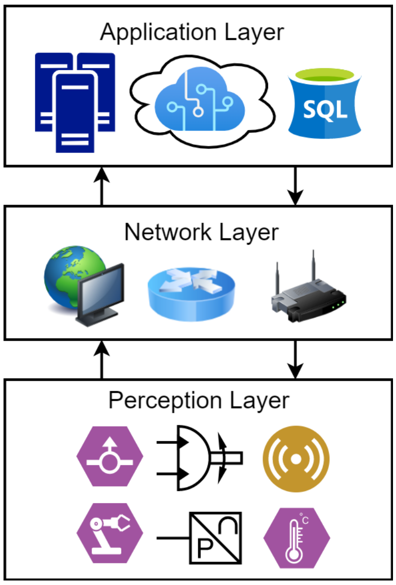


FIGURE 2.1: Basic architecture of IoT

According to Cisco's Annual Internet Report, 14.7 billion IoT devices will be connected to the Internet by 2023 [61]. More than 25.4 billion IoT devices will be connected to the Internet in 2030 [62], and IoT market is expected to worth about $1.6 trillion by 2025 [63]. The maturity of IoT and its wide adoption in critical infrastructure make it an attractive target for cyber-attackers. IoT networks and devices are subject to high security risks due to the lack of IoT standardization and interoperability, irregular security updates in many legacy IoT devices, low priority for security in the development cycle of IoT devices, inability to implement strong encryption mechanism due to power and memory constraints in IoT devices, weak login credentials, availability of open ports and lack of IoT-optimised intrusion detection and prevention systems [64], [65].

IoT has become the primary target of malicious botnet operators due to their proliferation and distributed nature [66]–[68]. Cyber attackers use this complex hacking technique to propagate malware and launch cyber attacks against IoT-enabled systems. The first malicious botnet, named *Eggdrop*, uses the concept of Internet Relay Chat (IRC) technology to launch Denial of Service (DoS) and DDoS attacks against IRC users and servers [69]. *GTbot* is another example of an IRC-based botnet [70]. Botnets became a major threat to Internet security when *AgoBot* [71] and *SDBot* [72], which employ a complex communication protocol, emerged. Other sophisticated botnets have been reported in recent times including *Mirai* [24], [25] and *BASHLITE* [73], [74].

In a typical botnet, a cyber attacker, also known as a botmaster, controls the activities of the bots remotely using a Command and Control (C&C) communication channel, as shown in Figure 2.2. The life cycle of a botnet involves five phases, namely initial injection, secondary injection, connection, malicious activities, and maintenance and upgrading [67]. First, the botmaster infect IoT devices with malware. Then, the infected devices download malware binary files from a specific network database using IRC, File Transfer Protocol (FTP), Hypertext Transfer Protocol (HTTP) or P2P communication protocols. The new bots establish connections with the C&C server to receive instructions and updates. The botmaster instructs the bots to perform malicious activities. Finally, the botmaster maintains its hold on the bots by updating the malware frequently. Advanced botnets employ P2P and hybrid C&C architectures to avoid single point of failure and evade detection [75], [76]. A P2P botnet has no dedicated C&C server, while a

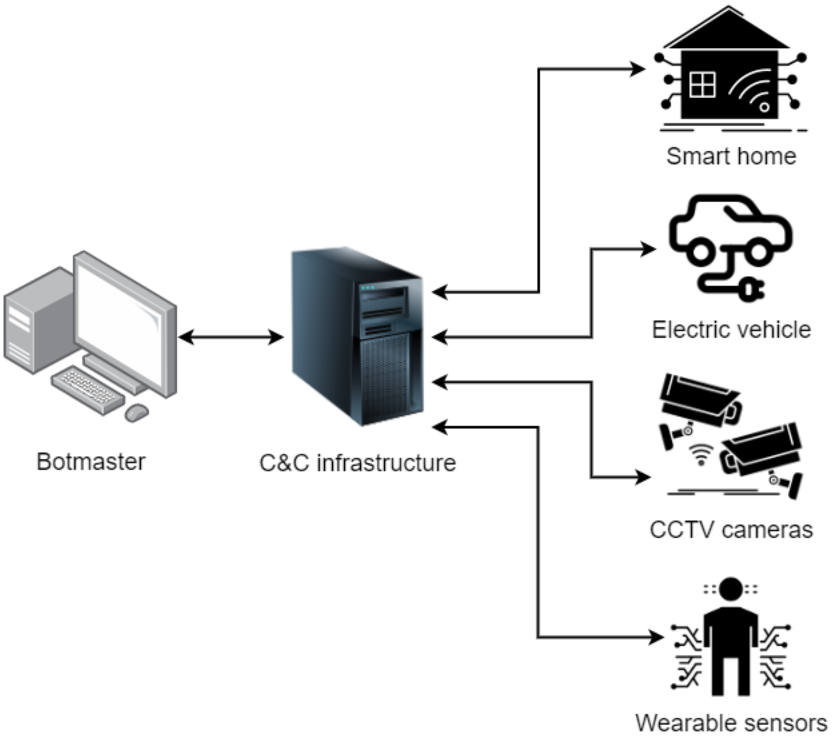


FIGURE 2.2: Basic architecture of a botnet

hybrid botnet combines both centralised and P2P architectures. Botnets have significantly widened the attack vulnerability landscape of IoT [77].

2.2 Datasets for Botnet Detection in IoT Network

Although several datasets are available for network intrusion detection, they have various challenges, including lack of reliable labels, low attack diversity, redundancy of network traffic, and missing ground truth. For instance, KDD Cup99 and NSL-KDD datasets are popularly used, but they are outdated, and they do not reflect current normal and attack scenarios [78], [79]. The application of the DEFCON-8 dataset is limited because of the low number of benign traffic samples [78]. The attack scenarios in the UNIBS dataset are limited to DoS; the network traffic data are presented in packets with no extracted features, and the labels were not provided [80]. CAIDA datasets have no ground truth information about the attack samples [80]. Network traffic samples in the LBNL dataset were not labeled, and the features were not extracted from the packet files [80]. Attack samples in the UNSW-NB15 dataset were generated in a synthetic environment [81]. ISCX and CICIDS2017 datasets were generated based on the concept of profiling, and this can be due to their innate complexity. Also, the ground truth of these datasets is not available to enhance the labeling process. Not much information is given about the botnet scenarios that were used in most datasets [78], [80]–[82]. Also, IoT network traffic data was not included in related datasets [78], [81], [83]–[86].

Bot-IoT dataset [87] is publicly and freely available for cyber security research. It contains benign IoT network traffic and four botnet attack scenarios including DoS, DDoS, reconnaissance (Recon), and data theft (Theft). The testbed that generated the benign IoT network traffic data comprised a weather station, a smart fridge, motion-activated lights, a remote-controlled garage door, and a smart thermostat. Koroniotis *et al.* [87] proposed a method for network packet capturing and feature extraction. In this method, network packets were captured using Tshark[1] while network traffic features were extracted using Argus[2]. Also, new features were generated based on transaction flows of network connections over a sliding window of 100. Forty-three features were extracted from a network packet to describe the behaviour of a network traffic sample. The list and description of these features are available in [87]. This data set has 477 benign IoT network traffic samples and 3,668,045 botnet attack samples. Figure 2.1 shows the total network traffic samples in the Bot-IoT dataset. These samples can be categorised into binary classes, 5 classes, or 11 classes. In the binary classification scenario, the samples are categorised as benign (Norm) or malicious (Attack) traffic. In the 5-class classification scenario, the samples are categorised as DDoS, DoS, Norm, Recon, or Theft traffic. In the 11-class classification scenario, the samples are categorised as DDoS-HTTP (DD-H), DDoS-TCP (DD-T), DDoS-UDP (DD-U), DoS-HTTP (D-H), DoS-TCP (D-T), DoS-UDP (D-U), Norm, Operating System Fingerprinting (OSF), Service Scanning (SS), data ex-filtration (DE), or keylogging (KL) traffic.

N-BaIoT dataset [74] is also publicly and freely available for cyber security research. The IoT testbed that generated this data set comprised two doorbells, a thermostat, a baby monitor, four security cameras, and a webcam. These commercial IoT devices were infected with *Mirai* and BASHLITE botnets, and 115 statistical features that represent the behaviour snapshots of the network traffic were extracted from the network packets over several temporal windows. The details of the data collection and the

[1] https://www.wireshark.org/docs/man-pages/tshark.html

[2] https://openargus.org/

TABLE 2.1: Total network traffic samples in the Bot-IoT dataset

Type	Class	Number of samples
2-class	Norm	477
	Attack	3668045
5-class	DDoS	1926624
	DoS	1650260
	Norm	477
	Recon	91082
	Theft	79
11-class	DD-H	989
	DD-T	977380
	DD-U	948255
	D-H	1485
	D-T	615800
	D-U	1032975
	Norm	477
	OSF	17914
	SS	73168
	DE	6
	KL	73

feature extraction can be found in [74]. This data set contains benign IoT network traffic and IoT botnet scenarios. Figure 2.2 shows the total network traffic samples in the N-BaIoT dataset. These samples can be categorised into binary classes or 10 classes. In the binary classification scenario, the samples are categorised as Norm or Attack traffic. In the 10-class classification scenario, the samples are categorised as Norm, *g_combo*, *g_junk*, *g_scan*, *g_udp*, *m_ack*, *m_scan*, *m_syn*, *m_udp*, or *m_udpp* traffic.

TABLE 2.2: Total network traffic samples in the N-BaIoT dataset

Type	Class	Number of samples
2-class	Norm	555932
	Attack	5646824
10-class	Norm	555932
	g_combo	515156
	g_junk	261789
	g_scan	255111
	g_udp	946366
	m_ack	643821
	m_scan	537979
	m_syn	733299
	m_udp	1229999
	m_udpp	523304

Therefore, all the models in this thesis are trained, validated, and tested with the network traffic data in the Bot-IoT and N-BaIoT datasets.

2.3 Centralised Learning for Botnet Detection

Botnet attack detection in IoT networks can be formulated as a classification problem. For binary classification, each sample in a network traffic packet is classified as either benign or malicious based on certain pre-defined features. On the other hand, the specific category of botnet attack is identified in multi-class classification. Different ML and DL models have been developed to classify network traffic data in IoT networks. These models learn the discriminating features of benign traffic and malicious traffic using different architectures. With the advent of cloud computing, each IoT device transmits its data to a central server on the Cloud, where different pre-processes and analyses can be performed on the aggregated data. In view of this, CDL method has been extensively proposed for network-based botnet attack detection in large IoT network traffic data with good classification performance. For example, Apruzzese et al [88] proposed a method that can prevent adversarial attacks using Deep Reinforcement Learning (DRL). Also, Zhao et al [89] proposed a lightweight dynamic AE network method for network intrusion detection in resource-constrained devices of a wireless sensor network. Shafiq *et al* [90] proposed a bijective soft set method to select the most effective ML algorithm among BN, DT, NB, RF, and RT for binary classification. Ferrag *et al* [91] combined REPTree, JRip, and Forest PA algorithms to produce a hybrid model. The first two models were trained to perform binary classification. The outputs of these models were combined with the original feature set to train the third model for 11-class classification. It took 195.5 seconds to train the model with the samples in the training set, and it spent 2.27 seconds to classify the samples in the testing set. The hybrid model achieve good classification performance. However, 22.53% of the samples in the D-H class were misclassified.

However, there are still some challenges that need to be addressed, which include the determination of optimal model hyperparameters, low classification performance due to imbalanced sample distribution in the training set, high memory space requirement, inability to detect zero-day attacks, and lack of data privacy.

2.3.1 Model Hyperparameter Selection

The classification performance of a ML/DL model largely depends on the selection of the right set of hyperparameters [50]. For a neural network, the hyperparameters include the numbers of hidden layers and the hidden units, the learning rate, the optimiser, the activation function, the batch size, and the number of epochs.

Koroniotis *et al* [87] developed RNN and LSTM models, each of which comprised four hidden layers with 20, 60, 80, and 90 hidden units, respectively. The models employed *tanh* activation function at the hidden layers, and they were trained using a batch size of 100 and 4 epochs. However, the authors did not report the learning rate and the optimiser that were used for the training. Koroniotis *et al* [92] developed DNN model, which had six hidden layers with 20, 40, 60, 80, 40, and 10 hidden units, respectively. The model was trained with a learning rate of 0.0015, Rectified Linear Unit (ReLU) activation function, a batch size of 732, and 12 epochs. However, the authors did not report the optimiser that was used for the training.

Ibitoye *et al* [93] developed Feedforward Neural Network (FNN) and Self-normalising Neural Network (SNN) models, which had three hidden layers with 16 hidden units each. The FNN model employed ReLU activation function, while the SNN model employed Scaled Exponential Linear Unit (SELU) activation function. However, the authors did not report the learning rate, optimiser, batch size, and the number of epochs. Susilo and Sari [94] trained DNN and CNN models with ReLU activation function and Adam optimiser. The authors investigated the effectiveness of different batch sizes (32, 64, and 128) and different number of epochs (10, 30, and 50). However, there was no clear information about the number of hidden layers and the learning rate.

Ferrag *et al* [41] developed three deep discriminative models (DNN, RNN, and CNN) and four generative models (RBM, DBN, DBM, and AE). Each of the models had a single hidden layer with 100 hidden units. The models were trained using sigmoid activation function, a learning rate of 0.5, a batch size of 1000, and 100 epochs. However, the authors did not report the optimiser that was used for the training. In another work, Ferrag *et al* [95] developed RNN model, which comprised a single hidden layer with 60 hidden units. The model was trained using a batch size of 100 and 5 epochs. However, the details of the learning rate, optimiser, and activation function were not provided. Alkadi *et al* [96] developed BLSTM model, but the information about the hyperparameters was not provided.

Ge *et al* [97] developed DNN model, which had three hidden layers with 512 hidden units each. The model was trained with ReLU activation function, Adam optimiser, a batch size of 500, a learning rate of 0.001, and 20 epochs. However, ten different binary classification scenarios were considered, where the Attack class contained only one class of botnet attack in each scenario. Also, in the multi-class classification scenario, the DDoS and DoS attack samples were grouped together to form a four-class classification scenario.

Currently, there is no definite procedure to justify the selection of an optimal set of hyperparameters for DL-based botnet attack detection in IoT networks. Therefore, in Chapter 3, a hyperparameter optimisation method is developed to determine the most appropriate combination of the numbers of hidden layers and hidden units, the learning rate, the optimiser, the activation function, the batch size, and the number of epochs for DL-based botnet attack detection models to achieve an optimal classification performance. The proposed method will help in choosing the best sets of hyperparameters that are most suitable to train efficient DNN-based botnet attack detection models to perform binary, 5-class, 10-class, and 11-class classification of network traffic data using the Bot-IoT and N-BaIoT datasets.

2.3.2 Class Imbalance in the Training Set

Similar to most real-life classification tasks [98], [99], network traffic data follows a long tail distribution [87]. For instance, samples of network traffic generated by botnets will be far more than those generated by real IoT devices in case of DDoS attack. In such case, the combined network traffic data available for ML and DL model training will be imbalanced. In ML and DL methods, training data is considered to be highly imbalanced when the class imbalance ratio[3] is greater than 10:1 [100]. Class imbalance problem affects the effectiveness of state-of-the-art ML and DL methods such that the models are biased in favour of class(es) with majority samples [101]. That is, a large percentage of samples in the minority class(es) may not be correctly classified. Therefore, the traditional ML/DL models may not be able to

[3] Class imbalance ratio of a given class is the number of samples in the class divided by the total number of samples in the remaining classes

handle class imbalance problem in the network traffic data generated by IoT-enabled critical infrastructure.

TABLE 2.3: Sample distribution of network traffic data in the training set of Bot-IoT dataset

Ref.	**Class**										
	DD-H	**DD-T**	**DD-U**	**D-H**	**D-T**	**D-U**	**Norm**	**OSF**	**SS**	**DE**	**KL**
[102]	594	498602	484127	942	317899	526487	4000	9002	36700	102	106
[41]	1582	1563808	1517208	2376	985280	1652759	7634	28662	117069	94	1175
[91]	1582	1563808	1517208	2376	985280	1652759	7634	28662	117069	94	1175
[103]	786	781468	759163	1191	492581	826475	385	14101	51351	5	66
This thesis	**588**	**586393**	**568760**	**906**	**369965**	**619414**	**290**	**10795**	**43949**	**4**	**48**

Table 2.3 shows the degree of class imbalance in the training data used for model development. Class imbalance is the ratio of the total number of samples in a particular class to the total number of samples in all other classes. Although the methods in related works considered the effect of class imbalance on classification performance, the degree of class imbalance was limited to $1 : 25528$ and $1 : 62527$ in 5-class and 11-class classification scenarios, respectively. In this thesis, the models handle high-class imbalance in 5-class and 11-class classification scenarios with class imbalance ratio of $1 : 45049$ and $1 : 641967$. Also, the effectiveness of these methods was not assessed in all the three classification scenarios (i.e. binary, 5-class, and 11-class). In this thesis, the models' performance was evaluated in all three classification scenarios.

SMOTE is a method that can effectively handle class imbalance problem in training data, but it must be combined with the right classifier. Pokhrel et al [104] proposed SMOTE-kNN to address the class imbalance problem in botnet detection. However, the study focused on binary classification, and the IR in the training data was 1:208. Bagui and Li [105] studied the effects of Random Under-sampling (RU), Random Over-sampling (RO), RU-RO, RU-SMOTE, and RU with Adaptive Synthetic (ADSYN) methods on the performance of the ANN model. Qaddoura et al [106] combined SVM-SMOTE with DNN to handle a class imbalance in binary classification. Derhab et al [107] employed the combination of SMOTE and temporal CNN to address the class imbalance in 5-class classification. However, SMOTE method has not been previously combined with the RNN, LSTM, and GRU models. Also, previous applications of SMOTE focused on binary and 5-class classification, but none of them applied it to solve the 11-class classification problem.

Table 2.3 shows that the distribution of network traffic samples in the training set is highly imbalanced across the 11 classes. The numbers of samples in the minority classes (DD-H, D-H, Norm, DE, and KL) are relatively fewer than those in the majority classes (DD-T, DD-U, D-T, D-U, OSF, and SS). For majority classes, high class imbalance in the training set degraded the classification performance of state-of-the-art ML and DL models. Therefore, state-of-the-art ML and DL models may not detect DD-H, D-H, Norm, DE, and KL correctly in IoT networks. In this thesis, the numbers of samples in the Norm, DE, and KL classes are relatively few compared to those in the previous studies. So, the class imbalance problem in the present study is more challenging. Also, in the previous related works [41], [91], [96], [102], the recall values for the Norm class were not reported, and the authors did not present the accuracy, precision, and F1 score of the ML/DL models for each of the 11 classes.

Therefore, in Chapter 4, a framework, which combines SMOTE with each of the DNN, RNN, LSTM, and GRU architectures, is proposed to improve the classification performance of DL-based botnet attack detection models when the network traffic data in the training set is highly imbalanced.

2.3.3 Feature Dimensionality Reduction

DL is an efficient method for botnet attack detection. In order to develop an efficient DL method for botnet detection in IoT networks, sufficiently large network traffic information is needed to guarantee efficient classification performance [108]. However, the volume of network traffic data and memory space required is usually large. Therefore, it is almost impossible to implement the DL method in memory-constrained IoT devices. Processing and analyzing high-dimensional network traffic data can lead to *curse of dimensionality* [109]. High-dimensional data processing is complex and requires huge computational resources and storage capacity [110], [111]. IoT devices do not have sufficient memory space to store big network traffic data required for DL. Given a high feature dimensionality in the training data, high network bandwidth and a large memory space will be needed to transmit and store the data, respectively in IoT back-end server or cloud platform for DL. For example, the memory sizes of the training data in the Bot-IoT and N-BaIoT datasets are more than 600 MB and 3 GB, respectively. Therefore, there is a need for end-to-end DL-based botnet detection method that will reduce high dimensionality of big network traffic features and also detect complex and recent botnet attacks accurately based on low-dimensional network traffic information.

Feature dimensionality reduction is mostly achieved by applying either linear or non-linear transformation technique to high-dimensional feature set. Principal Component Analysis (PCA) [112] is one of the common linear transformation methods while kernel methods [113], spectral methods [114] and DL methods [115] employ non-linear transformation techniques. AE is an unsupervised DL method that produces latent-space representation of input data at the hidden layer. Different AE architectures have been proposed to reduce the feature dimensionality in most popular network intrusion datasets. These methods were implemented and evaluated with the network traffic data in publicly available datasets which include KDD-Cup99 [116]–[123], NSL-KDD [116], [124]–[127], UNSW-NB15 [117], [126], [128] and CICIDS2017 [129]. Table 2.4 shows the AE-based feature dimensionality reduction techniques in the literature. The original feature dimensionality, feature reduction method, new feature dimensionality, classifier and classification scenarios were provided. LSTM is a variant of RNN, and it has the capacity to learn long-term dependencies in network traffic features [130]–[132]. However, none of the proposed AE-based methods in Table 2.4 was implemented and validated with the Bot-IoT and N-BaIoT dataset.

Different feature selection methods have been proposed to reduce the dimensionality of network traffic features. Table 2.5 presents an overview of state-of-the-art feature dimensionality reduction methods that are related to botnet attack detection in IoT networks. Koroniotis et al. [87] employed Pearson Correlation and Entropy (PCE) method for feature selection. The authors reported that 10 optimal network traffic features were selected. SVM, RNN and LSTM models were trained with the optimal features to perform binary classification in general and specific attack detection scenarios. Asadi et al. [133] identified optimal feature clusters and eliminated outliers using X-Mean clustering with Particle Swarm Optimisation (XMP) technique. Most relevant network traffic features were selected based on the XMP method. The best binary classification performance was recorded when SVM, DNN and DT classifiers were trained with 10 network traffic features. Khraisat et al. [134] selected 13 network traffic features using Information Gain (IFG) method. An ensemble classifier, which comprised of DT and One-Class SVM (OCSVM) models,

TABLE 2.4: Dimensionality Reduction of Network Traffic Features Based on AE method

Dataset	Ref.	Input	Method	Output	Classifier	Classification scenarios
KDD-Cup99	[116]	41	SAE	28	RF	Binary, multi-class
	[117]	41	SAE	10	Softmax	Binary, multi-class
	[118]	122	AE	100	CNN	Multi-class
	[119]	41	AE	-	k-means	Binary
	[120]	41	SAE	13, 5	DT	Binary
	[121]	41	VAE	20	Dictionary	Binary
	[122]	41	AE	18	k-means	Multi-class
	[123]	39	AE	3	k-means	Binary
NSL-KDD	[116]	41	SAE	28	RF	Binary, multi-class
	[124]	122	SSAE	-	SVM	Binary, multi-class
	[125]	115	SSAE	10	Logistic	Binary, multi-class
	[126]	41	AE	3	DNN	Binary, multi-class
	[127]	52	AE	2	-	Multi-class
UNSW-B15	[117]	42	SAE	10	Softmax	Binary, multi-class
	[128]	207	AE	2	DT	Binary, multi-class
	[126]	41	AE	3	DNN	Binary, multi-class
CICIDS2017	[129]	81	SSAE	64	RF	Binary, multi-class

TABLE 2.5: Dimensionality Reduction of Network Traffic Features in Bot-IoT Dataset

Ref.	Method	Feature size	Classifier	Classification scenarios
[87]	PCE	10	SVM RNN LSTM	Binary
[133]	XMP	10	SVM DNN DT	Binary
[134]	IFG	13	DT, OCSVM	Multi-class

was trained with the selected network traffic features to perform multi-label classification.

Most of the feature dimensionality reduction methods that have been applied to botnet attack detection were based on feature selection techniques. These techniques include the filter method with PCE [87], XMP [133], and (IFG) [134]. Soe et al. [135] did not consider the DoS attack scenario. Also, the performance of the method in detecting benign network traffic was not reported. In a similar work [136], the authors did not evaluate the performance of the proposed method. In another work [137], Guerra-Manzanares et al. did not evaluate the performance of the proposed method with the network traffic data in the Bot-IoT dataset.

The state-of-the-art methods in the related work focused on the selection of specific features from available network traffic information. However, this approach may likely affect the efficiency of botnet attack detection in IoT networks because the classifiers will not have access to some relevant network information during training, validation, and testing. Consequently, the feature selection approach may lead to low botnet attack detection accuracy and a high false alarm rate in IoT networks. On the other hand, LAE reduces the dimensionality of big IoT network traffic data and produces a low-dimensional latent-space feature representation at the hidden layer without loosing useful intrinsic network information. There are different variants of AE including Stacked AE (SAE), Variational AE (AE), Stacked Sparse AE (SSAE), Convolutional AE (CAE), Denoising AE (DeAE), and Long Short-Term Memory AE (LAE). Unlike other variants of AE and similar to RNN, LAE uses LSTM to account for long-term dependencies among features while learning their representation and reducing the dimensionality. So, LAE is a good fit for feature dimensionality reduction in the botnet detection task.

Table 2.6 presents a summary of the state-of-the-art feature dimensionality reduction methods and class balance methods proposed for botnet detection in IoT networks. Koroniotis et al. [87] used PCE method to select the 10 most relevant features. SVM, RNN, and LSTM models were trained with these features to perform binary classification. The reduction in the number of features shortened the time taken to train the ML and DL models, but the classification performance was lower than when the full features were used for model training. Furthermore, the authors did not evaluate the performance of the feature selection method in a multi-class classification scenario. The same set of features was also used for ML-based intrusion detection in [93], [138]–[143].

Kumar et al. [144] proposed a hybrid feature selection method, which combined Pearson Correlation Coefficient (PCC) with Random Forest Mean Decrease Accuracy (RFMDA) and Gain Ratio (GR), to select the 10 most important features. RF, kNN, and Extreme Gradient Boosting (XGBoost) models were trained with these features to perform 5-class classification. Kumar et al. [145] used a mutual information-based feature selection

TABLE 2.6: Review of dimensionality reduction and class balance methods

Dimensionality Reduction Method	Class Balance Method	References
Feature selection	None	[87], [92], [93], [133], [138]–[147]
PCA	None	[148]–[150]
t-SNE	None	[151]
None	Up-sampling	[152]
None	SMOTE	[52], [105], [107], [153]
Feature selection	Focal loss	[154]

method to select the 10 most relevant features. RF and XGBoost models were trained with these features for the 5-class classification task. Shafiq et al. [146], [147] proposed a new feature selection algorithm based on the wrapper technique and Area Under Curve (AUC) metric. Then, DT, NB, RF, and SVM models were trained for 8-class classification. Koroniotis et al. [92] developed Multilayer Perceptron (MLP) and RNN models using 13 network traffic features. Asadi et al [133] proposed Particle Swarm Optimisation (PSO) algorithm to select 10 outstanding features. These features were used to train ML/DL models for binary classification. Other feature dimensionality reduction methods include Principal Component Analysis (PCA) in [148]–[150], and t-distributed Stochastic Neighbour Embedding (t-SNE) in [151].

Khan and Kim [155] proposed a hybrid intelligent model using both anomaly-based and misuse-based network intrusion detection approaches. At the first stage, Logistic Regression (LR) and XGBoost algorithms were used to develop anomaly-based network intrusion detection, while the LAE algorithm was used for misuse-based network intrusion detection in the second stage of the system. The effectiveness of the hybrid model was evaluated with the ISCX-2012 data set. Roopak et al. [156] investigated the effectiveness of MLP, CNN, LSTM, and CNN-LSTM models for DDoS attack detection in IoT networks. The authors simulated these models with the CICIDS2017 data set. Liaqat et al. [152] used the up-sampling method to increase the number of benign samples in the training data set. In [52], [105], [107], [153], SMOTE method was used to generate additional samples for the minority classes. Mulyanto et al. [154] performed feature selection to reduce dimensionality while focal loss function was used to address class imbalance problem. Similarly, Injadat et al. [157] selected the most relevant features and additional minority samples were generated using SMOTE.

Koroniotis *et al* [87] selected the best 10 features using two statistical analysis techniques, namely Pearson correlation coefficient and Shannon entropy. The feature selection method reduced the memory size of the network traffic data by 72.97%. SVM, RNN, and LSTM models were trained with the selected features. The SVM model classified all the benign samples correctly, but 11.63% of the malicious samples were misclassified as benign network traffic. SVM model is a traditional ML method, and it has very limited number of trainable parameters. Although each of the RNN and LSTM models classified 99.75% of the malicious samples correctly, they misclassified 68.76% and 73.38% of the benign samples, respectively. The authors could not determine the optimal sets of hyperparameters for the RNN and LSTM models. It took $1270.48 - 10482.19$ seconds to train each of the models, but the authors did not report the time taken to classify the samples in the testing set.

Khraisat *et al* [134] selected thirteen features using information gain method. The feature selection method reduced the memory size of the network traffic data by 64.87%. An ensemble of DT and OCSVM was trained with the selected features to produce a hybrid model, which achieved high accuracy and high recall of more than 97%. Almost all the samples in the DoS, Norm, Recon, and Theft classes were classified correctly. However, the precision and the F1 score of the model were less than 86%. About 12% of the DDoS attack samples were misclassified as reconnaissance attack. Also, the authors did not evaluate the computation efficiency of the model.

Shafiq *et al* [146], [147] proposed a feature selection method named CorrAUC, which is based on the combination of correlation attribute evaluation and AUC metric. A bijective soft set is applied to validate the selected features using Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) and Shannon entropy. DT, SVM, NB, and RF models with the selected features. However, the DD-H, DD-T, and DD-U classes of botnet attack were not included. Therefore, the validity of the method is limited to eight classes instead of eleven. Soe *et al* [135] proposed correlated-set thresholding on gain-ratio algorithm for feature selection. Tree-based classifiers were trained with the selected features, but the authors did not include the DoS attack samples in the study.

LAE is an effective DL method that produces a low-dimensional latent-space feature representation of a high-dimensional feature set at its hidden layer. To the best of our knowledge, this DL method has not been previously applied to reduce the dimensionality of the feature set in the Bot-IoT dataset. Deep Bidirectional Long Short-Term Memory (BLSTM) is a DL method that learns hierarchical feature representations and long-term inter-related changes directly from raw data using multiple hidden layers. Therefore, in Chapter 5, a hybrid DL method, which employs LAE and BLSTM architectures, is proposed to reduce the feature dimensionality of the network traffic data without any significant adverse effect on the classification performance. Consequently, the amount of memory space required to store the data on the central cloud server will be reduced.

2.4 Federated Learning for Botnet Detection

Modern IoT networks are fast becoming highly scalable. Therefore, due to network constraints, it may be difficult to offload massive distributed IoT network traffic data to a remote central cloud server for data processing in real-life use cases. Also, centralised ML/DL method takes longer time to train, it has high communication overhead, and its memory space requirement for data storage is high. Furthermore, Cloud data centers are often located far away from where IoT devices are deployed. This causes high latency in centralised botnet detection method. Without appropriate encryption, centralised method does not guarantee the privacy and security of IoT devices because it involves the transmission of network traffic features from all participating IoT devices to a central cloud server. Specifically, the use of a third-party cloud server for ML/DL will introduce a high risk of privacy leakage in IoT systems because the network traffic features may contain sensitive information about the owners of the IoT devices [42]–[45].

FL is an advanced AI technique which seeks to protect the privacy of participating nodes without a significant compromise in the classification performance and the generalisation ability of the DL models [46]–[49]. In FL, multiple distributed edge nodes collaboratively develop a robust DL model for botnet attack detection in IoT-enabled critical infrastructure without data privacy concerns. Participating edge nodes request and receive the global model parameters from a central cloud server. Each of the edge nodes trains a local DL model with its private network traffic data based on the global model parameters, and sends the model updates back to the central cloud server for model aggregation. In this way, all the participating nodes benefit from each other's experience without disclosing any private information. The FL method requires lower latency, power, and memory requirements since there is no need to transmit network traffic data to the central cloud server [158].

2.5 Research Gaps

The most recent FL methods that were proposed for cyber-attack detection in IoT and IoT-enabled critical infrastructure. Table 2.7 shows that none of the previous works addressed all the four challenges namely the determination of optimal model hyperparameters, class imbalance in the training data, high memory space requirement for network traffic data storage in resource-constrained edge nodes, and zero-day botnet attack scenarios.

TABLE 2.7: Review of related works

Ref.	Model	Dataset(s)	Hyperparameter Optimisation	Class Balance	Feature Dimensionality Reduction	Zero-Day IoT Botnet Attacks
[42]	ANN	NSL-KDD	✗	✗	✗	✗
[159]	CNN-GRU	GPWST	✗	✗	✗	✗
[160]	DNN, RNN, CNN	Bot-IoT, MQTTset, TON_IoT	✗	✗	✗	✗
[161]	CNN	NSL-KDD, UNSW-NB15	✗	✗	✗	✗
[162]	LAE-GRU	ToN_IoT	✗	✗	✓	✗
[163]	ANN	UNSW-NB15	✗	✓	✗	✗
[164]	ANN-RF	MQTT	✗	✗	✗	✗
[165]	CNN	Private	✗	✗	✗	✗
[166]	Transformer	ToN_IoT	✗	✗	✗	✗
[167]	DNN	Edge-IIoTset	✓	✓	✗	✗
[168]	GRU-SVM	KDD-Cup99, CICIDS2017, WSN-DS	✗	✗	✗	✗
[169]	AE-ANN	GPWST	✗	✗	✓	✗
[170]	LR	ToN_IoT	✗	✗	✗	✗
[171]	ANN	Bot-IoT	✗	✗	✓	✗
[172]	GRU-RF	Modbus	✗	✗	✗	✗
[173]	DQN, DNN	CICIDS2017	✗	✗	✗	✗
This Thesis	LAE-BLSTM	Bot-IoT, NBaIoT	✓	✓	✓	✓

Rahman et al [42] proposed a FL method for intrusion detection in IoT. This method uses ANN model architecture, which has a single hidden layer with 288 hidden units, for binary classification. However, the authors did not provide any provable justification for the choice of the numbers of hidden layers and the hidden units. Also, there was no information about the learning rate, the activation function at the hidden layers, the batch size, and the number of epochs. The effectiveness of the FL method was evaluated with the NSL-KDD dataset, and three use cases were considered namely data distribution per attack type, equal data distribution of attack types, and random data distribution of attack types. However, the authors did not consider class imbalance problem, high memory requirement for data storage, and zero-day botnet attack scenarios.

Li et al [159] proposed a FDL method for intrusion detection in industrial CPS. A hybrid of CNN and GRU architectures were used for local model training in multiple industrial agents. The CNN model comprised three convolutional blocks, while the GRU model had two hidden layers. The outputs of the two models were concatenated and fed into an MLP module, which comprised two hidden layers. The hybrid DL model employed Adam optimiser to ensure the convergence of the loss function. However, the authors did not provide any provable justification for the choice of the model's hyperparameters. Also, the number of hidden units in the hidden layers, the learning rate, the activation function, the batch size, and the number of epochs were not specified. Paillier public-key cryptosystem was used to establish a secure communication channel between the cloud server and the industrial agents. However, the authors did not propose any method to handle high class imbalance problem as well as the high memory space that the industrial agents would need to store the private training data locally. The performance of the FDL method was evaluated with the GPWST dataset, which contains neither IoT network traffic data nor botnet attack traffic data.

Ferrag et al [160] proposed a FDL method for cyber-attack detection in IoT. In this method, DNN, RNN, and CNN model architectures were used for local model training. The DNN model had two hidden layers with 25-60 hidden units each, and the RNN model had two LSTM layers with 22-60 hidden units each. The CNN model comprised two convolutional layers, 18-26 filters, a kernel size of 3, a pooling layer, and two hidden layers with 39-60 hidden units each. Each of the DL models was trained using a batch size of 1000, a single epoch, a learning rate of 0.01-0.5, a ReLU activation function, and Adam optimiser. However, there was no verifiable procedure to justify the selection of the models' hyperparameters. The performance of the FDL method was evaluated with the Bot-IoT, MQTTset, and ToN_IoT datasets. The authors trained the DL models using the original dimensionality of the network traffic features in the datasets, and this implies that a large memory space would be required to store the training data. The class imbalance ratios in the training sets of these datasets were 1:2371, 1:270, and 1:66, respectively. For the Bot-IoT dataset, the authors focused on the 5-class scenario, which only provides a general description of the botnet attacks. On the other hand, the 11-class scenario gives specific details about the botnet attacks. Also, the class imbalance ratio of the Bot-IoT dataset in 11-class scenario is 1:154854. This gives a better representation of a highly imbalance classification scenario. Although the authors considered both IID and non-IID data distribution cases, they did not consider zero-day attack scenario.

Cheng et al [161] proposed a federated transfer learning method for intrusion detection in mobile edge computing. Transfer learning was employed in FL to speed up the model training, reduce computational cost, increase communication efficiency, and improve classification performance. This involves selecting a well-trained model in a particular source domain and transferring it to the edge server in the target domain. A CNN model architecture, which comprised three convolutional layers, two max-pooling layers, a batch normalisation layer, a dropout layer, and two dense layers, was used for binary classification. The CNN model was trained using a batch size of 8, a learning rate of 0.00005, sigmoid activation function, and 100 epochs. However, the authors did not provide any provable justification for the choice of the model's hyperparameters. Reinforcement learning, based on Q-learning algorithm, was used to select only useful clients, while unreliable, low-quality, and malicious clients while exluded from participating in model training to achieve the highest accuracy and save costs. To evaluate the performance of the method, the NSL-KDD dataset was used for model training in the source domain, while the UNSW-NB15 dataset was used to complete the training in the target domain. However, the authors did not consider class imbalance problem, high memory requirement for data storage, and zero-day botnet attack scenarios.

Kumar et al [162] proposed a deep privacy-encoding-based FL framework for data security and privacy in smart agriculture. In this method, a perturbation-based encoding (feature mapping and feature normalisation) and an LAE-based transformation technique were used to prevent inference attacks. The LAE model has two hidden layers with 256 and 128 hidden units, and a latent layer with 12 hidden units. On the other hand, a GRU model, which comprised two hidden layers with 16 and 8 hidden units, was employed for the classification of the encoded data. However, the authors did not provide any provable justification for the choice of the numbers of hidden layers and the hidden units. Also, there was no information about the learning rate, the activation function at the hidden layers, the batch size, and the number of epochs. The performance of the method was evaluated with the ToN_IoT dataset, but the authors did not consider class imbalance problem, high memory requirement for data storage, and zero-day botnet attack scenarios.

Attota et al [164] proposed an ensemble multi-view FL method for intrusion detection in IoT. For each client, three ANN models are trained with the bidirectional flow, unidirectional flow, and packet views of the network traffic features. Grey Wolf Optimization (GWO) technique is used to select the best set of network traffic features for the ANN model training. This reduces the dimensionality of the network traffic features, and consequently minimises the amount of memory space required to store the training data. The outputs of the ANN models are fed into a RF model for attack prediction, but there was no information about the models' hyperparameters. Also, the method was not designed to handle class imbalance problem. Furthermore, the effectiveness of this method was evaluated with the MQTT dataset, which does not contain IoT botnet attack samples.

Chen et al [168] proposed a FL method for intrusion detection in wireless edge networks. This method uses the concept of attention mechanism to calculate the importance of uploaded model parameters, especially when limited bandwidth is available. This reduces the communication overhead while ensuring model convergence because selected clients have different weights. A combination of GRU and SVM model architectures were used for local training. The GRU model has a single hidden layer with 28 hidden units. The performance of the method was evaluated with the KDD Cup99 and CICIDS2017 datasets using a learning rate of 0.0001, and 20 epochs. However, the authors did not provide any provable justification for the choice of the model's hyperparameters. Also, the authors did not consider class imbalance problem, high memory requirement for data storage, and zero-day botnet attack scenarios.

Sedjelmaci and Ansari [163] proposed a cooperative federated GAN for attack detection in multi-access edge computing. The discriminator and generator of the GAN model were designed based on the ANN model architecture, which has five hidden layers. The GAN model was trained with the UNSW-NB15 dataset using a learning rate of 0.1 and 300 epochs. Sun et al [165] proposed segmented FL for adaptive intrusion detection in large-scale local area networks. This method uses a CNN model architecture, which has two convolutional layers, two max-pooling layers, and two dense layers with 200 hidden units each. The model was trained with a private dataset using the RMSprop optimiser, a learning rate of 0.00001, a batch size of 50, and a single epoch. Abdel-Basset et al [166] proposed a FDL method for security and privacy in heterogeneous blockchain-based smart transportation systems. A stack of context-aware transformer networks, which comprised an encoder and a decoder, was used to learn the spatial-temporal representations of vehicular traffic flows. Consortium blockchain was used to confirm the distributed local updates from participating vehicles, thereby stopping unreliable updates from being part of the FL process. The method was evaluated with the ToN_IoT dataset.

Aouedi et al [169] proposed a federated semi-supervised learning method for attack detection in industrial IoT. This method uses both labelled and unlabelled data in federated approach since data labelling is costly and time-consuming. AE model was used for feature representation learning and dimensionality reduction to reduce communication overhead. The encoder of the AE model has three hidden layers with 20, 15, and 10 hidden units, respectively. An ANN model architecture was used for classification. The performance of the proposed method was evaluated with the GPWST dataset using a ReLU activation function, Adam optimiser, a learning rate of 0.001, a batch size of 64, and 100 epochs. The joint-announcement protocol was used for random client selection to further reduce communication overhead. Ruzafa-Alcazar et al [13] evaluated the performance of different differential privacy techniques for FL-based intrusion detection in industrial IoT. The techniques include Laplace, Laplace truncated, Laplace bounded domain, Laplace bounded noise, Gaussian, Gaussian analytic, and uniform. The authors employed LR model for the classification of network traffic samples. The FL method investigated the effectiveness of FedAvg and Fed+ algorithms using the ToN_IoT dataset.

Huong et al [171] proposed an edge-cloud architecture for attack detection. To minimize the complexity of the detection model, PCA was employed for feature dimensionality reduction. ANN model, which has a single hidden layer with 6-46 hidden units, was used for multi-class classification. The method was evaluated with the 11-class Bot-IoT dataset with a class imbalance ratio of 1:172162. The results showed that the F1 score of the FL model in the minority classes (D-H, DD-H, and KL) was less than 80%. Also, the FL model could not detect any of the DE samples in the testing set due to class imbalance problem in the training set.

