Applying AI-based Techniques for DDoS Anomaly Detection and Classification Using Large-scale Datasets

Yuanyuan Wei

A thesis submitted in partial fulfilment of the requirements for the degree of
Doctor of Philosophy (Ph.D.) in Computer Science,
Massey University, 2023.

Abstract

A Distributed Denial-of-Service (DDoS) attack is a type of malicious attempt to disrupt the normal traffic of a targeted server, service, or network by sending a flood of traffic to overwhelm the target or its surrounding infrastructure. DDoS attacks expose significant security vulnerabilities in network devices, allowing for malicious propagation. This presents serious security risks, including potential data loss and financial consequences. In order to identify and mitigate the impact of DDoS attacks, Artificial Intelligence (AI)-based techniques (e.g. machine learning or deep learning) can be deployed with the aim of improving decision-making in networked infrastructures to enhance reliability, interoperability, trust, security, and stability. Many of the studies that have deployed detection frameworks for DDoS attacks have suffered from the limitations of low detection rates, high false alarm rates, and a lack of scalability. In this context, it is important to apply AI-based techniques for classification and anomaly detection that can detect, prevent, and mitigate DDoS attacks.

This research focuses on studying the detection of DDoS attacks. Traditional shallow machine learning-based techniques for DDoS attack classification tend to be ineffective when the volume and features of network traffic, potentially carrying malicious DDoS payloads, increase exponentially as they cannot extract high-importance features automatically. To overcome the limitations in extracting high-importance features, we first investigate the classification of different DDoS attacks based on a hybrid deep learning technique that combines Autoencoder (AE) and Multi-Layer Perceptron (MLP). We propose a hybrid deep learning-based approach to extract the most important features and feed them into the classifier to obtain a multi-class classification of different DDoS attacks. Then, we provide a hybrid deep learning anomaly detection technique Long Short-Term Memory and Autoencoder (LSTM-AE) based on multivariate time series sequences, which can effectively detect potential DDoS attacks. We evaluate the effectiveness of the DDoS attack classification and anomaly detection. To evaluate whether the proposed hybrid deep learning-based anomaly detection is more promising, we apply the aforementioned hybrid deep learning-based LSTM-AE anomaly detection technique based on time series sequence analysis, to the real-world IoT sensor data (the IoT sensor dataset of Indoor Air Quality (IAQ) from SKOol MOnitoring BOx (SKOMOBO) units deployed on a large scale across the classrooms of primary schools in New Zealand). We demonstrate the proposed hybrid deep learning-based techniques that can effectively detect anomalies in the large-scale IoT dataset.

Finally, the outcomes of machine learning or deep learning performance lack transparency, posing challenges in both explaining the results to users and instilling trust in them. To address this issue, we propose a framework that can efficiently classify legitimate traffic and malicious traffic, and explain the decision-making of the machine learning/deep learning models by deploying Explainable Artificial Intelligence (XAI) techniques.

I would like to dedicate this thesis to my lovely parents for their love, motivation, and encouragement.

Acknowledgements

First and foremost, I would like to take this opportunity to express my deepest gratitude to all of you who have helped, guided, and supported me throughout my Ph.D. programme at Massey University.

I would like to express my sincere gratitude to my supervisor, Associate Professor Julian Jang-Jaccard, who has guided and supported me throughout my Ph.D. doctoral journey. I am very grateful to her for giving me the opportunity to study with her. She has always gently and patiently guided me in academic reading, critical thinking, and writing and guided me in the right direction of my research. Thanks to her for encouraging me to overcome the challenges when I encountered difficulties and struggles. I am very fortunate to have had her supervision. Her extensive expertise and patience have been invaluable to me in the completion of my doctoral thesis. I am deeply thankful to Julian for supporting me and providing me with financial support for my Ph.D. I have greatly appreciated my great supervisor Julian.

I would also like to extend my gratitude to my co-supervisors Dr. Fariza Sabrina and Professor Ruili Wang, who have been instrumental in helping me with my research. It was very helpful to receive feedback and suggestions on my writing. With their support and knowledge sharing, I have gained a lot of great progress.

I would also like to extend my gratitude to my co-collaborator Seyit Camtepe, who provided me with many constructive suggestions, and ideas.

I am deeply thankful to my colleague in the Cybersecurity Lab, Dr. Hooman Alavizadeh, who has also been helped me to overcome many concerns and challenges. He has enthusiastically shared his knowledge and experience with me, which I have greatly appreciated.

I would also like to thank Dr. Timothy McIntosh, who has shared his knowledge and showed me the way to start when I started my Ph.D. life. I would like to thank Xu Wen and Amerdeep for many discussions and suggestions on the domain knowledge. I have learned a lot from them.

I would like to express my grateful acknowledge to Massey University for providing me with the opportunity to conduct my research at the School of Mathematical and

Computational Sciences and for supporting and providing the resources.

I would like to express my great appreciation to the people with whom I have collaborated and made friends within the School of Mathematical and Computational Sciences and the Cybersecurity Lab at Massey University: Sibghat, Libby, Jinting, Mehmood, Aeryn, Song Jian.

I am grateful to everyone who has supported me throughout this journey. I am also fortunate to have the company of many wonderful friends in New Zealand. I would like to thank my lovely New Zealand mother, Angeline Tuscher, who gave me so much love and support when I first came to New Zealand and made me feel at home in New Zealand. I would like to thank Frank for his insightful advice on domain knowledge and for helping me to address my concerns regarding challenges. I would also like to thank Jessie, Dan, Cathy, Rachel, Yufei, and Leo for all the happy times we have spent together.

Last but not least, I would like to give many thanks to my family, especially my dear father and mother for their love and support. I would like to thank my parents for everything. My parents have always been the backbone of my life and have taught me to be an independent, kind, and confident person. We share every moment of our lives like friends. I will never forget the happy time that we spent traveling to many places together. Without their encouragement and support, I would not have been able to complete this journey. Finally, my parents in my mind, are the most amazing parents in the world.

Publications Arising from this Thesis

The core of this thesis was based on the following peer-reviewed journals and conferences.

- Y. Wei, J. Jang-Jaccard, F. Sabrina, A. Singh, W. Xu and S. Camtepe, "AE-MLP: A Hybrid Deep Learning Approach for DDoS Detection and Classification," in IEEE Access, vol. 9, pp. 146810-146821, 2021, doi: 10.1109/ACCESS.2021.3123791.

- Y. Wei, J. Jang-Jaccard, F. Sabrina, S. Camtepe and A. Dunmore. *"Reconstruction-based LSTM-Autoencoder for Anomaly-based DDoS Attack Detection over Multivariate Time-Series Data" Submitted to IEEE Transactions on Network and Service Management. (Under review)*

- Y. Wei, J. Jang-Jaccard, W. Xu, F. Sabrina, S. Camtepe and M. Boulic, "LSTM-Autoencoder-Based Anomaly Detection for Indoor Air Quality Time-Series Data," in IEEE Sensors Journal, vol. 23, no. 4, pp. 3787-3800, 15 Feb.15, 2023, doi: 10.1109/JSEN.2022.3230361.

- Y. Wei, J. Jang-Jaccard, A. Singh, F. Sabrina, S. Camtepe. *"Classification and Explanation of Distributed Denial-of-Service (DDoS) Attack Detection using Machine Learning and Shapley Additive Explanation (SHAP) Methods"* Submitted to ArXiv.

- Y. Wei, J. Jang-Jaccard, F. Sabrina and H. Alavizadeh, "Large-Scale Outlier Detection for Low-Cost PM_{10} Sensors," in IEEE Access, vol. 8, pp. 229033-229042, 2020, doi: 10.1109/ACCESS.2020.3043421.

- Y. Wei, J. Jang-Jaccard, F. Sabrina and T. McIntosh, "MSD-Kmeans: A Hybrid Algorithm for Efficient Detection of Global and Local Outliers," 2021 IEEE 15th International Conference on Big Data Science and Engineering (BigDataSE), Shenyang, China, 2021, pp. 87-94, doi: 10.1109/BigDataSE53435.2021.00022.

Contents

List of Figures

List of Tables

List of Acronyms

AE	Autoencoder
AI	Artificial Intelligence
ARIMA	Autoregressive Integrated Moving Average
Acc	Accuracy
CNN	Convolutional Neural Network
CO	Carbon Monoxide
DNS	Domain Name System
DL	Deep Learning
FTP	File Transfer Protocol
F1	F1- score
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IP	Internet Protocol
IoT	Internet of Things
IAQ	Indoor Air Quality
KNN	K-nearest Neighbours
LSTM	Long Short-Term Memory
LSTM-AE	Long Short-Term Memory Autoencoder
ML	Machine Learning
MLP	Multilayer Perceptron
NTP	Network Time Protocol
NO2	Nitrogen Dioxide

NIWA	National Institute of Water and Atmospheric Research
OCSVM	One-Class Support Vector Machines
OECD	Organisation for Economic Co-operation and Development
PCA	Principal Component Analysis
Pre	Precision
PM	Particulate Matter
RF	Random Forest
Re	Recall
SSDP	Simple Service Discovery Protocol
SNMP	Simple Network Management Protocol
SVDD	Support Vector Data Description
SSH	Secure Socket Shell
SKOMOBO	SKOol MOnitoring BOx
TCP	Transmission Control Protocol
TFTP	Trivial File Transfer Protocol
XGBoost	Extreme Gradient Boosting

Chapter 1

Preface

A Distributed Denial-of-Service (DDoS) attack occurs when attackers make victims' system/network resources unavailable by sending massive amounts of requests to flood the system resources/bandwidth of the victim's system [14, 50]. DDoS attackers exploit vulnerabilities in the transport, network, and application layers (e.g., TCP, UDP, HTTP, and ICMP, etc.) to send malicious payloads (e.g., network packets). These malicious payloads are also referred to as botnet attacks - command and control infrastructure, which is a large number of malicious network-infected devices that are remotely controlled [43,74,99]. Typically, attackers control these compromised botnets (or a series of bots) utilizing TCP or UDP-based protocols at the application layer and launch a large number of HTTP requests from the source IP to the victim's IP address over the Internet, overwhelming the server and resulting in a DDoS, thereby making it unavailable for legitimate users. TCP-based attacks include MSSQL and SSSP attack types, while UDP-based attacks include NTP and TFTP. Certain attacks use both TCP and UDP protocols, including DNS, LDAP, NetBIOS, and SNMP [74]. A protocol attack attempts to overload the capacity of the network by exploiting a weakness in the transport and network layers, causing the target servers to become disrupted or inaccessible. These attacks can also be carried out via application layer protocols using TCP and UDP, including SYN flood, UDP flood, and UDP-lag [74]. Attacks launched from the transport and network layers are easy to detect because they have clear signatures, whereas application layer attacks are more sophisticated and difficult to detect [1].

DDoS attacks are not only conducted against online services, web applications, and information infrastructure to cause downtime but also to prevent legitimate users from purchasing products and using online services - such as emails, websites, and applications - and affecting program performance [66]. As a result of the COVID-19 lockdown in 2020, there has been an increase in attacks on education, online shopping, and office work, as a large number of people are now studying, working, and shopping online, giving hackers greater opportunities [6]. In [48] Azure Networking found there was a 25% increase in DDoS attacks in the first six months of 2021 when compared with the fourth quarter of

2020. Moreover, Azure mitigated approximately 35 thousand attacks against its global infrastructure in the last six months of 2021, which increased from 43% compared with the first six months of 2021. A white paper from Cisco [17] predicted that nearly 300 billion mobile applications would be downloaded by 2023 and that DDoS attacks would rise to 15.4 million globally by 2023. With the demand for Internet connectivity expanding rapidly to mobile devices and the Internet of Things (IoT) which is predicted to reach 500 billion by the end of 2030 [103], there is an increasing concern for developing effective DDoS defense techniques.

DDoS detection is becoming an urgent need because of the sophistication and diversification of attacks. For instance, difficult-to-track attackers and unknown or new types of attacks occur continuously (e.g., zero-day attacks) [2, 96]. Detecting DDoS attacks is becoming increasingly difficult not only because a large proportion of attack traffic is similar to legitimate traffic but also because of newer hybrid attack methods [20, 99]. Therefore, the detection and mitigation of DDoS attacks not only protects the network for legitimate users but also reduces financial loss for businesses [68]. To address this concern, many Artificial Intelligence (AI) methods, both using traditional statistic-based, machine learning-based and more advanced deep neural network-based, have been proposed in recent years to demonstrate the feasibility of such AI-based approaches to safeguard the networks from DDoS attacks. To detect and mitigate DDoS attacks, statistic-based and machine learning-based approaches have been proposed for signature thresholding methods to identify and discriminate DDoS attacks [9, 49, 84]. However, most traditional statistical and machine learning-based detection approaches require better-selected features, defined thresholds, and difficult to detect previously unseen DDoS attacks [20, 99]. In contrast to traditional detection techniques, deep learning-based DDoS attack detection - such as CNN, AE, or RNN and others - can provide better detection rates for DDoS network traffic, compared to traditional statistical and ML-based techniques [99]. However, some limitations of existing deep learning-based detection need to be addressed, AE models are sensitive to the anomalies in the training phase, and RNNs can better address historical sequence data, but face the shortcomings of the vanishing gradient problem, etc. In this context, research questions provide in this thesis to address the aforementioned problems using a state-of-the-art large-scale DDoS dataset

- Q1: What is it about the DDoS attacks that require attention in a detection strategy?
- Q2: What are the main influencing factors about datasets that affect DDoS detection rate?
- Q3: What are the strategies for adopting artificial intelligence (AI) that can better detect DDoS attacks?
- Q4: How we can explain the decision-making behind the model when the model has made a decision and obtained a result?

The main goals of this research are to evaluate DDoS attack classification and anomaly detection by adopting hybrid AI-based Deep learning techniques. The main contributions are as follows:

- We propose a hybrid deep learning model named "AE-MLP" not only to detect DDoS attacks but also to classify the attack into different DDoS attack types. This classification capability can provide an opportunity for cybersecurity professionals to devise an optimal and relevant response strategy as quickly as possible before disastrous damage is done by different DDoS attack types. The proposed model uses an Autoencoder (AE) to extract the most important features from a large-scale DDoS attack dataset. Finding the set of correlated, compressed, and reduced feature sets not only improves the accuracy of detection but can also effectively reduce expensive execution time. Our experimental results, comprehensively and extensively evaluated, demonstrate a very high detection rate for detecting DDoS attacks and classifying them into correct attack types.

- We propose a novel time-series anomaly detection architecture that leverages reconstruction-based LSTM-AE for efficient DDoS attack detection. The LSTM networks are to learn the long short-term correlation of data within a time series sequence while autoencoder is used to identify the optimal threshold based on the reconstruction error rates evaluated across all time-series sequences. The model is trained on normal time-based traffic flow features using a subset of traffic flow information over a fixed-time window length. An anomaly score (MAE value) of each traffic flow is computed, which can be calculated flexibly based on different fixed-time window lengths. We demonstrate that our proposed model is an effective and promising method for detecting anomalies in a timely manner.

- We provide a hybrid deep learning model that exploits the capacity of LSTM and Autoencoder (AE). Our model uses LSTM networks, which consist of multiple LSTM units interacting with each other, to learn the long-term dependencies of multiple indoor air data points within a time series sequence. In addition, our model uses Autoencoder (AE) to reduce data dimension for more efficient training and also computes an optimal threshold associated with each time-series sequence. Together, our model is also suitable for detecting contextual anomalies of real-world IoT sensor data (the IoT sensor dataset of Indoor Air Quality (IAQ) from SKOol MOnitoring BOx (SKOMOBO) units deployed on a large scale across the classrooms of primary schools in New Zealand). We demonstrate the proposed hybrid deep learning-based techniques that can effectively detect anomalies in the large-scale IoT dataset.

- We propose a novel XAI-based SHAP explanation framework to explain the classification result in DDoS attack detection, which consists of two components: DDoS classification using MLP classifier and XAI-based technique of SHAP to explain

the most contributing features of the classification of the benign and attack traffic. Our XAI-based explanation framework consists of two types of explanation: global and local explanation. The global explanation we conduct is based on summary plots and dependence plots. For the local explanation, we focus the analysis on a single traffic in four cases, which are benign and malicious traffic correctly classified, and misclassified traffic. This work helps the users to trust and have a better understanding of the prediction results of the proposed model.

The organization of this thesis is as follows.

Chapter 2 in this thesis presents the investigation of the classification of different attack types of DDoS attacks. To overcome the limitations in extracting highly significant features, we propose a hybrid approach named AE-MLP that combines two deep learning-based models for effective detection and classification of DDoS attacks. Our proposed model uses an Autoencoder (AE) to extract the most important features from a large-scale DDoS attack dataset. Finding the most relevant compressed and reduced feature sets to detect the classification of different DDoS attack types by the MLP classifier. Our experimental results demonstrate high and robust performance in DDoS classification (The work has been published in [89]).

In Chapter 3, we propose a reconstruction-based anomaly detection model named LSTM-Autoencoder (LSTM-AE) which combines two deep learning-based models for detecting multivariate time series anomalies in DDoS attacks. Our model is trained on normal time-based traffic flow features using a subset of traffic flow information over a fixed time window length. The largest computed MAE value from the training set is used as the threshold to identify the anomalies. In this study, we apply three types of DDoS attacks - DNS, LDAP, and SNMP - to effectively detect multivariable time series anomalies.

In Chapter 4, to evaluate whether the proposed hybrid deep learning-based anomaly detection is more promising, we present an extensive evaluation of the aforementioned hybrid deep learning-based LSTM-AE anomaly detection technique based on time-series sequence analysis, on the real-world IoT sensor data (the Indoor Air Quality IoT sensor dataset from SKOMOBO units deployed on a large scale across the classrooms of primary schools in New Zealand). We use an important feature value of the CO_2 time-series sequence data as the detecting data. We compare the evaluation in different time window lengths and demonstrate that the proposed model can effectively detect anomalies (This work has been published in [91]).

In Chapter 5, we propose a novel XAI-based SHAP interpretation framework to explain the model's classification decision, which consists of two components: DDoS classification and explanation. DDoS classification is the use of an MLP classifier to classify benign and malicious traffic, while the XAI-based technique of SHAP for explanation is to explain the most contributing features of the classification of the benign and attack

traffic. The decision explanation of the model is based on global and local explanations. The global explanation uses the SHAP value to illustrate how much each feature contributes globally to the prediction results. The local explanation is based on a single data sample and is also known to explain individual predictions. We conducted an extensive evaluation using the CICDDoS2019 dataset to find the most corresponding features, with the explanation attached as the best exploration of the most contributing features.

In Chapter 6, we conclude this thesis and introduce future work directions.

Chapter 2

AE-MLP: A Hybrid Deep Learning Approach for DDoS Detection and Classification

2.1 Abstact

Distributed Denial-of-Service (DDoS) attacks are increasing as the demand for Internet connectivity massively grows in recent years. Conventional shallow machine learning-based techniques for DDoS attack classification tend to be ineffective when the volume and features of network traffic, potentially carry malicious DDoS payloads, increase exponentially as they cannot extract high importance features automatically. To address this concern, we propose a hybrid approach named AE-MLP that combines two deep learning-based models for effective DDoS attack detection and classification. The Autoencoder (AE) part of our proposed model provides an effective feature extraction that finds the most relevant feature sets automatically without human intervention (e.g., knowledge of cybersecurity professionals). The Multi-layer Perceptron Network (MLP) part of our proposed model uses the compressed and reduced feature sets produced by the AE as inputs and classifies the attacks into different DDoS attack types to overcome the performance overhead and bias associated with processing large feature sets with noise (i.e., unnecessary feature values). Our experimental results, obtained through comprehensive and extensive experiments on different aspects of performance on the CICDDoS2019 dataset, demonstrate both a very high and robust accuracy rate and F1-score that exceed 98% which also outperformed the performance of many similar methods. This shows that our proposed model can be used as an effective DDoS defense tool against the growing number of DDoS attacks.

2.2 Introduction

A Distributed Denial-of-Service (DDoS) attack occurs when attackers make victims' system/network resources unavailable by sending massive amounts of requests to flood the system resources/bandwidth of the victim's system [14, 50]. Typically, attackers exploit vulnerabilities in transport, network, and application layer protocols (e.g., TCP, UDP, HTTP, and ICMP, etc.) [74, 99] to send malicious payloads (e.g., network packets). With the demand for Internet connectivity expand rapidly to mobile devices and Internet of Things (IoT) which is predicted to reach 500 billion by 2030 [103], there is an increasing concern for developing effective DDoS defense techniques.

Table 2.1: The summary of existing ML and DL based approaches

Paper	Techniques	Domain	Performance	Dataset
[46]	KNN, NB, RF, SVM	IDS	98.86 ≈ 99.54 (Acc)	CICDDoS2017
[81]	NB, LR, DT, RF	IoT	99.99 ≈ 100 (F1)	IoT Botnet
[26]	DT, NB, LR, SVM, KNN	IDS	97.72 ≈ 99.99 (Acc)	CICDDoS2019
[5]	XGBoost	SDN	99.9 (Accuracy)	CICDDoS2019
[35]	Stacking, Bagging, Boosting	Smart Grid	92.2 ≈ 93.4 (Acc)	CICDDoS2019
[59]	Gradient Boosting	IDS	96.8 (F1)	CICDDoS2019
[59]	CatBoost	IDS	96.9 (F1)	CICDDoS2019
[70]	Random Forest	IoT	99.97 (Acc)	CICDDoS2019
[83]	D3	SDN	84.54 (Acc)	CICDDoS2019
[60]	EFC	IDS	97.5 (F1)	CICDDoS2019
[77]	Bi-directional LSTM + Gaussian Mixture	IDS	98 (Acc)	CICDDoS2019
[70]	MLP	IDS	99.93 (Acc)	CICDDoS2019
[82]	Gated Recurrent Units (GRU)	IDS	99.69 ≈ 99.94 (Acc)	CICDDoS2019
[8]	Kalman Backpropagation Neural Network	IDS	94 (Acc)	CICDDoS2019
[69]	MLP	IDS	99.92 (Acc)	CICDDoS2019
[21]	RNN + Autoencoder	SDN	99 (F1)	CICDDoS2019
[30]	Sparse-Autoencoder	IDS	88.39 (Acc)	NSL-KDD
[63]	Autoencoder	IDS	88.98 (Acc)	NSL-KDD
[14]	MLP	IDS	79.39 (F1)	CICDDoS2019

To address this concern, many Artificial Intelligence (AI) methods, both using traditional shallow machine learning-based and more advanced deep neural network-based, have been proposed to demonstrate the feasibility of such AI-based approaches to safeguard our networks from DDoS attacks. One of the essential tasks in proposing the next generation of DDoS defense techniques is with the effectiveness of feature extraction techniques. Because it is infeasible and expensive to analyze the entire raw network traffic samples manually among the large feature sets when not all of them provide useful information for detecting malicious payloads. [39, 102]. Many state-of-the-art have proposed solutions to feature extraction for DDoS attack detection and classifications using different feature extraction methods [14, 39, 74]. Though these existing shallow machine learning (ML) approaches have been shown to achieve high detection accuracy, some limitations have been discussed. For instance, Nguyen and Reddi [55] pointed out the inefficiency

in using ML approaches in handling raw, unlabelled, or high dimensional data. Others [10, 42, 47] indicated that the accuracy of detection degrades with ML approaches when a large dataset requires some level of manual feature extraction. Effective feature extraction of network traffic samples that are most relevant to the detection and classification task does not only increases high accuracy but also can accelerate the execution time to analyze the data. In this study, we propose a hybrid deep learning technique that utilizes two deep neural network models for effective feature extraction and accurate DDoS attack detection and classification without human intervention. The contribution of our proposed model is summarized as follows:

- We propose a hybrid deep learning model named "AE-MLP" not only to detect DDoS attacks but also classify the attack into different DDoS attack types in a timely manner.
- Our proposed model uses an Autoencoder (AE) to extract the most important features from a large-scale DDoS attack dataset. Finding the set of compressed and reduced feature sets, most relevant to detect malicious payload, not only improves the accuracy of detection but can also effectively reduce expensive execution time.
- Our proposed model does not only detect potential DDoS attacks but can effectively classify different DDoS attack types. This classification capability can provide an opportunity for cybersecurity professionals to devise an optimal and relevant response strategy as quickly as possible before disastrous damage is done by different DDoS attack types.
- Our experimental results, comprehensively and extensively evaluated, demonstrate a very high and robust F1-score over 98% for detecting DDoS attacks and classifying them into correct attack types. Our results outperformed the performance of many similar methods.