Mothukuri et al [172] proposed a FL method for anomaly detection in IoT. The combination of GRU and RF models was used for local training. The performance of the method was evaluated with the Modbus dataset. Wei et al [173] proposed a FL-based end-edge-cloud cooperative method for attack detection in 5G heterogeneous networks. The authors employed Deep Q-Network (DQN) and DNN for local training in the end nodes and edge nodes, respectively. the effectiveness of the method was evaluated with the CICIDS2017 dataset. Ferrag et al [167] proposed a FL method for cyber-attack detection in industrial IoT. They used the SMOTE method to oversample the minority classes. DNN model, which has two hidden layers with 90 hidden units each, was used for training. A grid search algorithm was used for model hyperparameter optimisation. The performance of the method was evaluated with the Edge-IIoTset dataset using a batch size of 800, ReLU activation function, Adam optimiser, and 25 epochs.

The classification performance of the local models at the edge nodes depends on the choice of the right set of model hyperparameters, which vary for different application contexts [50]. These hyperparameters include the number of hidden layers and their respective hidden units, the learning rate, the hidden layers' activation functions, the batch size, the optimiser, and the number of epochs. However, the hyperparameters of the models in the previous works were randomly selected with little or no justification. Therefore, in Chapter 3 of this thesis, a hyperparameter optimisation algorithm is proposed for DL-based botnet attack detection in IoT-enabled critical infrastructure.

Huong *et al* [171] divided the network traffic data in the Bot-IoT dataset among four IoT edge nodes, but the authors did not provide the details of the sample distribution of the benign traffic and the botnet attack traffic in each of the nodes. The class imbalance in the 11-class Bot-IoT dataset adversely affected the classification performance of the model, especially in the minority classes. In Chapter 4 of this thesis, a framework, named

SMOTE-DL, is proposed to improve the classification performance of the model when the network traffic data in the training set is highly imbalanced. Also, the authors employed PCA method [174] to reduce the feature dimensionality of the network traffic samples in the dataset. However, they did not investigate and quantify the effects of different feature dimensionality on the classification performance of the model. Similarly, Zhang *et al* [175] and Rey *et al* [176] used N-BaIoT dataset to simulate the FL method that was proposed for anomaly detection in IoT. The authors did not reduce the feature dimensionality of the network traffic data. So, a large memory space is required to store the data for model training in resource-constrained IoT edge nodes. In Chapter 5 of this thesis, a hybrid DL method, named LAE-BLSTM, is proposed to reduce the feature dimensionality of the network traffic data, and its effects on the classification performance of the model is extensively investigated and quantified. Furthermore, zero-day botnet attack scenarios were not considered in [171], [175], [176]. Therefore, in Chapter 6 of this thesis, a FDL method is proposed to detect zero-day botnet attacks in IoT edge nodes.

2.6 Chapter Summary

In this chapter, the main concept of botnet was discussed in the context of IoT-enabled critical infrastructure. Then, an extensive review of the state-of-the-art ML, DL, and FL methods was conducted to identify current research gaps. It was established that there is a need to develop an efficient FDL method for botnet attack detection in IoT-enabled critical infrastructure. An optimisation algorithm is needed to determine the most appropriate sets of hyperparameters when the Bot-IoT and N-BaIoT datasets are used for model development in binary, 5-class, 10-class, and 11-class classification scenarios. Also, a method is required to improve the classification performance of the FDL-based botnet detection model when the network traffic data in the training set is highly imbalanced. Furthermore, a feature dimensionality reduction method is needed to minimise the amount of memory space required to store network traffic data in resource-constrained IoT devices. Finally, a privacy-preserving method is needed to detect zero-day botnet attack in IoT edge nodes.

Chapter 3

Model Hyperparameter Optimisation for Deep Learning-Based Botnet Detection

3.1 Introduction

DL is an AI technique that can be used to automatically learn the underlying features of the traffic patterns in IoT networks directly from raw network data using hierarchical representations. However, the performance of a DL model largely depends on the set of hyperparameters that is used for model development, as earlier discussed in Section 2.3.1. Therefore, in this chapter, an algorithm is developed to determine the optimal set of hyperparameters (the numbers of hidden layers and hidden units, the learning rate, the optimiser, the activation function, the batch size, and the number of epochs) for efficient DL-based botnet detection in IoT networks. The DL models employ a DNN architecture for binary and multi-class classification. DNN-based botnet detection models are developed and experiments are performed with the Bot-IoT and N-BaIoT data sets to test the effectiveness of the hyperparameter optimisation method. The classification performance of the DNN-based botnet detection models is evaluated based on accuracy, precision, recall, and F1 score, while their computation efficiency is evaluated based on training time and testing time.

The remaining sections of this chapter are organised as follows: in Section 3.2, the DNN model architecture and the hyperparameter optimsation algorithm are presented. In Section 3.3, experiments are performed to develop optimal DNN models for botnet detection in IoT networks. In Section 3.4, the results of the experiments are analysed and discussed. Finally, the major findings in this chapter are summarised in Section 3.5.

3.2 Model Hyperparameter Optimisation Method

DNN is a DL model architecture that is made up of an input layer, l_i, two or more densely connected hidden layers, l_h, and an output layer, l_o. Each of these layers has u units, which are fully connected. Figure 3.1 shows the architecture of a DNN model, which has three hidden layers with 12, 10, and 10 hidden units, respectively. The input layer has eight units, while the output layer has five units. In this thesis, a boldface upper-case alphabet and a boldface lower-case alphabet represent a matrix and a vector, respectively. For a DNN model, the number of units at the input layer is the same as the feature dimensionality[1], d, of the network traffic data, $\mathbf{X} \in \mathbb{R}^{d \times n}$, where n is the total number of samples in the training set. On the other hand, the number of units at the output layer depends on the type of classification task at hand. For binary classification, a single unit is required at the output layer, while the number of units at the output layer is equal to the number of unique class labels, c, in multi-class classification. For instance, in order to classify a network traffic sample as either DDoS, DoS, Recon, Theft, or Norm traffic, the output layer must have five units at the output layer.

A DNN model is trained with $\mathbf{X}$ and a corresponding class label vector, $\mathbf{y} \in \mathbb{R}^n$, using a supervised learning approach. A set of features representing a network traffic sample, $\mathbf{x} \in \mathbf{X}$, is propagated forward from the input layer to the output layer through the units in the hidden layer(s). A forward propagation through a single hidden unit is given by:

$$h_{out} = \sigma_h \Big[\sum_{i=1}^{d} (w_i x_i) + b \Big], \tag{3.1}$$

where h_{out} is the output of the hidden unit, σ_h is the activation function of the hidden unit, w_i is the weight associated with the network traffic feature, x_i, and b is the bias associated to the network traffic sample, $\mathbf{x}$. The activation function introduces non-linearity to the weighted sum of the input by transforming real numbers into a bounded output, and this helps to approximate arbitrarily complex functions. On the other hand, the bias term helps to shift the activation function either to the left or to the right,

[1] Feature dimensionality is the total number of features in a given network traffic data

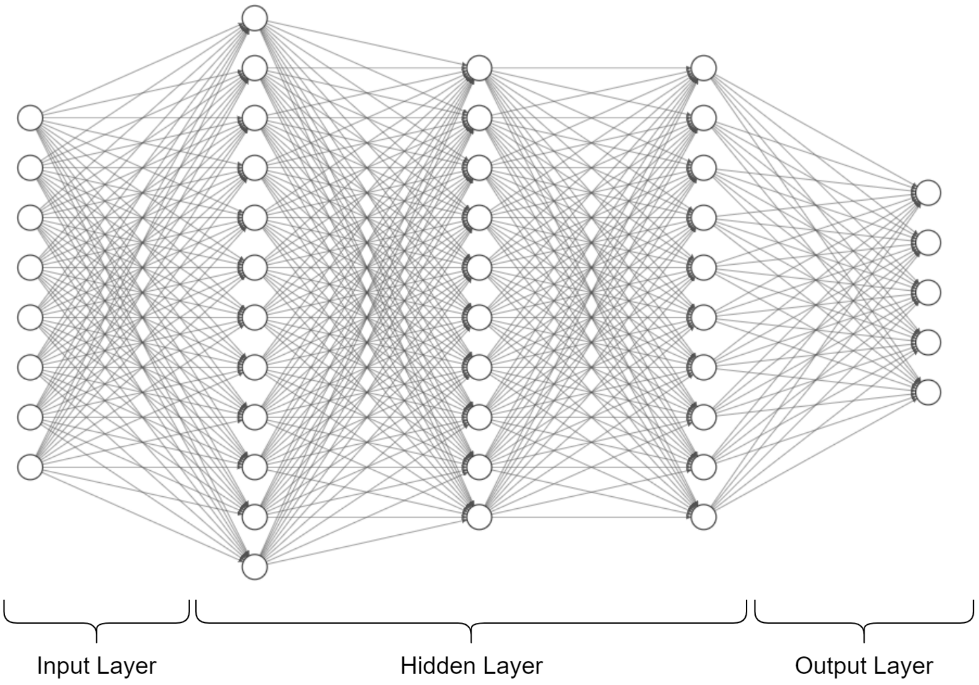


FIGURE 3.1: DNN model architecture

regardless of the value of $\mathbf{x}$. The forward propagation can be represented as:

$$h_{out} = \sigma_h(\mathbf{w}\mathbf{x} + b), \tag{3.2}$$

where $\mathbf{w} = [w_1, w_2, \ldots, w_d]$ and $\mathbf{x} = [x_1, x_2, \ldots, x_d]^T$.

At the first hidden layer, the DNN learns the representation of a mini-batch of the network traffic features, $\mathbf{X} \in \mathbb{R}^{d \times n_b}$ by transforming the input data with an initial random weight matrix, $\mathbf{W}_1 \in \mathbb{R}^{u_1 \times d}$, and the bias vector, $\mathbf{B}_1 \in \mathbb{R}^{u_1 \times n_b}$, as:

$$\mathbf{H}_1 = \sigma_h(\mathbf{W}_1\mathbf{X} + \mathbf{B}_1), \tag{3.3}$$

where n_b is the batch size, $\mathbf{H}_1 \in \mathbb{R}^{u_1 \times n_b}$ is the first hidden state vector, u_1 is the number of hidden units at the first hidden layer, $\mathbf{W}_1$ is the weight matrix of the first hidden layer, and $\mathbf{B}_1$ is the bias matrix of the first hidden layer. For any successive hidden layer, $\mathbf{H}_{i+1}$, the output of the current hidden layer, $\mathbf{H}_i$, is transformed as:

$$\mathbf{H}_{i+1} = \sigma_h(\mathbf{W}_i\mathbf{H}_i + \mathbf{B}_i), \tag{3.4}$$

where $\mathbf{W}_i$ is the weight matrix of the current hidden layer, and $\mathbf{B}_i$ is the bias matrix of the current hidden layer.

The predicted class label, $\tilde{\mathbf{y}}$, is obtained by transforming the output of the last hidden layer, $\mathbf{H}_j$:

$$\tilde{\mathbf{y}} = \sigma_y(\mathbf{H}_j), \tag{3.5}$$

where σ_y is the activation function at the output layer, and j is the number of hidden layers. In a binary classification scenario, σ_y is a sigmoid function, which is defined as:

$$\sigma_{y,binary}(\mathbf{H}_j) = \frac{1}{1+e^{-\mathbf{H}_j}}. \tag{3.6}$$

On the other hand, a softmax activation function is used for multi-class classification, and it is defined as:

$$\sigma_{y,multi}(\mathbf{H}_j) = \frac{e^{\mathbf{H}_j}}{\sum\limits_{\tau=1}^{c} e^{\mathbf{H}_\tau}}, \tag{3.7}$$

where $e^{\mathbf{H}_j}$ is the standard exponential function for the input hidden vector, and $e^{\mathbf{H}_\tau}$ is the standard exponential function for the output hidden vector. The difference between the probability distribution of $\tilde{\mathbf{y}}$ and $\mathbf{y}$ is measured using a cross-entropy loss function θ. In a binary classification scenario, θ is a binary cross-entropy function, which is defined as:

$$L_{binary} = \theta_{binary}(\mathbf{y}, \tilde{\mathbf{y}}) = -\frac{1}{n}\sum_{\tau=1}^{n} \left[y_\tau \log(\tilde{y_\tau}) + (1-y_\tau)\log(1-\tilde{y}) \right], \tag{3.8}$$

where L_{binary} is the binary cross-entropy loss. For multi-class classification, θ is a categorical cross-entropy loss function, and it is defined as:

$$L_{multi} = \theta_{multi}(\mathbf{y}, \tilde{\mathbf{y}}) = -\frac{1}{n}\sum_{\tau=1}^{n}\sum_{\omega=1}^{c}\left[y_{\tau,\omega} \log(\tilde{y}_{\tau,\omega}) \right], \tag{3.9}$$

where L_{multi} is a categorical cross-entropy loss. The performance of the DNN model is validated with an entirely different network traffic feature matrix and its corresponding class label vector in the validation set. An optimiser, ψ, is used to minimise the cross-entropy loss, L, of the DNN model over e epochs based on gradient descent algorithm [177]:

$$\mathbf{W} \leftarrow \mathbf{W} - r\frac{\partial\theta(\mathbf{W})}{\partial\mathbf{W}}, \tag{3.10}$$

$$\mathbf{W}^* = \psi(\mathbf{W}, L) = \underset{\mathbf{W}}{\operatorname{argmin}}\, L, \tag{3.11}$$

where $\mathbf{W}^*$ is the set of weight matrices that achieved the lowest L. Meanwhile, the classification performance of the DNN model depends on the learning rate r of the backpropagation[2] process [179]. For each epoch, the optimizer, ψ, updates $\mathbf{W}$ to minimize L.

As earlier discussed in Section 2.3.1, there is often no definite procedure to justify the selection of an optimal set of hyperparameters for DL-based botnet attack detection in IoT networks. Therefore, an algorithm is developed to determine the optimal number of hidden layers, l_{ho}, and their respective hidden units, u_{ho}, the learning rate, r_o, the optimiser, ψ_o, the activation function at the hidden layer, σ_{ho}, the batch size, n_{bo}, and the number of epochs, e_o, as shown in Algorithm 1. First, a set of values is defined for each model hyperparameter. The sets of hyperparameters include $L_H = [l_{h1}, l_{h2}, \ldots, l_{hn}]$ for the number of hidden layers, $U_H = [u_{h1}, u_{h2}, \ldots, u_{hn}]$ for the number of hidden units, $R = [r_1, r_2, \ldots, r_n]$ for the learning rate, $\Psi = [\psi_1, \psi_2, \ldots, \psi_n]$ for the optimiser, $\epsilon_H = [\sigma_{h1}, \sigma_{h2}, \ldots, \sigma_{hn}]$ for the activation function, $N_B = [n_{b1}, n_{b2}, \ldots, n_{bn}]$ for the batch size, and $E = [e_1, e_2, \ldots, e_n]$ for the number of epochs. From these sets of hyperparameters, r_d, ψ_d, σ_{hd}, n_{bd}, e_d are randomly chosen as the default values of learning rate, optimiser, activation function, batch size, and number of epochs, respectively.

Given the default values, DNN models are developed with the different numbers of hidden layers in L_H and the different numbers of hidden units in U_H. The combination of the number of hidden layers and the number of hidden units that produce the DNN model with the best classification performance is selected as l_{ho} and u_{ho}, respectively. Given l_{ho}, u_{ho}, ψ_d, σ_{hd}, n_{bd}, and e_d, DNN models are developed with the different learning rates in R. The learning rate that produce the DNN model with the best classification performance is selected as r_o. Given l_{ho}, u_{ho}, r_o, σ_{hd}, n_{bd}, and e_d, DNN models are developed with the different optimisers in Ψ. The optimiser that produce the DNN model with the best classification performance is selected as ψ_o. Given l_{ho}, u_{ho}, r_o, ψ_o, n_{bd}, and e_d, DNN models are developed with the different activation functions in ϵ_H. The activation function that produce the DNN model with the best classification performance is selected as σ_{ho}. Given l_{ho}, u_{ho}, r_o, ψ_o, σ_{ho}, and e_d, DNN models are developed with the different batch sizes in N_B. The batch size that produce the DNN model with the best classification performance is

[2] Backpropagation refers to the backward propagation of the cross-entropy loss [178].

Algorithm 1: Hyperparameter Optimisation Algorithm

```
Input: $\mathbf{X}, \mathbf{y}$
Output: $l_{ho}, u_{ho}, r_o, \psi_o, \sigma_{ho}, n_{bo}, e_o$
Initialization: $r_d, \psi_d, \sigma_{hd}, n_{bd}, e_d$
function DNN($X, y$):
    for $e$ in $E$ do
        for $n_b$ in $N_B$ do
            $\mathbf{H}_1 = \sigma_h(\mathbf{W}_1\mathbf{X} + \mathbf{B}_1)$
            $\mathbf{H}_{i+1} = \sigma_h(\mathbf{W}_i\mathbf{H}_i + \mathbf{B}_i)$
            $\tilde{\mathbf{y}} = \sigma_y(\mathbf{H}_j)$
            $L \leftarrow \theta(\mathbf{y}, \tilde{\mathbf{y}})$
        end
        $\mathbf{W}^*, L^* \leftarrow \psi(\mathbf{W}, L)$
    end
    return metrics
end function
for $l_h$ in $L_H$ do
    for $u_h$ in $U_H$ do
        Metrics ← DNN(X, y)
    end
end
$l_{ho}, u_{ho} \leftarrow best \in Metrics$
for $rinR$ do
    Metrics ← DNN(X, y)
end
$r_o \leftarrow best \in Metrics$
for $\psi$ in $\Psi$ do
    Metrics ← DNN(X, y)
end
$\psi_o \leftarrow best \in Metrics$
for $\sigma_h \in \epsilon_H$ do
    Metrics ← DNN(X, y)
end
$\sigma_{ho} \leftarrow best \in Metrics$
for $n_b$ in $N_B$ do
    Metrics ← DNN(X, y)
end
$n_{bo} \leftarrow best \in Metrics$
for $e$ in $E$ do
    Metrics ← DNN(X, y)
end
$e_o \leftarrow best \in Metrics$
```

selected as n_{bo}. Given l_{ho}, u_{ho}, r_o, ψ_o, σ_{ho}, and e_d, DNN models are developed with the different batch sizes in N_B. The batch size that produce the DNN model with the best classification performance is selected as n_{bo}. Given l_{ho}, u_{ho}, r_o, ψ_o, σ_{ho}, and n_{bo}, DNN models are developed with the different number of epochs in E. The number of epochs that produce the DNN model with the best classification performance is selected as e_o.

3.3 Model Development and Experiment

The framework for the development of an optimal DNN-based botnet detection model involves data pre-processing and hyperparameter optimisation, as shown in Figure 3.2. For the pre-processing of the network traffic data, feature selection, feature normalisation, label encoding, and data splitting were performed to prepare the data for model development. Then, the method that was earlier proposed in Section 3.2 is used to determine the optimal sets of hyperparameters for binary and multi-class classification using the Bot-IoT and N-BaIoT data sets.

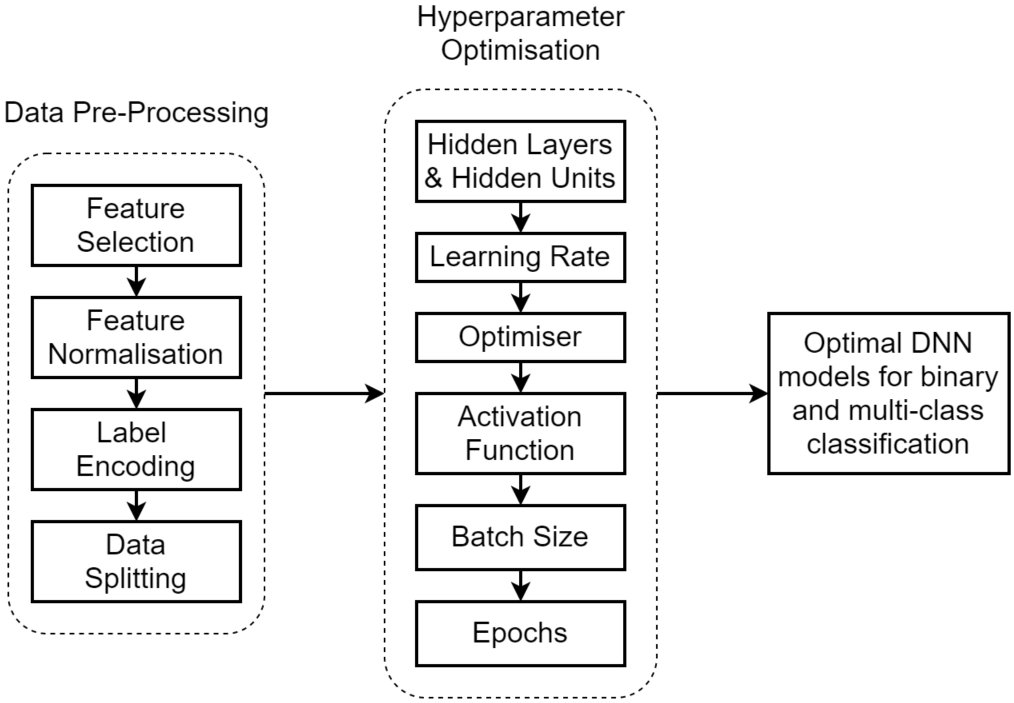


FIGURE 3.2: Framework for the development of an optimal DNN-based botnet detection model

The Bot-IoT dataset, which was previously described in Section 2.2, contains 43 network traffic features and three categories of label for binary, 5-class, and 11-class classification, respectively. Six redundant features (*pkSeqID*, *saddr*, *daddr*, *proto*, *state*, and *flgs*) were identified and removed from the data set. Specifically, *pkSeqID* is the sequence identification number that the network traffic capturing tool, Argus, assigned to each sample in the network packet; *saddr* and *daddr* are device-specific; while *proto*, *state*, and *flgs* give the same information as *proto_number*, *state_number*, and *flgs_number*. Therefore, in this thesis, the remaining 37 features were used to characterise a network traffic sample in a typical IoT network. On the other hand, the N-BaIoT data set contains 115 network traffic features and two categories of label for binary and 10-class classification, respectively. The details of this data set was earlier discussed in Section 2.2.

For all the features to contribute equally to the classification outcomes of the DNN models, their values were scaled to numbers between 0 and 1 using min-max normalisation [180], [181] given by:

$$\mathbf{x}_{norm} = \frac{\mathbf{x} - \mathbf{x}_{min}}{\mathbf{x}_{max} - \mathbf{x}_{min}}, \tag{3.12}$$

where $\mathbf{x}$ is a network traffic feature vector, while $\mathbf{x}_{min}$ and $\mathbf{x}_{max}$ are the minimum and maximum values of $\mathbf{x}$ respectively. Feature normalisation eliminates any form of bias in favour of a particular feature. On the other hand, the categorical labels were encoded with numbers for ease of computation. For binary classification, the *Norm* and *Attack* classes were represented by 0 and 1, respectively. Similarly, the labels in the 5-class, 10-class, and 11-class categories were represented by $0-4$, $0-9$, and $0-10$ respectively.

The hold-out validation method was employed to evaluate the classification performance of the DNN-based botnet detection models. The complete network traffic data was randomly split into training set (60%), validation set (20%), and testing set (20%). Tables 3.1 and 3.2 present the distribution of network traffic samples in the Bot-IoT and N-BaIoT data sets, respectively. In the Bot-IoT data set, there were 2201112 samples in the training set, 733705 samples in the validation set, and 733705 samples in the testing set. On the other hand, in the N-BaIoT data set, there were 3721653 samples in the training set, 1240551 samples in the validation set, and 1240552 samples in the testing set. The samples in the training set were used to train DNN-based botnet detection models. The samples in the validation set were used to evaluate the robustness of the model against under-fitting and over-fitting based on the cross-entropy loss. The samples in the testing set were used to evaluate the generalisation ability of the models based on the

TABLE 3.1: Distribution of the network traffic samples in the Bot-IoT dataset

Type	Class	Training	Validation	Testing
2-class	Norm	290	86	101
	Attack	2200822	733619	733604
5-class	DDoS	1155741	385317	385566
	DoS	990285	329944	330031
	Norm	290	86	101
	Recon	54744	18343	17995
	Theft	52	15	12
11-class	DD-H	588	197	204
	DD-T	586393	195713	195274
	DD-U	568760	189407	190088
	D-H	906	311	268
	D-T	369965	122861	122974
	D-U	619414	206772	206789
	Norm	290	86	101
	OSF	10795	3537	3582
	SS	43949	14806	14413
	DE	4	1	1
	KL	48	14	11

TABLE 3.2: Distribution of the network traffic samples in the N-BaIoT dataset

Type	Class	Training	Validation	Testing
2-class	Norm	333390	111133	111409
	Attack	3388263	1129418	1129143
10-class	Norm	333390	111133	111409
	g_combo	308666	103471	103019
	g_junk	156937	52590	52262
	g_scan	153146	51061	50904
	g_udp	567451	189423	189492
	m_ack	386965	128166	128690
	m_scan	322833	107336	107810
	m_syn	440419	146858	146022
	m_udp	738149	245529	246321
	m_udpp	313697	104984	104623

accuracy (A), precision (P), recall (R), and F1 score (F1) given by:

$$A = \frac{TP + TN}{TP + TN + FP + FN} \times 100\%, \tag{3.13}$$

$$P = \frac{TP}{TP + FP} \times 100\%, \tag{3.14}$$

$$R = \frac{TP}{TP + FN} \times 100\%, \tag{3.15}$$

$$F1 = \frac{2 \times TP}{(2 \times TP) + FP + FN} \times 100\%, \tag{3.16}$$

where TP is the number of the samples in the positive class that are correctly classified, FP is the number of samples in the positive class that are misclassified, TN is the number of samples in the negative class that are correctly classified, and FN is the number of samples in the negative class that are misclassified. Accuracy (A) is the ratio of the number of samples correctly classified as either positive or negative to the total number of samples in all classes. Precision (P) is the ratio of the number of samples in the positive class that are correctly classified as positive to the total number of samples that are either correctly classified or misclassified as positive. Recall (R) is the ratio of the number of samples in the positive class that are correctly classified as positive to the total number of samples in the positive class. F1 score (F1) refers to the harmonic mean of precision and recall. It is also known as F-measure.

TABLE 3.3: DNN Model Hyperparameters

Hyperparameter	Values
Hidden layer(s)	1, 2, 3, 4
Hidden units	16, 32, 64, 128
Learning rate	0.1, 0.01, 0.001, 0.0001
Optimiser	Adam, SGD, RMSprop, Adadelta, Adagrad, Adamax, Nadam
Activation function	ReLU, *tanh*, SELU, ELU
Batch size	64, 128, 256, 512, 1024
Epochs	5, 10, 15, 20

The proposed method in Section 3.2 was implemented to evaluate its effectiveness for botnet attack detection in IoT networks. The set of hyperparameters and their elements are presented in Table 3.3. Only the most popular optimisers - Adam [182], SGD [183], Root Mean Squared Propagation (RMSprop) [184], Adadelta [185], Adagrad [186], Adamax [182], Nadam [177] - and the advanced activation functions - ReLU [187], *tanh* [188], SELU [189], ELU [190] - were considered. The default hyperparameters include a learning rate of 0.001, Adam optimiser, ReLU activation function, a batch size of 512, and 5 epochs.

3.4 Result Analysis and Discussion

In this section, the effectiveness of the proposed hyperparameter optimisation method in binary and multi-class classification scenarios is evaluated with the Bot-IoT and N-BaIoT datasets. The classification performance of the models is analysed based on accuracy, precision, recall, and F1 score. F1 score is the harmonic mean of precision and recall, and it accounts for both detection rate and false alarm rate. Therefore, the optimal classification performance is determined based on the F1 score. The robustness of the optimal models against under-fitting and over-fitting is evaluated based on the cross-entropy losses during training and validation. Also, the computation efficiency of the optimal models is evaluated based on the time spent on training and testing the models.

3.4.1 Optimal Numbers of Hidden Layers and Hidden Units

The optimal numbers of hidden layers and hidden units in binary and multi-class classification scenarios are determined when the DNN models were developed with the Bot-IoT and N-BaIoT datasets. For each classification scenario, sixteen DNN models were developed with different numbers of hidden layers and hidden units, while the default values of the other hyperparameters (a learning rate of 0.001, Adam optimiser, ReLU activation function, a batch size of 512, and 5 epochs) were maintained.

Table 3.4 presents the performance in binary classification scenario when the sixteen DNN models were tested with the network traffic samples in the testing set of the Bot-IoT dataset. Three DNN models achieved the highest F1 score of 99.03%. However, the DNN model architecture which comprised two hidden layers with 128 and 16 hidden units, respectively was considered the optimal among the three models because it had the least total number of hidden units. The smaller the number of hidden units in the model architecture, the smaller the number of trainable parameters, and the lesser the computational complexity. The model achieved 100% accuracy, 98.1% precision, and 100% recall. Figure 3.3 shows that all the benign samples in the Norm class were correctly classified, and only four out of 733604 malicious samples in the Attack class were misclassified as benign network traffic. This implies that the model had a good classification performance. Figure 3.4 shows that the training loss of the model reduced by 99.52%, while the validation loss reduced by 61.47%. This implies that

TABLE 3.4: Binary classification performance for different hidden layers and hidden units based on the Bot-IoT dataset

Hidden Units				Classification Performance (%)			
HL1	HL2	HL3	HL4	A	P	R	F1
16	-	-	-	100.00	98.90	94.06	96.35
32	-	-	-	100.00	98.95	96.04	97.45
64	-	-	-	100.00	98.97	97.03	98.97
128	-	-	-	100.00	98.98	97.52	98.24
128	**16**	-	-	100.00	98.10	100.00	**99.03**
128	32	-	-	100.00	98.10	100.00	99.03
128	64	-	-	100.00	97.64	100.00	98.79
128	128	-	-	100.00	97.60	99.01	98.29
128	128	16	-	100.00	98.10	100.00	99.03
128	128	32	-	100.00	97.20	100.00	98.56
128	128	64	-	100.00	98.10	100.00	99.03
128	128	128	-	100.00	96.76	100.00	98.33
128	128	128	16	100.00	96.76	100.00	98.33
128	128	128	32	100.00	97.64	100.00	98.79
128	128	128	64	100.00	97.64	100.00	98.79
128	128	128	128	100.00	96.76	100.00	98.33

the model neither under-fitted nor over-fitted the network traffic data in the training set. It took 39.25 seconds to train the model with the network traffic samples in the training set, and the model spend 441 milliseconds to classify the network traffic samples in the testing set. This implies that the model was computationally efficient. Therefore, in binary classification scenario, two hidden layers with 128 and 16 hidden units, respectively are recommended for developing DNN-based botnet detection model with the Bot-IoT dataset.

Table 3.5 presents the performance in 5-class classification scenario when the sixteen DNN models were tested with the network traffic samples in the testing set of the Bot-IoT dataset. The DNN model architecture which comprised four hidden layers with 128, 128, 128, and 16 hidden units, respectively has the highest F1 score of 98.81%. The model achieved 99.97% accuracy, 99.95% precision, and 97.79% recall. Figure 3.5 shows that the model correctly classified all the samples in the Recon and Theft classes, more than 99.85% of the samples in the DDoS and DoS classes, and 89.11% of the samples in the Norm class. This shows that the model had a good classification performance. Figure 3.6 shows that the training loss of the model reduced by 91.73%, while the validation loss reduced by 96.95%. This shows that the model neither under-fitted nor over-fitted the network traffic

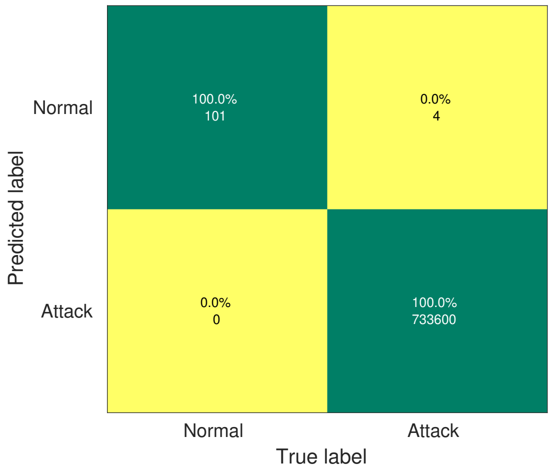


FIGURE 3.3: Confusion matrix of binary DNN model with optimal numbers of hidden layers and hidden units based on the Bot-IoT dataset

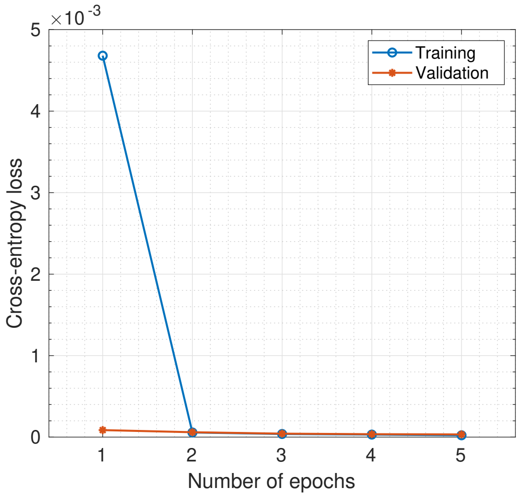


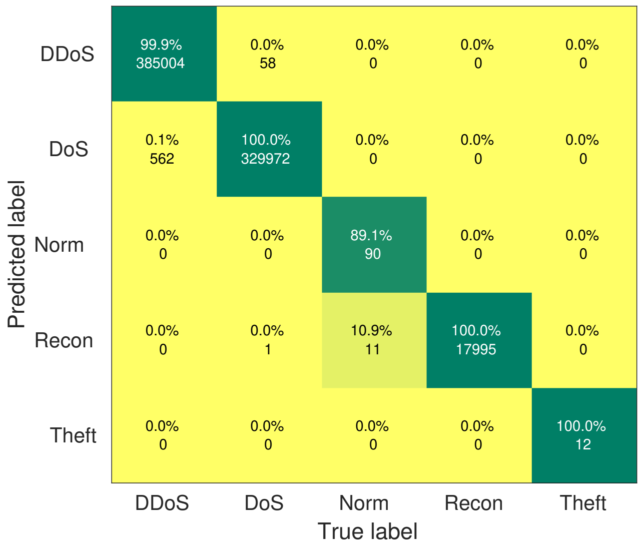
FIGURE 3.4: Training and validation losses of binary DNN model with optimal numbers of hidden layers and hidden units based on the Bot-IoT dataset

data in the training set. It took 32.59 seconds to train the model with the network traffic samples in the training set, and the model spend 585 milliseconds to classify the network traffic samples in the testing set. This shows that the model was computationally efficient. Therefore, in 5-class classification scenario, four hidden layers with 128, 128, 128, and 16 hidden units, respectively are recommended for developing DNN-based botnet detection model with the Bot-IoT dataset.

Table 3.6 presents the performance in 11-class classification scenario when the sixteen DNN models were tested with the network traffic samples in the

TABLE 3.5: 5-class classification performance for different hidden layers and hidden units based on the Bot-IoT dataset

Hidden Units				Classification Performance (%)			
HL1	HL2	HL3	HL4	A	P	R	F1
16	-	-	-	98.89	78.80	71.15	74.05
32	-	-	-	99.19	76.40	74.21	75.23
64	-	-	-	99.43	78.94	77.06	77.94
128	-	-	-	99.68	98.81	86.24	90.11
128	16	-	-	99.95	78.61	76.77	77.64
128	32	-	-	99.96	99.12	93.18	95.78
128	64	-	-	99.95	99.01	93.05	95.78
128	128	-	-	99.96	98.55	93.37	95.61
128	128	16	-	99.98	99.37	95.66	97.36
128	128	32	-	99.87	97.70	95.92	96.62
128	128	64	-	99.98	99.17	97.32	98.21
128	128	128	-	99.98	99.56	97.13	98.30
128	**128**	**128**	**16**	99.97	99.95	97.79	**98.81**
128	128	128	32	99.98	99.53	96.13	97.76
128	128	128	64	99.93	99.50	96.47	97.92
128	128	128	128	99.99	97.25	98.30	97.64



	DDoS	DoS	Norm	Recon	Theft
DDoS	99.9% 385004	0.0% 58	0.0% 0	0.0% 0	0.0% 0
DoS	0.1% 562	100.0% 329972	0.0% 0	0.0% 0	0.0% 0
Norm	0.0% 0	0.0% 0	89.1% 90	0.0% 0	0.0% 0
Recon	0.0% 0	0.0% 1	10.9% 11	100.0% 17995	0.0% 0
Theft	0.0% 0	0.0% 0	0.0% 0	0.0% 0	100.0% 12

FIGURE 3.5: Confusion matrix of 5-class DNN model with optimal numbers of hidden layers and hidden units based on the Bot-IoT dataset

testing set of the Bot-IoT dataset. The DNN model architecture which comprised four hidden layers with 128, 128, 128, and 64 hidden units, respectively has the highest F1 score of 88.06%. The model achieved 99.99% accuracy, 88.59% precision, and 87.55% recall. Figure 3.7 shows that 99.6% of the samples in the DD-T, DD-U, D-T, D-U, OSF and SS classes were

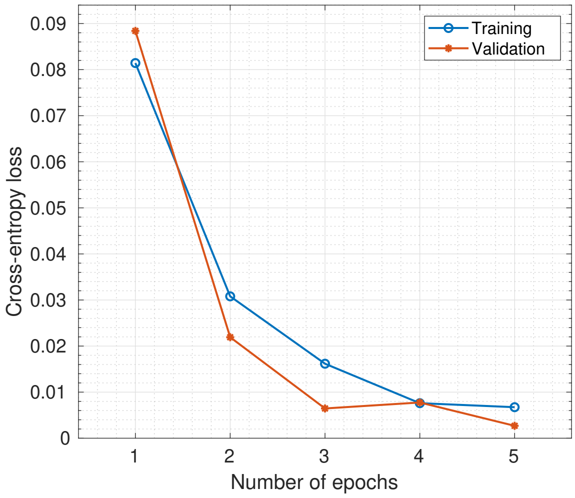
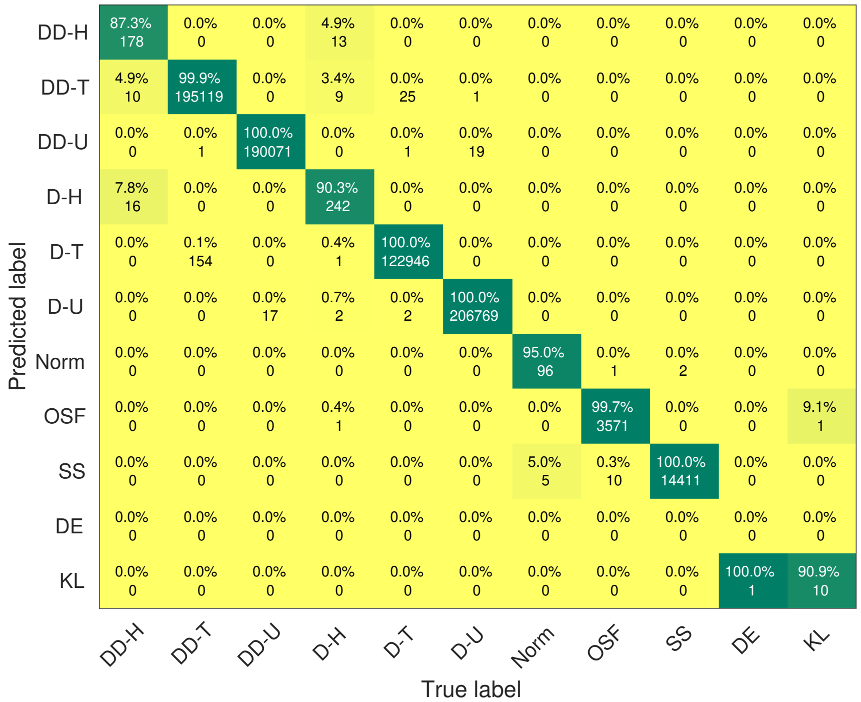


FIGURE 3.6: Training and validation losses of 5-class DNN model with optimal numbers of hidden layers and hidden units based on the Bot-IoT dataset

TABLE 3.6: 11-class classification performance for different hidden layers and hidden units based on the Bot-IoT dataset

Hidden Units				Classification Performance (%)			
HL1	HL2	HL3	HL4	A	P	R	F1
16	-	-	-	99.46	78.18	61.95	64.39
32	-	-	-	99.65	72.98	67.01	69.20
64	-	-	-	99.77	82.25	78.53	79.82
128	-	-	-	99.85	84.43	78.90	80.79
128	16	-	-	99.95	83.97	78.39	80.36
128	32	-	-	99.98	75.79	73.55	73.85
128	64	-	-	99.99	84.95	83.30	83.52
128	128	-	-	99.96	85.46	83.92	83.74
128	128	16	-	99.97	85.48	79.47	81.32
128	128	32	-	99.98	86.79	80.43	82.13
128	128	64	-	99.99	86.44	84.58	84.88
128	128	128	-	99.99	88.08	84.79	86.09
128	128	128	16	99.99	85.71	83.69	83.62
128	128	128	32	99.99	87.85	85.87	86.76
128	**128**	**128**	**64**	99.99	88.59	87.55	**88.06**
128	128	128	128	99.99	86.33	86.55	86.29

classified correctly. However, less than 96% of the samples in the DD-H, D-H, Norm, DE, and KL classes were classified incorrectly due to the high class imbalance in the training set, as shown in Table 3.1. Figure 3.8 shows that the training loss of the model reduced by 91.55%, while the validation loss reduced by 95.10%. This shows that the model neither under-fitted nor over-fitted the network traffic data in the training set. It took 36.14 seconds to train the model with the network traffic samples in the training set, and the model spend 638 milliseconds to classify the network traffic samples in the testing set. This shows that the model was computationally efficient. Therefore, in 11-class classification scenario, four hidden layers with 128, 128, 128, and 64 hidden units, respectively are recommended for developing DNN-based botnet detection model with the Bot-IoT dataset.