We organize the rest of the Chapter as follows. Section 2.3 examines the related work. Section 2.4 provides the details of the proposed AE-MLP model that contains the feature extraction and classification strategies as well as the algorithm involved. Section 2.5 illustrates the details of the dataset we used in our study and the methodologies we used for data pre-processing. We describe the experimental results in Section 2.6 including the experimental setup, the performance metrics we used, performance of our proposed model, and a comparison to other similar models. Section 2.7 provides a conclusion of our work and future work directions.

2.3 Related Work

We review the existing state-of-the-art in addressing DDoS detection and classification using Artificial Intelligence techniques, both shallow machine learning and deep learning-based neural network here. The summary of these related works are shown in Table 2.1.

2.3.1 machine learning based approaches

Many classical shallow machine learning techniques have been used for DDoS classification. The authors in [26, 46, 81] presents the performance of many classic machine learning techniques, such as Naïve Bayes, Logistic Regression, Decision Tree, Random Forest, Support Vector Machine, and K-Nearest-Neighbour against various DDoS datasets with the detection accuracy reaching near 99%.

Ensemble-based techniques were proposed by [5, 35, 59] by utilizing different techniques involved in bagging, boosting, and stacking, and the results show that some of these techniques outperforming Random forest, Naïve Bayes, and KNN in detecting DDoS attacks in different application contexts (e.g., Smart Grid, IoT).

The authors [70] propose a lightweight approach to detect DDoS attacks aimed at resource-constrained environments such as IoT and show that their lightweight random forest technique can achieve as high as 99% of detection accuracy. Varghese et al. [83] proposed a statistical anomaly detection algorithm implemented in the data plane of Software Defined Network (SDN) to detect DDoS attacks near real-time as a part of an Intrusion Detection System (IDS). Pontes et al. [60] propose an Energy-based Flow Classifier (EFC) which utilizes inverse statistics to infer anomaly scores based on labeled benign examples. The anomaly scores are then used as classifying different DDoS attacks. Their approach achieved 97.5% F1-score while the outcomes of other performance metrics were not presented.

Despite often very high detection rate that achieves 99% accuracy, however, many argued [10, 42, 47, 55] that the detection accuracy degrades with the increase of the size of dataset often containing high dimensional features. In addition, these approaches become impractical when they require raw or unlabelled datasets that require manual feature extraction. To address this limitation, a number of deep learning-based neural networks that can detect DDoS attacks have been proposed.

2.3.2 deep learning based approaches

Shieh et al. [77] demonstrate a Bi-directional LSTM model along with a Gaussian Mixture Model to detect and classify 6 different types of DDoS attacks with an accuracy of 98%. Sanchez et al. [70] proposed a standalone Multi-Layer Perceptron (MLP) achieving the 99.93% accuracy and 99.96% F1-score. Rehman et al. [82] proposed a Gated Recurrent Units (GRU) model to detect DDoS attacks based on CICDDoS2019 dataset. They

achieved the highest accuracy of 99.69% for reflection attacks and 99.94% for exploitation attacks. Almaini et al. [8] proposed Kalman Backpropagation Neural Network where the Kalman algorithm is used to fine-tune weight metrics while backpropagation was utilized to tune biases. The performance evaluation results of their proposed model achieved a performance of 94% accuracy with a low false alarm rate (0.0952). Samom and Taggu [69] proposed an MLP model to detect 4 different DDoS attack types (i.e., SYN, NTP, Portmap, and UDPLag) and compared the results with other machine learning methods. Their study uses the Chi-Squared Function as a feature extractor to select 20 features and then uses the PCA technique for dimension reduction. Their proposal showed that their model achieved 99.92% accuracy on the CICDDoS2019 dataset. Though some of these existing works appear to provide good performance near 99%, they often only offer binary classification where it only detects whether network traffic contains a DDoS attack or not but does not offer to classify what type of DDoS attack it is.

Elsayed et al. [21] proposed a hybrid method named DDoSNet that combines a Recurrent Neural Network (RNN) with an Autoencoder to detect DDoS attack at the Software-Defined Networking (SDN) layer. The evaluation result of their proposal showed that the DDoSNet model achieved the highest performance metrics based on Confusion Matrix but again they also offer only binary classification. Javaid et al. [30] proposed a sparse-autoencoder for feature learning and soft-max regression-based neural architecture for classification and they achieved 88.39% accuracy. The authors in [63] automated threshold learning for anomaly detection in an autoencoder-based model by combining it with unsupervised learning technique isolation forest and got 88.98% accuracy. Can et al. [14] proposed DDoSNet which utilizes an automatic Feature Selection (FS) technique based on the context of the whole feature set then classifies them with fully connected MLP. This proposed method achieved 91.16% precision, 79.41% recall, and 79.39% F1-score for multi-class classification. The authors emphasized that the limitation of their approach was low performance as they were using the whole feature set for classification.

2.4 Our model

Our proposed AE-MLP model consists of two phases: 1) The first phase involves feature extraction via an AE; 2) The second phase involves DDoS attack type classification via an MLP. During the first phase, we build an AE model by using traffic samples as input to train the model. Once AE is trained, the features from the bottleneck layer are extracted. These extracted features from the bottleneck layer of the AE model are fed as inputs to the second phase where MLP uses it to classify different DDoS attack types. Figure 2.1 illustrates the overall approach that is used by our proposed AE-MLP algorithm.

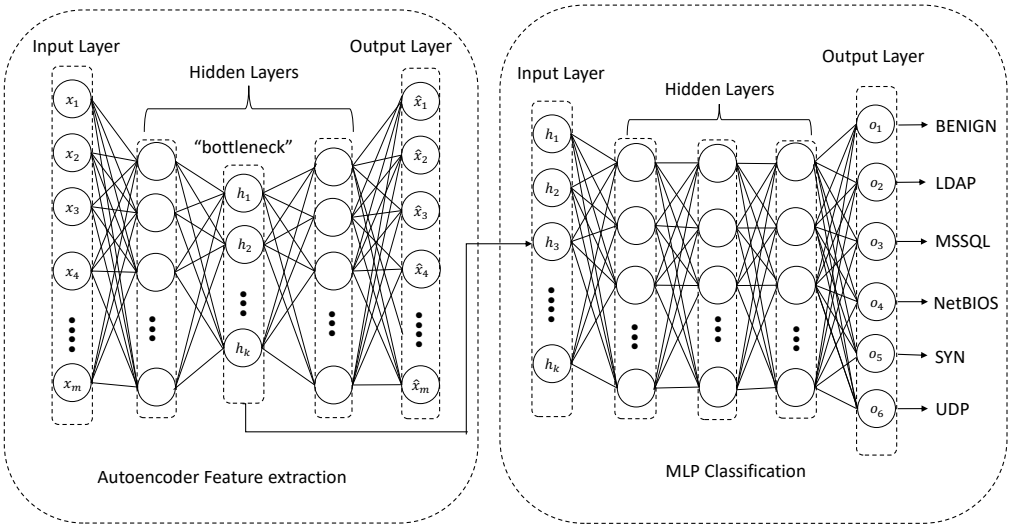


Figure 2.1: The overview of AE-MLP

2.4.1 Autoencoder and Feature extraction

Our AE is an unsupervised feed-forward neural network. It composes of an input layer, an output layer, and several hidden layers. It has a symmetrical pattern – the output layer has the same number of neurons as the input layer while any hidden layer generally has fewer neurons than the input and output layers. The bottleneck layer, also referred to as a latent space, is one of the hidden layers that has the smallest number of neurons. The latent space contains the compressed representation of the input. Typically, the main aim of AE is to reconstruct the input from the output, i.e. $\hat{x} \approx x$ where x indicates the input while $\hat{x}$ indicates the output.

In our approach, the input is reconstructed using AE while doing so we get the hidden layer feature embedding with the minimum number of neurons which represents a lower-dimensional projection of input features. This hidden layer embedding captures data characteristics in the lower dimension, thus we have used it as input features to the MLP classification model. To use AE as a feature extraction engine, our model goes through the following steps.

2.4.1.1 Encoding

In the encoding operation (E_ϕ), any input sample x is a m dimensional vector ($x \in \mathbb{R}^m$) and is mapped to the bottleneck layer representation (h), as shown in Equation (2.1).

$$h = f_1(w_1 x + b_1) \tag{2.1}$$

where w_1 is the weight matrix, b_1 is a bias and f_1 is an activation function.

2.4.1.2 Decoding

In the decoding operation (D_θ), the bottleneck layer representation of (h) is mapped back into a reconstruction of x, as shown in Equation (2.2):

$$\hat{x} = f_2(w_2 h + b_2) \tag{2.2}$$

where f_2 is an activation function for the decoder. w_2 is the wight matrix, b_2 represents a bias and $\hat{x}$ represent reconstructed input sample.

2.4.1.3 Loss Function

To minimize reconstruct error on x with non-linear functions, the loss reconstruction (L) is calculated from Equation (2.3).

$$L(x, \hat{x}) = \frac{1}{n} \sum_{i=1}^{n} (x_i - \hat{x_i})^2 \tag{2.3}$$

where n represents the number of training samples.

2.4.1.4 Feature Extraction

The equations (2.1), (2.2) and ((2.3)) represent working of single hidden layer auto-encoder where h represent bottleneck layer feature embedding (encoding) of AE. This embedding size is dependent on the number of neurons (k) in the hidden layer, in general, its size is less compared to input dimension ($k << m$). AE model applies backpropagation to obtain optimal values for the weight matrix $w_1 \in R^{m \times k}$ and $w_2 \in R^{k \times m}$ and bias $b_1 \in R^{m \times 1}$ and $b_2 \in R^{k \times 1}$ in equations (2.1) and (2.2) respectively to minimize the difference between input x and output $\hat{x}$. Mostly rectified linear unit (ReLU) is used as a non-linear activation function in the hidden layer. In practice, multiple hidden layers are used, where each layer has its own encoding and decoding function described in equation (2.1) and (2.2) respectively. In our work, we have used the hidden layer with the lowest number of neurons as a feature vector for our Multi-layer Perceptron classification model.

2.4.2 Multi-layer Perceptron Network and Classification

MLP is also a feed-forward network, unlike AE its output layer is equal to the number of classes (p). The MLP has an input layer (our case equal to the size of AE bottleneck layer), multiple hidden layers, and an output layer (equal to the sum of attacks and benign classes, $p = 6$). Similar to AE, the hidden layer uses non-linear activation function (in our case, we used relu function) to extract information from input features as shown in Equation (2.4)

$$y_z = f_{relu}(h_z w_j + b_j) \tag{2.4}$$

where h_z represent latent space embedding from AE as feature vector, w_j is the weight matrix , b_j is bias vector, f_{relu} non-linear activation function of hidden layer and y_z is information extracted at hidden layer of MLP.

By processing the input vector h_z, the hidden layers of MLP produce the vector y_z. This vector y_z is fed as input to output layer to predict output class. The output layer mostly use softmax function for multiclass problems.

Finally, output class ($\hat{y} \in \mathbb{R}^p$) can be predicted using equation (2.5).

$$\hat{y} = softmax(y_z w_y + b_y) \tag{2.5}$$

where w_y and b_y are weights matrix and bias vector for the output layer.

2.4.3 AE-MLP Algorithm

The algorithm for our proposed AE-MLP model for DDoS attack detection and classification is shown in Algorithm 1.

In our proposed model, we use AE as a feature extraction tool that can transform the original data (e.g., network traffic) from the high dimensional space to the non-linear low dimensional space. By doing this, the latent space at the AE now contains the number of features that can be best represented to detect if network traffic contains a malicious DDoS payload and further classify what type of DDoS attack payload it carries. To determine the best features for DDoS detection, our AE goes through the following steps:

- We first use the unsupervised learning mode of the AE to train on the training dataset for dimensionality reduction purposes.
- We have experimented on the number of different AE architecture in terms of the number of input, hidden, and output layers and corresponding hyperparameters. The best optimized AE architecture was the one that uses 77 encoded features as input, a single hidden layer with 32 neurons, and the latent space that represents the 24 features as the last hidden layer.
- The 24 features at the latent space are extracted.

The extracted features are then fed into the MLP model as inputs and are now used to train the MLP model as a classifier to detect different DDoS attack types. To classify different DDoS attack types, our MLP goes through the following steps:

- We use the supervised learning mode of the MLP to train on the training dataset using the label contained in the training dataset.
- We have experimented on the number of different MLP architectures. The best optimized MLP architecture was the one that uses 5-layers – 1 input layer, 3 hidden layers, and 1 output layer.

Algorithm 1: AE-MLP Classification

Input: Training dataset $X = \{x_1, x_2, x_3, \ldots, x_n\}$
Testing dataset $X' = \{x'_1, x'_2, x'_3, \ldots, x'_n\}$
Training Label $Y = \{y_1, y_2, \ldots, y_n\}$
Testing Label $Y' = \{y'_1, y'_2, \ldots, y'_n\}$
Encoder E_ϕ; Decoder D_θ; MLP M_δ
Output: $O(\hat{y'}|y')$

begin
  /* Phase 1: AE feature extraction */
  /* Training AE in mini-batch */
  $\phi, \theta \leftarrow$ Initialize parameters
  for *number of training iterations* **do**
    sample mini-batch of k samples $\{X_1, X_2, X_3, \ldots, X_k\}$ from X
    /* Calculating loss */
    $V(E, D) = \frac{1}{k} \sum_{i=1}^{k} (X_i - D_\theta(E_\phi(X_i)))^2$
    $\phi, \theta \leftarrow$ Update parameters using Stochastic Gradient Descent of V
  end
  /* Phase 2: MLP Classification */
  $\delta \leftarrow$ Initialize parameters
  /* Training MLP */
  for *each* $(x, y) \in (X, Y)$ **do**
    /* get latent presentations from trained AE */
    $a \leftarrow E_\phi(x)$
    /* trained MLP M_δ */
    $O(\hat{y}|y) \leftarrow M_\delta((a), y)$
    $\delta \leftarrow$ Update parameters using Stochastic Gradient Descent
  end
  /* Testing */
  for *each* $(x', y') \in (X', Y')$ **do**
    $a' \leftarrow E_\phi(x')$
    $O(\hat{y'}|y') \leftarrow M_\delta((a'), y')$
  end
end

- The activation function "relu" was used for hidden layers while "softmax" function was used at the last output layer for classification.

2.5 Data and Methodologies

In this section, we provide the details of the data we used for our study, the methodology we employed for data processing, and the workflow of our proposed model. The CIC-DDOS dataset has two datasets, training, and testing datasets, respectively. As seen in Figure 2.2, we first use only the training dataset after applying a number of data pre-processing techniques such as data cleaning, removing irrelevant features, label encoding,

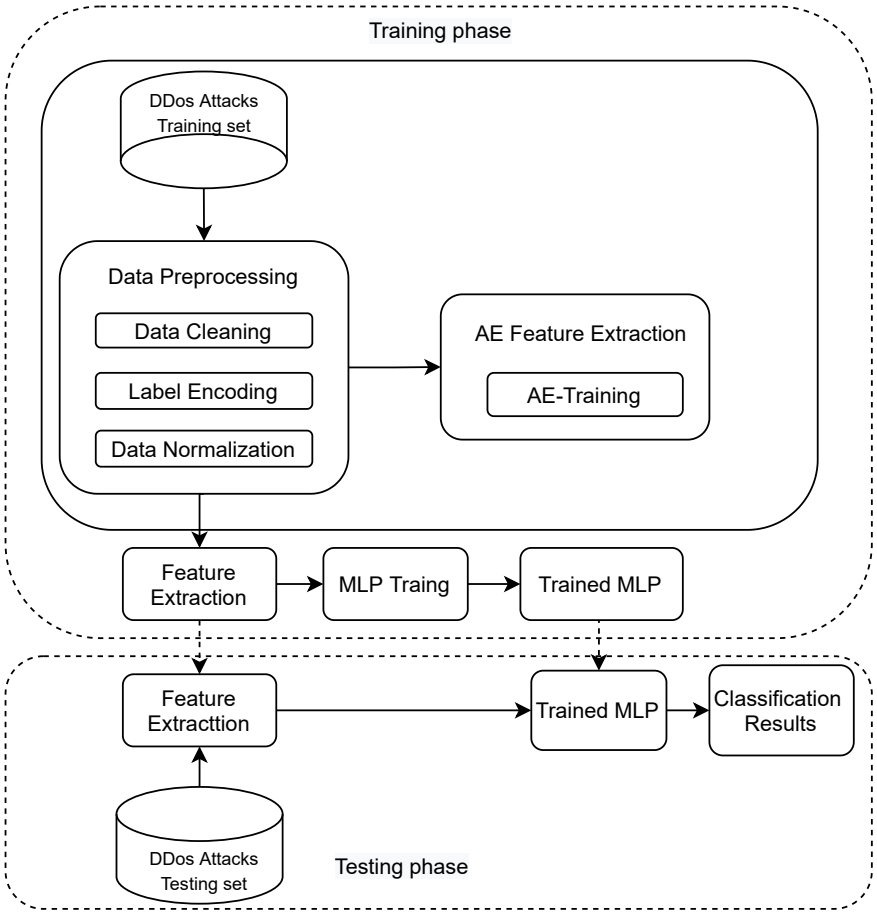


Figure 2.2: AE-MLP classification

and normalizing the dataset by scaling them to fit in the range of [0, 1]. After preprocessing the training dataset, we fit the dataset into our proposed model for AE training and subsequent feature extraction. The extracted features are then fed into the MLP. Another training by MLP proceeds to train the MLP model. Once our proposed model is well trained, we use the testing dataset first fed into the AE for the feature extraction using the hyperparameters that were trained during the AE training phase, the extracted features then are fed into the trained MLP for a classification task to categorize different DDoS attack types.

2.5.1 CICDDoS2019 Dataset

In this study, we use the CICDDoS2019 [74] dataset that has been widely used for DDoS attack detection and classification. The dataset contains a large amount of up-to-date realistic DDoS attack samples as well as benign samples. The total number of records contained in CICDDoS2019 is depicted in Table 2.2.

Table 2.2: The number of records in CICDDoS2019

dataset	**total**	**benign**	**malicious**
Training day	50,063,112	56,863	50,006,249
Testing day	20,364,525	56,965	20,307,560

Each record of the dataset contains 88 statistical features (e.g., timestamp, source and destination IP addresses, source and destination port numbers, the protocol used for the attack, and a label for a type of DDoS attack). The training dataset contains a total of 12 different DDoS attacks (i.e., NTP, DNS, LDAP, MSSQL, NetBIOS, SNMP, SSDP, UDP, UDP-Lag, WebDDoS, SYN, and TFTP) while only 7 DDoS attacks are included in the testing dataset (i.e., PortScan, NetBIOS, LDAP, MSSQL, UDP, UDP-Lag and SYN). The details of the date and number of records collected for different DDoS attack types are present in Table 2.3.

Table 2.3: Daily label of data collection

Days	Attacks	Attack times	Flow Count
Training Set	LDAP	11:22 - 11:32	2,179,930
	MSSQL	11:36 - 11:45	4,522,492
	NetBIOS	11:50 - 12:00	4,093,279
	UDP	12:45 - 13:09	3,134,645
	SYN	13:29 - 13:34	1,582,289
Testing Set	NetBIOS	10:00 - 10:09	3,657,497
	LDAP	10:21 - 10:30	1,915,122
	MSSQL	10:33 - 10:42	5,787,453
	UDP	10:53 - 11:03	3,867,155
	SYN	11:28 - 17:35	4,891,500

These DDoS attacks cover two different categories, some belong to Reflection-based and others belong to Exploitation-based.

Reflection-based attacks: The attackers of the DDoS attack in this category typically send malicious network packets to reflector servers with the source IP address set to target the victim's IP address so that the victim is overwhelmed to send an enormous number of response packets. These attacks are typically carried out through application layer protocols. In terms of our CICDDoS2019 dataset, any traffic with the (application layer) protocol defined for MSSQL, SSDP, NTP, TFTP, DNS, LDAP, NetBIOS, and SNMP be reflection-based attacks.

Exploitation-based attacks: The attackers of the DDoS attack in this category exploit a particular protocol used in the network, transport, and application level of the Open Systems Interconnection (OSI) model or TCP/IP 5-layer model. Transport layer protocols such as TCP or UDP are typically used to overwhelm the victim's IT resources (e.g., SYN flood, UDP flood, and UDP-Lag) by sending a massive number of TCP or UDP packets. The dataset labeled with SYN, UDP, and UDP-lag in CIDDDOS2019 belongs to this category. The DDoS attack categorization is seen in Table 2.3

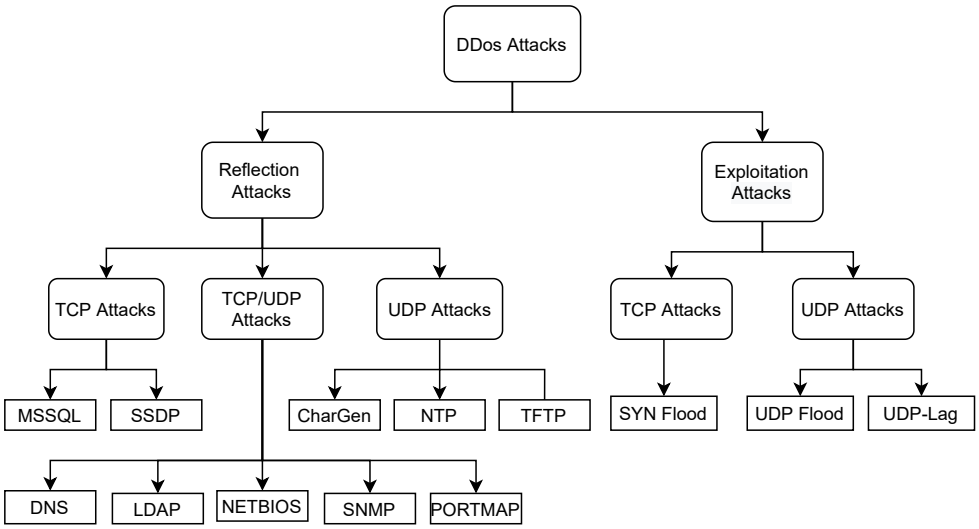


Figure 2.3: DDos attacks categorization and hierarchy [74]

In our study, we use 5 DDoS attack types (i.e., LDAP, MSSQL, NetBIOS, SYN, UDP) and benign traffic samples to train and test our proposed model. The high-level description of the nature of the DDoS attack used in our study is summarised as follows.

- **LDAP Attack**: In this DDoS attack, an application layer protocol, Lightweight Directory Access Protocol (LDAP) typically used to obtain a human-readable URL (e.g., google.com), is exploited by an attacker to send requests to a publicly available but vulnerable LDAP server to generate large responses.
- **NetBIOS Attack**: In this DDoS attack, Network Basic Input/Output System (NetBIOS) is exploited by an attacker that sends spoofed "Name Release" or "Name Conflict" messages to a victim machine in order to refuse all NetBIOS network traffic.
- **MSSQL Attack**: An attacker exploits the vulnerabilities in Microsoft Structured Query Language (MSSQL) where the attacker pretends to be a legitimate MSSQL client by executing the scripted requests using a forged IP address to the MSSQL Server to appear as coming from the target server.
- **SYN Attack**: The SYN flood attack exploits the TCP-three-way handshake by sending a massive number of repeated SYN packets to the target machine until the server crashes/malfunctions.
- **UDP Attack:** In the UDP flood attack, UDP packets are sent to random ports on the target machine at a very high rate. As a result, the available bandwidth of the network gets exhausted, the system crashes and performance degrades. The firewall protecting the target server can be exhausted as a result.

2.5.2 Data Pre-processing

In this section, we discuss the methodologies we used to process our dataset to feed into our proposed AE-MLP model.

2.5.2.1 Data Cleaning

The original dataset contained 88 features. As suggested by [8], we also removed the features not contributing to detecting DDoS attacks. These include the feature such as "Unnamed", "Flow ID", "Source IP", "Destination IP", "Source Port", "Destination Port", "Timestamp", "Flow Bytes", "Flow Packets", and "SimilarHTTP". After the exclusion of these 10 features, we have 78 features to work with. Following the recommendation of the work by [69], we further cleaned up the values containing NaN (not a number), blank, and infinity values to set 0.

2.5.2.2 Label Encoding

We had to substitute the categorical labels as deep models only operate on float/numeric values. One categorical value we had to convert was the attack label (i.e., benign and the five attack types). We used a 6-bit feature vector to indicate different labels, for example [1,0,0,0,0,0] indicates benign, [0,1,0,0,0,0] indicates LDAP attack type, and [0,0,1,0,0,0] indicates MSSQL attack, etc. With an additional 6 feature vectors added, we had a total of 83 features (i.e., 77 representing the original features plus 6 features for an attack label).