	DD-H	DD-T	DD-U	D-H	D-T	D-U	Norm	OSF	SS	DE	KL
DD-H	87.3% 178	0.0% 0	0.0% 0	4.9% 13	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0
DD-T	4.9% 10	99.9% 195119	0.0% 0	3.4% 9	0.0% 25	0.0% 1	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0
DD-U	0.0% 0	0.0% 1	100.0% 190071	0.0% 0	0.0% 1	0.0% 19	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0
D-H	7.8% 16	0.0% 0	0.0% 0	90.3% 242	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0
D-T	0.0% 0	0.1% 154	0.0% 0	0.4% 1	100.0% 122946	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0
D-U	0.0% 0	0.0% 0	0.0% 17	0.7% 2	0.0% 2	100.0% 206769	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0
Norm	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	95.0% 96	0.0% 1	0.0% 2	0.0% 0	0.0% 0
OSF	0.0% 0	0.0% 0	0.0% 0	0.4% 1	0.0% 0	0.0% 0	0.0% 0	99.7% 3571	0.0% 0	0.0% 0	9.1% 1
SS	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	5.0% 5	0.3% 10	100.0% 14411	0.0% 0	0.0% 0
DE	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0
KL	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	100.0% 1	90.9% 10

FIGURE 3.7: Confusion matrix of 11-class DNN model with optimal numbers of hidden layers and hidden units based on the Bot-IoT dataset

Table 3.7 presents the performance in binary classification scenario when the sixteen DNN models were tested with the network traffic samples in the testing set of the N-BaIoT dataset. The DNN model architecture which comprised three hidden layers with 128, 128, and 16 hidden units, respectively has the highest F1 score of 99.98%. The model achieved 99.99% accuracy, 99.97% precision, and 99.98% recall. Figure 3.9 shows that nearly all the samples in the Norm and Attack classes were classified correctly. This implies that the model had a good classification performance. Figure 3.10 shows that the training loss of the model reduced by 86.75%, while the validation loss reduced by 49.98%. This shows that the model neither under-fitted nor over-fitted the network traffic data in the training set. It took 39.68 seconds to train the model with the network traffic samples in the

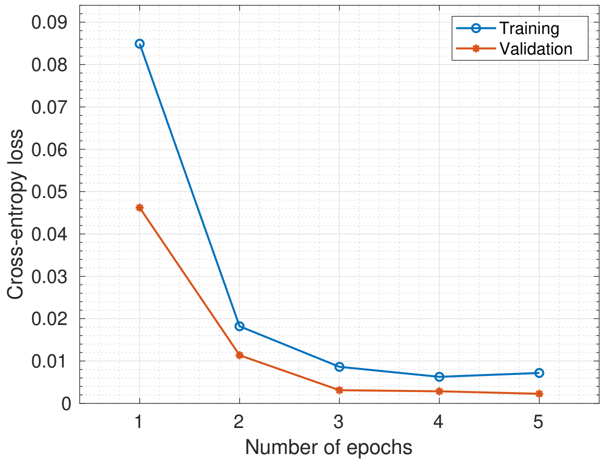


FIGURE 3.8: Training and validation losses of 11-class DNN model with optimal numbers of hidden layers and hidden units based on the Bot-IoT dataset

TABLE 3.7: Binary classification performance for different hidden layers and hidden units based on the N-BaIoT dataset

Hidden Units				Classification Performance (%)			
HL1	HL2	HL3	HL4	A	P	R	F1
16	-	-	-	99.97	99.91	99.90	99.90
32	-	-	-	99.97	99.93	99.91	99.92
64	-	-	-	99.98	99.96	99.95	99.95
128	-	-	-	99.99	99.96	99.95	99.96
128	16	-	-	99.99	99.96	99.95	99.96
128	32	-	-	99.99	99.95	99.97	99.96
128	64	-	-	99.98	99.91	99.97	99.94
128	128	-	-	99.99	99.97	99.96	99.96
128	**128**	**16**	**-**	99.99	99.97	99.98	**99.98**
128	128	32	-	99.99	99.97	99.97	99.97
128	128	64	-	99.99	99.96	99.98	99.97
128	128	128	-	99.98	99.91	99.97	99.94
128	128	128	16	99.99	99.96	99.97	99.97
128	128	128	32	99.98	99.96	99.95	99.95
128	128	128	64	99.99	99.97	99.96	99.97
128	128	128	128	99.98	99.92	99.98	99.95

training set, and the model spend 1.02 seconds to classify the network traffic samples in the testing set. This shows that the model was computationally efficient. Therefore, in 11-class classification scenario, three hidden layers with 128, 128, and 16 hidden units, respectively are recommended for developing DNN-based botnet detection model with the N-BaIoT dataset.

Table 3.8 presents the performance in 10-class classification scenario when

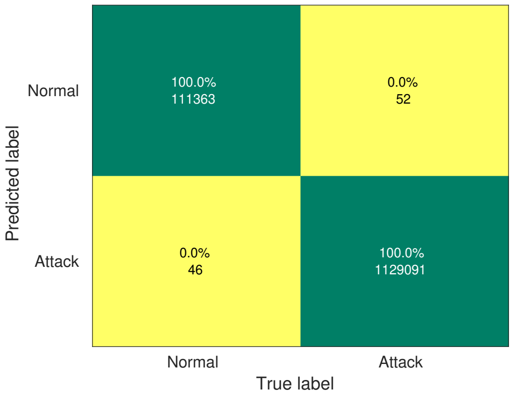


FIGURE 3.9: Confusion matrix of binary DNN model with optimal numbers of hidden layers and hidden units based on the N-BaIoT dataset

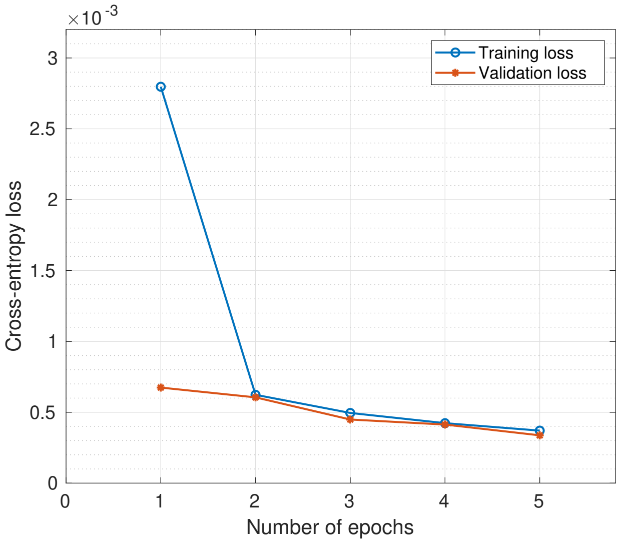


FIGURE 3.10: Training and validation losses of binary DNN model with optimal numbers of hidden layers and hidden units based on the N-BaIoT dataset

the sixteen DNN models were tested with the network traffic samples in the testing set of the N-BaIoT dataset. The DNN model architecture which comprised two hidden layers with 128 and 32 hidden units, respectively has the highest F1 score of 99.87%. The model achieved 99.98% accuracy, 99.86% precision, and 99.89% recall. Figure 3.11 shows that nearly all the samples in each of the 10 classes were classified correctly. This implies that the model had a good classification performance. Figure 3.12 shows that the training loss of the model reduced by 94.01%, while the validation loss reduced by 87.54%. This shows that the model neither under-fitted nor over-fitted the network traffic data in the training set. It took 144.46 seconds to train the

TABLE 3.8: 10-class classification performance for different hidden layers and hidden units based on the N-BaIoT dataset

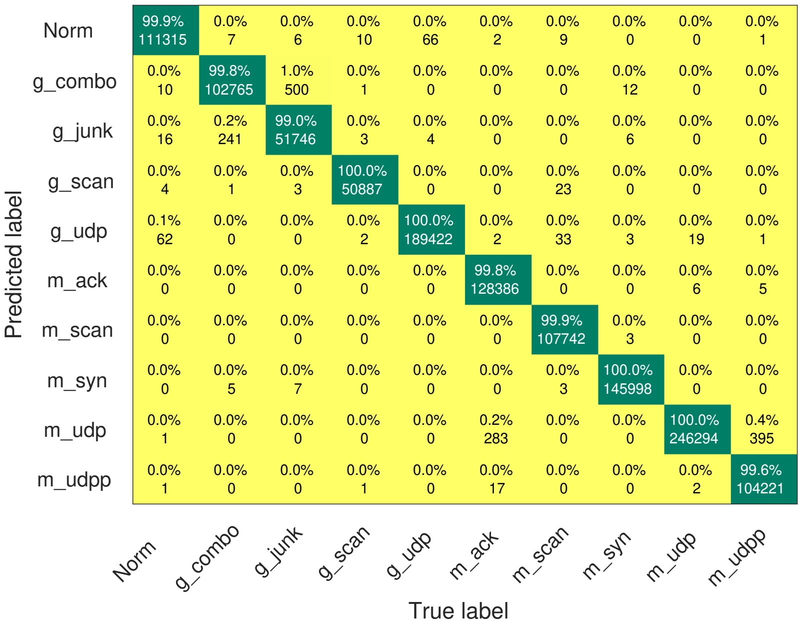
Hidden Units				Classification Performance (%)			
HL1	HL2	HL3	HL4	A	P	R	F1
16	-	-	-	99.76	98.48	98.09	98.27
32	-	-	-	99.75	98.69	98.57	98.63
64	-	-	-	99.88	99.25	99.08	99.16
128	-	-	-	99.94	99.58	99.63	99.60
128	16	-	-	99.94	99.70	99.54	99.62
128	**32**	**-**	**-**	99.98	99.86	99.89	**99.87**
128	64	-	-	99.98	99.85	99.79	99.82
128	128	-	-	99.98	99.85	99.77	99.81
128	128	16	-	99.95	99.71	99.49	99.59
128	128	32	-	99.98	99.86	99.86	99.86
128	128	64	-	99.96	99.72	99.55	99.63
128	128	128	-	99.96	99.77	99.63	99.70
128	128	128	16	99.96	99.76	99.68	99.72
128	128	128	32	99.97	99.83	99.75	99.79
128	128	128	64	99.97	99.81	99.71	99.76
128	128	128	128	99.94	99.66	99.43	99.54

model with the network traffic samples in the training set, and the model spend 2.01 seconds to classify the network traffic samples in the testing set. This shows that the model was computationally efficient. Therefore, in 10-class classification scenario, three hidden layers with 128 and 32 hidden units, respectively are recommended for developing DNN-based botnet detection model with the N-BaIoT dataset.

3.4.2 Optimal Learning Rates

The optimal learning rates in binary and multi-class classification scenarios are determined when the DNN models were developed with the Bot-IoT and N-BaIoT datasets. For each classification scenario, four DNN models were developed with learning rates of 0.1, 0.01, 0.001, and 0.0001, respectively. The optimal numbers of hidden layers and hidden units in Section 3.4.1 and the default values of the other hyperparameters (Adam optimiser, ReLU activation function, a batch size of 512, and 5 epochs) were maintained.

Table 3.9 presents the performance in binary and multi-class classification scenarios when DNN models are developed with different learning rates and tested with the Bot-IoT and N-BaIoT datasets. In all use cases, the



	Norm	g_combo	g_junk	g_scan	g_udp	m_ack	m_scan	m_syn	m_udp	m_udpp
Norm	99.9% 111315	0.0% 7	0.0% 6	0.0% 10	0.0% 66	0.0% 2	0.0% 9	0.0% 0	0.0% 0	0.0% 1
g_combo	0.0% 10	99.8% 102765	1.0% 500	0.0% 1	0.0% 0	0.0% 0	0.0% 0	0.0% 12	0.0% 0	0.0% 0
g_junk	0.0% 16	0.2% 241	99.0% 51746	0.0% 3	0.0% 4	0.0% 0	0.0% 0	0.0% 6	0.0% 0	0.0% 0
g_scan	0.0% 4	0.0% 1	0.0% 3	100.0% 50887	0.0% 0	0.0% 0	0.0% 23	0.0% 0	0.0% 0	0.0% 0
g_udp	0.1% 62	0.0% 0	0.0% 0	0.0% 2	100.0% 189422	0.0% 2	0.0% 33	0.0% 3	0.0% 19	0.0% 1
m_ack	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	99.8% 128386	0.0% 0	0.0% 0	0.0% 6	0.0% 5
m_scan	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	99.9% 107742	0.0% 3	0.0% 0	0.0% 0
m_syn	0.0% 0	0.0% 5	0.0% 7	0.0% 0	0.0% 0	0.0% 0	0.0% 3	100.0% 145998	0.0% 0	0.0% 0
m_udp	0.0% 1	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.2% 283	0.0% 0	0.0% 0	100.0% 246294	0.4% 395
m_udpp	0.0% 1	0.0% 0	0.0% 0	0.0% 1	0.0% 0	0.0% 17	0.0% 0	0.0% 0	0.0% 2	99.6% 104221

FIGURE 3.11: Confusion matrix of 10-class DNN model with optimal numbers of hidden layers and hidden units based on the N-BaIoT dataset

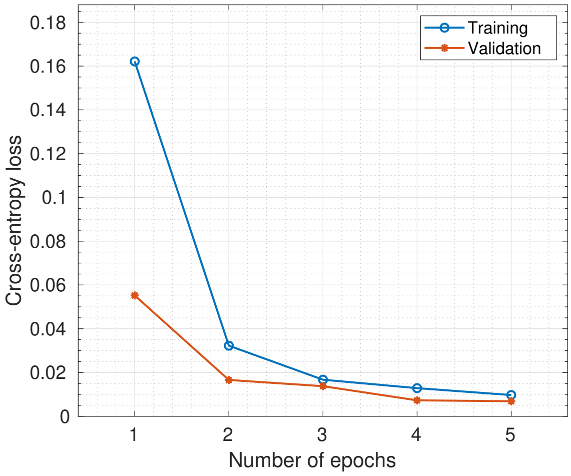


FIGURE 3.12: Training and validation losses of 10-class DNN model with optimal numbers of hidden layers and hidden units based on the N-BaIoT dataset

default learning rate of 0.001 produced the DNN model that had the highest F1 score. Therefore, this learning rate is recommended for developing DNN-based botnet detection model with the Bot-IoT and N-BaIoT datasets.

TABLE 3.9: Classification performance of DNN models for different learning rates

Dataset	Scenario	Learning rate	Classification Performance (%)			
			A	P	R	F1
Bot-IoT	Binary	0.1	100.00	98.85	92.08	95.21
		0.01	100.00	95.05	99.50	97.17
		0.001	100.00	98.10	100.00	**99.03**
		0.0001	100.00	100.00	90.59	94.81
	5-class	0.1	81.02	10.51	20.00	13.78
		0.01	99.94	59.80	59.94	59.87
		0.001	99.97	99.95	97.79	**98.81**
		0.0001	99.60	79.52	70.50	73.66
	11-class	0.1	86.94	2.56	9.09	4.00
		0.01	99.98	74.86	74.12	74.48
		0.001	99.99	88.59	87.55	**88.06**
		0.0001	99.88	76.21	70.16	72.25
N-BaIoT	Binary	0.1	91.02	45.51	50.00	47.65
		0.01	99.98	99.93	99.97	99.95
		0.001	99.99	99.97	99.98	**99.98**
		0.0001	99.98	99.96	99.95	99.95
	10-class	0.1	89.84	41.74	36.08	29.54
		0.01	99.83	98.82	99.29	99.04
		0.001	99.98	99.86	99.89	**99.87**
		0.0001	99.71	98.34	97.99	98.16

3.4.3 Optimisers

The most appropriate optimisers in binary and multi-class classification scenarios are determined when the DNN models were developed with the Bot-IoT and N-BaIoT datasets. For each classification scenario, seven DNN models were developed with Adam, SGD, RMSprop, Adadelta, Adagrad, Adamax, and Nadam optimisers, respectively. The optimal numbers of hidden layers and hidden units in Section 3.4.1, the optimal learning rates in Section 3.4.2, and the default values of the other hyperparameters (ReLU activation function, a batch size of 512, and 5 epochs) were maintained.

Table 3.10 presents the performance in binary and multi-class classification scenarios when DNN models are developed with different learning rates and tested with the Bot-IoT dataset. In the binary classification scenario, Nadam optimiser produced the DNN model that had the highest F1 score of 99.27%. The model achieved 100% accuracy, 98.56% precision, and 100% recall. Figure 3.13 shows that all the benign samples in the Norm class were correctly classified, and only three out of 733604 malicious samples in the

TABLE 3.10: Classification performance of DNN models for different optimisers based on the Bot-IoT dataset

Scenario	Optimiser	Classification Performance (%)			
		A	P	R	F1
Binary	Adam	100.00	98.10	100.00	99.03
	SGD	99.99	49.99	50.00	50.00
	RMSprop	100.00	98.35	93.56	95.83
	Adadelta	99.99	49.99	50.00	50.00
	Adagrad	99.99	49.99	50.00	50.00
	Adamax	100.00	98.84	91.58	94.92
	Nadam	100.00	98.56	100.00	**99.27**
5-class	**Adam**	99.97	99.95	97.79	**98.81**
	SGD	94.39	54.40	42.00	44.93
	RMSprop	99.93	79.92	77.54	78.66
	Adadelta	89.02	29.04	29.56	29.24
	Adagrad	97.04	56.92	56.39	56.57
	Adamax	99.84	78.30	74.48	76.17
	Nadam	99.99	99.35	90.07	93.60
11-class	**Adam**	99.99	88.59	87.55	**88.06**
	SGD	96.67	29.52	30.46	29.86
	RMSprop	99.97	86.94	82.24	84.28
	Adadelta	94.94	26.44	27.29	26.49
	Adagrad	98.51	40.96	42.41	41.59
	Adamax	99.91	76.51	72.90	73.89
	Nadam	99.99	85.23	83.72	84.10

Attack class were misclassified as benign network traffic. This implies that the model had a good classification performance. Figure 3.14 shows that the training loss of the model reduced by 99.52%, while the validation loss reduced by 62.02%. This implies that the model neither under-fitted nor over-fitted the network traffic data in the training set. It took 33.49 seconds to train the model with the network traffic samples in the training set, and the model spend 433 milliseconds to classify the network traffic samples in the testing set. This implies that the model was computationally efficient. Therefore, in binary classification scenario, Nadam optimiser is recommended for developing DNN-based botnet detection model with the Bot-IoT dataset. In the 5-class and 11-class classification scenarios, the default Adam optimiser produced the DNN model that had the highest F1 score of 98.81% and 88.06%, respectively. Therefore, the default Adam optimiser is recommended for developing 5-class and 11-class DNN-based botnet detection models with the Bot-IoT dataset.

Table 3.11 presents the performance in binary and multi-class classification

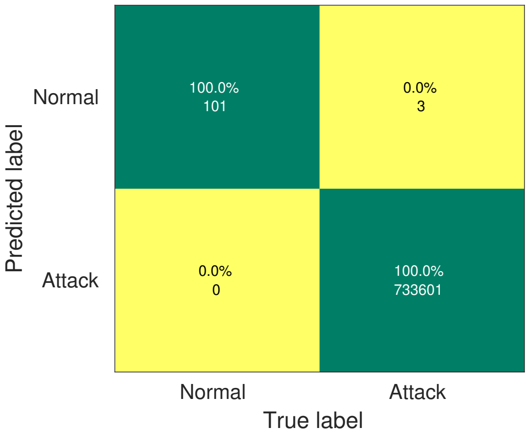


FIGURE 3.13: Confusion matrix of binary DNN model with Nadam optimiser based on the Bot-IoT dataset

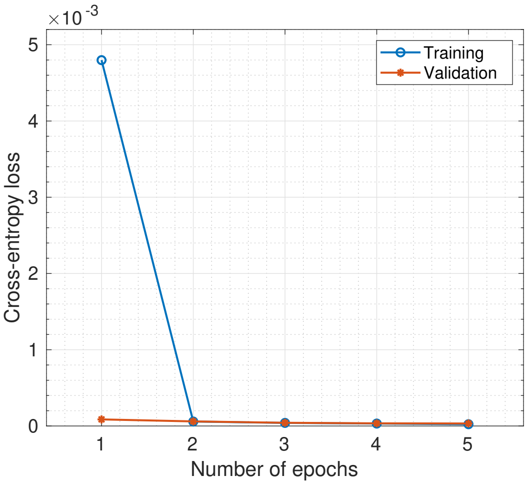


FIGURE 3.14: Training and validation losses of binary DNN model with Nadam optimiser based on the Bot-IoT dataset

scenarios when DNN models are developed with different learning rates and tested with the N-BaIoT dataset. In the binary classification scenario, the default Adam optimiser produced the DNN model that had the highest F1 score of 98.81% and 88.06%, respectively. Therefore, the Adam optimiser is recommended for developing binary DNN-based botnet detection model with the Bot-IoT dataset. In the 10-class classification scenario, Nadam optimiser produced the DNN model that had the highest F1 score of 99.92%. The model achieved 99.99% accuracy, 99.92% precision, and 99.93% recall. Figure 3.15 shows that nearly all the samples in each of the 10 classes were classified correctly. This implies that the model had a good classification performance. Figure 3.16 shows that the training loss of the model reduced by 94.97%, while the validation loss reduced by 89.68%. This implies that

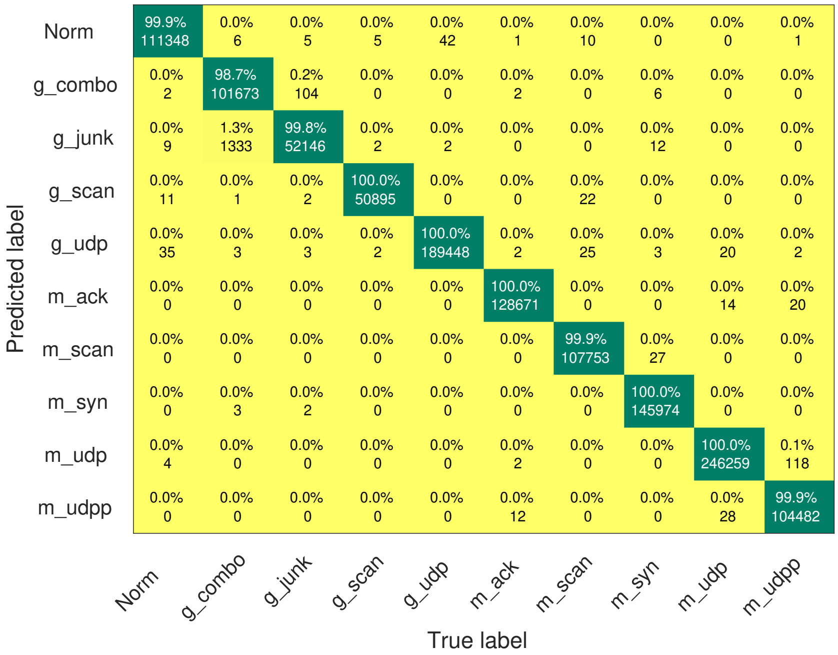
TABLE 3.11: Classification performance of DNN models for different optimisers based on the N-BaIoT dataset

Scenario	Optimiser	Classification Performance (%)			
		A	P	R	F1
Binary	**Adam**	99.99	99.97	99.98	**99.98**
	SGD	99.80	99.26	99.54	99.40
	RMSprop	99.98	99.95	99.94	99.94
	Adadelta	99.80	99.25	99.54	99.40
	Adagrad	99.83	99.36	99.60	99.48
	Adamax	99.98	99.96	99.95	99.95
	Nadam	99.99	99.98	99.96	99.97
10-class	Adam	99.98	99.86	99.89	99.87
	SGD	96.08	79.07	74.97	74.90
	RMSprop	99.48	97.66	98.41	97.90
	Adadelta	95.65	68.56	70.25	67.77
	Adagrad	96.23	80.63	76.39	76.51
	Adamax	99.95	99.72	99.71	99.72
	Nadam	99.99	99.92	99.93	**99.92**

the model neither under-fitted nor over-fitted the network traffic data in the training set. It took 191.46 seconds to train the model with the network traffic samples in the training set, and the model spend 1.11 seconds to classify the network traffic samples in the testing set. This implies that the model was computationally efficient. Therefore, in binary classification scenario, Nadam optimiser is recommended for developing DNN-based botnet detection model with the Bot-IoT dataset. In the 5-class and 11-class classification scenarios, the default Adam optimiser produced the DNN model that had the highest F1 score of 98.81% and 88.06%, respectively. Therefore, the default Adam optimiser is recommended for developing 5-class and 11-class DNN-based botnet detection models with the Bot-IoT dataset.

3.4.4 Optimal Activation Functions

The optimal activation functions in binary and multi-class classification scenarios are determined when the DNN models were developed with the Bot-IoT and N-BaIoT datasets. For each classification scenario, four DNN models were developed with ReLU, tanh, SELU, and ELU activation functions, respectively. The optimal numbers of hidden layers and hidden units in Section 3.4.1, the optimal learning rates in Section 3.4.2, the most appropriate optimisers in Section 3.4.3, and the default values of the other



	Norm	g_combo	g_junk	g_scan	g_udp	m_ack	m_scan	m_syn	m_udp	m_udpp
Norm	99.9% 111348	0.0% 6	0.0% 5	0.0% 5	0.0% 42	0.0% 1	0.0% 10	0.0% 0	0.0% 0	0.0% 1
g_combo	0.0% 2	98.7% 101673	0.2% 104	0.0% 0	0.0% 0	0.0% 2	0.0% 0	0.0% 6	0.0% 0	0.0% 0
g_junk	0.0% 9	1.3% 1333	99.8% 52146	0.0% 2	0.0% 2	0.0% 0	0.0% 0	0.0% 12	0.0% 0	0.0% 0
g_scan	0.0% 11	0.0% 1	0.0% 2	100.0% 50895	0.0% 0	0.0% 0	0.0% 22	0.0% 0	0.0% 0	0.0% 0
g_udp	0.0% 35	0.0% 3	0.0% 3	0.0% 2	100.0% 189448	0.0% 2	0.0% 25	0.0% 3	0.0% 20	0.0% 2
m_ack	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	100.0% 128671	0.0% 0	0.0% 0	0.0% 14	0.0% 20
m_scan	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	99.9% 107753	0.0% 27	0.0% 0	0.0% 0
m_syn	0.0% 0	0.0% 3	0.0% 2	0.0% 0	0.0% 0	0.0% 0	0.0% 0	100.0% 145974	0.0% 0	0.0% 0
m_udp	0.0% 4	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 2	0.0% 0	0.0% 0	100.0% 246259	0.1% 118
m_udpp	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 12	0.0% 0	0.0% 0	0.0% 28	99.9% 104482

FIGURE 3.15: Confusion matrix of 10-class DNN model with Nadam optimiser based on the N-BaIoT dataset

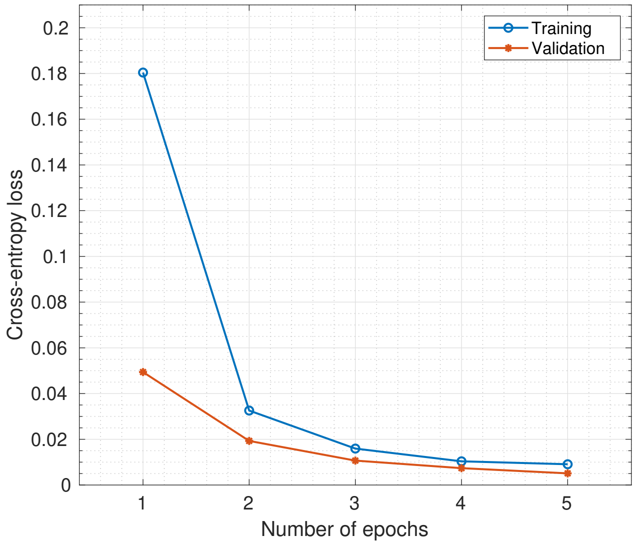


FIGURE 3.16: Training and validation losses of 10-class DNN model with Nadam optimiser based on the N-BaIoT dataset

hyperparameters (a batch size of 512, and 5 epochs) were maintained.

Table 3.12 presents the performance in binary and multi-class classification scenarios when DNN models are developed with different activation functions and tested with the Bot-IoT and N-BaIoT datasets. In all use cases, the default activation function, ReLU, produced the DNN model that had

TABLE 3.12: Classification performance of DNN models for different activation functions

Dataset	Scenario	Activation function	Classification Performance (%)			
			A	P	R	F1
Bot-IoT	Binary	**ReLU**	100.00	98.56	100.00	**99.27**
		tanh	100.00	98.92	95.05	96.91
		SELU	100.00	98.92	95.05	96.91
		ELU	100.00	98.02	98.02	98.02
	5-class	**ReLU**	99.97	99.95	97.79	**98.81**
		tanh	99.99	99.97	97.01	92.05
		SELU	99.97	97.86	96.23	90.75
		ELU	99.97	96.75	95.74	93.82
	11-class	**ReLU**	99.99	88.59	87.55	**88.06**
		tanh	99.99	88.16	82.08	84.03
		SELU	99.99	87.28	85.35	86.20
		ELU	99.99	85.40	85.28	85.18
N-BaIoT	Binary	**ReLU**	99.99	99.97	99.98	**99.98**
		tanh	99.99	99.96	99.96	99.96
		SELU	99.99	99.95	99.97	99.96
		ELU	99.99	99.95	99.97	99.96
	10-class	**ReLU**	99.99	99.92	99.93	**99.92**
		tanh	99.97	99.78	99.85	99.82
		SELU	99.97	99.76	99.82	99.78
		ELU	99.98	99.85	99.86	99.85

the highest F1 score. Therefore, this learning rate is recommended for developing DNN-based botnet detection model with the Bot-IoT and N-BaIoT datasets.

3.4.5 Optimal Batch Sizes

The optimal batch sizes in binary and multi-class classification scenarios are determined when the DNN models were developed with the Bot-IoT and N-BaIoT datasets. For each classification scenario, five DNN models were developed with batch sizes of 64, 128, 256, 512, and 1024, respectively. The optimal numbers of hidden layers and hidden units in Section 3.4.1, the optimal learning rates in Section 3.4.2, the most appropriate optimisers in Section 3.4.3, the optimal activation functions in Section 3.4.5, and the default number of epochs (i.e. 5 epochs) were maintained.

Table 3.13 presents the performance in binary and multi-class classification scenarios when DNN models are developed with different batch sizes and tested with the Bot-IoT and N-BaIoT datasets. In all use cases, the default

TABLE 3.13: Classification performance of DNN models for different batch sizes

Dataset	Scenario	Batch size	Classification Performance (%)			
			A	P	R	F1
Bot-IoT	Binary	64	100.00	97.62	99.50	98.54
		128	100.00	97.20	100.00	98.56
		256	100.00	98.10	100.00	99.03
		512	100.00	98.56	100.00	**99.27**
		1024	100.00	99.01	99.01	99.01
	5-class	64	99.98	98.98	90.65	93.72
		128	99.98	99.97	96.53	98.18
		256	99.99	99.77	97.13	98.40
		512	99.97	99.95	97.79	**98.81**
		1024	99.71	79.67	76.94	78.20
	11-class	64	99.98	86.08	85.53	85.73
		128	99.98	87.18	86.38	86.68
		256	100.00	88.27	87.64	87.92
		512	99.99	88.59	87.55	**88.06**
		1024	99.96	82.76	79.08	80.10
N-BaIoT	Binary	64	99.99	99.96	99.96	99.96
		128	99.99	99.96	99.98	99.97
		256	99.99	99.96	99.97	99.97
		512	99.99	99.97	99.98	**99.98**
		1024	99.99	99.96	99.96	99.96
	10-class	64	99.99	99.89	99.92	99.91
		128	99.99	99.89	99.93	99.91
		256	99.99	99.89	99.92	99.90
		512	99.99	99.92	99.93	**99.92**
		1024	99.97	99.79	99.84	99.82

batch size of 512 produced the DNN model that had the highest F1 score. Therefore, this batch size is recommended for developing DNN-based botnet detection model with the Bot-IoT and N-BaIoT datasets.

3.4.6 Optimal Number of Epochs

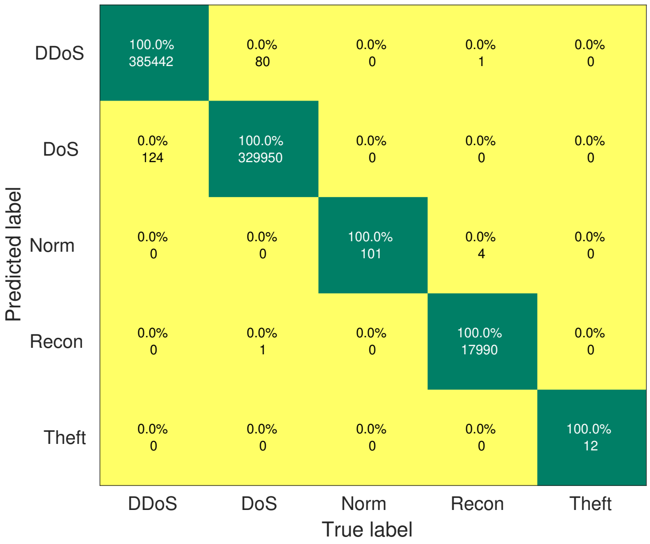
The optimal number of epochs in binary and multi-class classification scenarios are determined when the DNN models were developed with the Bot-IoT and N-BaIoT datasets. For each classification scenario, four DNN models were developed with 5, 10, 15, and 20 epochs, respectively. The optimal numbers of hidden layers and hidden units in Section 3.4.1, the optimal learning rates in Section 3.4.2, the most appropriate optimisers in Section 3.4.3, the optimal activation functions in Section 3.4.5, and the optimal batch sizes in Section 3.4.5 were maintained.

TABLE 3.14: Classification performance of DNN models for different number of epochs

Dataset	Scenario	Batch size	Classification Performance (%)			
			A	P	R	F1
Bot-IoT	Binary	**5**	100.00	98.56	100.00	**99.27**
		10	100.00	98.10	100.00	99.03
		15	100.00	98.10	100.00	99.03
		20	100.00	98.10	100.00	99.03
	5-class	5	99.97	99.95	97.79	98.81
		10	99.99	99.23	99.98	**99.60**
		15	100.00	99.20	99.20	99.20
		20	100.00	99.22	99.79	99.50
	11-class	5	99.99	88.59	87.55	88.06
		10	99.96	87.23	87.15	86.90
		15	100.00	92.60	96.29	**92.19**
		20	100.00	89.65	94.51	89.48
N-BaIoT	Binary	**5**	99.99	99.97	99.98	**99.98**
		10	99.99	99.98	99.98	99.98
		15	99.99	99.96	99.99	99.98
		20	99.99	99.98	99.99	99.98
	10-class	5	99.99	99.92	99.93	99.92
		10	99.99	99.91	99.94	99.92
		15	99.99	99.93	99.92	**99.93**
		20	99.99	99.91	99.88	99.89

Table 3.14 presents the performance in binary and multi-class classification scenarios when DNN models are developed with different number of epochs and tested with the Bot-IoT and N-BaIoT datasets. In the binary classification scenarios, the default number of epochs produced the DNN models that had the highest F1 score of 98.27% and 99.98%, respectively. Therefore, this number of epoch is recommended for developing DNN-based botnet detection model with the Bot-IoT and N-BaIoT datasets in binary classification scenarios.

In 5-class classification scenario, 10 epochs produced the DNN model that had the highest F1 score of 99.60%. The model achieved 99.99% accuracy, 99.23% precision, and 99.98% recall. Figure 3.17 shows that the model correctly classified nearly all the samples in all the five classes. This shows that the model had a good classification performance. Figure 3.18 shows that the training loss of the model reduced by 93.39%, while the validation loss reduced by 98.77%. This shows that the model neither under-fitted nor over-fitted the network traffic data in the training set. It took 65.22 seconds



Predicted label \ True label	DDoS	DoS	Norm	Recon	Theft
DDoS	100.0% 385442	0.0% 80	0.0% 0	0.0% 1	0.0% 0
DoS	0.0% 124	100.0% 329950	0.0% 0	0.0% 0	0.0% 0
Norm	0.0% 0	0.0% 0	100.0% 101	0.0% 4	0.0% 0
Recon	0.0% 0	0.0% 1	0.0% 0	100.0% 17990	0.0% 0
Theft	0.0% 0	0.0% 0	0.0% 0	0.0% 0	100.0% 12

FIGURE 3.17: Confusion matrix of 5-class DNN model with 10 epochs based on the Bot-IoT dataset

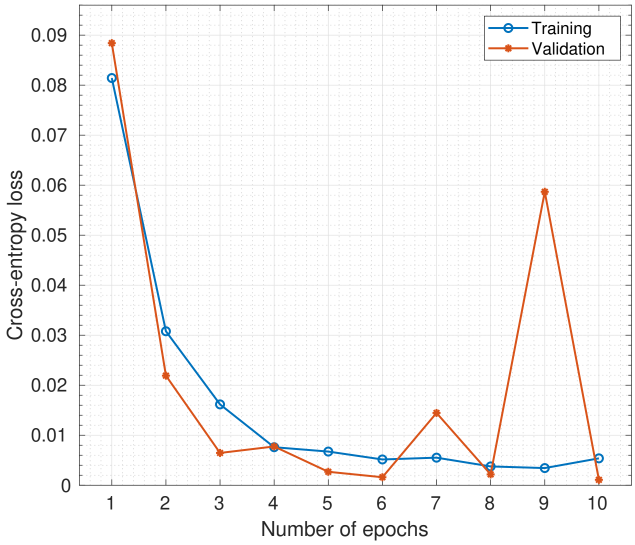


FIGURE 3.18: Training and validation losses of 5-class DNN model with 10 epochs based on the Bot-IoT dataset

to train the model with the network traffic samples in the training set, and the model spend 604 milliseconds to classify the network traffic samples in the testing set. This implies that the model was computationally efficient. Therefore, in 5-class classification scenario, 10 epochs are recommended for developing DNN-based botnet detection model with the Bot-IoT dataset.

In 11-class classification scenario, 15 epochs produced the DNN model that had the highest F1 score of 92.19%. The model achieved 100% accuracy,

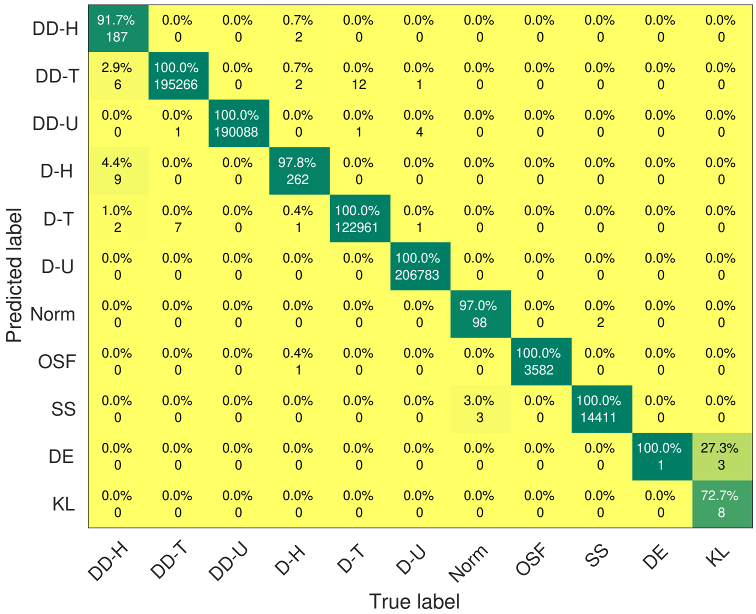


FIGURE 3.19: Confusion matrix of 11-class DNN model with 15 epochs based on the Bot-IoT dataset

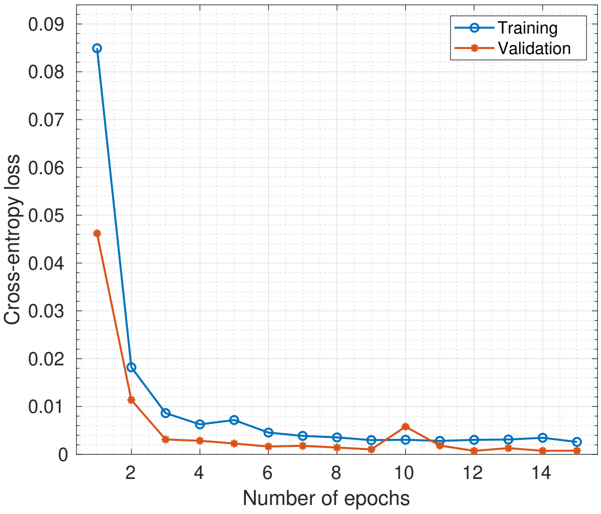
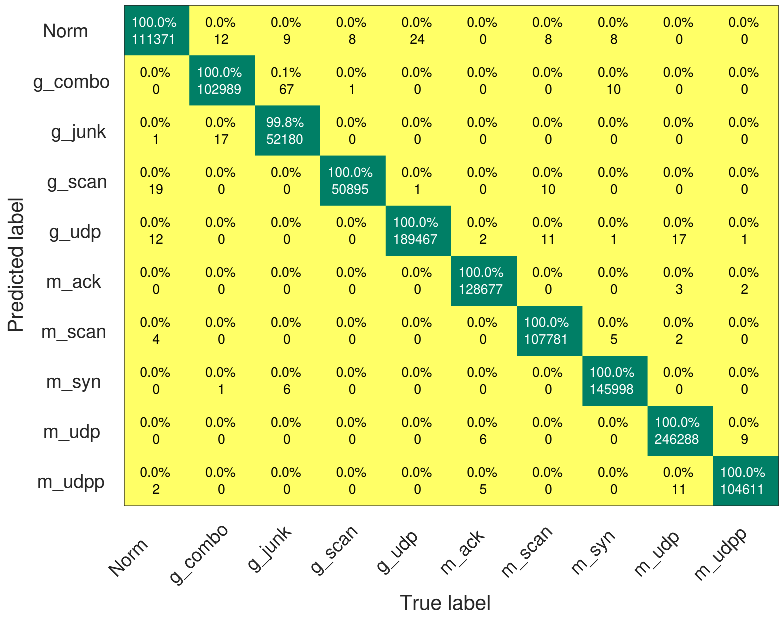


FIGURE 3.20: Training and validation losses of 11-class DNN model with 15 epochs based on the Bot-IoT dataset

92.60% precision, and 96.29% recall. Figure 3.19 shows that nearly all the samples in the DD-T, DD-U, D-T, D-U, OSF, SS, and DE classes were classified correctly. However, less than 98% of the samples in the DD-H, D-H, Norm, DE, and KL classes were classified incorrectly due to the high class imbalance in the training set, as shown in Table 3.1. Figure 3.20 shows that the training loss of the model reduced by 96.94%, while the validation loss reduced by 98.33%. This shows that the model neither under-fitted nor over-fitted the network traffic data in the training set. It took 106.82 seconds to train the model with the network traffic samples in the training set, and the model spend 638 milliseconds to classify the network traffic samples in the testing set. This implies that the model was computationally efficient. Therefore, in 5-class classification scenario, 15 epochs are recommended for developing DNN-based botnet detection model with the Bot-IoT dataset.