2.5.2.3 Data Normalization

The CICDDoS2019 datasets contain some features with very high variance in terms of value between the minimum and the maximum (e.g., "Flow Duration", "Flow IAT Std", "Flow IAT Max", "Bwd IAT min"). We applied a normalization strategy to eliminate the impacts of big variance of the values across the features thus reducing the execution time for model training and improving accuracy. There are several widely used methods to perform feature scaling, including Z Score, standardization, and normalization. As proposed by [82], we use MinMax-based normalization for our feature scaling. This method maps the original range of each feature into a new range with Equation (2.6)

$$Z_i = \frac{Z_i - min}{max - min} \tag{2.6}$$

where Z_i donates all the normalized numeric values ranging between [0-1]; max and min donates the maximum and minimum values from all data points.

2.6 Experimental Results

In this section, we provide the details of the experiment including the environment setup, analysis of results, and discussion.

2.6.1 Experiment Setup

Our experiments were carried out using the following system setup shown in Table 2.4.

Table 2.4: Implementation environment specification

Unit	**Description**
Processor	3.4GH$_z$ Inter Core i5
RAM	16GB
OS	MacOS Big Sur 11.4
Packages used	tensorflow 2.0.0, sklearn 0.24.1

To evaluate the performance of our proposed model, we use classification accuracy, precision, recall, and F1 score as performance metrics. Table 2.5 illustrates the confusion matrix, where:

- True Positive (TP) indicates anomalous traffic correctly classified as anomalous.
- True Negative (TN) indicates normal traffic correctly classified as normal.
- False Positive (FP) indicates normal traffic incorrectly classified as anomalous.
- False Negative (FN) indicates anomalous traffic incorrectly classified as normal.

Table 2.5: Confusion Matrix

Total Population		Predicted Condition	
		Normal	Anomaly
Actual Condition	Normal	TN	FP
	Anomaly	FN	TP

Based on the aforementioned terms, the evaluation metrics are calculated as follows. True Positive Rate (also known as Recall) estimates the ratio of the correctly predicted samples of the class to the overall number of instances of the same class. It can be computed using Equation (2.7). Higher TPR $\in [0, 1]$ value indicates the good performance of the machine learning model.

$$TPR(Recall) = \frac{TP}{TP + FN} \tag{2.7}$$

False Positive Rate (FPR) presents the proportion of data points correctly classified as anomalous, which can be calculated in Equation (2.8).

$$FPR = \frac{FP}{FP + TN} \tag{2.8}$$

Precision (Pre) measures the quality of the correct predictions. Mathematically, it is the ratio of correctly predicted samples to the number of all the predicted samples for that particular class as shown in Equation (2.9). Precision is usually paired with Recall to evaluate the performance of the model. Sometimes pair can appear contradictory thus comprehensive measure F1-score is considered.

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

F1-Score computes the trade-off between precision and recall. Mathematically, it is the harmonic mean of precision and recall as shown in Equation (2.10).

$$F1 = 2 \times \left(\frac{Precision \times Recall}{Precision + Recall} \right) \tag{2.10}$$

Accuracy (Acc) measures the total number of data samples correctly classified, as shown in Equation (2.11).

$$A_{CC} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.11}$$

The area under the curve (AUC) computes the area under the receiver operating characteristics (ROC) curve which is plotted based on the trade-off between the true positive rate on the y-axis and the false positive rate on the x-axis across different thresholds. Mathematically, AUC is computed as shown in Equation (2.12).

$$AUC_{ROC} = \int_0^1 \frac{TP}{TP + FN} d \frac{FP}{TN + FP} \tag{2.12}$$

The training parameters are shown in Table 2.6.

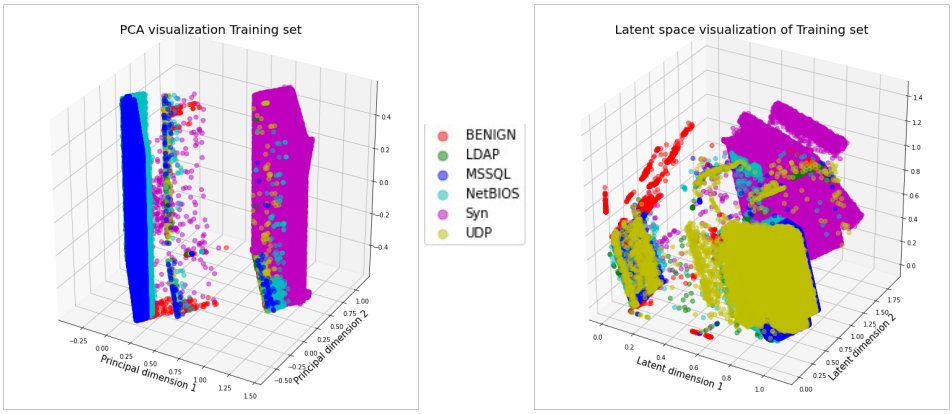
2.6.2 Performance Of Our Proposed Model

We used 5% of the original CICDDoS2019 dataset from the day one collection for training as it was not feasible to use the full dataset due to performance consideration. Figure 2.4 shows the PCA and latent space visualizations of the training dataset. In the PCA visualization, it is very difficult to identify different clusters of attacks, while the latent space components can identify most clusters. This exploratory analysis suggests that the latent space embedding better captures the features in a lower dimension. We can think of

Table 2.6: Training parameters

Algorithms	Hyperparameters	values
AE	Mini-batch	32
	Learning rate	0.001
	N-iterations	19137
	Epoch	20
MLP	activation	relu
	solver	adam
	hidden layers size	[23,15,10]

latent space as a coordinate system in which similar points are placed together. Therefore, feature embedding from the hidden layer of AE is more suitable for classification than raw features, as can be seen from the classification results. We would also like to point out that although the PCA components do not show a complete picture of the feature space, they still increase the interpretability of the data.



(a) Training set visualisation

(b) Training latent visualisation

Figure 2.4: The PCA visualization of training set and latent space visualization (bottleneck layer of AE) of training set

From Figure 2.4, it is visible that there is no balance in the number of traffic samples between benign (red color) and DDoS attack (other colors) as the number of benign is significantly less.

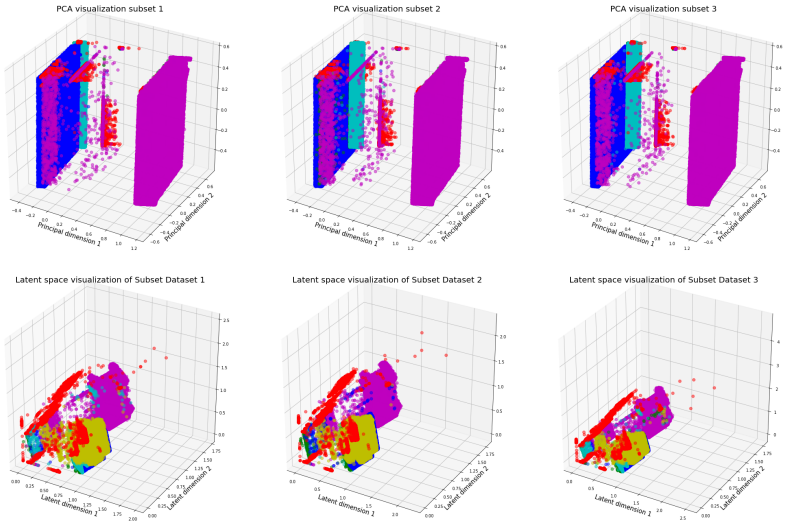
Our model used 80% of this data as the training dataset to fine-tune the model while 20% was used as a validation dataset to fine-tune the model's hyperparameters. We used 5% of the original CICDDoS2019 dataset for each subset from the day two collection as the testing dataset.

Table 2.7: Performance Metrics on different DDoS attack types

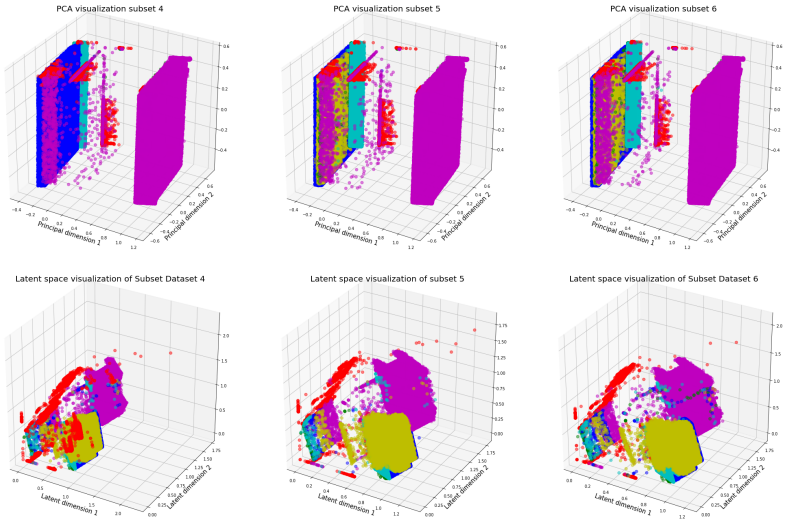
Testing Subdatasets	Attack types	performance metrics			
		Accuracy	Precision	Recall	F1
subdataset 1	BENIGN	99.95	92.12	88.19	90.11
	LDAP	99.21	93.91	98.30	96.05
	MSSQL	97.41	98.05	93.16	95.54
	NetBIOS	**99.96**	99.97	**99.82**	**99.90**
	SYN	99.94	**99.97**	99.76	99.86
	UDP	98.14	92.76	98.08	95.34
Subdataset 2	BENIGN	99.91	90.15	75.79	82.35
	LDAP	99.17	93.60	98.26	95.87
	MSSQL	98.50	98.06	96.89	97.47
	NetBIOS	**99.95**	99.92	**99.81**	**99.87**
	SYN	99.92	**99.93**	99.72	99.83
	UDP	99.24	98.11	97.96	98.03
Subdataset 3	BENIGN	99.90	72.34	95.74	82.41
	LDAP	99.18	93.71	98.30	95.95
	MSSQL	98.46	97.99	96.80	97.39
	NetBIOS	**99.85**	99.31	**99.90**	**99.60**
	SYN	99.76	**99.97**	98.93	99.45
	UDP	98.96	96.81	97.87	97.38
Subdataset 4	BENIGN	99.95	94.32	87.09	90.56
	LDAP	99.19	93.75	98.35	96.00
	MSSQL	97.71	98.05	94.18	96.08
	NetBIOS	**99.96**	99.92	**99.88**	**99.90**
	SYN	99.95	**99.98**	99.80	99.89
	UDP	98.46	94.28	97.99	96.10
Subdataset 5	BENIGN	99.95	95.26	85.62	90.18
	LDAP	99.18	93.79	98.24	95.96
	MSSQL	98.45	98.01	96.75	97.38
	NetBIOS	**99.96**	99.93	**99.87**	**99.90**
	SYN	99.94	**99.96**	99.79	99.87
	UDP	99.23	98.04	97.98	98.01
Subdataset 6	BENIGN	99.91	76.91	94.71	84.89
	LDAP	99.20	93.76	98.44	96.04
	MSSQL	97.63	97.99	93.98	95.94
	NetBIOS	**99.94**	99.84	**99.87**	**99.86**
	SYN	99.90	**99.99**	99.57	99.77
	UDP	98.37	94.09	97.75	95.87

In our study, we limit the classification of benign and 5 DDoS attacks - LDAP, MSSQL, NetBIOS, SYN, and UDP – during the testing phase to avoid any implication of biases due to an imbalanced dataset.

The plots in the top layer of Figure 2.5 show the PCA visualization of the distribution of data points of each test dataset in their raw form. As it is shown in the PCA visualization, there are more samples of DDoS attacks compared to the benign samples at each set - the similar pattern of data distribution we witnessed in the training dataset. The plots in the bottom layer of Figure 2.5 show the distribution of data points of each test set at the bottleneck layer of the trained AE which eventually becomes the inputs to the MLP model. As can be seen in the latent space visualization, there are distinct clusters around benign and each DDoS attack type. The size of the data points (i.e., number of features)



(a) Subdataset 1 (b) Subdataset 2 (c) Subdataset 3



(d) Subdataset 4 (e) Subdataset 5 (f) Subdataset 6

Figure 2.5: The top layer shows PCA visualization of test datasets where axes indicate principal components of PCA embedding. The bottom layer shows latent space projection of test datasets through the bottleneck layer of trained AE where axes indicate latent space components of projection. (●: benign, ●: LDAP, ● : MSSQL, ● NetBios, ●: Syn, ●: UDP)

that represent the benign and each DDoS attack type appear to be similar across all sub-datasets. The detection and classification performance based on the performance metrics of Accuracy, Precision, Recall, and F1-score on the five different DDoS attack types are shown in Table 2.7.