	Norm	g_combo	g_junk	g_scan	g_udp	m_ack	m_scan	m_syn	m_udp	m_udpp
Norm	100.0% 111371	0.0% 12	0.0% 9	0.0% 8	0.0% 24	0.0% 0	0.0% 8	0.0% 8	0.0% 0	0.0% 0
g_combo	0.0% 0	100.0% 102989	0.1% 67	0.0% 1	0.0% 0	0.0% 0	0.0% 0	0.0% 10	0.0% 0	0.0% 0
g_junk	0.0% 1	0.0% 17	99.8% 52180	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0
g_scan	0.0% 19	0.0% 0	0.0% 0	100.0% 50895	0.0% 1	0.0% 0	0.0% 10	0.0% 0	0.0% 0	0.0% 0
g_udp	0.0% 12	0.0% 0	0.0% 0	0.0% 0	100.0% 189467	0.0% 2	0.0% 11	0.0% 1	0.0% 17	0.0% 1
m_ack	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	100.0% 128677	0.0% 0	0.0% 0	0.0% 3	0.0% 2
m_scan	0.0% 4	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	100.0% 107781	0.0% 5	0.0% 2	0.0% 0
m_syn	0.0% 0	0.0% 1	0.0% 6	0.0% 0	0.0% 0	0.0% 0	0.0% 0	100.0% 145998	0.0% 0	0.0% 0
m_udp	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 6	0.0% 0	0.0% 0	100.0% 246288	0.0% 9
m_udpp	0.0% 2	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 5	0.0% 0	0.0% 0	0.0% 11	100.0% 104611

FIGURE 3.21: Confusion matrix of 10-class DNN model with 15 epochs based on the Bot-IoT dataset

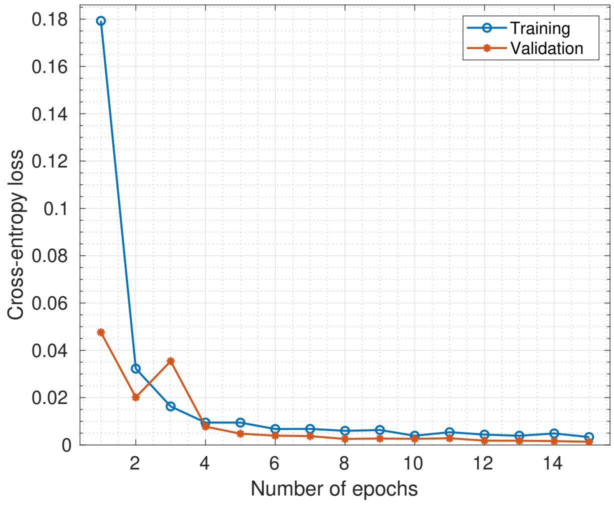


FIGURE 3.22: Training and validation losses of 11-class DNN model with 15 epochs based on the Bot-IoT dataset

In 10-class classification scenario, 15 epochs produced the DNN model that had the highest F1 score of 99.93%. The model achieved 99.99% accuracy, 99.93% precision, and 99.92% recall. Figure 3.21 shows that nearly all the samples in each of the 10 classes were classified correctly. This implies that the model had a good classification performance. Figure 3.22 shows that the training loss of the model reduced by 98.13%, while the validation loss reduced by 97.12%. This shows that the model neither under-fitted nor over-fitted the network traffic data in the training set. It took 168.85 seconds to train the model with the network traffic samples in the training set, and the model spend 1.16 seconds to classify the network traffic samples in the testing set. This implies that the model was computationally efficient. Therefore, in 10-class classification scenario, 15 epochs are recommended for developing DNN-based botnet detection model with the N-BaIoT dataset.

TABLE 3.15: Optimal sets of hyperparameters for DNN models

Data set	Type	Hyperparameters						
		HL	HU	LR	Optimiser	AF	BS	Epoch
Bot-IoT	Binary	2	128, 16	0.001	Nadam	ReLU	512	5
	5-Class	4	128, 128, 128, 16	0.001	Adam	ReLU	512	10
	11-Class	4	128, 128, 128, 64	0.001	Adam	ReLU	512	15
N-BaIoT	Binary	3	128, 128, 16	0.001	Adam	ReLU	512	5
	10-Class	2	128, 32	0.001	Nadam	ReLU	512	15

TABLE 3.16: Classification performance of the optimal DNN models

Data set	Type	Classification Performance (%)				Time (s)	
		A	P	R	F1	Training	Testing
Bot-IoT	Binary	100.00	98.56	100.00	99.27	33.49	0.43
	5-class	99.96	99.20	99.96	99.57	57.78	0.55
	11-class	100.00	92.60	96.29	92.19	168.85	1.16
N-BaIoT	2-class	99.99	99.97	99.98	99.98	39.68	1.02
	10-class	99.99	99.91	99.88	99.89	105.80	1.03

3.5 Chapter Summary

In this chapter, an empirical method was proposed to determine the optimal set of hyperparameters (the numbers of hidden layers and hidden units, learning rate, optimiser, activation function, batch size, and the number of epochs) for efficient DL-based botnet detection in IoT networks. DNN models were trained, validated, and tested with the Bot-IoT and N-BaIoT data sets to evaluate the performance of the proposed method. The Bot-IoT data has three label categories (binary, 5-class, and 11-class), while the N-BaIoT data set has two label categories (binary and 10-class). The optimal sets of hyperparameters for the DNN model are presented in Table 3.15. Experiment results showed that the proposed method produced DNN models that achieved $99.99 \pm 0.02\%$ accuracy, $97.85 \pm 3.77\%$ precision, $98.72 \pm 2.77\%$ recall, and $97.72 \pm 4.51\%$ F1 score, as shown in Table 3.16. Also, it took $14.35 - 260.29$ seconds to train the DNN models with the samples in the training sets, and $0.38 - 1.38$ seconds to classify the samples in the testing sets. Therefore, the proposed method will help cybersecurity experts to develop DNN-based botnet detection models that can detect cyber attacks in IoT and IIoT networks with high accuracy and low false alarm rate.

Chapter 4

SMOTE-DL: An Algorithm for Imbalanced Network Traffic Classification

4.1 Introduction

An optimised DL method was proposed in Chapter 3 to detect botnet attacks in IoT-enabled critical infrastructure. However, when a DL model was developed with highly imbalanced network traffic data in the training set, the classification performance of the model was low i.e. some of the network traffic samples in the testing set were misclassified. If this model is deployed in real-life, that means the botnet attack traffic patterns that were misclassified as benign network traffic will bypass the security system and invade the IoT-enabled critical infrastructure. On the other hand, the benign network traffic patterns that were misclassified as botnet attack traffic will be denied access to the IoT-enabled critical infrastructure because of the false alarm that was raised by the DL-based security system.

In this chapter, a framework named SMOTE-DL is proposed to improve the classification performance of DL-based botnet attack detection models when the network traffic data in the training set is highly imbalanced. First, SMOTE method generates synthetic network traffic samples for the minority classes in the training set to achieve class balance. Then, the optimised DL method, which was proposed in Chapter 3, learns the hierarchical feature representations of the balanced network traffic data for a more accurate classification. The DL architectures that were considered in this chapter include DNN, RNN, LSTM, and GRU. SMOTE-DNN, SMOTE-RNN, SMOTE-LSTM, and SMOTE-GRU models are developed with the highly imbalanced network traffic data in the Bot-IoT dataset to evaluate the effectiveness of the proposed framework. The classification performance of the models is evaluated and compared with that of DNN, RNN, LSTM, and GRU models based on accuracy, precision, recall, and F1 score, while their computation efficiency is evaluated and compared based on sampling time, training time, and testing time.

The rest of the chapter is organised as follows: the details of the SMOTE-DL framework is presented in Section 4.2; the information about the development of the SMOTE-DL models is provided in Section 4.3; the classification performance and the computation efficiency of the models are analysed and discussed in Section 4.4; and finally, the main findings of the chapter is summarised in Section 4.5.

4.2 SMOTE-DL for Imbalanced Network Traffic Classification

In this section, a framework named SMOTE-DL is proposed to improve the classification performance of DL-based botnet detection models when the network traffic data that is used for the training is highly imbalanced. SMOTE-DL framework is a combination of SMOTE method and a DL method. SMOTE method, which was originally developed by Chawla et al [191], generates synthetic network traffic samples for the minority classes to achieve class balance, while the DL method learns the hierarchical representation of the balanced data in the training set to perform network traffic classification with the aim of detecting botnet attacks in IoT-enabled critical infrastructure. Four DL architectures are considered for the classification of the balanced network traffic data, namely DNN, RNN, LSTM, and GRU.

4.2.1 Synthetic Minority Oversampling Technique

A highly imbalanced network traffic data comprised a feature matrix X and a class label vector y. The feature matrices of the majority classes and the minority classes are represented with X_{ma} and X_{mi}, respectively. Unlike the methods in [192], [193], which over-samples the minority classes with replacement, SMOTE creates synthetic network traffic data $X_s \in \mathbb{R}^{n_s \times d}$ for the minority classes to achieve class balance [191]. The number of samples in the synthetic data set n_s is the N_s% of the number of samples in the minority classes n_{mi}. If N_s is less than 100%, a proportion of X_{mi} is selected randomly to generate X_s. Otherwise, n_{mi} is multiplied by a factor of n_{multi} (i.e. $N_s/100$) to generate n_s synthetic samples for a minority class. The euclidean distance between the samples in X_{mi} is computed without considering the nearest neighbour for same sample. One of the k nearest neighbour is randomly selected, and the difference between two closely related samples is calculated. Then, n_s synthetic samples are created along the line segments that joined k nearest neighbours to the minority class.

Algorithm 2: SMOTE Algorithm

```
Input: $X_{mi}$, $N_s$, $k$
Output: $X_s$
if $N_s < 100$ then
    $n_s = (N_s/100) \times n_{mi}$
    $X_r = random(X_{mi}, n_s)$
    $N_s = 100$
end
if $N_s >= 100$ then
    $n_{mult} = int(N_s/100)$
    $n_s = n_{mult} \times n_{mi}$
end
for $index \leftarrow 1$ to $n_s$ do
    $nn\_array = kNN(index, k)$
end
while $N_s \neq 0$ do
    $rn = \text{random}(1, k)$
    $s\_index = 0$
    for $feature = 1$ to $d$ do
        $var1 = X_{mi}[nn\_array[rn]][feature] - X_{mi}[index][feature]$
        $var2 = \text{random}(0, 1)$
        $X_s[s\_index][feature] = X_{mi}[index][feature] + (var1 \times var2)$
    end
    $s\_index = s\_index + 1$
    $N_s = N_s - 1$
end
```

4.2.2 Recurrent Neural Network

Unlike the DNN method that was presented earlier in Section 3.2, RNN uses an additional hidden state vector, $\mathbf{H}_r$, at the input layer of the neural network to learn the temporal dynamics among the network traffic features. The mini-batch of the highly imbalanced network traffic features, x_k, and the initial hidden state $\mathbf{H}_{init}$ are transformed with trainable parameters as follows:

$$\mathbf{H}_1 = \sigma_h \left(\mathbf{W}_x \mathbf{X}_k + \mathbf{W}_h \mathbf{H}_r + \mathbf{B}_h \right), \tag{4.1}$$

where $\mathbf{h}_{1k}$ is the new hidden state when RNN is trained with the k^{th} mini-batch; $\mathbf{W}_x$ and $\mathbf{W}_h$ are the weights used for linear transformation of $\mathbf{X}_k$ and $\mathbf{h}_{init}$ respectively; and $\mathbf{b}_h$ is the bias. A complete information about the RNN algorithm is presented in Algorithm 3.

Algorithm 3: RNN Algorithm

```
Input: $\mathbf{X}$
Target: $\mathbf{y}$
Output: $\tilde{\mathbf{y}}$
$\mathbf{h}_0 = \mathbf{h}_{init}$
for $e = 1$ to $u$ do
    for $k = 1$ to $n$ do
        $\mathbf{h}_{1k} = \sigma_h \left( \mathbf{W}_{x1} \mathbf{x}_k + \mathbf{W}_{h1} \mathbf{h}_0 + \mathbf{b}_{1h} \right)$
        for $m = 2$ to $(d + 1)$ do
            $\mathbf{h}_{mk} = \sigma_h \left( \mathbf{W}_{hm} \mathbf{h}_{(m-1)k} + \mathbf{b}_{mh} \right)$
        end
        $\tilde{\mathbf{y}}_k = \sigma_y \left( \mathbf{W}_y \mathbf{h}_{mk} + \mathbf{b}_y \right)$
        $L_k = \theta(\mathbf{y}_k, \tilde{\mathbf{y}}_k)$
    end
    $L = \sum_{k=1}^{n} L_k$
    $\mathbf{W}'_{(\cdot)}, \mathbf{b}'_{(\cdot)} = \psi \left( \mathbf{W}_{(\cdot)}, \mathbf{b}_{(\cdot)} \right)$
end
```

4.2.3 Long Short-Term Memory

LSTM was proposed to address vanishing and exploding gradient problem in RNN [194]. Input gate ($\mathbf{i}_k$), cell state ($\mathbf{c}_k$) and output gate ($\mathbf{o}_k$) were introduced to learn long-term dependencies in the temporal dynamics of $\mathbf{X}_k$. The input gate determines the new information to be stored in the cell state while the output gate determines the output information based on the cell state. Forget gate ($\mathbf{f}_k$) was added to determine the information to be removed from the cell state [195]. The performance of LSTM improves as the bias of the forget gate, $\mathbf{b}_f$, increases [196]. LSTM layer is represented by

Eq. 4.2:

$$\begin{cases} \mathbf{i}_k = \sigma_h \left(\mathbf{W}_{ix}\mathbf{x}_k + \mathbf{W}_{ih}\mathbf{h}_0 + \mathbf{b}_i\right), \\ \mathbf{f}_k = \sigma_h \left(\mathbf{W}_{fx}\mathbf{x}_k + \mathbf{W}_{fh}\mathbf{h}_0 + \mathbf{b}_f\right), \\ \tilde{\mathbf{c}}_k = \sigma_h \left(\mathbf{W}_{\tilde{c}x}\mathbf{x}_k + \mathbf{W}_{\tilde{c}h}\mathbf{h}_0 + \mathbf{b}_{\tilde{c}}\right), \\ \mathbf{c}_k = \mathbf{i}_k \odot \tilde{\mathbf{c}}_k, \\ \mathbf{o}_k = \sigma_h \left(\mathbf{W}_{ox}\mathbf{x}_k + \mathbf{W}_{oh}\mathbf{h}_0 + \mathbf{b}_o\right), \\ \mathbf{h}_{1k} = \mathbf{o}_k \odot \sigma_h \left(\mathbf{c}_k\right). \end{cases} \tag{4.2}$$

A complete information about the LSTM algorithm is presented in Algorithm 4.

Algorithm 4: LSTM Algorithm

```
Input: $\mathbf{X}$
Target: $\mathbf{y}$
Output: $\tilde{\mathbf{y}}$
$\mathbf{h}_0 = \mathbf{h}_{init}$
for $e = 1$ to $u$ do
    for $k = 1$ to $n$ do
        $\mathbf{i}_k = \sigma_h \left(\mathbf{W}_{ix}\mathbf{x}_k + \mathbf{W}_{ih}\mathbf{h}_0 + \mathbf{b}_i\right)$
        $\mathbf{f}_k = \sigma_h \left(\mathbf{W}_{fx}\mathbf{x}_k + \mathbf{W}_{fh}\mathbf{h}_0 + \mathbf{b}_f\right)$
        $\tilde{\mathbf{c}}_k = \sigma_h \left(\mathbf{W}_{\tilde{c}x}\mathbf{x}_k + \mathbf{W}_{\tilde{c}h}\mathbf{h}_0 + \mathbf{b}_{\tilde{c}}\right)$
        $\mathbf{c}_k = \mathbf{i}_k \odot \tilde{\mathbf{c}}_k$
        $\mathbf{o}_k = \sigma_h \left(\mathbf{W}_{ox}\mathbf{x}_k + \mathbf{W}_{oh}\mathbf{h}_0 + \mathbf{b}_o\right)$
        $\mathbf{h}_{1k} = \mathbf{o}_k \odot \sigma_h \left(\mathbf{c}_k\right)$
        for $m = 2$ to $(d + 1)$ do
            $\mathbf{h}_{mk} = \sigma_h \left(\mathbf{W}_{hm}\mathbf{h}_{(m-1)k} + \mathbf{b}_{mh}\right)$
        end
        $\tilde{\mathbf{y}}_k = \sigma_y \left(\mathbf{W}_y\mathbf{h}_{mk} + \mathbf{b}_y\right)$
        $L_k = \theta(\mathbf{y}_k, \tilde{\mathbf{y}}_k)$
    end
    $L = \sum_{k=1}^{n} \theta(\mathbf{y}_k, \tilde{\mathbf{y}}_k)$
    $\mathbf{W}^{'}_{(\cdot)}, \mathbf{b}^{'}_{(\cdot)} = \psi \left(\mathbf{W}_{(\cdot)}, \mathbf{b}_{(\cdot)}\right)$
end
```

4.2.4 Gated Recurrent Unit

GRU is an extension of LSTM with a simplified gated mechanism [197]. The input and forget gates were replaced with an update gate, $\mathbf{z}_k$), to reduce the computational burden without significantly degrading classification performance. Also, a reset gate, $\mathbf{r}_k$, was introduced to control the amount of information that will be eliminated in $\mathbf{h}_{1k}$ before it is combined with $\mathbf{x}_k$ to produce a new hidden state vector. The update gate determines the percentage of $\mathbf{h}_{1k}$ and $\tilde{\mathbf{h}}_{1k}$ that will be in the next hidden state vector. The outputs of the two gates depend on past hidden states and the current input, $\mathbf{x}_k$. GRU layer is represented by Eq. (4.3):

$$\begin{cases} \mathbf{r}_k = \sigma_h \left(\mathbf{W}_{rx}\mathbf{x}_k + \mathbf{W}_{rh}\mathbf{h}_0 + \mathbf{b}_r \right), \\ \mathbf{z}_k = \sigma_h \left(\mathbf{W}_{zx}\mathbf{x}_k + \mathbf{W}_{zh}\mathbf{h}_k + \mathbf{b}_z \right), \\ \tilde{\mathbf{h}}_k = \sigma_h \left(\mathbf{W}_{\tilde{h}x}\mathbf{x}_k + \mathbf{W}_{\tilde{h}h} \left(\mathbf{r}_k \odot \mathbf{h}_k \right) + \mathbf{b}_z \right), \\ \mathbf{h}_{1k} = (1 - \mathbf{z}_k) \odot \mathbf{h}_k + \mathbf{z}_k \odot \tilde{\mathbf{h}}_k. \end{cases} \tag{4.3}$$

A complete information about GRU algorithm is presented in Algorithm 5.

Algorithm 5: GRU Algorithm

```
Input: $\mathbf{X}$
Target: $\mathbf{y}$
Output: $\tilde{\mathbf{y}}$
$\mathbf{h}_0 = \mathbf{h}_{init}$
for $e = 1$ to $u$ do
    for $k = 1$ to $n$ do
        $\mathbf{r}_k = \sigma_h \left( \mathbf{W}_{rx}\mathbf{x}_k + \mathbf{W}_{rh}\mathbf{h}_0 + \mathbf{b}_r \right)$
        $\mathbf{z}_k = \sigma_h \left( \mathbf{W}_{zx}\mathbf{x}_k + \mathbf{W}_{zh}\mathbf{h}_k + \mathbf{b}_z \right)$
        $\tilde{\mathbf{h}}_k = \sigma_h \left( \mathbf{W}_{\tilde{h}x}\mathbf{x}_k + \mathbf{W}_{\tilde{h}h} \left( \mathbf{r}_k \odot \mathbf{h}_k \right) + \mathbf{b}_z \right)$
        $\mathbf{h}_{1k} = (1 - \mathbf{z}_k) \odot \mathbf{h}_k + \mathbf{z}_k \odot \tilde{\mathbf{h}}_k$
        for $m = 2$ to $(d + 1)$ do
            $\mathbf{h}_{mk} = \sigma_h \left( \mathbf{W}_{hm}\mathbf{h}_{(m-1)k} + \mathbf{b}_{mh} \right)$
        end
        $\tilde{\mathbf{y}}_k = \sigma_y \left( \mathbf{W}_y\mathbf{h}_{mk} + \mathbf{b}_y \right)$
        $L_k = \theta(\mathbf{y}_k, \tilde{\mathbf{y}}_k)$
    end
    $L = \sum_{k=1}^{n} \theta(\mathbf{y}_k, \tilde{\mathbf{y}}_k)$
    $\mathbf{W}^{'}_{(\cdot)}, \mathbf{b}^{'}_{(\cdot)} = \psi \left( \mathbf{W}_{(\cdot)}, \mathbf{b}_{(\cdot)} \right)$
end
```

4.3 Model Development and Experiment

In this section, the proposed SMOTE-DL framework is implemented with the highly imbalanced 11-class network traffic data in the Bot-IoT dataset, as shown in Figure 4.1. The sample distribution of the data and the data preprocessing procedure were presented earlier in Section 3.3.

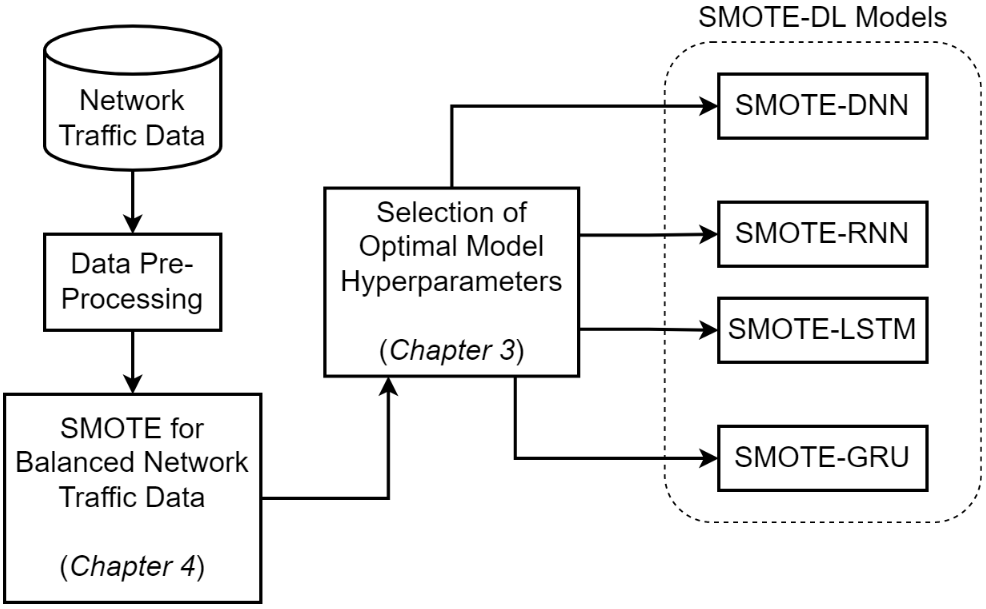


FIGURE 4.1: Framework for the development of SMOTE-DL model

Table 4.1 shows that the *DD-H*, *D-H*, *Norm*, *DE*, and *KL* classes have lower class imbalance ratios ($< 1 : 57$), compared to the other six classes. Therefore, these five classes are referred to as the minority classes. The SMOTE method was used to generate synthetic network traffic data for the minority classes such that the class imbalance ratio of each of these classes will increase to at least 1:57. The highly imbalanced network traffic data was used to train four DL models i.e. DNN, RNN, LSTM, and GRU models. On the other hand, the balanced network traffic data was used to train four SMOTE-DL models named SMOTE-DNN, SMOTE-RNN, SMOTE-LSTM, and SMOTE-GRU models. The optimal set of hyperparameters that were recommended for 11-class classification scenario in Chapter 3 (Table 3.15) were used for the model development in this chapter. The robustness of the DL and SMOTE-DL models was evaluated using the validation set, while the generalisation ability of these model was evaluated using the testing set. The sample distribution of the network traffic data in the validation and testing sets are presented in Table 3.1.

TABLE 4.1: Class-imbalance ratio of 11-Class Bot-IoT dataset

Class	No. of Samples	Class Imbalance Ratio
DD-H	588	1:1053
DD-T	586393	1:1
DD-U	568760	1:1
D-H	906	1:684
D-T	369965	1:2
D-U	619414	1:1
Norm	290	1:2136
OSF	10795	1:57
SS	43949	1:14
DE	4	1:154854
KL	48	1:12904

4.4 Result Analysis and Discussion

In this section, the results of the model development and the experiment are analysed to evaluate the evaluate the effectiveness of the proposed SMOTE-DL algorithm. The sample distribution of the newly generated network traffic data is presented. The robustness of the models against under-fitting and over-fitting is evaluated based on the cross-entropy loss during training and validation. The classification performance of the models is evaluated based on accuracy, precision, recall, and F1 score. Finally, the computation efficiency of the models is evaluated based on the training time and the testing time.

4.4.1 Generation of Synthetic Network Traffic Data

Table 4.2 presents the sample distribution of the highly imbalanced as well as the balanced network traffic data in the training set of the Bot-IoT dataset. The SMOTE method generated a total of 52139 synthetic samples to increase the low class imbalance ratio of the number of samples in a minority class to the number of samples in the majority class. In the *DD-H* class, 10207 synthetic samples were generated, and this increased the class imbalance ratio from 1:1053 to 1:57. In the *D-H* class, 9889 synthetic samples were generated, and this increased the class imbalance ratio from 1:684 to 1:57. In the *Norm* class, 10505 synthetic samples were generated, and this increased the class imbalance ratio from 1:2136 to 1:57. In the *DE* class, 10791 synthetic samples were generated, and this increased the class imbalance ratio from 1:154854 to 1:57. In the *KL* class, 10747 synthetic samples were generated, and this increased the class imbalance ratio from 1:12904 to 1:57. The increase in the class imbalance ratio helped the SMOTE-DL models to achieve high classification performance, as discussed later in Section 4.4.3. The synthetic data generation experiment was performed at four different times to determine the computation complexity of the process. Figure 4.2 shows that the sampling time for the synthetic data generation was 880 − 930 milliseconds. Therefore, the process will not increase the computation complexity of the DL-based botnet attack detection models.

TABLE 4.2: New training set for 11-class classification

Class	Training data		
	Original	Generated	New
DD-H	588	10207	10795
DD-T	586393	0	586393
DD-U	568760	0	568760
D-H	906	9889	10795
D-T	369965	0	369965
D-U	619414	0	619414
Norm	290	10505	10795
OSF	10795	0	10795
SS	43949	0	43949
DE	4	10791	10795
KL	48	10747	10795

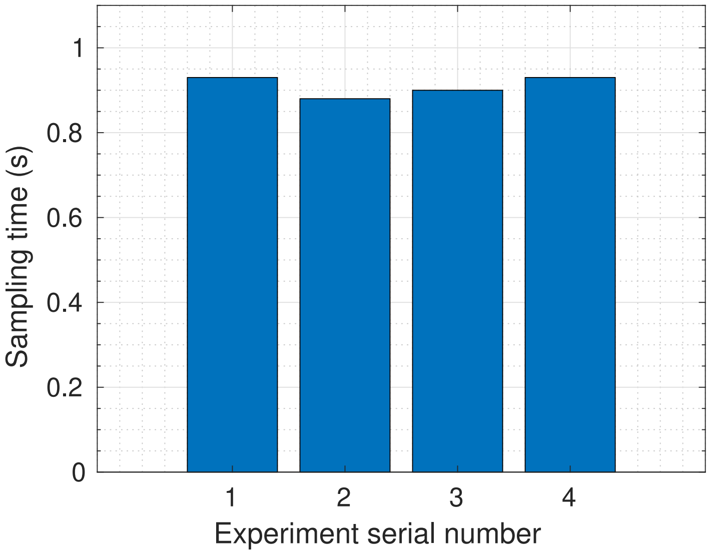


FIGURE 4.2: Sampling time of the SMOTE method

4.4.2 Robustness against Under-fitting and Over-fitting

Figure 4.3 shows the cross-entropy losses of DL models during training and validation. The training and validation losses of the models reduced significantly as the number of epochs increased from 1 to 15. Specifically, the training loss of the DNN model reduced by 96.94%, while its validation loss reduced by 98.33%. The training loss of the RNN model reduced by 95.74%, while its validation loss reduced by 98.52%. The training loss of the LSTM model reduced by 97.60%, while its validation loss reduced by 95.64%. The training loss of the GRU model reduced by 96.56%, while its validation loss reduced by 95.54%. The significant reduction in the training and validation losses implies that the DL models neither under-fit nor over-fit the highly imbalanced network traffic data in the training set.

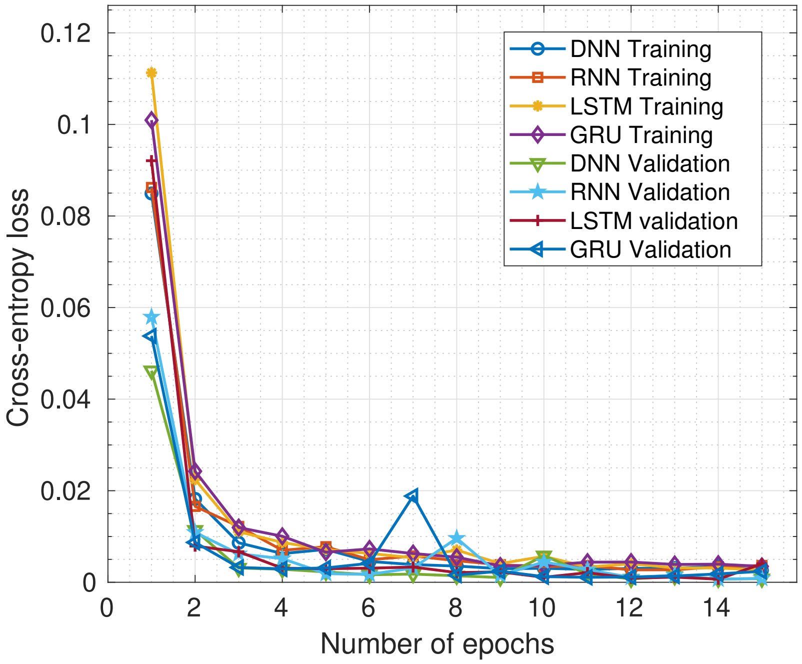


FIGURE 4.3: Cross-entropy loss of DL models during training and validation

Figure 4.4 shows the cross-entropy losses of SMOTE-DL models during training and validation. The training and validation losses of the models reduced significantly as the number of epochs increased from 1 to 15. Specifically, the training loss of the SMOTE-DNN model reduced by 97.60%, while its validation loss reduced by 97.26%. The training loss of the RNN model reduced by 97.23%, while its validation loss reduced by 96.65%. The training loss of the LSTM model reduced by 97.65%, while its validation loss reduced by 97.17%. The training loss of the GRU model reduced by 96.45%, while its validation loss reduced by 98.27%. The significant reduction in the training and validation losses implies that the SMOTE-DL models neither under-fit nor over-fit the balanced network traffic data in the training set.

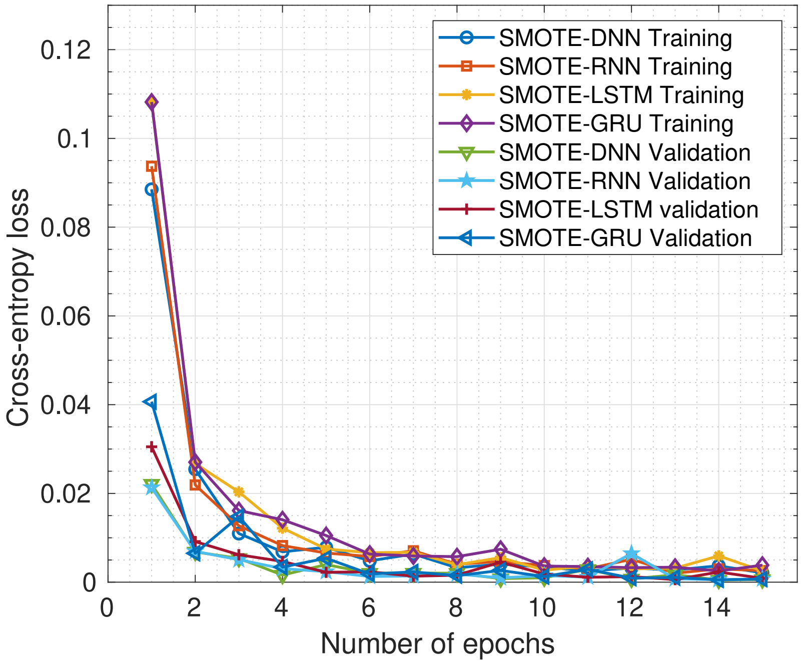


FIGURE 4.4: Cross-entropy loss of SMOTE-DL models during training and validation

4.4.3 Performance of the DL and SMOTE-DL Models

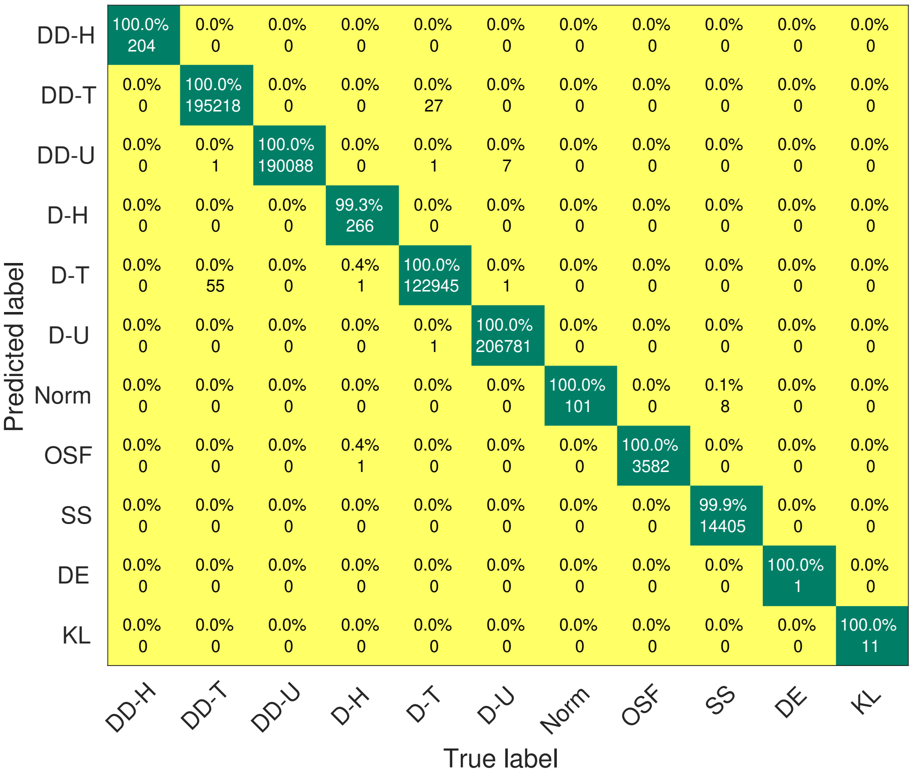
Table 4.3 presents the classification performance of the DL and SMOTE-DL models. The DL models were trained with the highly imbalanced network traffic data, while the SMOTE-DL models were trained with the balanced network traffic data. The SMOTE-DL models achieved a better classification performance than the DL models. The precision, recall, and F1 score of the SMOTE-DL models were higher than those of the DL models by $6.4 - 13.48\%$, $3.12 - 12.81\%$, and $7.01 - 13.27\%$, respectively. The SMOTE-LSTM model achieved the best classification performance with 100% accuracy, 94.20% precision, 98.99% recall, and 95.83% F1 score.

The confusion matrix in Figure 4.5 shows that the SMOTE-LSTM model correctly classified all the network traffic samples in the testing set, except very few samples in the *D-H* and *SS* classes. In the *D-H* class, 99.3% of the attack traffic samples were correctly classified, one sample was misclassified as *D-T* attack traffic, and another sample was misclassified as *OSF* attack traffic. In the *SS* class, 99.9% of the attack traffic samples were correctly classified, and only 0.1% of the samples were misclassified as benign network traffic.

Figure 4.6 shows the time taken to train the DL and SMOTE-DL models with

TABLE 4.3: Classification performance of the DL and SMOTE-DL models

Model	Classification Performance (%)			
	A	P	R	F1
DNN	100.00	91.14	93.76	89.67
RNN	99.98	85.17	89.11	86.58
LSTM	100.00	92.13	97.33	93.17
GRU	100.00	91.96	96.73	92.08
SMOTE-DNN	100.00	98.65	99.87	99.23
SMOTE-RNN	100.00	98.40	99.67	99.01
SMOTE-LSTM	100.00	94.20	98.99	95.83
SMOTE-GRU	100.00	98.83	99.79	99.28



	DD-H	DD-T	DD-U	D-H	D-T	D-U	Norm	OSF	SS	DE	KL
DD-H	100.0% 204	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0
DD-T	0.0% 0	100.0% 195218	0.0% 0	0.0% 0	0.0% 27	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0
DD-U	0.0% 0	0.0% 1	100.0% 190088	0.0% 0	0.0% 1	0.0% 7	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0
D-H	0.0% 0	0.0% 0	0.0% 0	99.3% 266	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0
D-T	0.0% 0	0.0% 55	0.0% 0	0.4% 1	100.0% 122945	0.0% 1	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0
D-U	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 1	100.0% 206781	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0
Norm	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	100.0% 101	0.0% 0	0.1% 8	0.0% 0	0.0% 0
OSF	0.0% 0	0.0% 0	0.0% 0	0.4% 1	0.0% 0	0.0% 0	0.0% 0	100.0% 3582	0.0% 0	0.0% 0	0.0% 0
SS	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	99.9% 14405	0.0% 0	0.0% 0
DE	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	100.0% 1	0.0% 0
KL	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	0.0% 0	100.0% 11

FIGURE 4.5: Confusion matrix of the SMOTE-LSTM model

the highly imbalanced network traffic data and the balanced network traffic data, respectively. The training of SMOTE-DL models took slightly longer time than that of DL models. The DL models spent 106.82 − 214.59 seconds to learn the feature representation of 2201112 network traffic samples, while the SMOTE-DL models spent 109.11 − 220.54 seconds to learn the feature representation of 2253251 network traffic samples. The DNN model had the least training time of 106.82 seconds, while the SMOTE-LSTM model had the slowest training time of 220.54 seconds. SMOTE-DL models took longer time because the number of samples in the balanced network traffic data is greater than the number of samples in the highly imbalanced network traffic data that was used to train DL models.

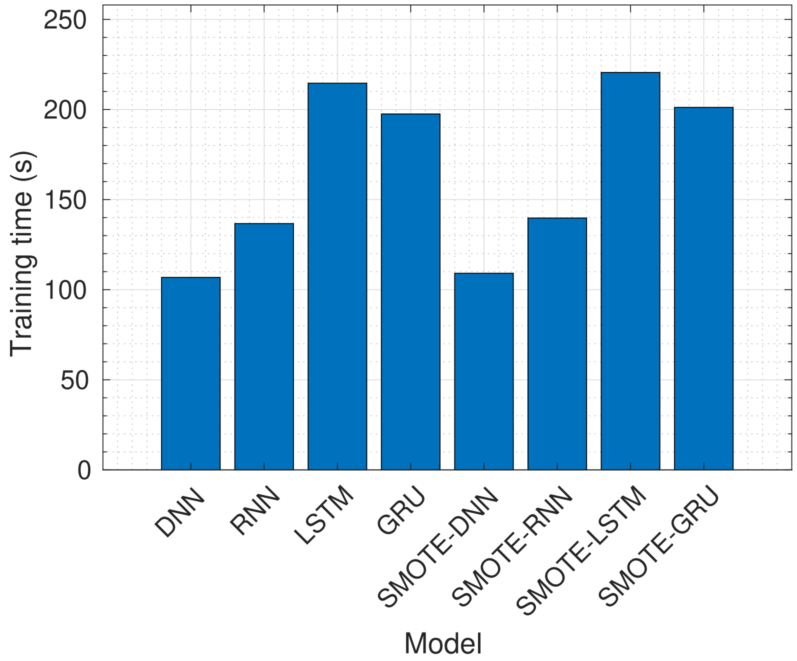


FIGURE 4.6: Training time of DL and SMOTE-DL models

Figure 4.7 shows the time taken by DL and SMOTE-DL models to classify 733705 network traffic samples in the testing set. There is no significant difference between the testing time of DL and SMOTE-DL models. The DL models spent $0.64 - 1.3$ seconds, while the SMOTE-DL models spent $0.64 - 1.31$ seconds. Therefore, these models are suitable for near real-time botnet attack detection in IoT-enabled critical infrastructure.

4.5 Chapter Summary

In this Chapter, a framework named SMOTE-DL was proposed to address the problem of high class imbalance in the training set that is used to develop DL-based botnet attack detection models for IoT-enabled critical infrastructure. The framework was implemented with the 11-class network traffic data in the Bot-IoT dataset. Four DL models (DNN, RNN, LSTM, and GRU) were trained with highly imbalanced network traffic data. SMOTE method was used to generate synthetic network traffic data to increase the

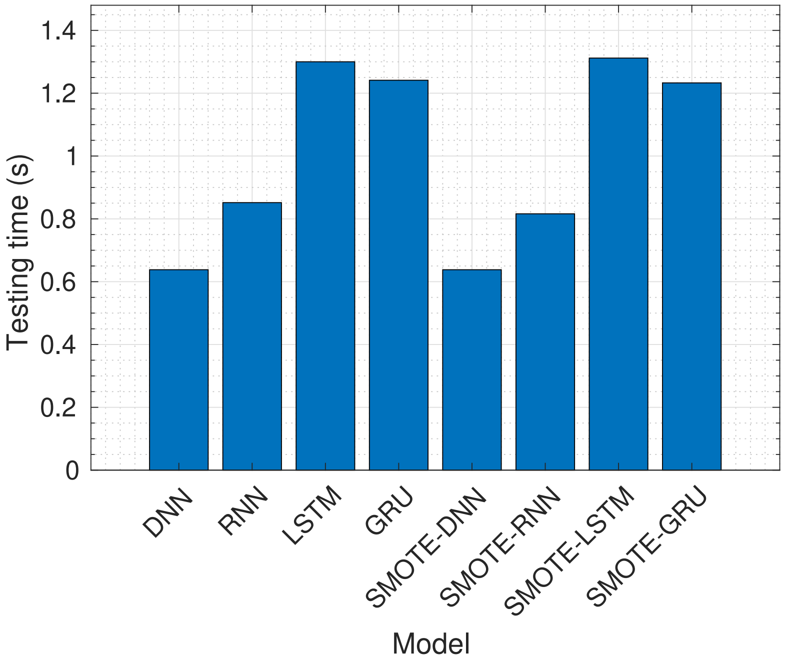


FIGURE 4.7: Testing time of DL and SMOTE-DL models

class imbalance ratio in the minority classes and achieve class balance. Four SMOTE-DL models (SMOTE-DNN, SMOTE-RNN, SMOTE-LSTM, and SMOTE-GRU) were trained with the balanced network traffic data. Results showed that SMOTE method increased the class imbalance ratio from 1:154854 to 1:57. Also, the DL and SMOTE-DL models were robust against under-fitting and over-fitting. More importantly, the SMOTE-DL models achieved a better classification performance than the DL models. The precision, recall, and F1 score of the SMOTE-DL models were higher than those of the DL models by $6.4 - 13.48\%$, $3.12 - 12.81\%$, and $7.01 - 13.27\%$, respectively. Specifically, the SMOTE-LSTM model achieved the best classification performance with 100% accuracy, 94.20% precision, 98.99% recall, and 95.83% F1 score. Interestingly, the SMOTE method did not introduce any significant computation complexity to the DL-based botnet attack detection process. Therefore, SMOTE-LSTM architecture will be used for near real-time botnet attack detection in subsequent chapters of this thesis.

Chapter 5

Hybrid Deep Learning for Memory-Efficient Botnet Detection

5.1 Introduction

In Chapters 3 and 4, optimised DL and SMOTE-DL algorithms were proposed, respectively for botnet attack detection in IoT-enabled critical infrastructure. The optimised DL method is suitable for network traffic classification when the class imbalance ratio is less or equal to 42328, while the SMOTE-DL method helps to improve the classification performance when the network traffic data in the training set is highly imbalanced. However, these methods rely on high-dimensional network traffic data to train, validate, and test the botnet attack detection models. Consequently, a large memory space will be needed to store the data either on a central server or an IoT edge node.

In this chapter, a hybrid DL method, which combines LAE and LSTM architectures, is proposed for memory-efficient botnet attack detection in IoT-enabled critical infrastructure. The LAE method is used to produce a latent-space feature representation of the high-dimensional network traffic data without loosing useful intrinsic network information, while BLSTM method is used to learn the hierarchical feature representations and long-term inter-related changes directly from the low-dimensional data to distinguish botnet attack traffic from benign traffic in IoT-enabled critical infrastructure. The hybrid DL method is implemented using the Bot-IoT and the N-BaIoT datasets to evaluate the effectiveness of the method in binary, 5-class, 10-class, and 11-class classification scenarios. The robustness of the LAE-BLSTM models against under-fitting and over-fitting is evaluated based on cross-entropy loss during training and validation. The classification performance of the models is evaluated based on accuracy, precision, recall, and F1 score. The computation efficiency of the models is evaluated based on the encoding time, training time, and testing time.

The rest of the chapter is organised as follows: the details of the hybrid DL framework is presented in Section 5.2; the information about the development of the LAE-BLSTM models is provided in Section 5.3; the classification performance and the computation efficiency of the models are analysed and discussed in Section 5.4; and finally, the main findings of the chapter is summarised in Section 5.5.

5.2 Hybrid Deep Learning Framework

In this section, a hybrid DL framework, named LAE-BLSTM, is proposed for efficient botnet attack detection in IoT-enabled critical infrastructure. The description of LAE and deep BLSTM methods is presented in Algorithms 6 and 7, respectively. In this paper, boldface uppercase alphabets and boldface lower case alphabets represent matrices and column vectors respectively.

Algorithm 6: LAE algorithm

```
Input: $\mathbf{X}$
Initialization: $\mathbf{h}_0(j) = \mathbf{x}(j), \forall j \in [1, k]$
        $h_d(0) = x(d), \forall d \in [1, n]$
Output: $\tilde{\mathbf{X}}$
for $d = 1$ to $n$ do
    for $j = 1$ to $k$ do
        $\mathbf{i}_j = \sigma_r\left(\mathbf{W}_{ix}\mathbf{x}_j + \mathbf{W}_{ih}\mathbf{h}_{j-1} + \mathbf{b}_i\right)$
        $\mathbf{f}_j = \sigma_r\left(\mathbf{W}_{fx}\mathbf{x}_j + \mathbf{W}_{fh}\mathbf{h}_{j-1} + \mathbf{b}_f\right)$
        $\tilde{\mathbf{c}}_j = \sigma_h\left(\mathbf{W}_{\tilde{c}x}\mathbf{x}_j + \mathbf{W}_{\tilde{c}h}\mathbf{h}_{j-1} + \mathbf{b}_{\tilde{c}}\right)$
        $\mathbf{c}_j = \mathbf{D}_i \odot \tilde{\mathbf{c}}_j + \mathbf{D}_f \odot \mathbf{c}_{j-1}$
        $\tilde{\mathbf{x}}_j = k_\phi(\mathbf{x}_j, \mathbf{c}_{j-1})$
    end
end
$\tilde{\mathbf{X}} = \left\{\{\tilde{\mathbf{x}}_{d,j}\}_{j=1}^{k}\right\}_{d=1}^{n}$
$L = \theta\left(\mathbf{X}, \tilde{\mathbf{X}}\right) = \left[\theta\left(\mathbf{X}_d, \tilde{\mathbf{X}}_d\right)\right]_{d=1}^{n}$
```

5.2.1 LSTM Autoencoder

Autoencoder is an unsupervised DL method which analyses the dynamic relationships between the features of high-dimensional data and produces a

Algorithm 7: Deep BLSTM algorithm

```
Input: $\tilde{\mathrm{X}}$
Target: $\mathbf{y}$
Initialization: $\mathbf{h}_0(j) = \mathbf{x}(j), \forall j \in [1, u]$
        $h_d(0) = x(d), \forall d \in [1, n]$
Output: $\tilde{\mathbf{y}}$
for $d = 1$ to $n$ do
    for $j = 1$ to $u$ do
        %% Forward direction
        $\overrightarrow{\mathbf{i}}_j = \sigma_r \left( \overrightarrow{\mathbf{W}}_{ix} \tilde{\mathbf{x}}_j + \overrightarrow{\mathbf{W}}_{ih} \overrightarrow{\mathbf{h}}_{1,j-1} + \overrightarrow{\mathbf{b}}_i \right)$
        $\overrightarrow{\mathbf{f}}_j = \sigma_r \left( \overrightarrow{\mathbf{W}}_{fx} \tilde{\mathbf{x}}_j + \overrightarrow{\mathbf{W}}_{fh} \overrightarrow{\mathbf{h}}_{1,j-1} + \overrightarrow{\mathbf{b}}_f \right)$
        $\overrightarrow{\tilde{\mathbf{c}}}_j = \sigma_h \left( \overrightarrow{\mathbf{W}}_{\tilde{c}x} \tilde{\mathbf{x}}_j + \overrightarrow{\mathbf{W}}_{\tilde{c}h} \overrightarrow{\mathbf{h}}_{1,j-1} + \overrightarrow{\mathbf{b}}_{\tilde{c}} \right)$
        $\overrightarrow{\mathbf{c}}_j = \overrightarrow{\mathbf{i}}_j \odot \overrightarrow{\tilde{\mathbf{c}}}_j + \overrightarrow{\mathbf{f}}_j \odot \overrightarrow{\mathbf{c}}_{j-1}$
        $\overrightarrow{\mathbf{o}}_j = \sigma_r \left( \overrightarrow{\mathbf{W}}_{ox} \tilde{\mathbf{x}}_j + \overrightarrow{\mathbf{W}}_{oh} \overrightarrow{\mathbf{h}}_{1,j-1} + \overrightarrow{\mathbf{b}}_o \right)$
        $\overrightarrow{\mathbf{h}}_{1,j} = \overrightarrow{\mathbf{o}}_j \odot \sigma_h \left( \overrightarrow{\mathbf{c}}_j \right)$
        $\overrightarrow{\mathbf{h}}_{2,j} = \sigma_h \left( \overrightarrow{\mathbf{W}}_{hx1} \tilde{\mathbf{x}}_j + \overrightarrow{\mathbf{W}}_{hh1} \overrightarrow{\mathbf{h}}_{1,j} + \overrightarrow{\mathbf{b}}_{h1} \right)$
        $\overrightarrow{\mathbf{h}}_{3,j} = \sigma_h \left( \overrightarrow{\mathbf{W}}_{hx2} \tilde{\mathbf{x}}_j + \overrightarrow{\mathbf{W}}_{hh2} \overrightarrow{\mathbf{h}}_{2,j} + \overrightarrow{\mathbf{b}}_{h2} \right)$
        $\overrightarrow{\mathbf{h}}_{4,j} = \sigma_h \left( \overrightarrow{\mathbf{W}}_{hx3} \tilde{\mathbf{x}}_j + \overrightarrow{\mathbf{W}}_{hh3} \overrightarrow{\mathbf{h}}_{3,j} + \overrightarrow{\mathbf{b}}_{h3} \right)$
    end
    %% Backward direction
    $\overleftarrow{\mathbf{i}}_j = \sigma_r \left( \overleftarrow{\mathbf{W}}_{ix} \tilde{\mathbf{x}}_j + \overleftarrow{\mathbf{W}}_{ih} \overleftarrow{\mathbf{h}}_{1,j-1} + \overleftarrow{\mathbf{b}}_i \right)$
    $\overleftarrow{\mathbf{f}}_j = \sigma_r \left( \overleftarrow{\mathbf{W}}_{fx} \tilde{\mathbf{x}}_j + \overleftarrow{\mathbf{W}}_{fh} \overleftarrow{\mathbf{h}}_{1,j-1} + \overleftarrow{\mathbf{b}}_f \right)$
    $\overleftarrow{\tilde{\mathbf{c}}}_j = \sigma_h \left( \overleftarrow{\mathbf{W}}_{\tilde{c}x} \tilde{\mathbf{x}}_j + \overleftarrow{\mathbf{W}}_{\tilde{c}h} \overleftarrow{\mathbf{h}}_{1,j-1} + \overleftarrow{\mathbf{b}}_{\tilde{c}} \right)$
    $\overleftarrow{\mathbf{c}}_j = \overleftarrow{\mathbf{i}}_j \odot \overleftarrow{\tilde{\mathbf{c}}}_j + \overleftarrow{\mathbf{f}}_j \odot \overleftarrow{\mathbf{c}}_{j-1}$
    $\overleftarrow{\mathbf{o}}_j = \sigma_r \left( \overleftarrow{\mathbf{W}}_{ox} \tilde{\mathbf{x}}_j + \overleftarrow{\mathbf{W}}_{oh} \overleftarrow{\mathbf{h}}_{1,j-1} + \overleftarrow{\mathbf{b}}_o \right)$
    $\overleftarrow{\mathbf{h}}_{1,j} = \overleftarrow{\mathbf{o}}_j \odot \sigma_h \left( \overleftarrow{\mathbf{c}}_j \right)$
    $\overleftarrow{\mathbf{h}}_{2,j} = \sigma_h \left( \overleftarrow{\mathbf{W}}_{hx1} \tilde{\mathbf{x}}_j + \overleftarrow{\mathbf{W}}_{hh1} \overleftarrow{\mathbf{h}}_{1,j} + \overleftarrow{\mathbf{b}}_{h1} \right)$
    $\overleftarrow{\mathbf{h}}_{3,j} = \sigma_h \left( \overleftarrow{\mathbf{W}}_{hx2} \tilde{\mathbf{x}}_j + \overleftarrow{\mathbf{W}}_{hh2} \overleftarrow{\mathbf{h}}_{2,j} + \overleftarrow{\mathbf{b}}_{h2} \right)$
    $\overleftarrow{\mathbf{h}}_{4,j} = \sigma_h \left( \overleftarrow{\mathbf{W}}_{hx3} \tilde{\mathbf{x}}_j + \overleftarrow{\mathbf{W}}_{hh3} \overleftarrow{\mathbf{h}}_{3,j} + \overleftarrow{\mathbf{b}}_{h3} \right)$
end
$\tilde{\mathbf{y}} = \sigma_y \left( \overrightarrow{\mathbf{W}}_y \overrightarrow{\mathbf{h}_4} + \overleftarrow{\mathbf{W}}_y \overleftarrow{\mathbf{h}_4} + \mathbf{b}_y \right)$ $\tilde{\mathbf{y}} = \{\tilde{\mathbf{y}}_d\}_{d=1}^{n}$
$\theta(\mathbf{y}, \tilde{\mathbf{y}}) = \left[ \theta(\mathbf{y}_d, \tilde{\mathbf{y}}_d) \right]_{d=1}^{n}$
```

low-dimensional latent-space representation. Unlike the conventional Autoencoder, LAE employs LSTM units to model the long-term inter-related changes in network traffic features. In this subsection, LAE method is developed to reduce feature dimensionality of big IoT network traffic data and obtain a low-dimensional latent-space feature representation with minimum reconstruction error.

A sequence of network traffic features is represented by a three-dimensional matrix, $\mathbf{X} \in \mathbb{R}^{n \times 1 \times k}$, in Equation 5.1:

$$\mathbf{X} = \left\{ \left\{ \mathbf{x}_{d,j} \right\}_{j=1}^{k} \right\}_{d=1}^{n}, \tag{5.1}$$

where $\{\mathbf{X}_d\}_{d=1}^{n} = \mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n$, $k \in \mathbb{Z}^+$ is the number of network traffic features, and $n \in \mathbb{Z}^+$ is the total instances of network traffic available in the dataset. An instance of network traffic is represented by Equation 5.2:

$$\{\mathbf{x}_j\}_{j=1}^{k} = \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k, \tag{5.2}$$

where $\mathbf{x}_j \in \mathbb{R}^n$ is a network traffic feature vector.

The encoder part of a single hidden layered LSTM autoencoder [198] was used to compress the network traffic feature matrix given by Equation 5.2 with the aim of reducing its dimensionality without loosing the information contained in the original data. LSTM has the capability to learn long time-dependencies with its feedback connections and a recurrent memory unit that is controlled by three gates, namely, input gate, a forget gate and an output gate [194]. LAE accepts high-dimensional network traffic feature set, $\mathbf{X}$, and produces a low-dimensional latent-space representation, $\tilde{\mathbf{X}}$, at the hidden layer. The input gate vector ($\mathbf{i}_j$), forget gate vector ($\mathbf{f}_j$), memory cell state vector ($\mathbf{c}_j$), output gate vector ($\mathbf{o}_j$) and hidden state vector ($\mathbf{h}_j$) were formed based on the LAE algorithm presented in Algorithm 6. The dimension of the column vectors is represented by $\mathbf{i}_j$, $\mathbf{f}_j$, $\mathbf{c}_j$, $\mathbf{o}_j$, $\mathbf{h}_j \in \mathbb{R}^u$, where u is the number of LSTM hidden units that represent the desired network traffic feature dimensionality.

Weight matrices and bias vectors were obtained by training the LAE using the Back Propagation Through Time (BPTT) algorithm [199]. The weight matrices for the connections between the input and the recurrent gates are given as $\mathbf{W}_{ix}, \mathbf{W}_{fx}, \mathbf{W}_{\tilde{c}x}, \mathbf{W}_{ox} \in \mathbb{R}^{u \times n}$, and these are the weight matrices of input-to-input gate connection, input-to-forget gate connection, input-to-cell connection and input-to-output gate connection respectively. Similarly, weight matrices for connections between the recurrent gates and the hidden state are given as $\mathbf{W}_{ih}, \mathbf{W}_{fh}, \mathbf{W}_{\tilde{c}h}, \mathbf{W}_{oh} \in \mathbb{R}^{u \times u}$, and these are the weight matrices of input gate-to-hidden state connection, forget gate-to-hidden state connection, cell state-to-hidden state connection and output gate-to-hidden state connection respectively. On the other hand, $\mathbf{b}_i, \mathbf{b}_f, \mathbf{b}_{\tilde{c}}, \mathbf{b}_o \in \mathbb{R}^u$ are the bias vectors of input gate, forget gate, cell state and output gate respectively. Recurrent activation function is a sigmoid function and it is represented by σ_r. Hidden layer activation function is represented by σ_h. Diagonal matrices, $\mathbf{D}_i$ and $\mathbf{D}_f$, are formed with input gate vector and forget gate vectors as represented by Equations 5.3 and 5.4 respectively:

$$\mathbf{D}_i = diag(\mathbf{i}) = \begin{bmatrix} i_1 & & \\ & \ddots & \\ & & i_h \end{bmatrix}, \tag{5.3}$$

$$\mathbf{D}_f = diag(\mathbf{f}) = \begin{bmatrix} f_1 & & \\ & \ddots & \\ & & f_h \end{bmatrix}. \tag{5.4}$$

For $\mathbf{x}_j$, the encoded output is given in Equation 5.5.

$$\tilde{\mathbf{x}}_j = k_{\phi}(\mathbf{x}_j, \mathbf{c}_{j-1}), \tag{5.5}$$

where $\mathbf{c}_{j-1}$ is the previous cell state vector and k_{ϕ} is the encoding function. Finally, the low-dimensional network traffic feature matrix is represented by Equation 5.6:

$$\tilde{\mathbf{X}} = \left\{ \left\{ \tilde{\mathbf{x}}_{d,j} \right\}_{j=1}^{u} \right\}_{d=1}^{n}. \tag{5.6}$$

5.2.2 Bidirectional LSTM

Conventional LSTM is unidirectional and it can only capture the dependence of the current state based on previous context [200]. Meanwhile, BLSTM has full access to both past and future sequential information using two LSTM hidden layers to scan input data sequence in positive and negative time directions respectively [201]. Therefore, a deep BLSTM method is developed to efficiently detect IoT botnet attack traffic by analysing the long-term inter-related changes in low-dimensional features produced by LAE.

Reduced network traffic feature set, $\tilde{\mathbf{X}}$, and its corresponding target vector, $\tilde{\mathbf{y}}$, were fed into a deep BLSTM model to produce input gate vectors ($\overrightarrow{\mathbf{i}}_j$, $\overleftarrow{\mathbf{i}}_j$), forget gate vectors ($\overrightarrow{\mathbf{f}}_j$, $\overleftarrow{\mathbf{f}}_j$), memory cell state vectors ($\overrightarrow{\mathbf{c}}_j$, $\overleftarrow{\mathbf{c}}_j$), output gate vectors ($\overrightarrow{\mathbf{o}}_j$, $\overleftarrow{\mathbf{o}}_j$) and hidden state vectors ($\overrightarrow{\mathbf{h}}_j$, $\overleftarrow{\mathbf{h}}_j$) based on the BLSTM algorithm provided in Algorithm 7. These column vectors are represented as $\overrightarrow{\mathbf{i}}_j$, $\overleftarrow{\mathbf{i}}_j$, $\overrightarrow{\mathbf{f}}_j$, $\overleftarrow{\mathbf{f}}_j$, $\overrightarrow{\mathbf{c}}_j$, $\overleftarrow{\mathbf{c}}_j$, $\overrightarrow{\mathbf{o}}_j$, $\overleftarrow{\mathbf{o}}_j$, $\overrightarrow{\mathbf{h}}_j$, $\overleftarrow{\mathbf{h}}_j$. Weight matrices ($\overrightarrow{\mathbf{W}}_{(\cdot)}$, $\overleftarrow{\mathbf{W}}_{(\cdot)}$) and bias vectors ($\overrightarrow{\mathbf{b}}_{(\cdot)}$, $\overleftarrow{\mathbf{b}}_{(\cdot)}$) were obtained by training the BLSTM using the BPTT algorithm. Recurrent activation function is a sigmoid function and it is represented by σ_r; hidden layer activation function is represented by σ_h; while output layer activation function is a softmax function and it is represented by σ_y. The parameters of the forward LSTM hidden layer are computed from the past to the present input data sequence while those of the backward LSTM hidden layer are calculated starting from the future to the present input data sequence. The LSTM hidden layers in the positive and negative time directions are jointly connected to the output layer. The difference between the predicted output of LAE-BLSTM, $\tilde{\mathbf{y}}$, and the target output, $\mathbf{y}$, is minimized in Equation 5.7.

$$\theta\left(\mathbf{y}, \tilde{\mathbf{y}}\right) = \left[\theta\left(\mathbf{y}_d, \tilde{\mathbf{y}}_d\right)\right]_{d=1}^{n}, \tag{5.7}$$

where θ is either a binary cross-entropy loss function for binary classification or a categorical cross-entropy loss function for multi-class classification scenarios.

5.3 Model Development and Experiment

In this section, the proposed hybrid DL method is implemented with the Bot-IoT and the N-BaIoT datasets. Figure 5.1 shows the framework for the development of the LAE-BLSTM models. The network traffic data is pre-processed as earlier described in Section 3.3. The SMOTE method is used to achieve class balance, as earlier described in Section 4.3, when the network traffic data is highly imbalanced.

The LAE models were developed to reduce the feature dimensionality of the network traffic data. The original feature dimensionality of the Bot-IoT and the N-BaIoT datasets are 37 and 115, respectively. Therefore, the number of hidden units at the input layer of the LAE model was 37 for BoT-IoT dataset and 115 for N-BaIoT dataset. The network traffic data is fed into the LSTM encoder, which transformed the data to produce a low-dimensional output

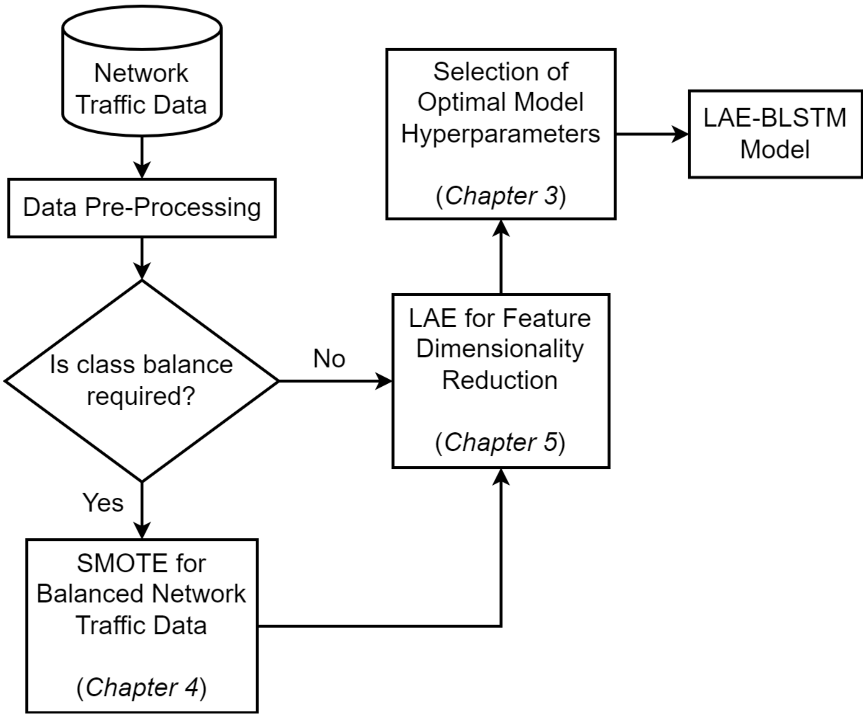

Network Traffic Data
Data Pre-Processing
Is class balance required?
No
Yes
SMOTE for Balanced Network Traffic Data
(Chapter 4)
LAE for Feature Dimensionality Reduction
(Chapter 5)
Selection of Optimal Model Hyperparameters
(Chapter 3)
LAE-BLSTM Model

FIGURE 5.1: Framework for the development of the LAE-BLSTM models

at the latent space. Then, the LSTM decoder reproduced the original network traffic data based on the low-dimensional output data from the LSTM encoder. ReLU activation functions were used for the LSTM encoder and decoder. The LAE models were trained using the following model hyperparameters: a learning rate of 0.001, a batch size of 512, and 10 epochs. Adam optimiser was used to reduce the MSE during model training and validation.

The LAE-BLSTM models were developed to classify the low-dimensional network traffic data. The optimal model hyperparameters that were presented in Table 3.15 were used for binary, 5-class, 10-class, and 11-class classification. The original feature dimensionality of the network traffic data was reduced to 2, 4, 6, 8, and 10. Experiments were performed to investigate the robustness of the models against under-fitting and over-fitting, their classification performance, as well as their computation efficiency.

5.4 Result Analysis and Discussion

In this section, the results of the experiments are analysed and discussed to evaluate the effectiveness of the LAE-BLSTM models in binary, 5-class, 10-class, and 11-class classification scenarios. The robustness of the models against under-fitting and over-fitting is evaluated based on the MSE during training and validation. The classification performance of the models is evaluated based on accuracy, precision, recall, and F1 score. The computation efficiency of the models is evaluated based on the encoding time, training time, and testing time.

5.4.1 Feature Dimensionality Reduction

5.4.1.1 The Original Bot-IoT Dataset

Figure 5.2 shows the training loss of the LAE model when it was trained with the original Bot-IoT dataset. The training loss reduced as the number of epochs increased from 1 to 10. Specifically, the training loss reduced by 13.59%, 64.97%, 12.34%, 98.58%, and 32.95% when LAE model reduced the feature dimensionality of the network traffic data from 37 to 2, 4, 6, 8, and 10, respectively. The LAE model achieved the lowest training loss of 7.36×10^{-5} when the feature dimensionality of the data was reduced from 37 to 8. Therefore, the LAE model did not under-fit the network traffic data in the original Bot-IoT dataset.

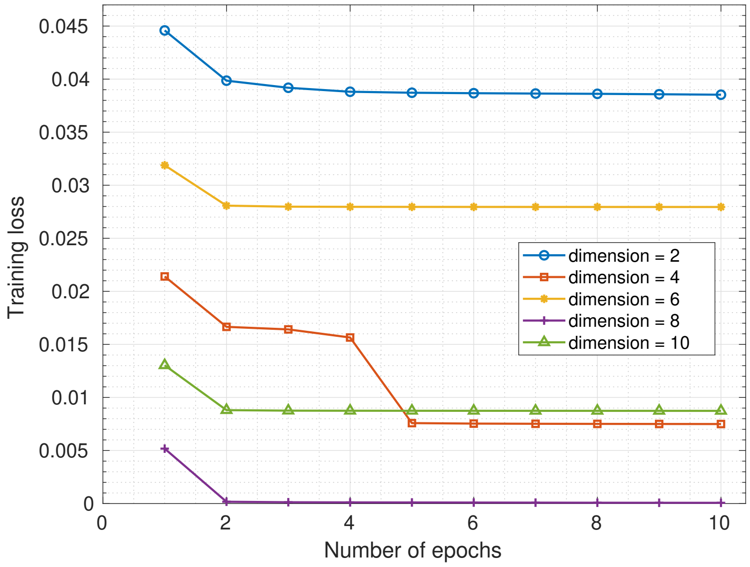


FIGURE 5.2: Training loss of the LAE model based on the original Bot-IoT dataset

Figure 5.3 shows the validation loss of the LAE model when it was trained with the original Bot-IoT dataset. The validation loss reduced as the number of epochs increased from 1 to 10. Specifically, the validation loss reduced by 4.22%, 55.76%, 1.30%, 66.44%, and 1.37% when LAE model reduced the feature dimensionality of the network traffic data from 37 to 2, 4, 6, 8, and 10, respectively. The LAE model achieved the lowest validation loss of 7.83×10^{-5} when the feature dimensionality of the data was reduced from 37 to 8. Therefore, the LAE model did not over-fit the network traffic data in the original Bot-IoT dataset. Figure 5.4 shows that the LAE model spent 146.03 ± 6.08 seconds to reduce the feature dimensionality of the network traffic data in the original Bot-IoT dataset.

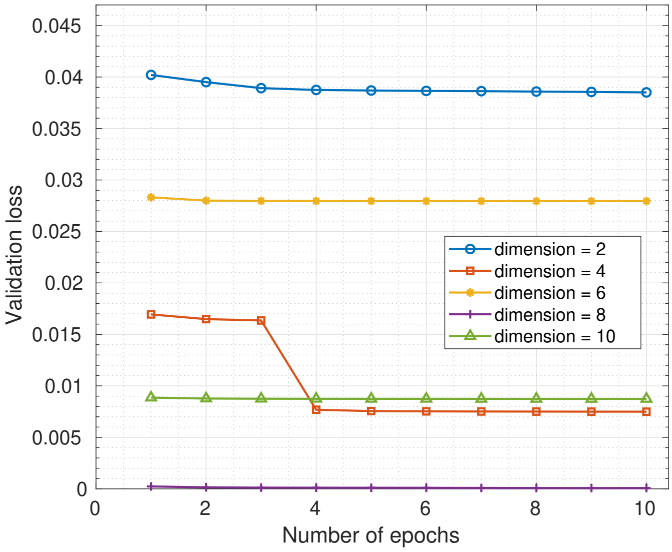


FIGURE 5.3: Validation loss of the LAE model based on the original Bot-IoT dataset

5.4.1.2 The Balanced Bot-IoT Dataset

Figure 5.5 shows the training loss of the LAE model when it was trained with the balanced Bot-IoT dataset. The training loss reduced as the number of epochs increased from 1 to 10. Specifically, the training loss reduced by 14%, 66.63%, 12.70%, 98.22%, and 31.79% when LAE model reduced the feature dimensionality of the network traffic data from 37 to 2, 4, 6, 8, and 10, respectively. The LAE model achieved the lowest training loss of 8.85×10^{-5} when the feature dimensionality of the data was reduced from 37 to 8. Therefore, the LAE model did not under-fit the network traffic data in the balanced Bot-IoT dataset.

Figure 5.6 shows the validation loss of the LAE model when it was trained with the balanced Bot-IoT dataset. The validation loss reduced as the number of epochs increased from 1 to 10. Specifically, the validation loss

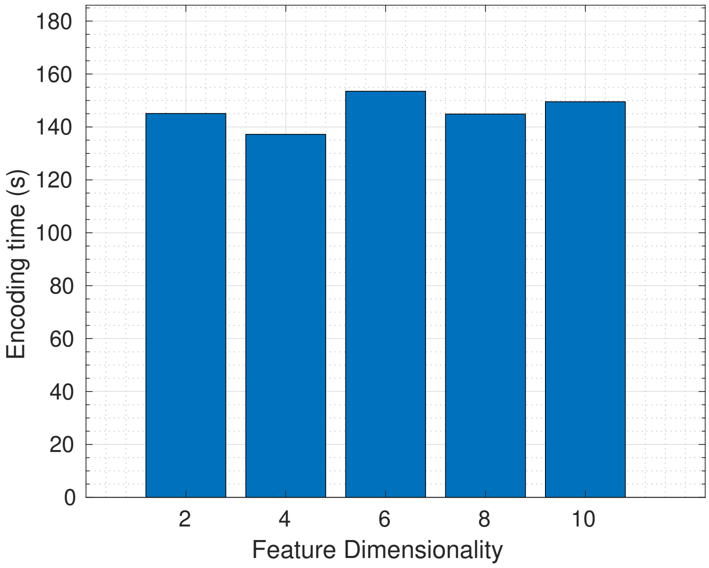


FIGURE 5.4: Encoding time of the LAE model based on the original Bot-IoT dataset

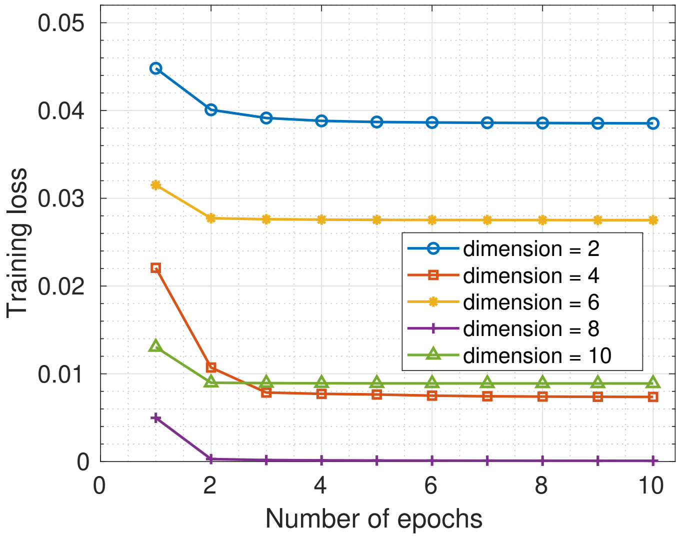


FIGURE 5.5: Training loss of the LAE model based on the balanced Bot-IoT dataset

reduced by 4.72%, 58.42%, 1.19%, 80.14%, and 1.27% when LAE model reduced the feature dimensionality of the network traffic data from 37 to 2, 4, 6, 8, and 10, respectively. The LAE model achieved the lowest validation loss of 6.07×10^{-5} when the feature dimensionality of the data was reduced from 37 to 8. Therefore, the LAE model did not over-fit the network traffic data in the balanced Bot-IoT dataset. Figure 5.7 shows that the LAE model spent 158.65 ± 8.22 seconds to reduce the feature dimensionality of the network traffic data in the balanced Bot-IoT dataset.

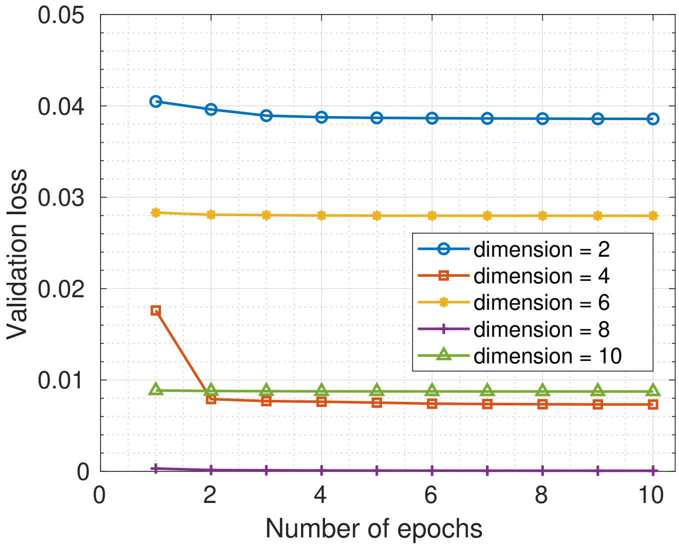


FIGURE 5.6: Validation loss of the LAE model based on the balanced Bot-IoT dataset

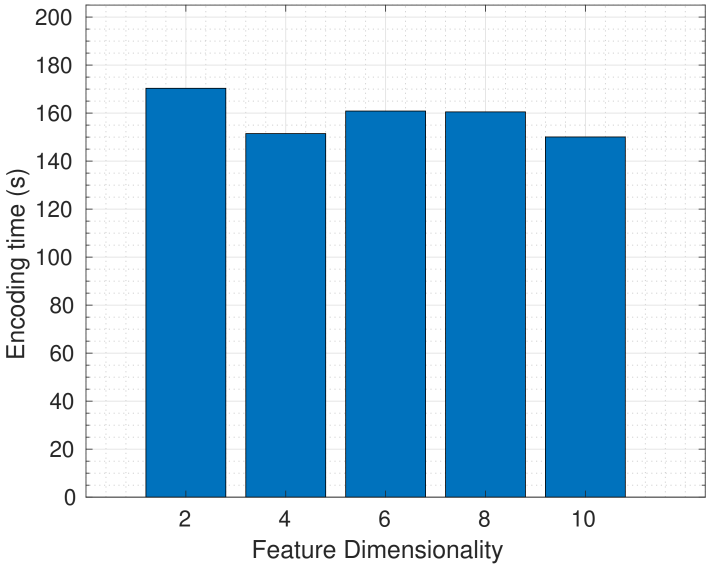


FIGURE 5.7: Encoding time of the LAE model based on the balanced Bot-IoT dataset

5.4.1.3 The N-BaIoT Dataset

Figure 5.8 shows the training loss of the LAE model when it was trained with the N-BaIoT dataset. The training loss reduced as the number of epochs increased from 1 to 10. Specifically, the training loss reduced by 15.89%, 19.65%, 6.79%, 37.74%, and 13.49% when LAE model reduced the feature dimensionality of the network traffic data from 115 to 2, 4, 6, 8, and 10, respectively. The LAE model achieved the lowest training loss of

7.52×10^{-3} when the feature dimensionality of the data was reduced from 37 to 10. Therefore, the LAE model did not under-fit the network traffic data in the N-BaIoT dataset.

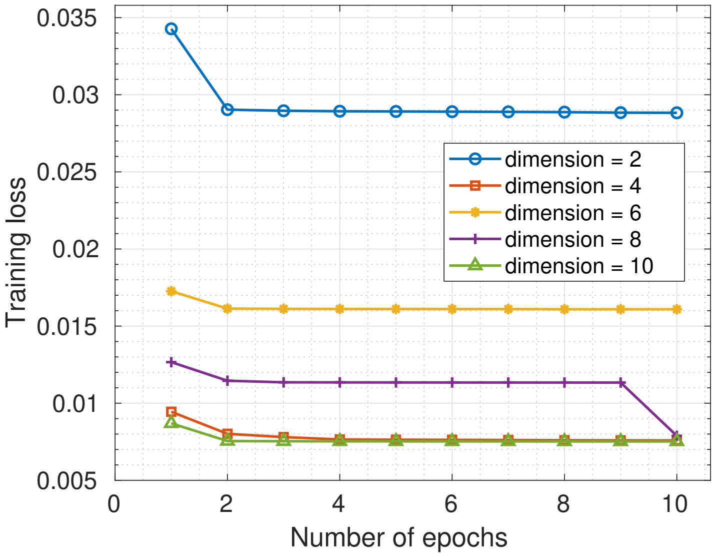


FIGURE 5.8: Training loss of the LAE model based on the N-BaIoT dataset

Figure 5.9 shows the validation loss of the LAE model when it was trained with the N-BaIoT dataset. The validation loss reduced as the number of epochs increased from 1 to 10. Specifically, the validation loss reduced by 0.98%, 6.08%, 0.40%, 32.07%, and 0.64% when LAE model reduced the feature dimensionality of the network traffic data from 115 to 2, 4, 6, 8, and 10, respectively. The LAE model achieved the lowest validation loss of 7.53×10^{-3} when the feature dimensionality of the data was reduced from 37 to 10. Therefore, the LAE model did not over-fit the network traffic data in the N-BaIoT dataset. Figure 5.10 shows that the LAE model spent 325.69 ± 5.99 seconds to reduce the feature dimensionality of the network traffic data in the N-BaIoT dataset.