All different sub-datasets show very similar trends of the performance metrics which

confirms that our proposed model does not overfit/underfit. Closely observing the performance metrics of each DDoS attack type, almost all DDoS attack types achieved above 97% classification accuracy. The NetBIOS attack type however showed the highest accuracy rate very close to almost 100%.

Figure 2.6 illustrates the exact number of records classified for different performance metrics for five DDoS attack types based on the confusion matrix. Similar to the results presented in Table 2.7, the DDoS attack type "SYN" has the most number of TPR where the number of FPR is almost negligent (around a few hundred records misclassified). In comparison, the DDoS attack type "MSSQL" shows the worst performance where there is a large number of FPR that goes beyond thousand records.

The average performance metrics of Accuracy, Precision, Recall, and F1-score on different sub-datasets are shown in Table 2.8. The number of traffic samples contained in different sub-dataset differs from $> 960,000$ (i.e., subdataset 1) to close to a million (i.e., subdataset 6). Regardless of the number of traffic samples, the different subdataset show a very similar pattern across all 6 subdatasets. The accuracy is in the range of 97% and 98% while a similar pattern in the Precision, Recall, and F1 scores are shown.

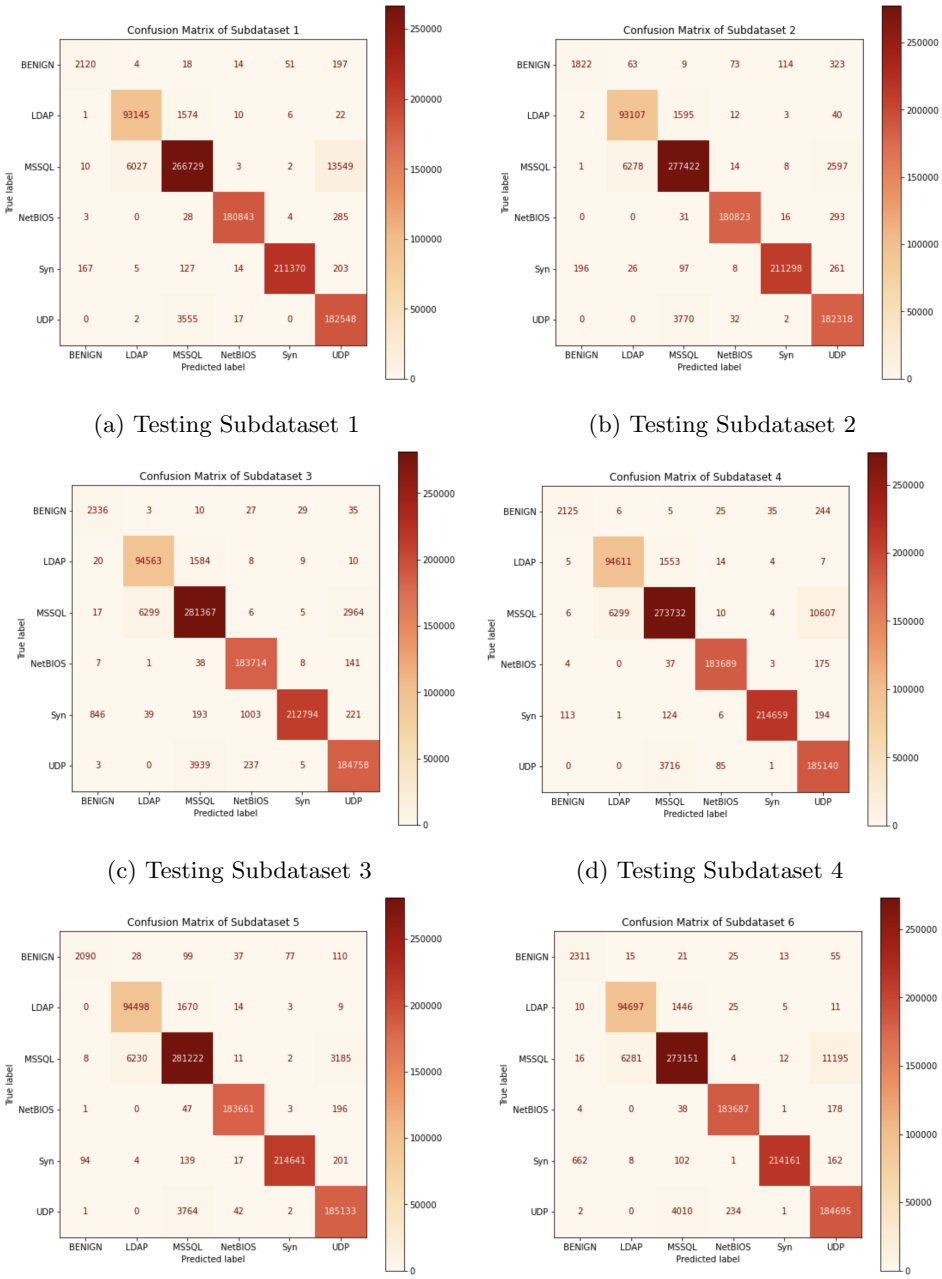
Figure 2.7 shows the AUC-ROC depicted on different DDoS attack types across all 6 subdatasets. The AUC-ROC value is more than 0.99 in all DDoS attack types in all subdatasets which confirms that our proposed AE-MLP model is highly effective in detecting and classifying different DDoS attack types with very high TPR while FPR stays very low.

2.6.3 Comparison To Other Similar Methods

Table 2.9 shows the performance comparison of our proposed AE-MLP model with other similar methods both from shallow machine learning and deep learning-based neural network approaches. As the results show, our approach shows the best performance in terms of all aspects of performance metrics reaching the average of 98.34% accuracy while the precision, recall, and F1-score all remain very competitive at 97.91%, 98.48%, and 98.18% respectively. In general, shallow machine learning approaches do not perform as well as deep learning-based counterparts unless they are extended, for example, Data Jungle proposed by Rajagopal et al. [61] which combines several decision trees to achieve a higher accuracy rate. A deep learning-based approach using a standalone classifier, such as LSTM, CNN, and MLP tends to achieve more than 90% accuracy and demonstrate that they are suitable to provide an effective classifier to detect and classify different DDoS attack types. In the realm of hybrid approaches, AE combined with a shallow machine learning-based classifier such as using linear regression or an isolation forest tends to work less than when two deep learning models are combined like ours.

Table 2.8: Average performance on six subdatasets

		Testing Subdataset					
	Attack types	Subdataset 1	Subdataset 2	Subdataset 3	Subdataset 4	Subdataset 5	Subdataset 6
No. of Instance	BENIGN	2,402	2,404	2,440	2,440	2,441	2,440
	LDAP	94,758	94,759	96,194	96,194	96,194	96,194
	MSSQL	286,320	286,320	290,658	290,658	290,658	290,659
	NetBIOS	181,163	181,163	183,909	183,908	183,908	183,908
	SYN	211,886	211,886	215,096	215,097	215,096	215,096
	UDP	186,122	186,122	188,942	188,942	188,942	188,942
Total performance mertrics		Acc = 97.31 Pre = 96.13 Recall = 96.22 F1 = 96.14	Acc = 98.35 Pre = 96.63 Recall = 94.74 F1 = 95.57	Acc = 98.19 Pre = 93.59 Recall = 97.91 F1 = 95.47	Acc = 97.62 Pre = 96.71 Recall = 96.21 F1 = 96.42	Acc = 98.36 Pre = 97.50 Recall = 96.37 F1 = 96.88	Acc = 97.49 Pre = 93.76 Recall = 97.39 F1 = 95.40



(a) Testing Subdataset 1

(b) Testing Subdataset 2

(c) Testing Subdataset 3

(d) Testing Subdataset 4

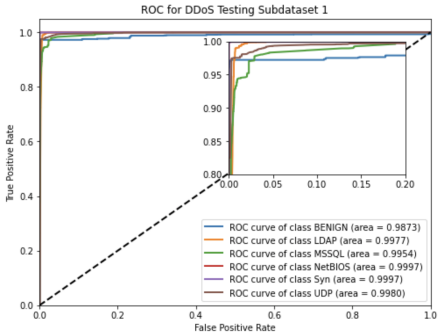
(e) Testing Subdataset 5

(f) Testing Subdataset 6

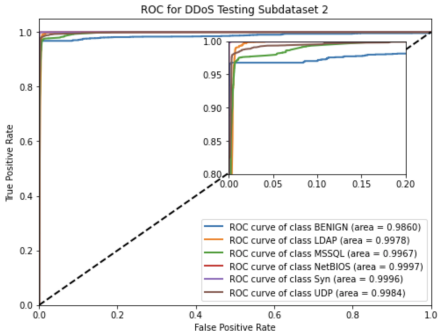
Figure 2.6: Performance of classification on different DDoS attack types based on Confusion Matrix

2.7 Conclusion

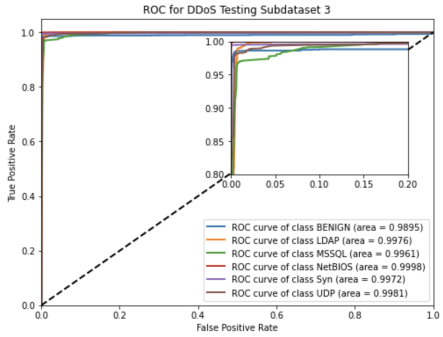
In this study, we show that DDoS attacks can be detected and classified with high accuracy using a combination of deep learning-based techniques. Our proposed hybrid model AE-



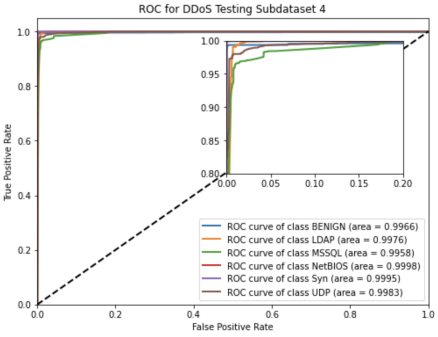
(a) ROC for Testing Subdataset 1



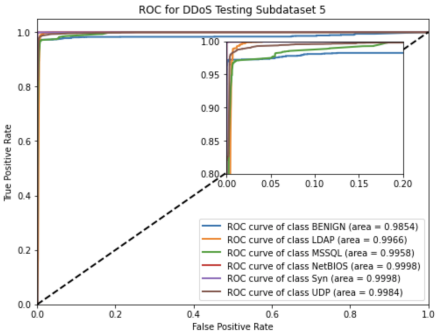
(b) ROC for Testing Subdataset 2



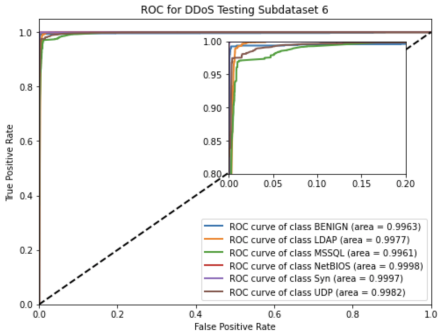
(c) ROC for Testing Subdataset 3



(d) ROC for Testing Subdataset 4



(e) ROC for Testing Subdataset 5



(f) ROC for Testing Subdataset 6

Figure 2.7: Performance based on AUC-ROC metric

MLP consists of an Autoencoder (AE) and a Multi-layer Perceptron Network (MLP). The AE in our proposed model extracts the most important and relevant features to find malicious DDoS network payloads from a large-scale network traffic sample. The compressed and reduced features produced by the AE model are then fed to the MLP to effectively classify different DDoS attack types. This hybrid approach is not only effective in detecting DDoS attacks in a timely manner but is also effective in classifying what DDoS attack family the detected DDoS payload belongs to. Our proposed model can be an effective DDoS defense tool to detect a massively growing number of DDoS attacks in

Table 2.9: Comparison to other similar methods

Paper	Techniques	Acc	Pre	Recall	F1
Sharafaldin et al. [74]	Random Forrest	-	77	56	62
Rajagopal et al. [61]	Extended Decision Tree	97	99.0	97.0	97.8
Gohil et al. [26]	Extended Naive Bayes	96.25	96	96	96
Shieh et al. [77]	Bi-LSTM	98.18	97.93	99.84	-
De Assis et al. [18]	CNN	95.4	93.3	92.4	92.8
De Assis et al. [18]	MLP	92.5	84.4	94.2	89.0
Javaid et al. [30]	AE + Regression	88.39	85.44	95.95	90.4
Sadaf et al. [63]	AE + Isolation Forest	88.98	87.92	93.48	90.61
Can et al. [14]	FS + MLP	-	91.16	79.41	79.39
Our Proposal	AE-MLP	98.34	97.91	98.48	98.18

recent times. Our proposed model, comprehensively and extensively tested against many subsets of large DDoS attack samples, demonstrates high performances against many performance metrics such as Precision (97.91%), Recall (98.48%), F1-score (98.18%), and Accuracy (98.34%) which outperformed many other similar methods.