5.4.2 Classification Performance of the LAE-BLSTM Models

5.4.2.1 Binary Classification: The Original Bot-IoT Dataset

Figure 5.11 shows the training loss of the LAE-BLSTM model when it was trained with the low-dimensional network traffic feature sets of the Bot-IoT dataset for binary classification. The training loss reduced as the number of epochs increased from 1 to 5. Specifically, the training loss reduced by

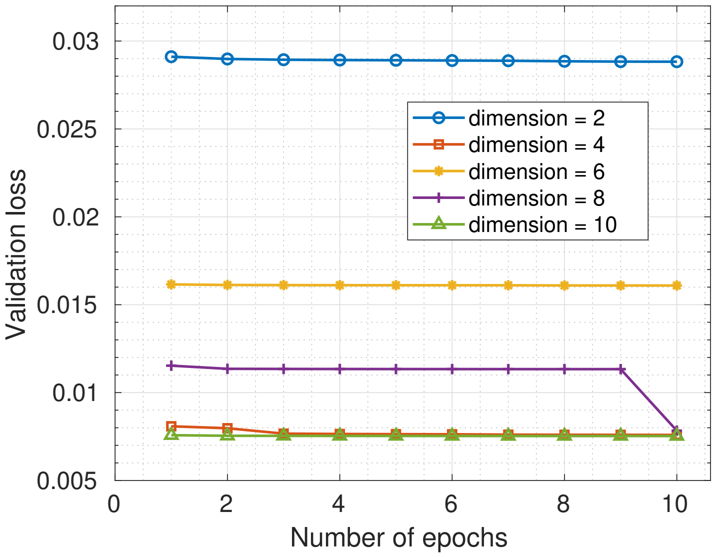


FIGURE 5.9: Validation loss of the LAE model based on the N-BaIoT dataset

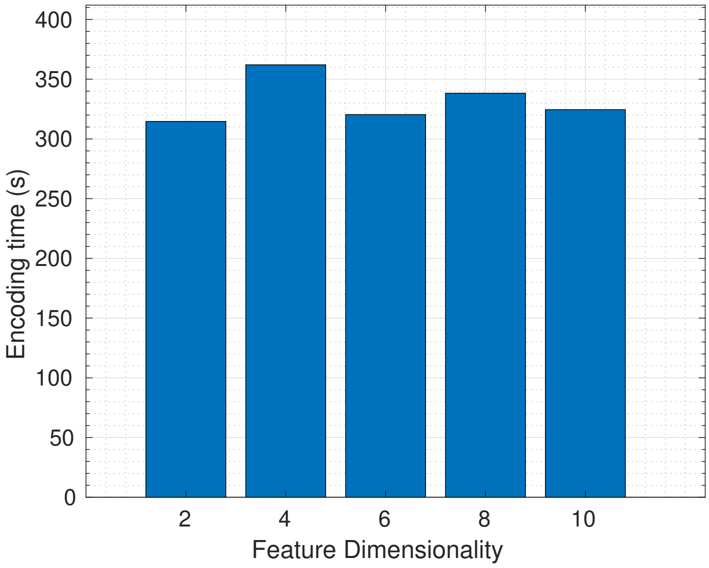


FIGURE 5.10: Encoding time of the LAE model based on the N-BaIoT dataset

89.72%, 97.69%, 98.50%, 98.71%, and 96.93% when LAE-BLSTM model reduced the feature dimensionality of the network traffic data from 37 to 2, 4, 6, 8, and 10, respectively. The LAE-BLSTM model achieved the lowest training loss of 2.21×10^{-5} when the feature dimensionality of the data was reduced from 37 to 8. Therefore, the LAE-BLSTM model did not under-fit the low-dimensional network traffic data in the binary Bot-IoT dataset.

Figure 5.12 shows the validation loss of the LAE-BLSTM model when it was

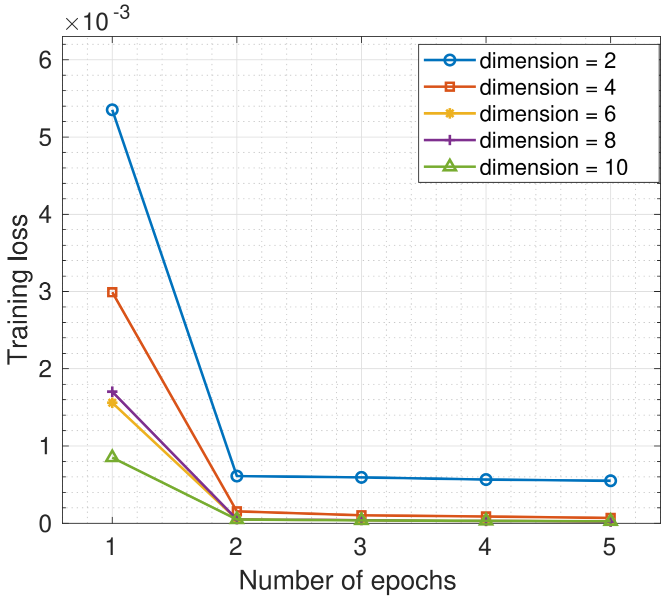


FIGURE 5.11: Training loss of the binary LAE-BLSTM model based on the original Bot-IoT dataset

trained with the low-dimensional network traffic feature sets of the Bot-IoT dataset for binary classification. The validation loss reduced as the number of epochs increased from 1 to 5. Specifically, the validation loss reduced by 13.70%, 67.88%, 53.87%, 33.24%, and 11.02% when LAE-BLSTM model reduced the feature dimensionality of the network traffic data from 37 to 2, 4, 6, 8, and 10, respectively. The LAE-BLSTM model achieved the lowest validation loss of 5.26×10^{-5} when the feature dimensionality of the data was reduced from 37 to 6. Therefore, the LAE-BLSTM model did not over-fit the low-dimensional network traffic data in the binary Bot-IoT dataset.

Table 5.1 presents the performance of the binary LAE-BLSTM model based on the original Bot-IoT dataset. The BLSTM model, which was developed with the 37-dimensional network traffic data, achieved the best classification performance with 100% accuracy, 99.03% precision, 100% recall, and 99.51% F1 score. However, large memory spaces of 651 MB, 217 MB, and 217 MB are required to store the data on a central server or IoT edge node for model training, validation, and testing, respectively. On the other hand, the LAE-BLSTM model, which was developed with 8-dimensional network traffic data, reduced the memory space requirements by 89.19%, without a significant decrease in the classification performance. The model achieved 100% accuracy, 98.48% precision, 97.52% recall, and 98% F1 score. It took 59.01 ± 2.72 seconds to train the LAE-BLSTM model, and the model spent 990 ± 28 milliseconds to classify the network traffic data in the testing set.

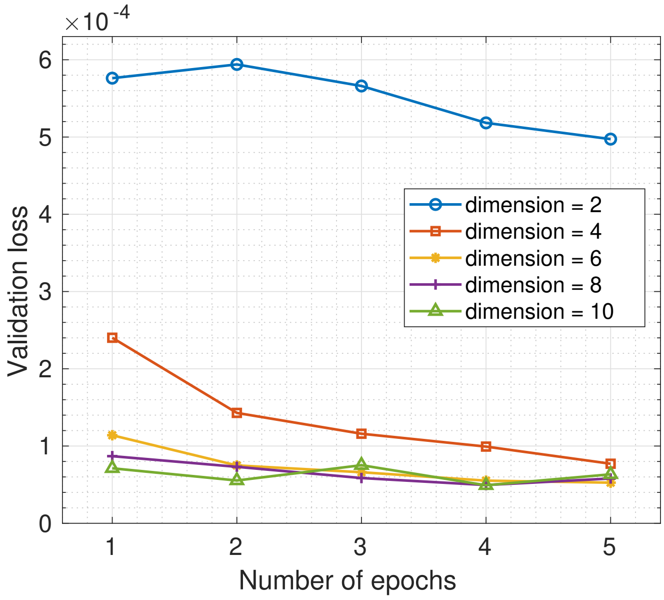


FIGURE 5.12: Validation loss of the binary LAE-BLSTM model based on the original Bot-IoT dataset

TABLE 5.1: Performance of the binary LAE-BLSTM model based on the original Bot-IoT dataset

Metrics		Feature Dimensionality					
		2	4	6	8	10	37
Performance (%)	Accuracy	99.99	100.00	100.00	100.00	100.00	100.00
	Precision	84.44	97.47	96.30	98.48	96.00	99.03
	Recall	65.35	96.53	99.50	97.52	95.54	100.00
	F1 Score	71.23	97.00	97.85	98.00	95.77	99.51
Data Size (MB)	Training	17.61	35.22	52.83	70.43	88.04	651.52
	Validation	5.87	11.74	17.61	23.48	29.35	217.18
	Testing	5.87	11.74	17.61	23.48	29.35	217.18
Time (s)	Training	58.08	57.39	57.53	63.84	58.20	60.89
	Testing	1.01	1.01	0.95	1.00	0.96	1.09

5.4.2.2 Binary Classification: The N-BaIoT Dataset

Figure 5.13 shows the training loss of the LAE-BLSTM model when it was trained with the low-dimensional network traffic feature sets of the N-BaIoT dataset for binary classification. The training loss reduced as the number of epochs increased from 1 to 5. Specifically, the training loss reduced by 55.55%, 80.15%, 78.52%, 80.57%, and 76.78% when LAE-BLSTM model reduced the feature dimensionality of the network traffic data from 115 to 2, 4, 6, 8, and 10, respectively. The LAE-BLSTM model achieved the lowest training loss of 9.74×10^{-4} when the feature dimensionality of the data was reduced from 115 to 4. Therefore, the LAE-BLSTM model did not under-fit the low-dimensional network traffic data in the binary N-BaIoT dataset.

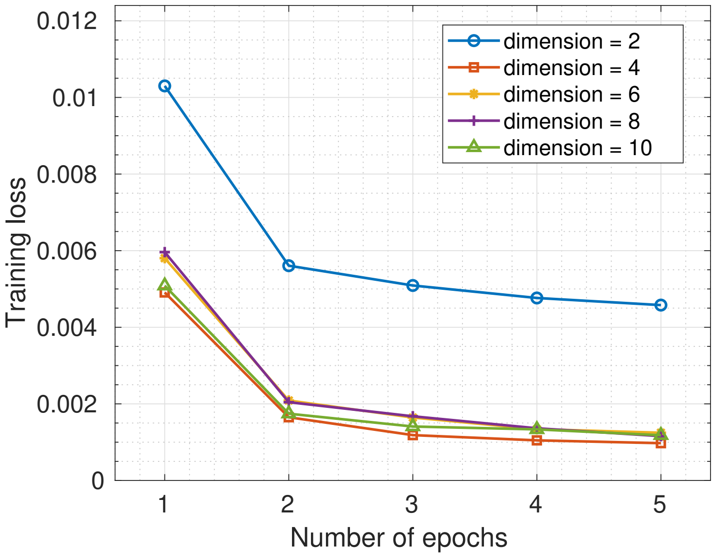


FIGURE 5.13: Training loss of the binary LAE-BLSTM model based on the N-BaIoT dataset

Figure 5.14 shows the validation loss of the LAE-BLSTM model when it was trained with the low-dimensional network traffic feature sets of the N-BaIoT dataset for binary classification. The validation loss reduced as the number of epochs increased from 1 to 5. Specifically, the validation loss reduced by 23.20%, 62.49%, 61.27%, 59.61%, and 61.95% when LAE-BLSTM model reduced the feature dimensionality of the network traffic data from 115 to 2, 4, 6, 8, and 10, respectively. The LAE-BLSTM model achieved the lowest validation loss of 8.47×10^{-4} when the feature dimensionality of the data was reduced from 115 to 4. Therefore, the LAE-BLSTM model did not over-fit the low-dimensional network traffic data in the binary N-BaIoT dataset.

Table 5.2 presents the classification performance of the binary LAE-BLSTM model based on the N-BaIoT dataset. The BLSTM model, which was developed with the 115-dimensional network traffic data, achieved the best classification performance with 99.99% accuracy, 99.97% precision, 99.96% recall, and 99.96% F1 score. However, large memory spaces of 3.42 GB, 1.14 GB, and 1.14 GB are required to store the data on a central server or IoT edge node for model training, validation, and testing, respectively. On the other hand, the LAE-BLSTM model, which was developed with 4-dimensional network traffic data, reduced the memory space requirements by 98.26%, without a significant decrease in the classification

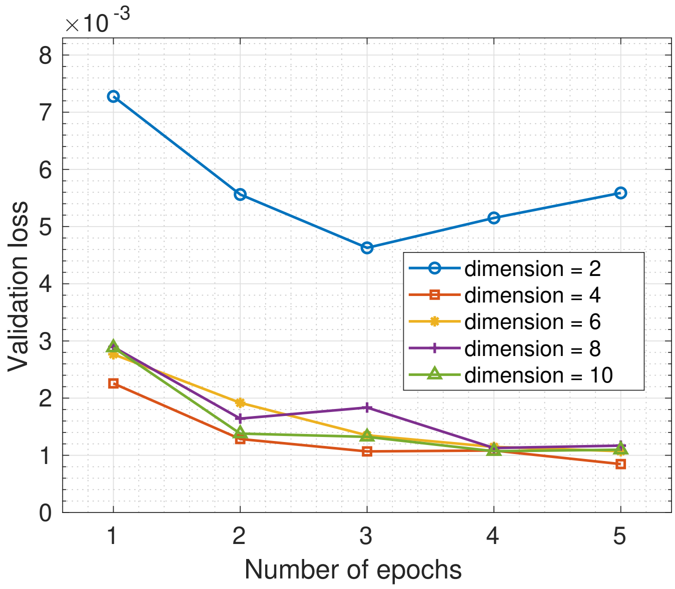


FIGURE 5.14: Validation loss of the binary LAE-BLSTM model based on the N-BaIoT dataset

performance. The model achieved 99.97% accuracy, 99.91% precision, 99.93% recall, and 99.92% F1 score. It took 262 ± 22.78 seconds to train the LAE-BLSTM model, and the model spent 911 ± 9 milliseconds to classify the network traffic data in the testing set.

TABLE 5.2: Performance of the binary LAE-BLSTM model based on the N-BaIoT dataset

Metrics		Feature Dimensionality					
		2	4	6	8	10	115
Performance (%)	Accuracy	99.85	99.97	99.97	99.97	99.97	99.99
	Precision	99.36	99.91	99.92	99.94	99.95	99.97
	Recall	99.74	99.93	99.89	99.87	99.89	99.96
	F1 Score	99.55	99.92	99.91	99.91	99.92	99.96
Data Size (MB)	Training	29.77	59.55	89.32	119.09	148.87	3423.92
	Validation	9.92	19.85	29.77	39.70	49.62	1141.31
	Testing	9.92	19.85	29.77	39.70	49.62	1141.31
Time (s)	Training	297.50	265.7	235.75	260.64	251.49	253.37
	Testing	0.91	0.90	0.92	0.91	0.92	6.97

5.4.2.3 5-Class Classification: The Original Bot-IoT Dataset

Figure 5.15 shows the training loss of the LAE-BLSTM model when it was trained with the low-dimensional network traffic feature sets of the Bot-IoT dataset for 5-class classification. The training loss reduced as the number of epochs increased from 1 to 10. Specifically, the training loss reduced by

26.40%, 85.43%, 68.68%, 91.12%, and 88.82% when LAE-BLSTM model reduced the feature dimensionality of the network traffic data from 37 to 2, 4, 6, 8, and 10, respectively. The LAE-BLSTM model achieved the lowest training loss of 1.09×10^{-2} when the feature dimensionality of the data was reduced from 37 to 8. Therefore, the LAE-BLSTM model did not under-fit the low-dimensional network traffic data in the 5-class Bot-IoT dataset.

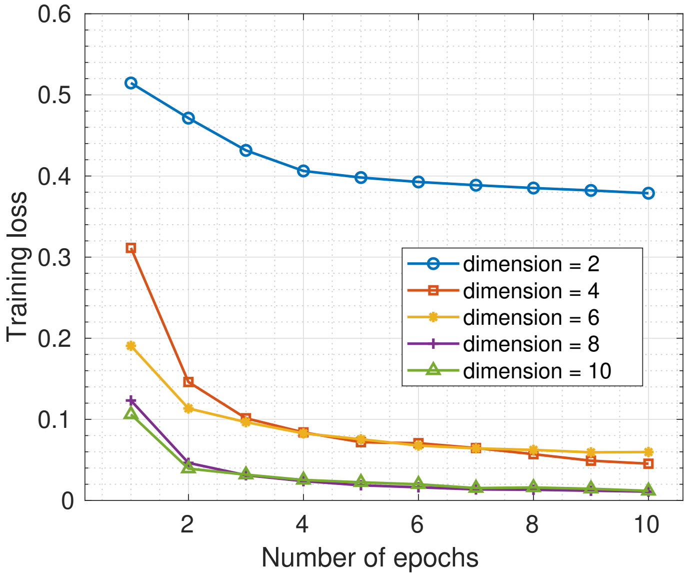


FIGURE 5.15: Training loss of the 5-class LAE-BLSTM model based on the original Bot-IoT dataset

Figure 5.16 shows the validation loss of the LAE-BLSTM model when it was trained with the low-dimensional network traffic feature sets of the Bot-IoT dataset for 5-class classification. The validation loss reduced as the number of epochs increased from 1 to 10. Specifically, the validation loss reduced by 19.40%, 77.14%, 46.83%, 50.38%, and 78.21% when LAE-BLSTM model reduced the feature dimensionality of the network traffic data from 37 to 2, 4, 6, 8, and 10, respectively. The LAE-BLSTM model achieved the lowest validation loss of 1.67×10^{-2} when the feature dimensionality of the data was reduced from 37 to 10. Therefore, the LAE-BLSTM model did not over-fit the low-dimensional network traffic data in the 5-class Bot-IoT dataset.

Table 5.3 presents the classification performance of the 5-class LAE-BLSTM model based on the original Bot-IoT dataset. The BLSTM model, which was developed with the 37-dimensional network traffic data, achieved the best classification performance with 99.99% accuracy, 99.58% precision, 99% recall, and 99.29% F1 score. However, large memory spaces of 651.52 MB, 217.18 MB, and 217.18 MB are required to store the data on a central server

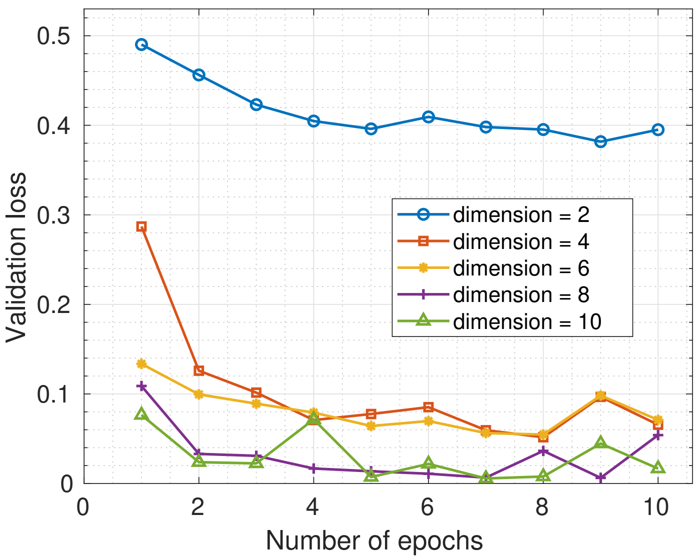


FIGURE 5.16: Validation loss of the 5-class LAE-BLSTM model based on the original Bot-IoT dataset

or IoT edge node for model training, validation, and testing, respectively. On the other hand, the LAE-BLSTM model, which was developed with 10-dimensional network traffic data, reduced the memory space requirements by 86.45%, without a significant decrease in the classification performance. The model achieved 99.71% accuracy, 99.71% precision, 97.69% recall, and 98.65% F1 score. It took 131.72 ± 0.57 seconds to train the LAE-BLSTM model, and the model spent 1.14 ± 0.01 seconds to classify the network traffic data in the testing set.

TABLE 5.3: Performance of the 5-class LAE-BLSTM model based on the original Bot-IoT dataset

Metrics		Feature Dimensionality					
		2	4	6	8	10	37
Performance (%)	Accuracy	91.99	99.05	98.80	99.40	99.71	99.99
	Precision	88.92	99.07	98.41	98.13	99.71	99.58
	Recall	60.22	92.75	97.12	97.67	97.69	99.00
	F1 Score	65.71	95.57	97.74	97.83	98.65	99.29
Data Size (MB)	Training	17.61	35.22	52.83	70.43	88.04	651.52
	Validation	5.87	11.74	17.61	23.48	29.35	217.18
	Testing	5.87	11.74	17.61	23.48	29.35	217.18
Time (s)	Training	131.81	131.44	131.43	131.26	132.68	123.41
	Testing	1.16	1.15	1.14	1.12	1.14	1.23

5.4.2.4 10-Class Classification: The N-BaIoT Dataset

Figure 5.17 shows the training loss of the LAE-BLSTM model when it was trained with the low-dimensional network traffic feature sets of the N-BaIoT dataset for 10-class classification. The training loss reduced as the number of epochs increased from 1 to 15. Specifically, the training loss reduced by 24.49%, 57.65%, 80.13%, 89.98%, and 91.59% when LAE-BLSTM model reduced the feature dimensionality of the network traffic data from 115 to 2, 4, 6, 8, and 10, respectively. The LAE-BLSTM model achieved the lowest training loss of 2.53×10^{-2} when the feature dimensionality of the data was reduced from 115 to 10. Therefore, the LAE-BLSTM model did not under-fit the low-dimensional network traffic data in the 10-class N-BaIoT dataset.

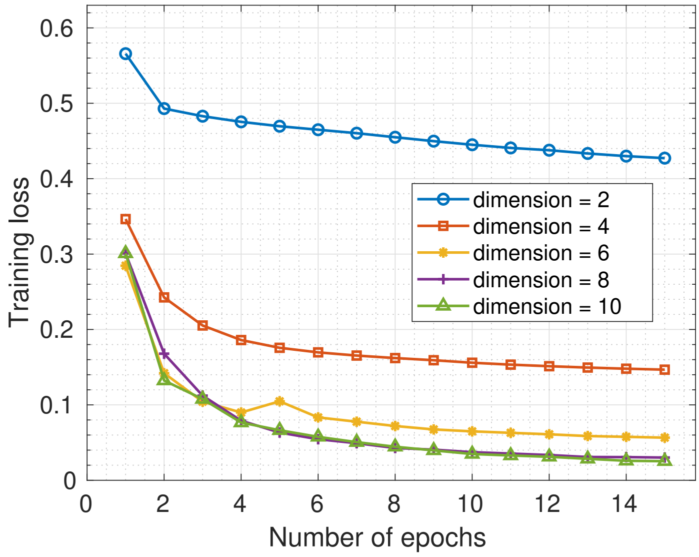


FIGURE 5.17: Training loss of the 10-class LAE-LSTM model based on the N-BaIoT dataset

Figure 5.18 shows the validation loss of the LAE-BLSTM model when it was trained with the low-dimensional network traffic feature sets of the Bot-IoT dataset for 10-class classification. The validation loss reduced as the number of epochs increased from 1 to 15. Specifically, the validation loss reduced by 15.24%, 46.70%, 69.10%, 88.10%, and 89.40% when LAE-BLSTM model reduced the feature dimensionality of the network traffic data from 115 to 2, 4, 6, 8, and 10, respectively. The LAE-BLSTM model achieved the lowest validation loss of 1.79×10^{-2} when the feature dimensionality of the data was reduced from 115 to 10. Therefore, the LAE-BLSTM model did not over-fit the low-dimensional network traffic data in the 10-class N-BaIoT dataset.

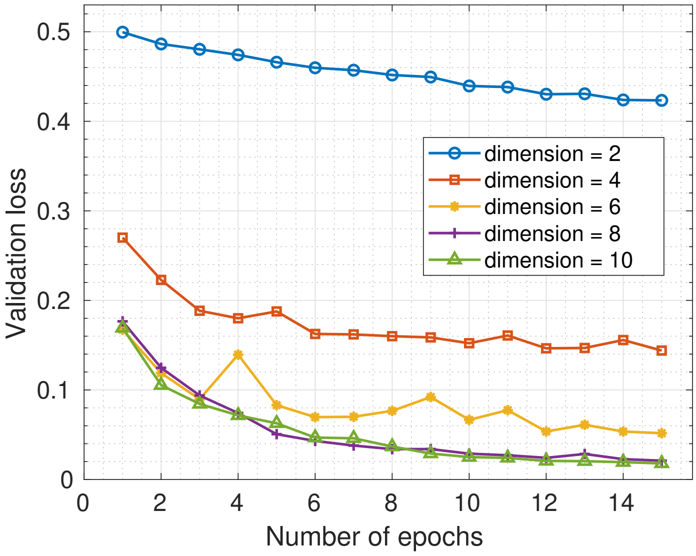


FIGURE 5.18: Validation loss of the 10-class LAE-LSTM model based on the N-BaIoT dataset

Table 5.4 presents the classification performance of the binary LAE-BLSTM model based on the N-BaIoT dataset. The BLSTM model, which was developed with the 115-dimensional network traffic data, achieved the best classification performance with 100% accuracy, 99.96% precision, 99.97% recall, and 99.97% F1 score. However, large memory spaces of 3.42 GB, 1.14 GB, and 1.14 GB are required to store the data on a central server or IoT edge node for model training, validation, and testing, respectively. On the other hand, the LAE-BLSTM model, which was developed with 8-dimensional network traffic data, reduced the memory space requirements by 96.52%, without a significant decrease in the classification performance. The model achieved 99.90% accuracy, 99.50% precision, 99.45% recall, and 99.47% F1 score. It took 926.57 ± 57.94 seconds to train the LAE-BLSTM model, and the model spent 900 ± 11 milliseconds to classify the network traffic data in the testing set.

5.4.2.5 11-Class Classification: The Original Bot-IoT Dataset

Figure 5.19 shows the training loss of the LAE-BLSTM model when it was trained with the low-dimensional network traffic feature sets of the Bot-IoT dataset for 11-class classification. The training loss reduced as the number of epochs increased from 1 to 15. Specifically, the training loss reduced by 35.92%, 84.02%, 88.68%, 94.02%, and 91.63% when LAE-BLSTM model reduced the feature dimensionality of the network traffic data from 37 to 2,

TABLE 5.4: Performance of the 10-class LAE-BLSTM model based on the N-BaIoT dataset

Metrics		Feature Dimensionality					
		2	4	6	8	10	115
Performance (%)	Accuracy	96.30	98.56	99.62	99.90	99.90	100.00
	Precision	81.94	94.02	98.34	99.50	99.47	99.96
	Recall	80.96	94.27	98.14	99.45	99.15	99.97
	F1 Score	80.87	94.13	98.23	99.47	99.30	99.97
Data Size (MB)	Training	29.77	59.55	89.32	119.09	148.87	3423.92
	Validation	9.92	19.85	29.77	39.70	49.62	1141.31
	Testing	9.92	19.85	29.77	39.70	49.62	1141.31
Time (s)	Training	1007.39	869.59	945.42	870.72	939.75	786.00
	Testing	0.89	0.92	0.90	0.89	0.89	6.78

4, 6, 8, and 10, respectively. The LAE-BLSTM model achieved the lowest training loss of 7.81 × 10^{-3} when the feature dimensionality of the data was reduced from 37 to 8. Therefore, the LAE-BLSTM model did not under-fit the low-dimensional network traffic data in the 11-class Bot-IoT dataset.

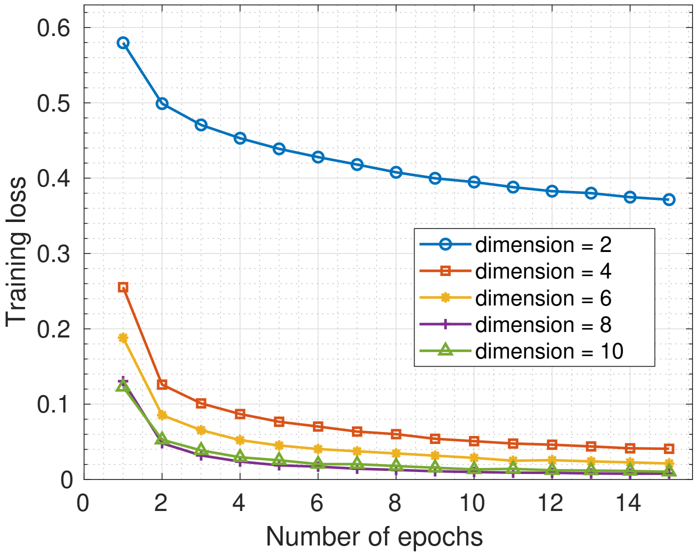


FIGURE 5.19: Training loss of the 11-class LAE-BLSTM model based on the balanced Bot-IoT dataset

Figure 5.20 shows the validation loss of the LAE-BLSTM model when it was trained with the low-dimensional network traffic feature sets of the Bot-IoT dataset for 11-class classification. The validation loss reduced as the number of epochs increased from 1 to 15. Specifically, the validation loss reduced by 28.15%, 78.76%, 83.29%, 92.44%, and 87.29% when LAE-BLSTM model reduced the feature dimensionality of the network traffic data from 37 to 2, 4, 6, 8, and 10, respectively. The LAE-BLSTM model achieved the lowest validation loss of 3.79×10^{-3} when the feature dimensionality of the data was reduced from 37 to 8. Therefore, the LAE-BLSTM model did not over-fit the low-dimensional network traffic data in the 11-class Bot-IoT dataset.

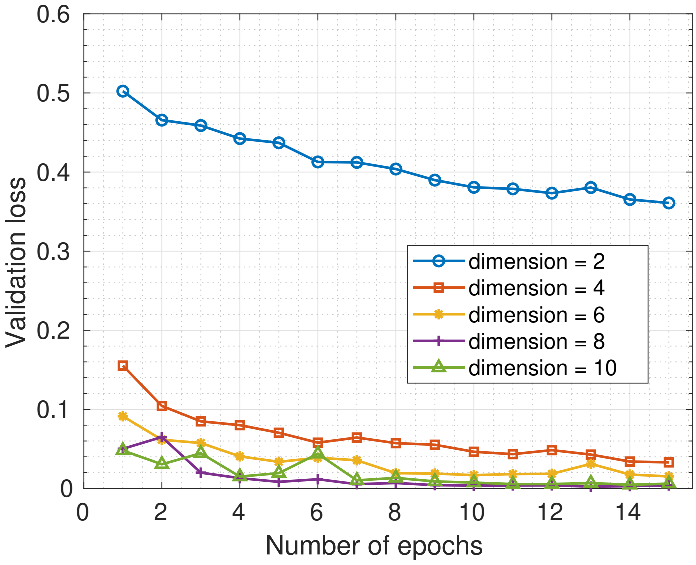


FIGURE 5.20: Validation loss of the 11-class LAE-BLSTM model based on the balanced Bot-IoT dataset

Table 5.5 presents the classification performance of the 11-class LAE-BLSTM model based on the balanced Bot-IoT dataset. The BLSTM model, which was developed with the 37-dimensional network traffic data, achieved the best classification performance with 100% accuracy, 99.32% precision, 99.92% recall, and 99.61% F1 score. However, large memory spaces of 666.96 MB, 217.18 MB, and 217.18 MB are required to store the data on a central server or IoT edge node for model training, validation, and testing, respectively. On the other hand, the LAE-BLSTM model, which was developed with 8-dimensional network traffic data, reduced the memory space requirements by 89.19%, without a significant decrease in the classification performance. The model achieved 99.98% accuracy, 99.03% precision, 99.53% recall, and 99.25% F1 score. It took 212.99 ± 0.66 seconds to train the LAE-BLSTM model, and the model spent 1.19 ± 0.01 seconds to classify the network traffic data in the testing set.

TABLE 5.5: Performance of the 11-class LAE-BLSTM model based on the original Bot-IoT dataset

Metrics		Feature Dimensionality					
		2	4	6	8	10	37
Performance (%)	Accuracy	97.02	99.80	99.91	99.98	99.96	100.00
	Precision	71.40	94.31	97.04	99.03	97.80	99.32
	Recall	83.97	99.04	98.52	99.53	99.44	99.92
	F1 Score	74.93	96.23	97.72	99.25	98.55	99.61
Data Size (MB)	Training	18.03	36.05	54.08	72.10	90.13	666.96
	Validation	5.87	11.74	17.61	23.48	29.35	217.18
	Testing	5.87	11.74	17.61	23.48	29.35	217.18
Time (s)	Training	1007.39	869.59	945.42	870.72	939.75	786.00
	Testing	0.89	0.92	0.90	0.89	0.89	6.78

5.5 Chapter Summary

In this chapter, a hybrid DL method, named LAE-BLSTM, is proposed to reduce the memory space requirements for DL-based botnet attack detection in IoT-enabled critical infrastructure, without a significant decrease in the classification performance. LAE-BLSTM models were developed with the Bot-IoT and N-BaIoT to perform binary and multi-class classification. Experiment results showed that the models were robust against under-fitting and over-fitting during training and validation. Also, the models reduced the sizes of memory space required to store network traffic data on the central server or IoT edge nodes for training, validation, and testing by $86.45 - 98.26\%$. Furthermore, the models achieved high classification performance with 99.91 ± 0.12 accuracy, 99.33 ± 0.57 precision, 98.82 ± 1.13 recall, and 99.06 ± 0.75 F1 score. Therefore, the LAE-BLSTM method can be used for memory-efficient botnet attack detection in IoT-enabled critical infrastructure.

Chapter 6

Federated Deep Learning for Zero-Day Botnet Attack Detection

6.1 Introduction

In Chapter 5, a hybrid DL method was proposed for memory-efficient botnet attack detection in IoT-enabled critical infrastructure. This method reduces the size of memory space required to store the network traffic data in a central server or IoT edge nodes for model training, validation, and testing. However, IoT-enabled critical infrastructure networks are fast becoming highly scalable. Due to network constraints, it may be difficult to offload massive distributed network traffic data to a remote central cloud server for data processing in real-life botnet attack detection scenarios. Also, the CDL method takes longer time to train, it has high communication overhead, and its memory space requirement for data storage is high. Cloud data centers are often located far away from where the IoT edge nodes are deployed. This causes high latency in the CDL-based botnet attack detection method. Furthermore, the CDL method does not guarantee users' privacy and security because it involves the transmission of network traffic features from all participating IoT devices to a central cloud server.

Edge computing can be combined with DL to bring intelligence closer to where data are being generated, thereby addressing the issues of data privacy, high communication cost, large memory space requirement, short training time, and high latency [202]. In IoT-enabled critical infrastructure, the network traffic data that are generated within the same local network are stored in an IoT edge node. However, IoT edge nodes have limited memory space for network traffic data storage and low computation resources for big data analytics. LDL method achieves edge intelligence without data aggregation [203]. However, the classification performance of LDL-based botnet attack detection method is very likely to be low because a single IoT edge node has limited network traffic samples. On the other hand, in zero-day botnet attacks, hackers use a network of compromised computing devices to exploit previously unknown vulnerabilities in IoT systems. Detecting zero-day cyber attacks is a very difficult task since there is no prior knowledge of such incidence [204]. Therefore, an efficient botnet attack detection method must be able to detect zero-day botnet attacks at edge nodes with high detection rate and low false alarm rate.

FL is a collaborative method for privacy-preserving DL based on the private data in distributed multiple devices [46]. This method enables collaborative DL in distributed IoT devices without sharing private network traffic data with the central cloud server. In this chapter, a FDL method is proposed for zero-day botnet attack detection in IoT edge nodes of critical infrastructure networks. The LAE-BLSTM method in Chapter 5 is used to develop local botnet attack detection models at the edge nodes. The local LAE-BLSTM models are trained independently in multiple IoT edge devices, while the FedAvg algorithm is used to aggregate local model updates. A global LAE-BLSTM model is produced based on the aggregation of local model updates in the central cloud server, and the model parameters are transmitted to all the IoT edge nodes. FDL models are developed with the Bot-IoT and N-BaIoT dataset to evaluate the effectiveness of the proposed method. The performance of the FDL models is compared with that of the CDL and LDL models.

The remaining parts of this paper are organised as follows: in Section 6.2, different zero-day botnet attack scenarios in IoT-enabled critical infrastructure are presented; in Section 6.3, the FDL method is proposed for zero-day botnet attack detection in IoT edge nodes; in Section 6.4, the effectiveness of the FDL method is evaluated and compared with that of the CDL and LDL methods; and the major findings of the chapter is summarised in Section 6.5.

6.2 Zero-Day Botnet Attack Scenarios

The network traffic patterns and the nature of botnet attack that is launched against the IoT edge nodes are usually different. In this study, zero-day botnet attack scenarios are modelled using the 11-class Bot-IoT and the 10-class N-BaIoT datasets.

TABLE 6.1: Sample distribution of the zero-day botnet attack traffic data based on the Bot-IoT dataset

Class	Edge Nodes									
	EN1	EN2	EN3	EN4	EN5	EN6	EN7	EN8	EN9	EN10
DD-H	0	539	1619	863	1295	215	971	1403	1731	2159
DD-T	117278	0	87958	46911	70367	11727	52775	76231	93827	29319
DD-U	113752	28438	0	45500	68251	11375	51188	73938	91004	85314
D-H	2159	539	1619	0	1295	215	971	1403	1731	863
D-T	73993	18498	55494	29597	0	7399	33296	48095	59198	44395
D-U	123882	30970	92912	49553	74329	0	55747	80523	99110	12388
Norm	2159	539	1619	863	1295	215	971	1403	1079	652
OSF	2159	539	1619	863	1295	215	0	1403	1731	971
SS	8789	2197	6592	3515	5273	878	3955	0	7037	5713
DE	2159	539	1619	863	1295	215	971	1403	0	1731
KL	2159	539	1619	863	1295	215	971	1403	1731	0

TABLE 6.2: Sample distribution of the zero-day botnet attack traffic data based on the N-BaIoT dataset

Class	Edge Nodes								
	EN1	EN2	EN3	EN4	EN5	EN6	EN7	EN8	EN9
Norm	33339	33339	33339	33339	33339	33339	33339	33339	33339
g_combo	37039	0	33953	33953	33953	33953	33953	33953	33953
g_junk	18832	17263	0	17263	17263	17263	17263	17263	17263
g_scan	18377	16846	16846	0	16846	16846	16846	16846	16846
g_udp	68094	62419	62419	62419	0	62419	62419	62419	62419
m_ack	46435	42566	42566	42566	42566	0	42566	42566	42566
m_scan	0	35511	35511	35511	35511	35511	35511	35511	35511
m_syn	52850	48446	48446	48446	48446	48446	0	48446	48446
m_udp	88577	81196	81196	81196	81196	81196	81196	0	81196
m_udpp	37643	34506	34506	34506	34506	34506	34506	34506	0

Table 6.1 presents the sample distribution of the zero-day botnet attack traffic data in ten IoT edge nodes based on the Bot-IoT dataset, and Table 6.2 presents the sample distribution in nine other IoT edge nodes based on the NBaIoT dataset. A class of botnet attack traffic was not included in each of the IoT edge nodes to model zero-day botnet attack scenario. For example, in Table 6.1, there is no sample of DD-H attack in EN1, and there is no sample of KL attack in EN10. Similarly, in Table 6.2, there is no sample of *g_junk* attack in EN3, and there is no sample of *m_syn* attack in EN7. In order to depict a real-life scenario, the distribution of botnet attack samples was unbalanced and non-identically distributed across the classes of network traffic and across the IoT edge nodes.

6.3 Federated Deep Learning Method

FDL method is proposed to detect zero-day botnet attacks in IoT-enabled critical infrastructure based on Algorithm 8. The FDL framework comprised of a model parameter server and K IoT edge nodes. The model parameter server coordinates the training of LAE-BLSTM[1] models in the IoT edge nodes. Also, it determines the number of training iterations/epochs (E), the batch size of training data (B), and the number of communication rounds (R). In this method, K LAE-BLSTM models are trained separately with local training data that are privately held in K IoT edge nodes. After each training of E epochs, all the edge IoT devices send their local model updates to the model parameter server for aggregation using FedAvg algorithm [46]. Model aggregation is performed by model parameter server in R communication rounds.

```
Algorithm 8: FDL algorithm
Input: $R, E, N, B, K$
Initialization: $W = W_0$
Output: $W_r$
function localUpdate($W$,$k$):
    for $e = 1$ to $E$ do
        for $b = 1$ to $\frac{N}{B}$ do
            $W_{k,b} = W_{k,b-1} - \gamma \Delta L(b, W_k)$
        end
    end
    return $W_k$
end function
for $r = 1$ to $R$ do
    for $k = 1$ to $K$ do
        $W_{r,k}$ = localUpdate($W_{r-1}, k$)
    end
    $W_r = \sum_{k=1}^{K} \frac{n_k}{N} W_{r,k}$
end
```

The FDL method was simulated with the Bot-IoT and N-BaIoT data sets to evaluate the effectiveness of this method for zero-day botnet attack detection in IoT edge nodes, as shown in Fig. 6.1. The network traffic data in the IoT edge nodes was pre-processed as earlier described in Section 3.3. The SMOTE method in Chapter 4 was used to achieve class balance when the network traffic data is highly imbalanced. The LAE-BLSTM method in Chapter 5 was used for feature dimensionality reduction. The optimisation method in Chapter 3 was used to select the most suitable set of

[1] The LAE-BLSTM method was discussed earlier in Chapter 5

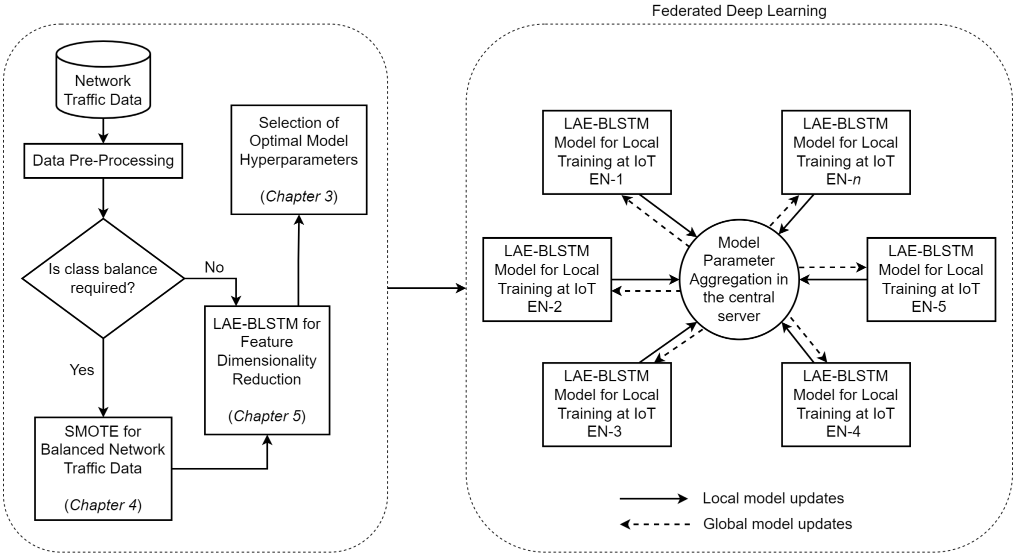

Network Traffic Data
Data Pre-Processing
Is class balance required?
No
Yes
SMOTE for Balanced Network Traffic Data (Chapter 4)
LAE-BLSTM for Feature Dimensionality Reduction (Chapter 5)
Selection of Optimal Model Hyperparameters (Chapter 3)
Federated Deep Learning
LAE-BLSTM Model for Local Training at IoT EN-1
LAE-BLSTM Model for Local Training at IoT EN-n
LAE-BLSTM Model for Local Training at IoT EN-2
Model Parameter Aggregation in the central server
LAE-BLSTM Model for Local Training at IoT EN-5
LAE-BLSTM Model for Local Training at IoT EN-3
LAE-BLSTM Model for Local Training at IoT EN-4
Local model updates
Global model updates

FIGURE 6.1: FDL architecture for zero-day botnet attack detection in IoT edge nodes

hyperparameters for local model training in the IoT edge nodes. IBM[2] FL framework was used to implement the proposed method. The models were trained using the Spyder[3] Integrated Development Environment (IDE) running on Ubuntu 16.04 LTS workstation with the following specifications: Random Access Memory (32 GB), Processor (Intel Core i7-9700K CPU @ 3.60GHz × 8), and 64-bit Operating System (OS). The deployment of the FDL model in IoT edge nodes was simulated using Linux terminals. The communication between the model parameter server and the IoT edge nodes was established using the Flask[4] web framework.

The performance of the FDL method was compared with that of CDL and LDL methods. For the CDL method, each of the IoT edge nodes transmitted its training data to a central server for aggregation. Therefore, the CDL model was trained with an aggregated data in the cloud. A copy of the CDL model was sent back to all the IoT edge devices for network traffic classification on the testing data. For the LDL method, model training was performed with the local training data in the edge IoT devices. Therefore, a unique LDL model was developed for each of the IoT edge devices. In the FDL method, a global LAE-BLSTM model was developed and a copy of this model was transmitted to all the IoT edge nodes for network traffic classification. The model parameter server receives further updates from the local models in the

[2] https://ibmfl.mybluemix.net/
[3] https://www.spyder-ide.org/
[4] https://palletsprojects.com/p/flask/

IoT edge nodes to improve the classification performance of the global FDL model.

6.4 Results Analysis and Discussion

In this section, the effectiveness of the CDL, LDL, and FDL models is evaluated with the network traffic data in the testing sets of the Bot-IoT and N-BaIoT datasets based on the following: (a) classification performance, (b) computation efficiency, (c) memory efficiency, (d) data privacy preservation, (e) communication cost, and (f) network latency. The sample distribution of the testing sets for the Bot-IoT and N-BaIoT datasets was presented earlier in Tables 3.1 and 3.2, respectively.

6.4.1 Centralised Deep Learning Models

A CDL-based botnet attack detection model, which employed the LAE-BLSTM architecture, was trained and tested with the Bot-IoT dataset. Table 6.3 shows that the model achieved a high classification performance with 99.98 ± 0.04% accuracy, 99.03 ± 2.94% precision, 99.53 ± 0.92% recall, and 99.25 ± 1.54% F1 score. All the network traffic samples in the *DD-T*, *D-T*, *D-U*, *Norm*, *DE*, and *KL* classes were classified correctly. This means that the CDL model can distinctively detect benign network traffic as well as DD-T, D-T, D-U, DE, and KL attack traffic in IoT-enabled critical infrastructure with 100% accuracy and zero false alarm rate.

TABLE 6.3: Classification performance of the CDL model based on the Bot-IoT dataset

Class	Classification Performance (%)			
	Accuracy	Precision	Recall	F1 Score
DD-H	100.00	100.00	98.04	99.01
DD-T	99.99	99.98	99.99	99.99
DD-U	99.89	99.97	99.60	99.78
D-H	100.00	99.62	97.39	98.49
D-T	99.99	99.98	99.98	99.98
D-U	99.89	99.63	99.97	99.80
Norm	100.00	90.18	100.00	94.84
OSF	100.00	99.94	99.92	99.93
SS	100.00	99.99	99.92	99.95
DE	100.00	100.00	100.00	100.00
KL	100.00	100.00	100.00	100.00

Although the CDL model could not classify all the network traffic samples in the remaining five classes correctly, the detection rates were very high and the false alarm rates were very low. In the *DD-H* class, 98% of the samples were correctly classified, and 2% were misclassified as DD-T attack traffic. In the *DD-U* class, 99.6% of the samples were correctly classified, and 0.4% were misclassified as D-U attack traffic. In the *D-H* class, 97.4% of the samples were correctly classified, 1.1% were misclassified as DD-T attack traffic, 1.1% were misclassified as D-T attack traffic, and a sample was misclassified as a benign network traffic. In the *OSF* class, 99.9% of the samples were correctly classified, and 0.1% were misclassified as SS attack traffic. In the *SS* class, 99.9% of the samples were correctly classified, and 0.1% were misclassified as benign network traffic. Therefore, the CDL model can also distinctively detect DD-H, DD-U, D-H, OSF, and SS attack traffic in IoT-enabled critical infrastructure with high accuracy and low false alarm rate. The computation efficiency of the CDL model was high. It took 212.75 seconds to train the model with 2253251 network traffic samples in the training set, while the model spent 1.18 seconds to classify 733705 network traffic samples in the testing set. However, the CDL method required a high memory space of 72.10 MB to store network traffic data in a central cloud server for model training.

TABLE 6.4: Classification performance of the CDL model based on the N-BaIoT dataset

Class	Classification Performance (%)			
	Accuracy	Precision	Recall	F1 Score
Norm	99.97	99.85	99.77	99.81
g_combo	99.88	99.93	98.68	99.30
g_junk	99.88	97.39	99.81	98.59
g_scan	99.99	99.81	99.83	99.82
g_udp	99.98	99.90	99.96	99.93
m_ack	99.57	96.37	99.64	97.98
m_scan	99.98	99.86	99.94	99.90
m_syn	99.98	99.96	99.90	99.93
m_udp	99.73	99.64	99.01	99.33
m_udpp	99.73	99.77	97.06	98.39

Another CDL-based botnet attack detection model was trained and tested with the N-BaIoT dataset. Table 6.4 shows that the model achieved a high classification performance with 99.87 ± 0.14% accuracy, 99.25 ± 1.27% precision, 99.36 ± 0.91% recall, and 99.30 ± 0.73% F1 score. Almost all the network traffic samples in the ten classes were classified correctly. In the

Norm class, 99.8% of the samples were classified correctly, and 0.2% were misclassified as *g_udp* attack. In the *g_combo* class, 98.7% of the samples were classified correctly, and 1.3% were misclassified as *g_junk* attack. In the *g_junk* class, 99.8% of the samples were classified correctly, and 0.2% were misclassified as *g_combo* attack. In the *g_scan* class, 99.8% of the samples were classified correctly, and 0.2% were misclassified as benign network traffic. In the *g_udp* class, all the samples were classified correctly. In the *m_ack* class, 99.6% of the samples were classified correctly, 0.2% were misclassified as *m_udp* attack, and 0.2% were misclassified as *m_udpp* attack. In the *m_scan* class, 99.9% of the samples were classified correctly, and 0.1% were misclassified as *g_udp* attack. In the *m_syn* class, 99.9% of the samples were classified correctly, and 0.1% were misclassified as *m_scan* attack. In the *m_udp* class, 99% of the samples were classified correctly, and 1% were misclassified as *m_ack* attack. In the *m_udpp* class, 97.1% of the samples were classified correctly, 2.3% were misclassified as *m_ack* attack, and 0.6% were misclassified as *m_udp* attack. Therefore, the CDL model can distinctively detect each class of botnet attack traffic in IoT-enabled critical infrastructure with high accuracy and low false alarm rate. The computation efficiency of the CDL model was high. It took 478.63 seconds to train the model with 3721653 network traffic samples in the training set, while the model spent 3.86 seconds to classify 1240552 network traffic samples in the testing set. Also, the CDL method required a high memory space of 119.09 MB to store network traffic data in a central cloud server for model training.

The CDL models achieved an excellent classification performance because they were trained with large and diverse data which covered all the benign network traffic patterns and all the categories of botnet attacks that were generated and transmitted from the ten IoT edge nodes to a central cloud server. However, the aggregation of the network traffic data from multiple IoT edge nodes will cause data privacy leakage in the IoT-enabled critical infrastructure. Furthermore, the cost of transmitting large network traffic data from the IoT edge nodes to a central cloud server is high, and this will likely increase network latency in the system.

6.4.2 Localised Deep Learning Models

Ten LDL-based botnet attack detection models, which employed LAE-BLSTM architecture, were trained and tested with the Bot-IoT network traffic data that are located in ten edge nodes (EN1-EN10), respectively.

TABLE 6.5: Classification performance of the LDL models based on the Bot-IoT dataset

Edge Node	Classification performance (%)			
	Accuracy	Precision	Recall	F1 Score
EN1	99.69	78.15	89.89	81.77
EN2	94.27	58.64	84.32	63.13
EN3	94.93	77.84	85.67	80.84
EN4	99.14	72.75	87.88	76.88
EN5	96.77	64.76	85.42	69.61
EN6	94.49	60.57	83.42	66.14
EN7	99.34	67.57	85.87	72.73
EN8	99.10	61.49	85.53	66.19
EN9	99.55	76.39	87.77	80.53
EN10	98.09	69.52	83.38	72.71

Table 6.5 shows that the LDL models achieved a low classification performance with 97.54 ± 2.23% accuracy, 68.77 ± 7.34% precision, 85.91 ± 2.07% recall, and 73.05 ± 6.77% F1 score. None of the models was able to detect any of the zero-day botnet attacks at the IoT edge nodes. At IoT edge node EN1, all the DD-H attack samples were misclassified: 66.7% as D-H attack, and 33.3% as DD-T attack. At IoT edge node EN2, all the DD-T attack samples were misclassified: 83.2% as D-T attack, 14.8% as DD-H attack, and 2% as D-H attack. At IoT edge node EN3, all the DD-U attack samples were misclassified as D-U attack. At IoT edge node EN4, all the D-H attack samples were misclassified: 82.8% as DD-H attack, 11.6% as D-T attack, 5.2% as DD-T attack, and 0.4% as KL attack. At IoT edge node EN5, all the D-T attack samples were misclassified: 97.7% as DD-T attack, 1.5% as D-H attack, 0.6% as D-U attack, and 0.2% as DD-H attack. At IoT edge node EN6, all the D-U samples were misclassified: 99.4% as DD-U attack, 0.5% as benign network traffic, and 0.1% as OSF attack. At IoT edge node EN7, all the OSF attack samples were misclassified: 98.9% as SS attack, 0.8% as benign network traffic, and 0.3% as DD-U attack. At IoT edge node EN8, all the SS attack samples were misclassified: 70.3% as OSF attack, and 29.7% as benign network traffic. At IoT edge node EN9, the only DE attack sample was misclassified as KL attack. At IoT edge node EN10, all the KL attack samples were misclassified: 63.6% as D-H attack, and 36.4% as DE attack.

The LDL models, which were developed based on the Bot-IoT dataset, had a faster training time than the CDL models. Figure 6.2 shows that it took $48.94 - 127.88$ seconds to train the models with training sets of different sizes,

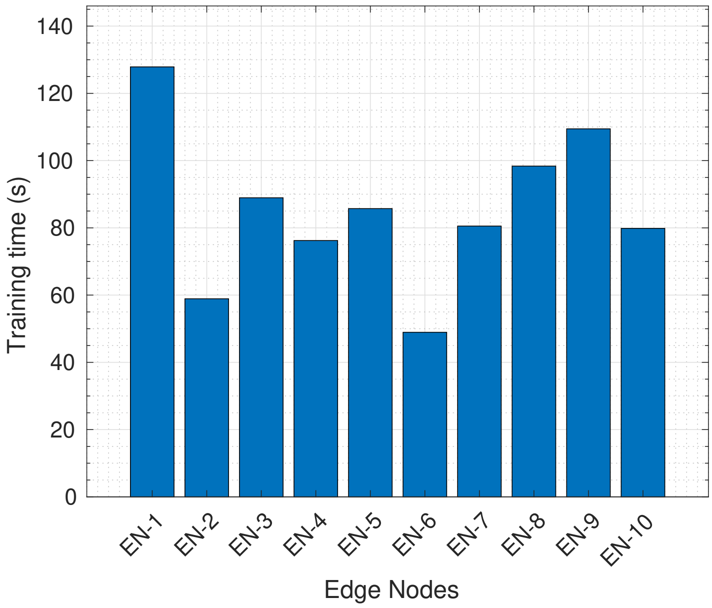


FIGURE 6.2: Training time of the LDL models based on the Bot-IoT dataset

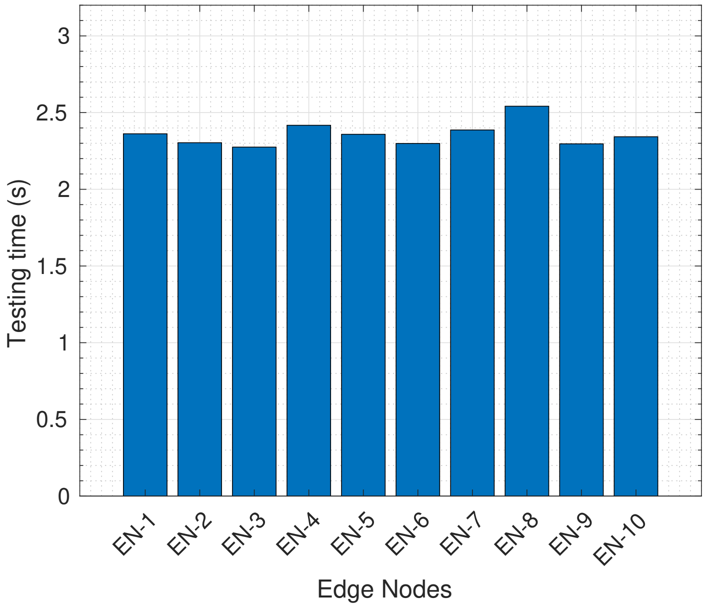


FIGURE 6.3: Testing time of the LDL models based on the Bot-IoT dataset

as shown in Table 6.1. However, the LDL models spent more time to classify the network traffic samples in the testing set, compared to the CDL models. Figure 6.3 shows that the LDL models spent $2.28 - 2.54$ seconds to classify 733705 network traffic samples in the testing set. Figure 6.4 shows that the LDL method required a smaller memory space of $2.1 - 28.7$ MB to store the network traffic data in the IoT edge nodes.

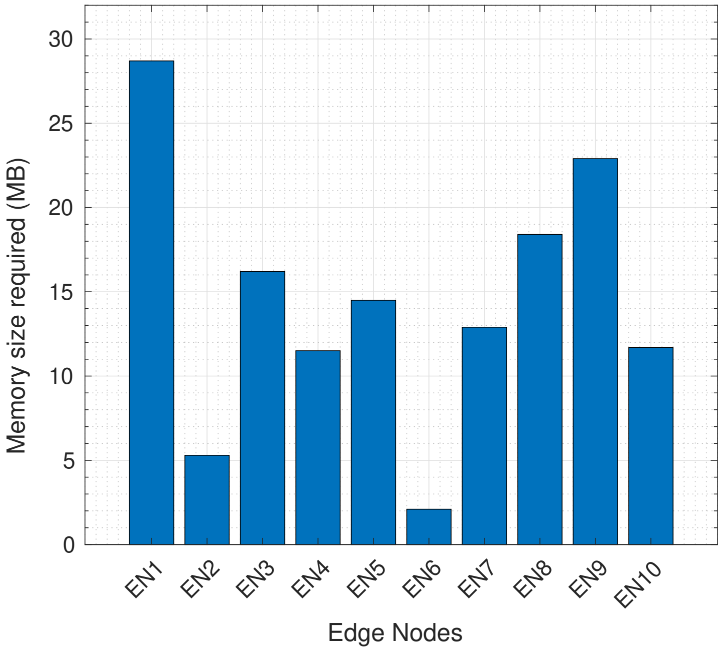


FIGURE 6.4: Memory sizes of the Bot-IoT dataset in the IoT edge nodes

Another nine LDL-based botnet attack detection models were trained and tested with the N-BaIoT network traffic data that are located in nine edge nodes (EN1-EN9), respectively.

TABLE 6.6: Classification performance of the LDL models based on the N-BaIoT dataset

Edge Node	Classification performance (%)			
	Accuracy	Precision	Recall	F1 Score
EN1	97.33	81.06	84.03	81.55
EN2	97.61	80.15	86.92	81.86
EN3	98.37	83.31	86.32	84.47
EN4	98.30	81.46	85.16	83.06
EN5	95.58	81.46	82.78	77.65
EN6	97.30	82.57	86.01	83.82
EN7	96.75	76.88	83.74	79.20
EN8	95.59	78.58	86.84	81.23
EN9	97.72	82.62	86.25	83.95

Table 6.6 shows that the LDL models achieved a low classification performance with 97.17 ± 1.03% accuracy, 80.90 ± 2.07% precision, 85.34 ± 1.49% recall, and 81.87 ± 2.28% F1 score. None of the models was able to detect any of the zero-day botnet attacks at the nine IoT edge nodes.

At IoT edge node EN1, all the *m_scan* attack samples were misclassified: 90.3% as *m_syn* attack, and 9.5% as *g_scan* attack. At IoT edge node EN2, all the *g_combo* attack samples were misclassified: 98.7% as *g_junk* attack, and 1.3% as *m_syn* attack. At IoT edge node EN3, all the *g_junk* attack samples were misclassified: 99.5% as *g_combo* attack, and 0.5% as *g_scan* attack. At IoT edge node EN4, all the *g_scan* attack samples were misclassified: 57% as *m_syn* attack, 39.6% as *m_scan* attack, and 3.4% as benign network traffic. At IoT edge node EN5, all the benign network traffic were misclassified. At IoT edge node EN6, all the *m_ack* attack samples were misclassified: 96.4% as *m_udp* attack, and 3.6% as *m_udpp* attack. At IoT edge node EN7, all the *m_syn* attack samples were misclassified: 69.3% as *g_combo* attack, 19.1% as *m_scan* attack, 9.7% as *g_junk* attack, and 1.9% as *m_udp* attack. At IoT edge node EN8, all the *m_udp* attack samples were misclassified: 88.4% as *m_ack* attack, and 11.6% as *m_udpp* attack. At IoT edge node EN9, all the *m_udpp* attack samples were misclassified: 81.9% as *m_ack* attack, and 18.1% as *m_udp* attack.

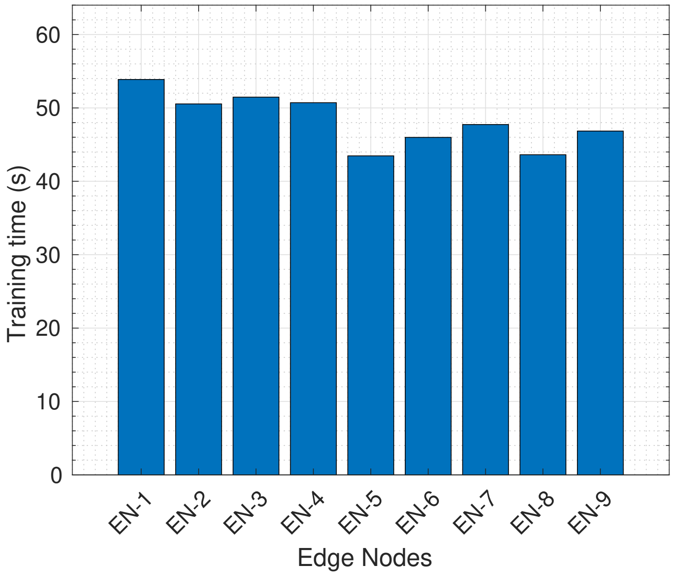


FIGURE 6.5: Training time of the LDL models based on the N-BaIoT dataset

The LDL models, which were developed based on the N-BaIoT dataset, had a faster training time than the CDL models. Figure 6.5 shows that it took 43.47 – 53.87 seconds to train the models with training sets of different sizes, as shown in Table 6.2. However, the LDL models spent more time to classify the network traffic samples in the testing set, compared to the CDL models. Figure 6.6 shows that the LDL models spent 3.77 – 4.64 seconds to classify

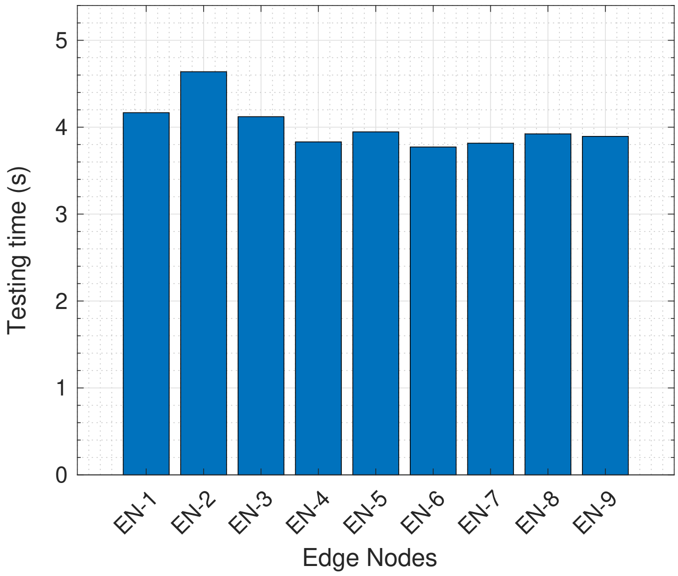


FIGURE 6.6: Testing time of the LDL models based on the N-BaIoT dataset

733705 network traffic samples in the testing set. Figure 6.7 shows that the LDL method required a smaller memory space of 20.8 − 25.7 MB to store the network traffic data in the IoT edge nodes.

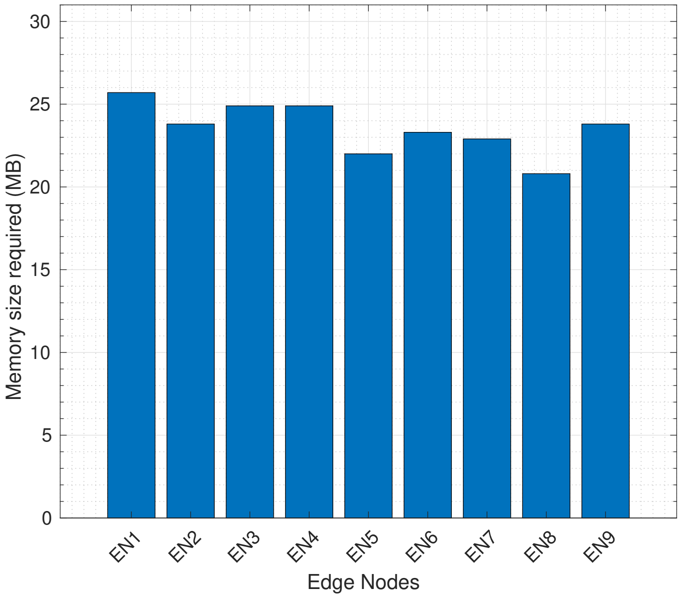


FIGURE 6.7: Memory sizes of the N-BaIoT dataset in the IoT edge nodes

In LDL method, the network traffic features of the IoT edge nodes were not shared with a third-party central cloud server to preserve the data privacy of IoT-enabled critical infrastructure users. The LDL models required a shorter training time and a lower memory space for data storage, and they incurred lower communication overhead. However, the classification performance of the LDL models was lower than that of the CDL models because each of former was trained with insufficient private network traffic and fewer botnet attack scenarios in a single IoT edge node. Therefore, the LDL method is not suitable for zero-day botnet attack detection in IoT-enabled critical infrastructure.

6.4.3 Federated Deep Learning Models

A FDL-based botnet attack detection model, which employed LAE-BLSTM architecture, was trained and tested with Bot-IoT network traffic data at ten IoT edge nodes.

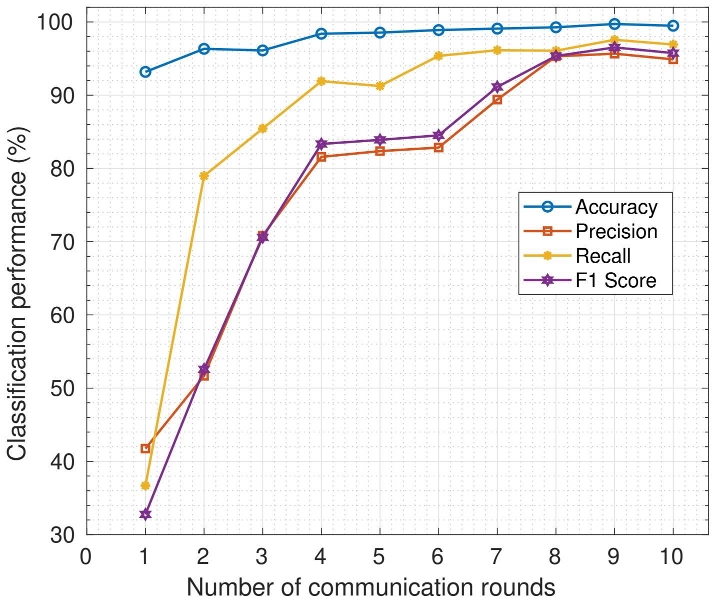


FIGURE 6.8: Classification performance of the FDL model based on the Bot-IoT dataset

Figure 6.8 shows that the classification performance of the FDL model improved as the number of communication rounds increased from 1 to 10. Specifically, the accuracy, precision, recall, and F1 score of the model increased by 6.54%, 53.92%, 60.88%, and 63.75%, respectively. The FDL model achieved the best classification performance at the end of the ninth communication round with 99.72% accuracy, 95.67% precision, 97.56% recall, and 96.52% F1 score. All the network traffic samples in the *DD-T*, *DD-U*, *D-U*, *Norm*, *OSF*, *SS*, *DE*, and *KL* classes were classified correctly. This means that the FDL model can distinctively detect benign network traffic as well as DD-T, DD-U, D-U, OSF, SS, DE, and KL attack traffic in IoT-enabled critical infrastructure with 100% accuracy and zero false alarm rate.

Although the FDL model could not classify all the network traffic samples in the *DD-H*, *D-H*, and *D-T* classes correctly, the detection rates were very high and the false alarm rates were very low. In the DD-H class, 90.7% of the samples were classified correctly, 5.9% were misclassified as D-H attack, and 3.4% were misclassified as DD-T attack. In the *D-H* class, 91.8% of the samples were classified correctly, 6.3% were misclassified as DD-H attack, and 1.9% were misclassified as DD-T attack. In the *D-T* class, 91.3% of the samples were classified correctly, and 8.7% were misclassified as DD-T attack. Therefore, the FDL model can also distinctively detect DD-H, DD-U, D-H, OSF, and SS attack traffic in IoT-enabled critical infrastructure with high accuracy and low false alarm rate.

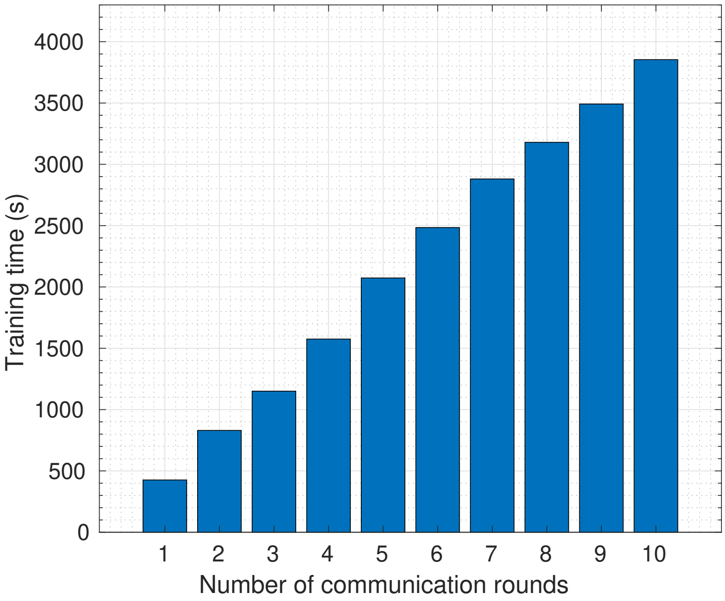


FIGURE 6.9: Training time of the FDL models based on the Bot-IoT dataset

Figure 6.9 shows that the time required to train the FDL model increased from 426 to 3853.64 seconds as the number of communication rounds increased from 1 to 10. The training time of the FDL model which achieved the best classification performance at the end of the ninth communication round was 3491.72 seconds. Figure 6.10 shows that the FDL model spent 3.7 – 5.3 seconds to classify 733705 network traffic samples in the testing set.

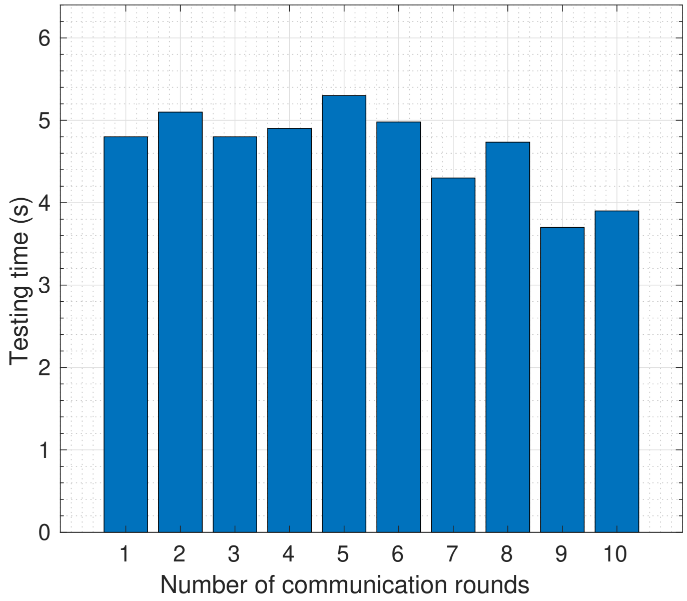


FIGURE 6.10: Testing time of the FDL models based on the Bot-IoT dataset

Another FDL-based botnet attack detection model, which employed LAE-BLSTM architecture, was trained and tested with N-BaIoT network traffic data at nine IoT edge nodes.

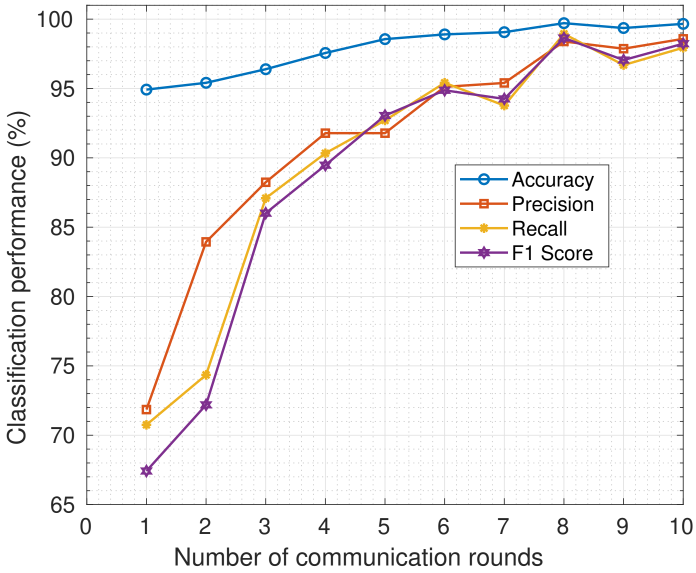


FIGURE 6.11: Classification performance of the FDL model based on the N-BaIoT dataset

Figure 6.11 shows that the classification performance of the FDL model improved as the number of communication rounds increased from 1 to 10.

Specifically, the accuracy, precision, recall, and F1 score of the model increased by 4.78%, 26.54%, 28.19%, and 31.22%, respectively. The FDL model achieved the best classification performance at the end of the eighth communication round with 99.71% accuracy, 98.39% precision, 98.94% recall, and 98.64% F1 score. All the network traffic samples in the *Norm*, *g_junk*, *g_scan*, *g_udp*, *m_ack*, *m_scan*, and *m_syn* classes were classified correctly. This means that the FDL model can distinctively detect benign network traffic as well as *g_junk*, *g_scan*, *g_udp*, *m_ack*, *m_scan*, and *m_syn* attack traffic in IoT-enabled critical infrastructure with 100% accuracy and zero false alarm rate.

Although the FDL model could not classify all the network traffic samples in the *g_combo*, *m_udp*, and *m_udpp* classes correctly, the detection rates were very high and the false alarm rates were very low. In the *g_combo* class, 98% of the samples were classified correctly, and 2% were misclassified as *g_junk* attack. In the *m_udp* class, 94.9% of the samples were classified correctly, 3.9% were misclassified as *m_ack* attack, 0.4% were misclassified as *m_udpp* attack, and 0.7% were misclassified as benign network traffic. In the *m_udpp* class, 98% of the samples were classified correctly, 1.6% were misclassified as *m_ack* attack, and 0.4% were misclassified as *m_udp* attack. Therefore, the FDL model can also distinctively detect D*g_combo*, *m_udp*, and *m_udpp* attack traffic in IoT-enabled critical infrastructure with high accuracy and low false alarm rate.

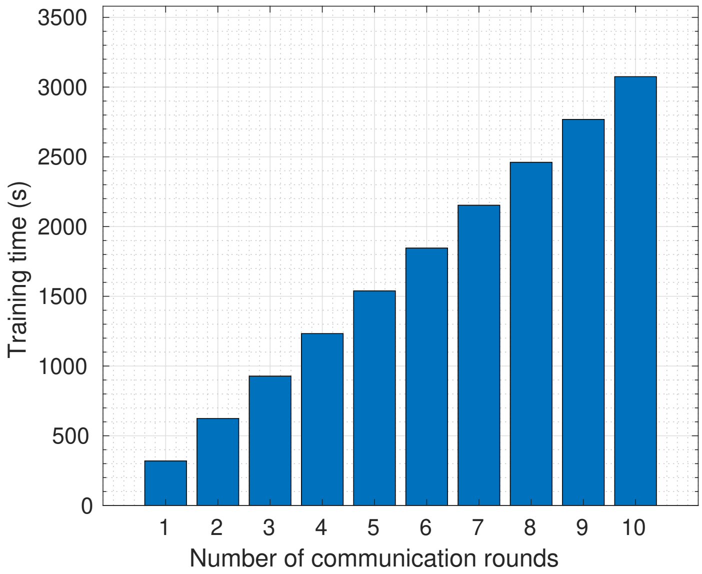


FIGURE 6.12: Training time of the FDL models based on the N-BaIoT dataset

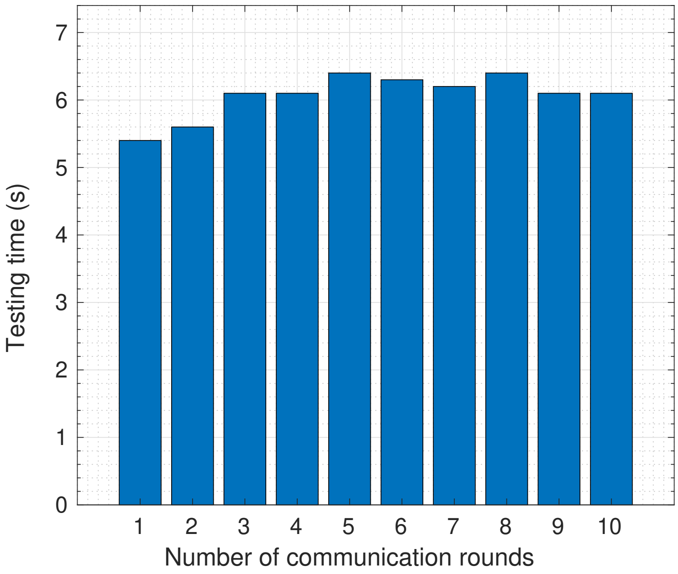


FIGURE 6.13: Testing time of the FDL models based on the N-BaIoT dataset

Figure 6.12 shows that the time required to train the FDL model increased from 319.95 to 3074.85 seconds as the number of communication rounds increased from 1 to 10. The training time of the FDL model which achieved the best classification performance at the end of the eighth communication round was 2460.92 seconds. Figure 6.13 shows that the FDL model spent 5.4 − 6.4 seconds to classify 733705 network traffic samples in the testing set.

TABLE 6.7: Summary of research findings

	CDL	LDL	FDL
Data aggregation	✓	✗	✗
Model parameter aggregation	✗	✗	✓
Classification performance	high	low	high
Training time	long	short	long
Data privacy	✗	✓	✓
Communication overhead	high	low	low
Memory requirement	high	low	low
Network latency	high	low	low

The overview of our findings in this chapter is presented in Table 6.7. FDL method detects zero-day botnet attacks with high classification performance; it guarantees data privacy and security; it has low communication overhead; it requires low memory space for the data storage; and it has low latency. The FDL model achieved a better classification performance than the LDL and DDL models because the central parameter server receives multiple local model updates from all the IoT edge devices. The only trade-off in FDL method is the time required to train its model. Therefore, FDL method is efficient for zero-day botnet attack detection in IoT edge devices.

6.5 Chapter Summary

In this chapter, FDL method is proposed for zero-day attack detection in IoT edge edge nodes of critical infrastructure. FDL model was developed with the Bot-IoT and N-BaIoT data sets, and its effectiveness was compared with the CDL, LDL, and DDL models. The CDL model involves data aggregation, and it achieved high classification performance. However, it did not preserve the privacy and security of network traffic data in IoT edge devices. Also, the CDL model had high communication overhead, large memory space requirement for data storage, high network latency, and it took long time to train the model. Although the LDL models overcame the limitations of the CDL model, their classification performance was very low. Interestingly, the FDL model outperformed the CDL and LDL models, except for the long training time. Therefore, the FDL method is most efficient for zero-day botnet attack detection in the IoT edge devices.

Chapter 7

Conclusion and Future Work

In this chapter, the major findings in this thesis are summarised, and recommendations are provided for possible future research.

7.1 Conclusion

In this thesis, a FDL method was developed for botnet attack detection in IoT-enabled critical infrastructure. This method addressed the challenges that were identified in Chapter 2 which include determining the optimal set of hyperparameters for DL-based botnet attack detection model, low classification performance due to imbalanced sample distribution in the training set, high memory space requirement for network traffic data storage, inability to detect zero-day attacks, and lack of data privacy.

7.1.1 Model Hyperparameter Optimisation

In Chapter 3, a method was developed to determine the optimal sets of hyperparameters for DL-based botnet attack detection model. DNN models were trained, validated, and tested with the network traffic features in the Bot-IoT and N-BaIoT datasets to perform binary and multi-class classification. A learning rate of 0.001, ReLU activation function, and a batch size of 512 were the most suitable in all cases. However, the numbers of hidden layers and their hidden units, the optimiser, and the number of epochs were different for each dataset and each classification scenario. Two hidden layers with 128 and 16 hidden units, Nadam optimiser, and 5 epochs produced the optimal classification performance when DNN models were developed with the Bot-IoT dataset for binary classification. Four hidden layers with 128, 128, 128, and 16 hidden units, Adam optimiser, and 10 epochs produced the optimal classification performance when DNN models were developed with the Bot-IoT dataset for 5-class classification. Four hidden layers with 128, 128, 128 and 64 hidden units, Adam optimiser, and 15 epochs produced the optimal classification performance when DNN models were developed with the Bot-IoT dataset for 11-class classification. Three hidden layers with 128, 128 and 16 hidden units, Adam optimiser, and 5 epochs produced the optimal classification performance when DNN models were developed with the N-BaIoT dataset for binary classification. Two hidden layers with 128 and 32 hidden units, Nadam optimiser, and 15 epochs produced the optimal classification performance when DNN models were developed with the N-BaIoT dataset for 10-class classification. Overall, the optimal DNN models achieved 99.99 ± 0.02% accuracy, 97.85 ± 3.77% precision, 98.72 ± 2.77% recall, and 97.72 ± 4.51% F1 score. It took $33.49 - 168.85$ seconds to train the models, and they spent $0.43 - 1.16$ seconds to classify the network traffic data in the testing set.

7.1.2 Imbalanced Network Traffic Classification

In Chapter 4, a method, named SMOTE-DL, was developed to improve the classification performance of DL models when the network traffic data in the training set is highly imbalanced. SMOTE algorithm was used to generate 52139 synthetic network traffic samples to increase the class imbalance ratio of the Bot-IoT dataset from as low as 1:154854 to 1:57. DNN, RNN, LSTM, and GRU models were trained with the balanced network traffic data. The optimal hyperparameters in Chapter 3 were used for the development of the SMOTE-DL models. The precision, recall, and F1 score of the models improved by $6.4 - 13.48\%$, $3.12 - 12.81\%$, and $7.01 - 13.27\%$, respectively. Specifically, the SMOTE-LSTM model achieved the best classification performance with 100% accuracy, 94.20% precision, 98.99% recall, and 95.83% F1 score. Interestingly, the SMOTE method did not introduce any significant computation complexity to the DL-based botnet attack detection process.

7.1.3 Memory-Efficient Botnet Attack Detection

In Chapter 5, a hybrid DL method, named LAE-BLSTM, was developed to reduce the memory space required to store network traffic data without compromising the classification performance. The optimal hyperparameters in Chapter 3 were used for model development, and the SMOTE method in Chapter 4 was used when the network traffic data in the training set is highly imbalanced. LAE models were trained to reduce the dimensionality of the features in the Bot-IoT and N-BaIoT datasets. This reduced the memory space required to store the data in a central cloud server or in an IoT node by $86.46 - 98.26\%$. BLSTM models were trained with the reduced feature sets to perform binary and multi-class classification, and they achieved 99.91 ± 0.12 accuracy, 99.33 ± 0.57 precision, 98.82 ± 1.13 recall, and 99.06 ± 0.75 F1 score. Therefore, the LAE-BLSTM method can be used for memory-efficient botnet attack detection in IoT-enabled critical infrastructure.

7.1.4 Zero-Day Botnet Attack Detection

In Chapter 6, a FDL method was developed to detect zero-day botnet attack in IoT edge nodes, and preserve the data privacy of IoT-enabled critical infrastructure users. Zero-day botnet attack scenarios were simulated with the Bot-IoT and N-BaIoT datasets, and the network traffic samples in these datasets were distributed among ten and nine IoT edge nodes, respectively. The LAE-BLSTM model architecture in Chapter 5 was used for local training at the IoT edge nodes. The method in Chapter 3 was used to optimise the hyperparameters of the LAE-BLSTM models. The SMOTE method in Chapter 4 was employed for class balance when the network traffic data in the training set is highly imbalanced. The FDL method eliminated the need to aggregate and transmit the network traffic data from multiple IoT edge nodes to a central cloud server for model training. This preserved users' data privacy, reduced communication overhead, and reduced network latency. Also, the FDL models achieved high classification performance, and they required lower memory space to store network traffic data at the edge nodes, compared to the centralised approach. Therefore, FDL method can detect known and unknown botnet attacks in IoT-enabled critical infrastructure with high detection accuracy and low false alarm rate.