Chapter 3

Reconstruction-based LSTM-Autoencoder for DDoS Attack Detection over Multivariate Time-Series Data

Abstract

A Distributed Denial-of-service (DDoS) attack is a malicious attempt to disrupt the regular traffic of a targeted server, service, or network by sending a flood of traffic to overwhelm the target or its surrounding infrastructure. As technology improves, new attacks have been developed by hackers. Traditional statistical and shallow machine learning techniques can detect superficial anomalies based on shallow data and feature selection, however, these approaches can not detect unseen DDoS attacks. In this context, we propose a reconstruction-based anomaly detection model named LSTM-Autoencoder (LSTM-AE) which combines two deep learning-based models for detecting DDoS attack anomalies. The proposed structure of long short-term memory (LSTM) networks provides units that work with each other to learn the long short-term correlation of data within a time series sequence. Autoencoders are used to identify the optimal threshold based on the reconstruction error rates evaluated on each sample across all time-series sequences. As such, a combination model LSTM-AE can not only learn delicate sub-pattern differences in attacks and benign traffic flows but also minimize reconstructed benign traffic to obtain a lower range reconstruction error, with attacks presenting a larger reconstruction error. In this research, we trained and evaluated our proposed LSTM-AE model on reflection-based DDoS attacks (DNS, LDAP, and SNMP). The results of our experiments demonstrate that our method performs better than other state-of-the-art methods, especially for LDAP attacks, with an accuracy of over 99%.

3.1 Introduction

Network traffic is increasing rapidly with the continued development of information and communication technology (ICT) due to advanced innovative technologies, including cloud computing, and big data. However, the rapid proliferation of innovative technologies and communication infrastructure brings the potential for cyberattacks and other threats to Internet users. In the area of cyber security attacks, one of the most dangerous threats is a distributed denial-of-service (DDoS) attack [27, 64, 99].

A DDoS attack is a form of network attack that attempts to overwhelm online services, websites, and web applications with malicious traffic from multiple compromised computer systems. It can also make simultaneous requests to the target server in order to exhaust the network resources and thereby deny normal online service to legitimate users or computer systems [20, 66, 79]. DDoS attacks are not only conducted against online services, web applications, and information infrastructure to cause downtime, but also to prevent legitimate users from purchasing products and using online services - such as emails, websites, and applications - and affecting program performance [66]. As a result of the COVID-19 lockdown in 2020, there has been an increase in attacks on education, online shopping, and office work, as a large number of people are now studying, working, and shopping online, giving hackers greater opportunities [6]. In [48] Azure Networking found there was a 25% increase in DDoS attacks in the first six months of 2021 when compared with the fourth quarter of 2020. Moreover, Azure mitigated approximately 35 thousand attacks against its global infrastructure in the last six months of 2021, which increased from 43% compared with the first six months of 2021. A white paper from Cisco [17] predicted that nearly 300 billion mobile applications would be downloaded by 2023 and that DDoS attacks would rise to 15.4 million globally by 2023.

DDoS detection is becoming an urgent need because of the sophistication and diversification of attacks. For instance, difficult-to-track attackers and unknown or new attack types occur continuously, for example, zero-day attacks [2, 96]. Detecting DDoS attacks becomes more and more difficult not only because a large proportion of attack traffic is similar to legitimate traffic, but also because of newer hybrid attack methods [20, 99]. Therefore, the detection and mitigation of DDoS attacks not only protects the network for legitimate users but also reduces financial loss for businesses [68]. In order to detect and mitigate DDoS attacks, statistical techniques have been proposed in [49] to identify DDoS attacks. Furthermore, some machine learning approaches for signature, threshold, and statistics-based measurements have been proposed to distinguish DDoS attack traffic [9, 84]. However, traditional statistical and machine learning can not detect previously unseen DDoS attacks [20]. Moreover, most traditional statistical and machine learning-based detection approaches require better-selected features or defined thresholds [20, 99]. In contrast to traditional detection techniques, deep learning-based DDoS attack detection - such as Convolutional Neural Networks (CNNs) [20], Recurrent Neural Networks (RNNs) [99], Autoencoders [68], and so forth - can offer better detection rates for DDoS

network traffic [99]. However, some limitations in existing deep learning-based detection need to be addressed, for example, Autoencoder models are sensitive to the anomalies in the training stage, and RNNs can better address historical sequence data, but face the shortcoming of the vanishing gradient problem. To address these issues, we propose a reconstruction-based hybrid deep learning model that combines the capabilities of long short-term memory (LSTM) and Autoencoders (AE) for detecting DDoS attacks, using the state-of-the-art CICDDoS2019 datasets.

In this research, the LSTM model aims to solve the time-series sequence problem of DDoS traffic flow, while the AE is used to calculate the reconstruction loss in order to define the threshold and detect DDoS attacks. In order to overcome the shortcoming caused by sensitivity to anomalies in the training process, we use only benign traffic from the DDoS dataset to train our model and minimize the reconstruction error. Furthermore, the LSTM can learn the time series sequence of DDoS network traffic continuously but learns the delicate difference between attacks and benign traffics based on the time window length section. A combination model LSTM-AE can learn delicate differences in sub-patterns between attacks and benign traffic while minimizing the reconstructed benign traffic to obtain a lower range reconstruction error. Our experimental results showed that the proposed LSTM-AE model achieves better performance in processing reconstruction-based time-series data than other comparable proposed models. The main contributions of our proposed model are as follows:

Summary of Original Contributions

- We propose a novel time-series anomaly detection architecture that leverages reconstruction-based LSTM-AE for efficient DDoS attack detection. In our proposed model, the LSTM networks are comprised of multiple LSTM units that work with each other to learn the long short-term correlation of data within a time series sequence. An autoencoder is used to identify the optimal threshold based on the reconstruction error rates evaluated across all time-series sequences. This threshold is used to identify anomalies.
- We apply our proposed LSTM-AE model against the reflection-based DDoS attacks - DNS, LDAP and SNMP. The model is trained on normal time-based traffic flow features using a subset of traffic flow information over a fixed-time window length.
- A novel anomaly score technique is proposed to calculate the MAE value of each traffic flow, which can be calculated flexibly based on different fixed-time window lengths.
- We performed tests on the state-of-the-art CICDDoS2019 dataset, and compared the performance of our proposed model with other similar approaches that use different aspects of LSTM and/or AE. Our experimental results, based on the comprehensive set of evaluation criteria, demonstrate that our proposed model can effectively detect anomalies reaching a detection accuracy exceeding 99%.

The rest of this Chapter is structured as follows: Section 3.2 introduces related works in the field of DDoS attack detection. Section 3.3 introduces our methodology. Section 3.4 illustrates the experimental setup and Section 3.5 details the analysis of our results evaluated on the various reflection-based attack types, including DNS, LDAP, and SNMP datasets. Section 3.6 concludes the paper with the planned future works.

3.2 Related Work

In recent years, anomaly detection has attracted extensive attention in literature exploring machine learning techniques. In this paper, we review the issues of variable length of DDoS anomaly detection, areas closely related to our contributions.

Sharafaldin et al. [74] generated a dataset titled CICDDoS2019 and classified benign traffic and attacks based on 4 machine learning methods, including ID3, Random Forest, Naïve Bayes, and Multinomial Logistic Regression. The evaluation result shows the highest accuracy from RF and ID3. Jia et al. [31] proposed an IoT DDoS defense technique named FlowGuard, and constructed LSTM and CNN techniques for DDoS identification and classification based on simulated data and CICDDoS2019 dataset to identify, classify, and mitigate DDoS attacks. They used a CNN to better classify all malicious flows, then employed the LSTM technique as an identification module. This was not only able to capture significant features of flows to identify benign samples but also to apply a softmax function on the output layer to distinguish between benign and malicious traffic. The above researches used flow-based statistical features, which were extracted from CICDDoS2019 dataset for DDoS attack detection.

Novaes et al. [56] introduced two scenarios for detecting anomalies and mitigating attacks on Software-Defined Networks (SDNs) using LSTM-FUZZY techniques. Firstly, the authors used an LSTM network semi-supervised learning technique to predict benign univariate time series behaviour of IP flows, followed by classifying attacks with a Fuzzy logic technique. There were two datasets used in this research, including the SDN dataset and the CICDDoS2019 dataset. Aydin et al. [12] proposed an LSTM-based system (LSTM-CLOUD) to detect and prevent DDoS attacks in a public cloud network environment through experimentation on CICDDoS2019 dataset. The authors built the LSTM models with two hidden layers, three dense and dropout layers, and used the sigmoid function to classify benign and DDoS attacks (anomalies). The model performed to a high accuracy rate of 99.83%, but this work only classified 3 attack types of attacks - UDP, MSSQL, SYN - as well as benign traffic samples.

Nezhad et al. [53] normalized two features (packets and source IP addresses) on a 1-minute time series interval by using a Box-Cox transformation. They then used a statistical time series analysis technique called ARIMA to predict the number of packets.

Ergan et al. [22] introduced LSTM-based neural networks to detect anomalies in a time series in an unsupervised manner. The author employs an LSTM-based network

technique to obtain the fixed-length sequence data. They utilized the OC-SVM and SVDD algorithms together with a scoring function to detect anomalies.

Salahuddin et al. [68] proposed a time-based Autoencoder technique named Chronos to detect DDoS traffic anomalies with aggregating features. One of the contributions in this research was the threshold they selected to highlight the efficiency of their anomaly detection system. They utilised threshold selection heuristic maximizes the F1 score. To detect anomalies effectively, they implemented various window sizes on different DDoS attacks. As a result, the proposed Chronos system achieves an F1-score of 99% for most attack types and over 95.86% for all attack type performance measurements by using two-time windows along with the selected heuristic threshold.

Fouladi et al. [23] used the CAIDA dataset to train and evaluate the K-SVD and BMP (Basic Matching Pursuit) algorithms and a SOM (Self-Organizing Map) model, using sparse coding and frequency domain for DDoS attack anomaly detection. They extracted normal time series data using a K-SVD algorithm and applied a BMP method to train and evaluate the benign data to estimate the sparse coefficients. They then used the normal data on the SOM model to obtain a SOM lattice. This enabled them to calculate the minimum Euclidean Distance between corresponding coefficients and use the SOM lattice to distinguish between normal and attack behaviors.

Although many researchers above have introduced statistical and machine learning techniques to detect anomalies in the DDoS datasets, it was also a challenging task worthy of further study, especially for distinguishing subtle differences in benign and attack traffics. In this research, we propose a hybrid deep learning reconstruction-based LSTM-Autoencoder model for anomaly-based DDoS attack detection. Our model uses a specific selection of features over multivariate time-series data analysis. Compared to the above statistical and machine learning-based studies, the differences in this research can be detailed as follows: 1) Our LSTM-AE neural network model conducts training and testing in an unsupervised manner (without labels), this is combined with the reconstruction error used to detect anomalies with several different time window lengths; 2) Anomaly detection in our proposed model gave experimental results for that DDoS attack anomaly detection with high accuracy on the CICDDoS2019 time-series dataset; 3) This experiment performed better in terms of the performance matrix - including precision, recall, and F1-score - than benchmark studies on the same CICDDoS2019 dataset, such as [74], which used machine learning techniques (including Random forest, Naive Bayes, etc.) to detect DDoS attacks.