7.2 Recommendations for Future Work

7.2.1 Implementation of FDL on Real IoT Hardware

The IoT edge nodes in this thesis were simulated using different Linux terminals on the same computer. In the future, the FDL models will be deployed on real IoT devices such as Raspberry Pi 4B using an open-source platform called FedIoT[1] [175].

7.2.2 Advanced Aggregation Methods for Robust FDL

The potentials of advanced model parameter aggregation algorithms such as Probabilistic Federated Neural Matching (PFNM) [205], Fed+ [206], and FedProx [207] will be explored to improve the classification performance of FDL models with less communication rounds. Furthermore, a decentralised FL approach will be investigated to eliminate the need for a central model parameter aggregation server. In this method, the IoT edge node with the best computation and energy resources will be nominated to perform model parameter aggregation, This will further reduce communication cost and network latency.

7.2.3 Securing FDL Models against Adversarial Attacks

Efforts will be made to protect the FDL models against adversarial attacks such as backdoor and poisoning attacks [208]. At the IoT edge nodes, an attacker can manipulate the network traffic features or insert corrupt features to 'poison' the training dataset. Also, the model parameter updates from the IoT edge nodes can be intercepted and contaminated when transmitted to the cloud server for aggregation. The potentials of blockchain technology [209] and differential privacy mechanisms [210] will be explored to achieve a trustworthy FDL model for botnet attack detection in IoT-enabled critical infrastructure.

[1] https://github.com/FedML-AI/FedIoT

Bibliography

[1] CPNI, *Critical national infrastructure*, Last accessed: 29 October 2021, 2021. [Online]. Available: `https://www.cpni.gov.uk/critical-national-infrastructure-0`.

[2] K. Shafique, B. A. Khawaja, F. Sabir, S. Qazi, and M. Mustaqim, "Internet of things (iot) for next-generation smart systems: A review of current challenges, future trends and prospects for emerging 5g-iot scenarios," *Ieee Access*, vol. 8, pp. 23 022–23 040, 2020.

[3] N. Mishra and S. Pandya, "Internet of things applications, security challenges, attacks, intrusion detection, and future visions: A systematic review," *IEEE Access*, 2021.

[4] Y. Yang, H. Wang, R. Jiang, X. Guo, J. Cheng, and Y. Chen, "A review of iot-enabled mobile healthcare: Technologies, challenges, and future trends," *IEEE Internet of Things Journal*, 2022.

[5] N. Pathak, P. K. Deb, A. Mukherjee, and S. Misra, "Iot-to-the-rescue: A survey of iot solutions for covid-19-like pandemics," *IEEE Internet of Things Journal*, 2021.

[6] Y. Dong and Y.-D. Yao, "Iot platform for covid-19 prevention and control: A survey," *IEEE Access*, vol. 9, pp. 49 929–49 941, 2021.

[7] S. A. A. Abir, A. Anwar, J. Choi, and A. Kayes, "Iot-enabled smart energy grid: Applications and challenges," *IEEE access*, vol. 9, pp. 50 961–50 981, 2021.

[8] M. S. Farooq, O. O. Sohail, A. Abid, and S. Rasheed, "A survey on the role of iot in agriculture for the implementation of smart livestock environment," *IEEE Access*, 2022.

[9] Y. Song, F. R. Yu, L. Zhou, X. Yang, and Z. He, "Applications of the internet of things (iot) in smart logistics: A comprehensive survey," *IEEE Internet of Things Journal*, 2020.

[10] D. C. Nguyen, M. Ding, P. N. Pathirana, *et al.*, "6g internet of things: A comprehensive survey," *IEEE Internet of Things Journal*, 2021.

[11] F. Guo, F. R. Yu, H. Zhang, X. Li, H. Ji, and V. C. Leung, "Enabling massive iot toward 6g: A comprehensive survey," *IEEE Internet of Things Journal*, 2021.

[12] J. Singh, T. Pasquier, J. Bacon, H. Ko, and D. Eyers, "Twenty security considerations for cloud-supported internet of things," *IEEE Internet of things Journal*, vol. 3, no. 3, pp. 269–284, 2015.

[13] G. S. Aujla, R. Chaudhary, K. Kaur, S. Garg, N. Kumar, and R. Ranjan, "Safe: Sdn-assisted framework for edge–cloud interplay in secure healthcare ecosystem," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 1, pp. 469–480, 2018.

[14] A. Singh, S. Garg, S. Batra, N. Kumar, and J. J. Rodrigues, "Bloom filter based optimization scheme for massive data handling in iot environment," *Future Generation Computer Systems*, vol. 82, pp. 440–449, 2018.

[15] S. Garg, A. Singh, K. Kaur, *et al.*, "Edge computing-based security framework for big data analytics in vanets," *IEEE Network*, vol. 33, no. 2, pp. 72–81, 2019.

[16] R. M. Lee, M. J. Assante, and C. T, "Analysis of the cyber attack on the ukrainian power grid," *Electricity Information Sharing and Analysis Center (E-ISAC)*, vol. 388, 2016.

[17] B. D. Davis, J. C. Mason, and M. Anwar, "Vulnerability studies and security postures of iot devices: A smart home case study," *IEEE Internet of Things Journal*, 2020.

[18] W. Zhou, Y. Jia, A. Peng, Y. Zhang, and P. Liu, "The effect of iot new features on security and privacy: New threats, existing solutions, and challenges yet to be solved," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1606–1616, 2018.

[19] M. Stoyanova, Y. Nikoloudakis, S. Panagiotakis, E. Pallis, and E. K. Markakis, "A survey on the internet of things (iot) forensics: Challenges, approaches and open issues," *IEEE Communications Surveys & Tutorials*, 2020.

[20] I. Stellios, P. Kotzanikolaou, M. Psarakis, C. Alcaraz, and J. Lopez, "A survey of iot-enabled cyberattacks: Assessing attack paths to critical infrastructures and services," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3453–3495, 2018.

[21] T. Qiu, N. Chen, K. Li, M. Atiquzzaman, and W. Zhao, "How can heterogeneous internet of things build our future: A survey," *IEEE*

Communications Surveys & Tutorials, vol. 20, no. 3, pp. 2011–2027, 2018.

[22] L. Yin, X. Luo, C. Zhu, L. Wang, Z. Xu, and H. Lu, "Connspoiler: Disrupting c&c communication of iot-based botnet through fast detection of anomalous domain queries," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 2, pp. 1373–1384, 2020.

[23] S. Walker-Roberts, M. Hammoudeh, O. Aldabbas, M. Aydin, and A. Dehghantanha, "Threats on the horizon: Understanding security threats in the era of cyber-physical systems," *The Journal of Supercomputing*, pp. 1–22, 2019.

[24] M. Antonakakis, T. April, M. Bailey, *et al.*, "Understanding the mirai botnet," in *26th {USENIX} Security Symposium ({USENIX} Security 17)*, 2017, pp. 1093–1110.

[25] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "Ddos in the iot: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[26] D. McMillen, W. Gao, and C. DeBeck, *A new botnet attack just mozied into town*, Last accessed: September 18, 2020, 2020. [Online]. Available: `https://securityintelligence.com`.

[27] S. Soltan, P. Mittal, and H. V. Poor, "Blackiot: Iot botnet of high wattage devices can disrupt the power grid," in *27th {USENIX} Security Symposium ({USENIX} Security 18)*, 2018, pp. 15–32.

[28] ——, "Protecting the grid against iot botnets of high-wattage devices," *arXiv preprint arXiv:1808.03826*, 2018.

[29] H. S. Lallie, L. A. Shepherd, J. R. Nurse, *et al.*, "Cyber security in the age of covid-19: A timeline and analysis of cyber-crime and cyber-attacks during the pandemic," *arXiv preprint arXiv:2006.11929*, 2020.

[30] S. N. John, O. A. Albert, K. Okokpujie, E. Noma-Osaghae, O. Osemwegie, and C. Okereke, "Mitigating threats in a corporate network with a taintcheck-enabled honeypot," in *Information Science and Applications*, Springer, 2020, pp. 73–83.

[31] J. Yu, *Heh, a new iot p2p botnet going after weak telnet services*, Last accessed on 19 November 2020, October 2020. [Online]. Available: `https://blog.netlab.360.com/heh-an-iot-p2p-botnet/`.

[32] S. Raza, L. Wallgren, and T. Voigt, "Svelte: Real-time intrusion detection in the internet of things," *Ad hoc networks*, vol. 11, no. 8, pp. 2661–2674, 2013.

[33] E. Bertino and N. Islam, "Botnets and internet of things security," *Computer*, no. 2, pp. 76–79, 2017.

[34] P. Mishra, V. Varadharajan, U. Tupakula, and E. S. Pilli, "A detailed investigation and analysis of using machine learning techniques for intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 686–728, 2018.

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

[36] Y. Xin, L. Kong, Z. Liu, *et al.*, "Machine learning and deep learning methods for cybersecurity," *IEEE Access*, vol. 6, pp. 35 365–35 381, 2018.

[37] D. Gümüşbaş, T. Yıldırım, A. Genovese, and F. Scotti, "A comprehensive survey of databases and deep learning methods for cybersecurity and intrusion detection systems," *IEEE Systems Journal*, 2020.

[38] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki, "Network intrusion detection for iot security based on learning techniques," *IEEE Communications Surveys & Tutorials*, 2019.

[39] G. Singh and N. Khare, "A survey of intrusion detection from the perspective of intrusion datasets and machine learning techniques," *International Journal of Computers and Applications*, pp. 1–11, 2021.

[40] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[41] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *Journal of Information Security and Applications*, vol. 50, p. 102 419, 2020.

[42] S. A. Rahman, H. Tout, C. Talhi, and A. Mourad, "Internet of things intrusion detection: Centralized, on-device, or federated learning?" *IEEE Network*, 2020.

[43] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, "Dïot: A federated self-learning anomaly detection system for iot," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, IEEE, 2019, pp. 756–767.

[44] S. Kim, H. Cai, C. Hua, P. Gu, W. Xu, and J. Park, "Collaborative anomaly detection for internet of things based on federated learning," in *2020 IEEE/CIC International Conference on Communications in China (ICCC)*, IEEE, 2020, pp. 623–628.

[45] X. Wang, S. Garg, H. Lin, *et al.*, "Towards accurate anomaly detection in industrial internet-of-things using hierarchical federated learning," *IEEE Internet of Things Journal*, 2021.

[46] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*, PMLR, 2017, pp. 1273–1282.

[47] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, and H. V. Poor, "Federated learning for internet of things: A comprehensive survey," *IEEE Communications Surveys Tutorials*, pp. 1–1, 2021. DOI: 10.1109/COMST.2021.3075439.

[48] O. A. Wahab, A. Mourad, H. Otrok, and T. Taleb, "Federated machine learning: Survey, multi-level classification, desirable criteria and future directions in communication and networking systems," *IEEE Communications Surveys Tutorials*, early access, 10.1109/COMST.2021.3058573, 2021. DOI: 10.1109/COMST.2021.3058573.

[49] W. Y. B. Lim, N. C. Luong, D. T. Hoang, *et al.*, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys Tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020. DOI: 10.1109/COMST.2020.2986024.

[50] S. I. Popoola, R. Ande, K. B. Fatai, and B. Adebisi, "Deep bidirectional gated recurrent unit for botnet detection in smart homes," *Machine Learning and Data Mining for Emerging Trend in Cyber Dynamics: Theories and Applications*, p. 29, 2021.

[51] S. I. Popoola, B. Adebisi, M. Hammoudeh, H. Gacanin, and G. Gui, "Stacked recurrent neural network for botnet detection in smart homes," *Computers & Electrical Engineering*, vol. 92, p. 107 039, 2021.

[52] S. I. Popoola, B. Adebisi, R. Ande, M. Hammoudeh, K. Anoh, and A. A. Atayero, "Smote-drnn: A deep learning algorithm for botnet detection in the internet-of-things networks," *Sensors*, vol. 21, no. 9, p. 2985, 2021.

[53] S. I. Popoola, B. Adebisi, M. Hammoudeh, G. Gui, and H. Gacanin, "Hybrid deep learning for botnet attack detection in the internet-of-things networks," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4944–4956, 2021. DOI: 10.1109/JIOT.2020.3034156.

[54] S. I. Popoola, B. Adebisi, R. Ande, M. Hammoudeh, and A. A. Atayero, "Memory-efficient deep learning for botnet attack detection in iot networks," *Electronics*, vol. 10, no. 9, p. 1104, 2021.

[55] S. I. Popoola, R. Ande, B. Adebisi, G. Gui, M. Hammoudeh, and O. Jogunola, "Federated deep learning for zero-day botnet attack detection in iot edge devices," *IEEE Internet of Things Journal*, vol. 9, no. 5, pp. 3930–3944, 2022. DOI: `10.1109/JIOT.2021.3100755`.

[56] S. I. Popoola, G. Gui, B. Adebisi, M. Hammoudeh, and H. Gacanin, "Federated deep learning for collaborative intrusion detection in heterogeneous networks," in *2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall)*, IEEE, 2021, pp. 1–6.

[57] G. Aceto, V. Persico, and A. Pescape, "A survey on information and communication technologies for industry 4.0: State-of-the-art, taxonomies, perspectives, and challenges," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3467–3501, 2019.

[58] J. M. Williams, R. Khanna, J. P. Ruiz-Rosero, *et al.*, "Weaving the wireless web: Toward a low-power, dense wireless sensor network for the industrial iot," *IEEE Microwave Magazine*, vol. 18, no. 7, pp. 40–63, 2017.

[59] H. Darvishi, D. Ciuonzo, E. R. Eide, and P. S. Rossi, "Sensor-fault detection, isolation and accommodation for digital twins via modular data-driven architecture," *IEEE Sensors Journal*, 2020.

[60] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE communications surveys & tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[61] Cisco, *Annual internet report (2018-2023)*, Last accessed: September 19, 2020, 2020. [Online]. Available: `https://www.cisco.com`.

[62] A. Holst, *Number of iot connected devices worldwide 2019-2030*, Accessed: 2021-02-20, 2021. [Online]. Available: `https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/`.

[63] Statista, *Global iot end-user spending worldwide 2017-2025*, Accessed: 2021-02-20, 2021. [Online]. Available: `https://www.statista.com/statistics/976313/global-iot-market-size/`.

[64] S. Walker-Roberts, M. Hammoudeh, O. Aldabbas, M. Aydin, and A. Dehghantanha, "Threats on the horizon: Understanding security threats in the era of cyber-physical systems," *The Journal of*

Supercomputing, Oct. 2019, ISSN: 1573-0484. DOI: 10.1007/s11227-019-03028-9. [Online]. Available: https://doi.org/10.1007/s11227-019-03028-9.

[65] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani, "Demystifying iot security: An exhaustive survey on iot vulnerabilities and a first empirical look on internet-scale iot exploitations," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2702–2733, 2019.

[66] G. Vormayr, T. Zseby, and J. Fabini, "Botnet communication patterns," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2768–2796, 2017.

[67] S. S. Silva, R. M. Silva, R. C. Pinto, and R. M. Salles, "Botnets: A survey," *Computer Networks*, vol. 57, no. 2, pp. 378–403, 2013.

[68] S. Khattak, N. R. Ramay, K. R. Khan, A. A. Syed, and S. A. Khayam, "A taxonomy of botnet behavior, detection, and defense," *IEEE communications surveys & tutorials*, vol. 16, no. 2, pp. 898–924, 2013.

[69] J. B. Grizzard, V. Sharma, C. Nunnery, B. B. Kang, and D. Dagon, "Peer-to-peer botnets: Overview and case study.," *HotBots*, vol. 7, no. 2007, 2007.

[70] G. Macesanu, T. Codas, C. Suliman, and B. Tarnauca, "Development of gtbot, a high performance and modular indoor robot," in *2010 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*, IEEE, vol. 1, 2010, pp. 1–6.

[71] L. McLaughlin, "Bot software spreads, causes new worries," *IEEE Distributed Systems Online*, vol. 5, no. 6, p. 1, 2004.

[72] T. Holz, "A short visit to the bot zoo [malicious bots software]," *IEEE Security & Privacy*, vol. 3, no. 3, pp. 76–79, 2005.

[73] A. Marzano, D. Alexander, O. Fonseca, *et al.*, "The evolution of bashlite and mirai iot botnets," in *2018 IEEE Symposium on Computers and Communications (ISCC)*, IEEE, 2018, pp. 00 813–00 818.

[74] Y. Meidan, M. Bohadana, Y. Mathov, *et al.*, "N-baiot: Network-based detection of iot botnet attacks using deep autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, 2018. DOI: 10.1109/MPRV.2018.03367731.

[75] M. Jelasity, V. Bilicki, *et al.*, "Towards automated detection of peer-to-peer botnets: On the limits of local approaches.," *LEET*, vol. 9, p. 3, 2009.

[76] P. Wang, S. Sparks, and C. C. Zou, "An advanced hybrid peer-to-peer botnet," *IEEE Transactions on Dependable and Secure Computing*, vol. 7, no. 2, pp. 113–127, 2008.

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

[78] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization.," in *ICISSP*, 2018, pp. 108–116.

[79] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, IEEE, 2009, pp. 1–6.

[80] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Towards generating real-life datasets for network intrusion detection.," *IJ Network Security*, vol. 17, no. 6, pp. 683–701, 2015.

[81] N. Moustafa and J. Slay, "Unsw-nb15: A comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 military communications and information systems conference (MilCIS)*, IEEE, 2015, pp. 1–6.

[82] P. Gogoi, M. H. Bhuyan, D. Bhattacharyya, and J. K. Kalita, "Packet and flow based network intrusion dataset," in *International Conference on Contemporary Computing*, Springer, 2012, pp. 322–334.

[83] E. Alomari, S. Manickam, B. Gupta, P. Singh, and M. Anbar, "Design, deployment and use of http-based botnet (hbb) testbed," in *16th International Conference on Advanced Communication Technology*, IEEE, 2014, pp. 1265–1269.

[84] C. Livadas, R. Walsh, D. Lapsley, and W. T. Strayer, "Usilng machine learning technliques to identify botnet traffic," in *Proceedings. 2006 31st IEEE Conference on Local Computer Networks*, IEEE, 2006, pp. 967–974.

[85] S. Bhatia, D. Schmidt, G. Mohay, and A. Tickle, "A framework for generating realistic traffic for distributed denial-of-service attacks and flash events," *computers & security*, vol. 40, pp. 95–107, 2014.

[86] S. Behal and K. Kumar, "Detection of ddos attacks and flash events using information theory metrics–an empirical investigation," *Computer Communications*, vol. 103, pp. 18–28, 2017.

[87] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset," *Future Generation Computer Systems*, vol. 100, pp. 779–796, 2019.

[88] G. Apruzzese, M. Andreolini, M. Marchetti, A. Venturi, and M. Colajanni, "Deep reinforcement adversarial learning against botnet evasion attacks," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 1975–1987, 2020.

[89] R. Zhao, J. Yin, Z. Xue, *et al.*, "An efficient intrusion detection method based on dynamic autoencoder," *IEEE Wireless Communications Letters*, 2021.

[90] M. Shafiq, Z. Tian, Y. Sun, X. Du, and M. Guizani, "Selection of effective machine learning algorithm and bot-iot attacks traffic identification for internet of things in smart city," *Future Generation Computer Systems*, vol. 107, pp. 433–442, 2020.

[91] M. A. Ferrag, L. Maglaras, A. Ahmim, M. Derdour, and H. Janicke, "Rdtids: Rules and decision tree-based intrusion detection system for internet-of-things networks," *Future Internet*, vol. 12, no. 3, p. 44, 2020.

[92] N. Koroniotis, N. Moustafa, and E. Sitnikova, "A new network forensic framework based on deep learning for internet of things networks: A particle deep framework," *Future Generation Computer Systems*, 2020.

[93] O. Ibitoye, O. Shafiq, and A. Matrawy, "Analyzing adversarial attacks against deep learning for intrusion detection in iot networks," *arXiv preprint arXiv:1905.05137*, 2019.

[94] B. Susilo and R. F. Sari, "Intrusion detection in iot networks using deep learning algorithm," *Information*, vol. 11, no. 5, p. 279, 2020.

[95] M. A. Ferrag and L. Maglaras, "Deepcoin: A novel deep learning and blockchain-based energy exchange framework for smart grids," *IEEE Transactions on Engineering Management*, vol. 67, no. 4, pp. 1285–1297, 2020. DOI: `10.1109/TEM.2019.2922936`.

[96] O. Alkadi, N. Moustafa, B. Turnbull, and K.-K. R. Choo, "A deep blockchain framework-enabled collaborative intrusion detection for protecting iot and cloud networks," *IEEE Internet of Things Journal*, 2020.

[97] M. Ge, X. Fu, N. Syed, Z. Baig, G. Teo, and A. Robles-Kelly, "Deep learning-based intrusion detection for iot networks," in *2019 IEEE*

24th Pacific Rim International Symposium on Dependable Computing (PRDC), IEEE, 2019, pp. 256–25 609.

[98] D. Yan, Y. Yang, and B. Li, "An improved fuzzy classifier for imbalanced data," *Journal of Intelligent & Fuzzy Systems*, vol. 32, no. 3, pp. 2315–2325, 2017.

[99] C. Gui, "Analysis of imbalanced data set problem: The case of churn prediction for telecommunication.," *Artif. Intell. Research*, vol. 6, no. 2, p. 93, 2017.

[100] A. Fernández, S. Garcıa, M. J. del Jesus, and F. Herrera, "A study of the behaviour of linguistic fuzzy rule based classification systems in the framework of imbalanced data-sets," *Fuzzy Sets and Systems*, vol. 159, no. 18, pp. 2378–2398, 2008.

[101] E. Lin, Q. Chen, and X. Qi, "Deep reinforcement learning for imbalanced classification," *Applied Intelligence*, pp. 1–15, 2020.

[102] M. A. Ferrag and L. Maglaras, "Deepcoin: A novel deep learning and blockchain-based energy exchange framework for smart grids," *IEEE Transactions on Engineering Management*, 2019.

[103] I. Idrissi, M. Boukabous, M. Azizi, O. Moussaoui, and H. El Fadili, "Toward a deep learning-based intrusion detection system for iot against botnet attacks," *IAES International Journal of Artificial Intelligence*, vol. 10, no. 1, p. 110, 2021.

[104] S. Pokhrel, R. Abbas, and B. Aryal, "Iot security: Botnet detection in iot using machine learning," *arXiv preprint arXiv:2104.02231*, 2021.

[105] S. Bagui and K. Li, "Resampling imbalanced data for network intrusion detection datasets," *Journal of Big Data*, vol. 8, no. 1, pp. 1–41, 2021.

[106] R. Qaddoura, A. Al-Zoubi, I. Almomani, and H. Faris, "A multi-stage classification approach for iot intrusion detection based on clustering with oversampling," *Applied Sciences*, vol. 11, no. 7, p. 3022, 2021.

[107] A. Derhab, A. Aldweesh, A. Z. Emam, and F. A. Khan, "Intrusion detection system for internet of things based on temporal convolution neural network and efficient feature engineering," *Wireless Communications and Mobile Computing*, vol. 2020, 2020.

[108] Y. Wang, L. Guo, Y. Zhao, *et al.*, "Distributed learning for automatic modulation classification in edge devices," *IEEE Wireless Communications Letters*, 2020.

[109] R. E. Bellman, *Adaptive control processes: a guided tour*. Princeton university press, 2015.

[110] F. Luo, B. Du, L. Zhang, L. Zhang, and D. Tao, "Feature learning using spatial-spectral hypergraph discriminant analysis for hyperspectral image," *IEEE transactions on cybernetics*, vol. 49, no. 7, pp. 2406–2419, 2018.

[111] J. Peng, W. Sun, and Q. Du, "Self-paced joint sparse representation for the classification of hyperspectral images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 2, pp. 1183–1194, 2018.

[112] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[113] T. Hofmann, B. Schölkopf, and A. J. Smola, "Kernel methods in machine learning," *The annals of statistics*, pp. 1171–1220, 2008.

[114] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[115] Y. Bengio *et al.*, "Learning deep architectures for ai," *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[116] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41–50, 2018.

[117] F. A. Khan, A. Gumaei, A. Derhab, and A. Hussain, "A novel two-stage deep learning model for efficient network intrusion detection," *IEEE Access*, vol. 7, pp. 30 373–30 385, 2019.

[118] Y. Xiao, C. Xing, T. Zhang, and Z. Zhao, "An intrusion detection model based on feature reduction and convolutional neural networks," *IEEE Access*, vol. 7, pp. 42 210–42 219, 2019.

[119] A. Dawoud, S. Shahristani, and C. Raun, "Dimensionality reduction for network anomalies detection: A deep learning approach," in *Workshops of the International Conference on Advanced Information Networking and Applications*, Springer, 2019, pp. 957–965.

[120] F. C. Schuartz, M. Fonseca, and A. Munaretto, "Improving threat detection in networks using deep learning," *Annals of Telecommunications*, pp. 1–10, 2020.

[121] J. Sun, X. Wang, N. Xiong, and J. Shao, "Learning sparse representation with variational auto-encoder for anomaly detection," *IEEE Access*, vol. 6, pp. 33 353–33 361, 2018.

[122] X. Wang and L. Wang, "Research on intrusion detection based on feature extraction of autoencoder and the improved k-means algorithm," in *2017 10th International Symposium on Computational Intelligence and Design (ISCID)*, IEEE, vol. 2, 2017, pp. 352–356.

[123] M. Z. Alom and T. M. Taha, "Network intrusion detection for cyber security using unsupervised deep learning approaches," in *2017 IEEE National Aerospace and Electronics Conference (NAECON)*, IEEE, 2017, pp. 63–69.

[124] M. Al-Qatf, Y. Lasheng, M. Al-Habib, and K. Al-Sabahi, "Deep learning approach combining sparse autoencoder with svm for network intrusion detection," *IEEE Access*, vol. 6, pp. 52 843–52 856, 2018.

[125] S. Gurung, M. K. Ghose, and A. Subedi, "Deep learning approach on network intrusion detection system using nsl-kdd dataset," *International Journal of Computer Network and Information Security (IJCNIS)*, vol. 11, no. 3, pp. 8–14, 2019.

[126] A.-H. Muna, N. Moustafa, and E. Sitnikova, "Identification of malicious activities in industrial internet of things based on deep learning models," *Journal of Information security and applications*, vol. 41, pp. 1–11, 2018.

[127] B. Abolhasanzadeh, "Nonlinear dimensionality reduction for intrusion detection using auto-encoder bottleneck features," in *2015 7th Conference on Information and Knowledge Technology (IKT)*, IEEE, 2015, pp. 1–5.

[128] D. C. Ferreira, F. I. Vázquez, and T. Zseby, "Extreme dimensionality reduction for network attack visualization with autoencoders," in *2019 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2019, pp. 1–10.

[129] R. Abdulhammed, H. Musafer, A. Alessa, M. Faezipour, and A. Abuzneid, "Features dimensionality reduction approaches for machine learning based network intrusion detection," *Electronics*, vol. 8, no. 3, p. 322, 2019.

[130] J. Yang, T. Li, G. Liang, W. He, and Y. Zhao, "A simple recurrent unit model based intrusion detection system with dcgan," *IEEE Access*, vol. 7, pp. 83 286–83 296, 2019.

[131] A. Liu and B. Sun, "An intrusion detection system based on a quantitative model of interaction mode between ports," *IEEE Access*, vol. 7, pp. 161 725–161 740, 2019.

[132] R.-H. Dong, X.-Y. Li, Q.-Y. Zhang, and H. Yuan, "Network intrusion detection model based on multivariate correlation analysis–long short-time memory network," *IET Information Security*, vol. 14, no. 2, pp. 166–174, 2020.

[133] M. Asadi, M. A. J. Jamali, S. Parsa, and V. Majidnezhad, "Detecting botnet by using particle swarm optimization algorithm based on voting system," *Future Generation Computer Systems*, vol. 107, pp. 95–111, 2020.

[134] A. Khraisat, I. Gondal, P. Vamplew, J. Kamruzzaman, and A. Alazab, "A novel ensemble of hybrid intrusion detection system for detecting internet of things attacks," *Electronics*, vol. 8, no. 11, p. 1210, 2019.

[135] Y. N. Soe, Y. Feng, P. I. Santosa, R. Hartanto, and K. Sakurai, "Towards a lightweight detection system for cyber attacks in the iot environment using corresponding features," *Electronics*, vol. 9, no. 1, p. 144, 2020.

[136] ——, "Rule generation for signature based detection systems of cyber attacks in iot environments," *Bulletin of Networking, Computing, Systems, and Software*, vol. 8, no. 2, pp. 93–97, 2019.

[137] A. Guerra-Manzanares, H. Bahsi, and S. Nõmm, "Hybrid feature selection models for machine learning based botnet detection in iot networks," in *2019 International Conference on Cyberworlds (CW)*, IEEE, 2019, pp. 324–327.

[138] S. Lee, A. Abdullah, N. Jhanjhi, and S. Kok, "Classification of botnet attacks in iot smart factory using honeypot combined with machine learning," *PeerJ Computer Science*, vol. 7, e350, 2021.

[139] Y. Zhang, J. Xu, Z. Wang, *et al.*, "Efficient and intelligent attack detection in software defined iot networks," in *2020 IEEE International Conference on Embedded Software and Systems (ICESS)*, IEEE, 2020, pp. 1–9.

[140] K. Filus, J. Domańska, and E. Gelenbe, "Random neural network for lightweight attack detection in the iot," in *Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems*, Springer, 2020, pp. 79–91.

[141] M. A. Lawal, R. A. Shaikh, and S. R. Hassan, "An anomaly mitigation framework for iot using fog computing," *Electronics*, vol. 9, no. 10, p. 1565, 2020.

[142] D. Oreški and D. Andročec, "Genetic algorithm and artificial neural network for network forensic analytics," in *2020 43rd International*

Convention on Information, Communication and Electronic Technology (MIPRO), IEEE, 2020, pp. 1200–1205.

[143] B. A. NG and S. Selvakumar, "Anomaly detection framework for internet of things traffic using vector convolutional deep learning approach in fog environment," *Future Generation Computer Systems*, vol. 113, pp. 255–265, 2020.

[144] P. Kumar, G. P. Gupta, and R. Tripathi, "Toward design of an intelligent cyber attack detection system using hybrid feature reduced approach for iot networks," *Arabian Journal for Science and Engineering*, pp. 1–30, 2021.

[145] P. Kumar, R. Kumar, G. P. Gupta, and R. Tripathi, "A distributed framework for detecting ddos attacks in smart contract-based blockchain-iot systems by leveraging fog computing," *Transactions on Emerging Telecommunications Technologies*, e4112, 2020.

[146] M. Shafiq, Z. Tian, A. K. Bashir, X. Du, and M. Guizani, "Corrauc: A malicious bot-iot traffic detection method in iot network using machine learning techniques," *IEEE Internet of Things Journal*, 2020.

[147] ——, "Iot malicious traffic identification using wrapper-based feature selection mechanisms," *Computers & Security*, p. 101 863, 2020.

[148] T. T. Huong, T. P. Bac, D. M. Long, B. D. Thang, T. D. Luong, and N. T. Binh, "An efficient low complexity edge-cloud framework for security in iot networks," in *2020 IEEE Eighth International Conference on Communications and Electronics (ICCE)*, IEEE, 2021, pp. 533–539.

[149] P. Kumar, G. P. Gupta, and R. Tripathi, "Tp2sf: A trustworthy privacy-preserving secured framework for sustainable smart cities by leveraging blockchain and machine learning," *Journal of Systems Architecture*, p. 101 954, 2020.

[150] M. Alshamkhany, W. Alshamkhany, M. Mansour, M. Khan, S. Dhou, and F. Aloul, "Botnet attack detection using machine learning," in *2020 14th International Conference on Innovations in Information Technology (IIT)*, IEEE, 2020, pp. 203–208.

[151] S. Sriram, R. Vinayakumar, M. Alazab, and K. Soman, "Network flow based iot botnet attack detection using deep learning," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, IEEE, 2020, pp. 189–194.

[152] S. Liaqat, A. Akhunzada, F. S. Shaikh, A. Giannetsos, and M. A. Jan, "Sdn orchestration to combat evolving cyber threats in internet of

medical things (iomt)," *Computer Communications*, vol. 160, pp. 697–705, 2020.

[153] Y. N. Soe, P. I. Santosa, and R. Hartanto, "Ddos attack detection based on simple ann with smote for iot environment," in *2019 Fourth International Conference on Informatics and Computing (ICIC)*, 2019, pp. 1–5. DOI: `10.1109/ICIC47613.2019.8985853`.

[154] M. Mulyanto, M. Faisal, S. W. Prakosa, and J.-S. Leu, "Effectiveness of focal loss for minority classification in network intrusion detection systems," *Symmetry*, vol. 13, no. 1, p. 4, 2021.

[155] M. A. Khan and Y. Kim, "Deep learning-based hybrid intelligent intrusion detection system," *Computers, Materials & Continua*, vol. 68, no. 1, pp. 671–687, 2021.

[156] M. Roopak, G. Yun Tian, and J. Chambers, "Deep learning models for cyber security in iot networks," in *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, 2019, pp. 0452–0457. DOI: `10.1109/CCWC.2019.8666588`.

[157] M. Injadat, A. Moubayed, and A. Shami, "Detecting botnet attacks in iot environments: An optimized machine learning approach," *arXiv preprint arXiv:2012.11325*, 2020.

[158] K. Yang, T. Jiang, Y. Shi, and Z. Ding, "Federated learning via over-the-air computation," *IEEE Transactions on Wireless Communications*, vol. 19, no. 3, pp. 2022–2035, 2020.

[159] B. Li, Y. Wu, J. Song, R. Lu, T. Li, and L. Zhao, "Deepfed: Federated deep learning for intrusion detection in industrial cyber–physical systems," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5615–5624, 2020.

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

[161] Y. Cheng, J. Lu, D. Niyato, B. Lyu, J. Kang, and S. Zhu, "Federated transfer learning with client selection for intrusion detection in mobile edge computing," *IEEE Communications Letters*, 2022.

[162] P. Kumar, G. P. Gupta, and R. Tripathi, "Pefl: Deep privacy-encoding based federated learning framework for smart agriculture," *IEEE Micro*, 2021.

[163] H. Sedjelmaci and N. Ansari, "On cooperative federated defense to secure multi-access edge computing," *IEEE Consumer Electronics Magazine*, 2022.

[164] D. C. Attota, V. Mothukuri, R. M. Parizi, and S. Pouriyeh, "An ensemble multi-view federated learning intrusion detection for iot," *IEEE Access*, vol. 9, pp. 117 734–117 745, 2021.

[165] Y. Sun, H. Esaki, and H. Ochiai, "Adaptive intrusion detection in the networking of large-scale lans with segmented federated learning," *IEEE Open Journal of the Communications Society*, 2020.

[166] M. Abdel-Basset, N. Moustafa, H. Hawash, I. Razzak, K. M. Sallam, and O. M. Elkomy, "Federated intrusion detection in blockchain-based smart transportation systems," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[167] M. A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, and H. Janicke, "Edge-iiotset: A new comprehensive realistic cyber security dataset of iot and iiot applications for centralized and federated learning," *IEEE Access*, 2022.

[168] Z. Chen, N. Lv, P. Liu, Y. Fang, K. Chen, and W. Pan, "Intrusion detection for wireless edge networks based on federated learning," *IEEE Access*, vol. 8, pp. 217 463–217 472, 2020.

[169] O. Aouedi, K. Piamrat, G. Muller, and K. Singh, "Federated semi-supervised learning for attack detection in industrial internet of things," *IEEE Transactions on Industrial Informatics*, 2022.

[170] P. Ruzafa-Alcazar, P. Fernandez-Saura, E. Marmol-Campos, *et al.*, "Intrusion detection based on privacy-preserving federated learning for the industrial iot," *IEEE Transactions on Industrial Informatics*, 2021.

[171] T. T. Huong, T. P. Bac, D. M. Long, *et al.*, "Lockedge: Low-complexity cyberattack detection in iot edge computing," *IEEE Access*, vol. 9, pp. 29 696–29 710, 2021.

[172] V. Mothukuri, P. Khare, R. M. Parizi, S. Pouriyeh, A. Dehghantanha, and G. Srivastava, "Federated learning-based anomaly detection for iot security attacks," *IEEE Internet of Things Journal*, 2021.

[173] Y. Wei, S. Zhou, S. Leng, S. Maharjan, and Y. Zhang, "Federated learning empowered end-edge-cloud cooperation for 5g hetnet security," *IEEE Network*, vol. 35, no. 2, pp. 88–94, 2021.

[174] A. Tharwat, "Principal component analysis-a tutorial," *International Journal of Applied Pattern Recognition*, vol. 3, no. 3, pp. 197–240, 2016.

[175] T. Zhang, C. He, T. Ma, L. Gao, M. Ma, and S. Avestimehr, "Federated learning for internet of things: A federated learning framework for on-device anomaly data detection," *arXiv preprint arXiv:2106.07976*, 2021.

[176] V. Rey, P. M. S. Sánchez, A. H. Celdrán, and G. Bovet, "Federated learning for malware detection in iot devices," *Computer Networks*, p. 108 693, 2022.

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

[178] Y. Chauvin and D. E. Rumelhart, *Backpropagation: theory, architectures, and applications*. Psychology press, 2013.

[179] X.-H. Yu and G.-A. Chen, "Efficient backpropagation learning using optimal learning rate and momentum," *Neural networks*, vol. 10, no. 3, pp. 517–527, 1997.

[180] L. Friedman and O. V. Komogortsev, "Assessment of the effectiveness of seven biometric feature normalization techniques," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 10, pp. 2528–2536, 2019.

[181] S. Patro and K. K. Sahu, "Normalization: A preprocessing stage," *arXiv preprint arXiv:1503.06462*, 2015.

[182] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[183] S.-i. Amari, "Backpropagation and stochastic gradient descent method," *Neurocomputing*, vol. 5, no. 4-5, pp. 185–196, 1993.

[184] M. C. Mukkamala and M. Hein, "Variants of rmsprop and adagrad with logarithmic regret bounds," in *International Conference on Machine Learning*, PMLR, 2017, pp. 2545–2553.

[185] M. D. Zeiler, "Adadelta: An adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[186] A. Lydia and S. Francis, "Adagrad—an optimizer for stochastic gradient descent," *Int. J. Inf. Comput. Sci*, vol. 6, no. 5, 2019.

[187] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Icml*, 2010.

[188] B. L. Kalman and S. C. Kwasny, "Why tanh: Choosing a sigmoidal function," in *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, IEEE, vol. 4, 1992, pp. 578–581.

[189] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Advances in neural information processing systems*, 2017, pp. 971–980.

[190] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289*, 2015.

[191] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[192] C. X. Ling and C. Li, "Data mining for direct marketing: Problems and solutions.," in *Kdd*, vol. 98, 1998, pp. 73–79.

[193] N. Japkowicz, "The class imbalance problem: Significance and strategies," in *Proc. of the International Conf. on Artificial Intelligence*, Citeseer, vol. 56, 2000.

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

[195] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," in *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, vol. 2, Sep. 1999, 850–855 vol.2. DOI: `10.1049/cp:19991218`.

[196] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," in *International Conference on Machine Learning*, 2015, pp. 2342–2350.

[197] K. Cho, B. Van Merriënboer, C. Gulcehre, *et al.*, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[198] F. Zhao, J. Feng, J. Zhao, W. Yang, and S. Yan, "Robust lstm-autoencoders for face de-occlusion in the wild," *IEEE Transactions on Image Processing*, vol. 27, no. 2, pp. 778–790, 2017.

[199] P. J. Werbos *et al.*, "Backpropagation through time: What it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[200] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural networks*, vol. 18, no. 5-6, pp. 602–610, 2005.

[201] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[202] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Communications Surveys Tutorials*, vol. 22, no. 2, pp. 869–904, 2020. DOI: `10.1109/COMST.2020.2970550`.

[203] J. Shu, L. Zhou, W. Zhang, X. Du, and M. Guizani, "Collaborative intrusion detection for vanets: A deep learning-based distributed sdn approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. early access, doi: 10.1109/TITS.2020.3027390, 2020. DOI: `10.1109/TITS.2020.3027390`.

[204] M. Zhang, L. Wang, S. Jajodia, A. Singhal, and M. Albanese, "Network diversity: A security metric for evaluating the resilience of networks against zero-day attacks," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 5, pp. 1071–1086, 2016.

[205] M. Yurochkin, M. Agarwal, S. Ghosh, K. Greenewald, N. Hoang, and Y. Khazaeni, "Bayesian nonparametric federated learning of neural networks," in *International Conference on Machine Learning*, PMLR, 2019, pp. 7252–7261.

[206] P. Yu, A. Kundu, L. Wynter, and S. H. Lim, "Fed+: A unified approach to robust personalized federated learning," *arXiv preprint arXiv:2009.06303*, 2020.

[207] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine Learning and Systems*, vol. 2, pp. 429–450, 2020.

[208] T. D. Nguyen, P. Rieger, M. Miettinen, and A.-R. Sadeghi, "Poisoning attacks on federated learning-based iot intrusion detection system," in *Proc. Workshop Decentralized IoT Syst. Secur.(DISS)*, 2020, pp. 1–7.

[209] S. K. Lo, Y. Liu, Q. Lu, *et al.*, "Towards trustworthy ai: Blockchain-based architecture design for accountability and fairness of federated learning systems," *IEEE Internet of Things Journal*, 2022.

[210] A. El Ouadrhiri and A. Abdelhadi, "Differential privacy for deep and federated learning: A survey," *IEEE Access*, 2022.