3.3 LSTM-Autoencoder Anomaly Detection

3.3.1 Overview of Our Framework

This section provides an overview of our reconstruction-based time-series anomaly detection system. One of the DDoS traffic characteristics is collecting correlated temporal

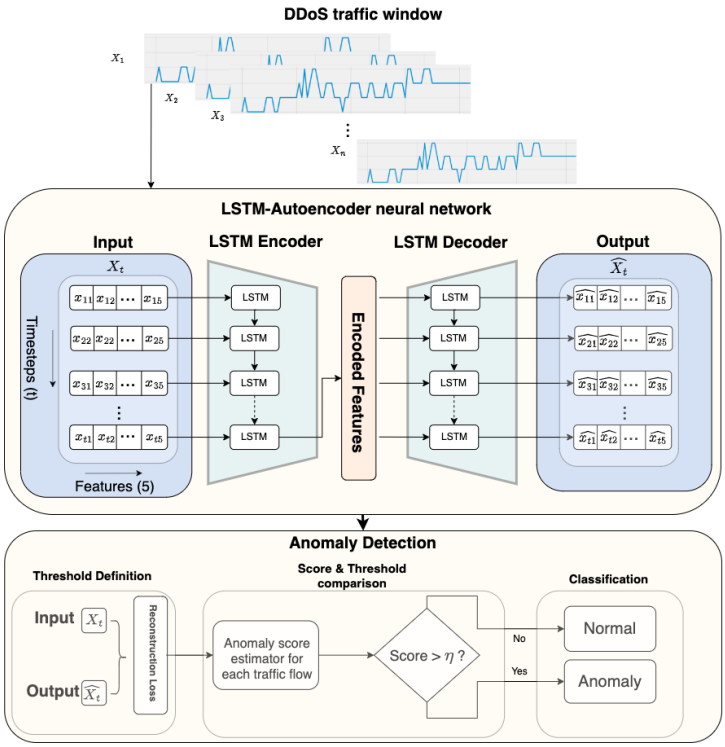


Figure 3.1: Overview of our proposed model

sequences, and the deep learning technique of LSTM is a good algorithm to deal with this temporal problem. Thus, we adopted the LSTM model for the LSTM-Autoencoder neural network for the Encoder and Decoder stages as its mechanism better captures DDoS flow information by feeding each flow at each time step. We propose a hybrid machine learning model which combines an LSTM neural network and an autoencoder. We apply our model to the multivariate time-series dataset CICDDoS2019.

As illustrated in Fig. 3.1, our LSTM-AE model builds the LSTM networks on the encoder and decoder schemes of an Autoencoder. The encoder obtains the sequence of the high-dimensional input data as a fixed-size vector. Using the memory cells of LSTM, the data processed by the encoder scheme retains the dependencies across multiple data points within a time-series sequence. This stage reduces the high-dimensional input vector representation into a low-dimensional representation. The decoder-LSTM reproduces the fixed-size input sequence from the reduced representation of the input data in the latent space, while reconstruction error rates determine the classification threshold. Fig. 3.1 depicts the operation of a model for DDoS anomaly detection. Phase 1 is data preprocessing based on the selected characterization of each traffic flow fed into the LSTM-AE neural network. Phase 2 is the training and testing process, in which the threshold is obtained based on minimizing the reconstruction error on benign traffic. The DDoS anomaly detection is the last phase in the diagram, which determines anomaly scores by

calculating the reconstruction error of each traffic flow between the time-series sequence's original input and the reconstructed output.

3.3.2 Feature usage and Input Sequence Data

Fig 3.1 shows an overview of our proposed LSTM-AE model. To obtain the DDoS traffic flow data, we used the public dataset CICDDoS2019 [74] from the Canadian Institute for Cybersecurity. In [74], the extracted network traffic features are stored in a CSV file, and each CSV file includes different DDoS attack types (anomalies) and benign traffic. In this research, we used three reflection-based DDoS attack types to detect anomalies, specifically DNS, SNMP and LDAP. As shown in the first part of the DDoS traffic window in Fig 3.1, all DDoS traffic flows are separated into n subsets based on a selected window length.

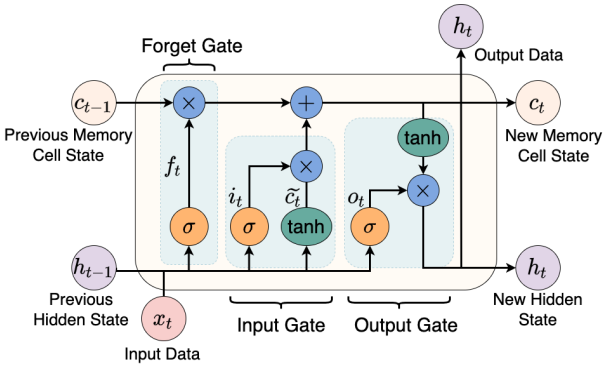
As our LSTM-AE model for detecting anomalies depends on learning the network traffic patterns, it is important to use high correlation features to obtain high performance in measurements such as accuracy. Using the research on the state-of-the-art CICDDoS2019 DDoS attack dataset in [74], the author specifies the importance of selecting the top-n features which correspond to the weight and mean value of different attack types. In this research, we have also used these top-n features as depicted in [74] from the CICDDoS2019 dataset. As such 'Max Packet Length', 'Fwd Packet Length Max', 'Fwd Packet Length Min', 'Average Packet Size', 'Min Packet Length' is used as selected features for our multivariate time-series anomaly detection. After feature selection, the input data has to be reshaped into a 2-dimensional vectors before the data is fed into the required LSTM encoder input layer. The original DDoS dataset is comprised of a series of time sequence $[X_1, X_2, X_3, ..., X_n]$. Each sequence X with a fixed T-length time window data $[x_1, x_2, x_3, ..., x_t]$ is created where $x_t \in R^m$ represents an m-features input at time-instance t. They are then reshaped into 2-D (2-dimensional) arrays, representing samples and time steps. For example, a sequence of the DDoS attack data is converted into a 2-D array where each dimension indicates the list of samples at 10 time steps with n features.

3.3.3 LSTM-AE Model Architecture

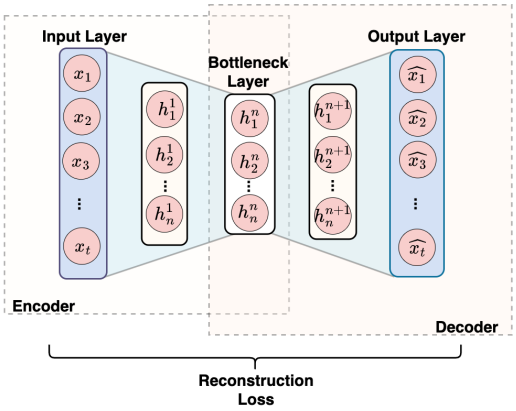
3.3.3.1 Long Short Term Memory

The LSTM network is a variation of a Recurrent Neural Network (RNN) that addresses the gradient vanishing and exploding problems of RNNs and can process long term sequences between data samples at any given time from many history time steps. The architecture of the LSTM network is suitable for processing time series sequence data and provides the capability of forgetting the historical data from each memory cell before updating the memory cell with new data. As illustrated in Fig. 3.2(a), the LSTM network contains memory cells c_t and multiple gate units, including the forget gate f_t, the input

gate i_t and the output gate o_t. These three gate units control the state of memory cells. For example, at time step t, the forget gate decides which bits of the cell state are useful given both the previous hidden state and new input data. The LSTM network can omit insignificant information (values) from the cell state of the forget gate and can recognize significant information (values) to keep and update in the cell state. Using this network architecture, the LSTM model performs to a high standard when capturing long-term patterns in time series sequence data.



(a) LSTM



(b) AE

Figure 3.2: Neural Networks architectures: (a) LSTM; (b) Autoencodre.

3.3.3.2 Autoencoder

An Autoencoder (AE) is an unsupervised neural network model that is not only used for feature selection and dimension reduction but also can be used for reconstruction-based Encoder-Decoders to detect anomalies. The typical architecture of an autoencoder is composed three components: an input layer, one or more hidden layer(s), and an output layer. The operations of an autoencoder for detecting anomalies can be divided into Encoding, Decoding, and Reconstruction Loss as illustrated in Fig. 3.2(b). The encoder

compresses the high dimension input data and maps it to low dimensional representations h, in the bottleneck layer while the decoder decompresses the encoded representation and reconstructs to the output $\hat{x}$. Typically, the autoencoder is trained by utilizing the MAE equation as a loss function to minimize the reconstruction error between the output $\hat{x_t}$ and input x_t.

3.3.3.3 LSTM-Autoencoder

We have addressed the issue of anomaly detection in capturing normal phenomena through time-based traffic flow from the DDoS attacks dataset. An LSTM-AE model need not only be applied to address the problem of feed-forward neural networks but also may be utilized to learn patterns in time-based sequence data, making them suitable for time-series anomaly detection [65,98]. Our proposed LSTM-AE model includes an autoencoder that utilizes the LSTM network as a hidden layer in both the Encoder and Decoder schemes. This is followed by an Encoded Features layer and an output layer respectively, which calculate the reconstruction error to detect anomalies. The role of the LSTM network in our proposed scheme is to learn the patterns of data based on time-series DDoS signals. We combine this with an autoencoder to learn the best encoder-decoder scheme to detect anomalies. The output of the LSTM Encoder and Decoder are then compared with the original input data and the reconstruction error is backpropagated through the architecture to update the weights of the neural network.

Our LSTM-AE model employs an autoencoder that utilizes the LSTM network as a hidden layer in both Encoder and Decoder schemes followed by an Encoded Features layer and an output layer respectively. These calculate the reconstruction error to detect anomalies. The role of the LSTM network in our proposed scheme is to learn the patterns of data based on selected time window length sequences. Our proposed LSTM-AE is composed of six layers - an Input, an LSTM Encoder, an LSTM Decoder, a RepeatVector, a TimeDistributed and an Output layer.

The LSTM Encoder obtains the sequence of high-dimensional input data as a fixed-size vector and compresses the input data into a low-dimensional hidden representation. Within an LSTM cell, for an input time series sequence data $X_1 = [x_1, x_2, ..., x_t]$ where t represents the time steps, each x_t calculation is performed using the following:

$$f_t = \sigma(w_f[H_{t-1}, x_t] + b_f) \tag{3.1}$$

$$i_t = \sigma(w_i[H_{t-1}, x_t] + b_i) \tag{3.2}$$

$$\tilde{C_t} = tanh(w_c[H_{t-1}, x_t] + b_c) \tag{3.3}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C_t} \tag{3.4}$$

$$o_t = \sigma(w_o[H_{t-1}, x_t] + b_o) \tag{3.5}$$

$$H_t = o_t \odot tanh(C_t) \tag{3.6}$$

where

- f_t represents the forget gate, i_t, $\tilde{C}_t$ and C_t represents the input gate, and o_t and H_t represents the output gate.
- w and b are the weights and the bias of the forget gate, input gate and output gate.
- H_{t-1} and x_t present the concatenation of the hidden state and the current input respectively.
- σ is the activation function of each gate, and it outputs numbers in the range of [0, 1].
- $\odot$ represents element-wise multiplication.

Next, after compressing input data into low-dimensional representation until it reaches the latent space (encoded features), all data presentation can be repeated t times on the RepeatVector layer to feed into the LSTM Decoder layer. In this LSTM Decoder layer, the decoder scheme uses the same number of features (equal to the encoder features on the LSTM Encoder layer) to map the latent space representation back to a high-dimensional representation. We add a TimeDistributed layer in order to generate the output of the LSTM Decoder in time sequence.

Encoder: To illustrate the LSTM encoder stage, if the time step is set to 10 - seen LSTM cells working theory at the time step t (show in Fig 3.2a) - the input includes the previous output of the hidden state (h_{t_1}) and the cell state (C_{t_1}), and the current input x_t while output includes the new hidden state (h_t) and the new cell state (C_t), and an output y_t. At the time step $t+1$, the new input becomes a new input x_{t+1} for the next set of cells, and the output is obtained from the last hidden state and cell state of the LSTM cell. This then becomes the new input of hidden state (h_t) and the cell state (C_t) information. Note that we discard the output (y_t) of each LSTM cell at each time step t in the Encoder and preserve the initial state which is from the hidden state and cell state. Finally, the output of the last time step (10) is the hidden state (h_{10}) and the cell state (c_{10}), which is a summary of the entire 10 time steps. Therefore, the input vector of LSTM encoder is 10×5, where 10 is the time steps (or time window length) and 5 is the number of features. If we set the number of input features (units) of LSTM to 16, after passing through the LSTM Encoder layer the size of the output vector is 1×16.

Our LSTM Encoder architecture (see Fig. 3.1: LSTM-Autoencoder neural network) is defined as follows:

- Each traffic flow in the Input layer has been reshaped into a 2D matrix. Note that each traffic flow is represented by a matrix $n \times t$, where n represents the traffic flows, and t represents the time steps. This form of a series of input data is able to capture DDoS traffic patterns based on the sliding window length.

- Each LSTM Encoder layer consists of 16 LSTM units with "tanh" activation functions.
- We added a dropout layer (0.2) to prevent over-fitting in the Encoder part.
- The encoded features have lower dimensions than the number of input features.

Decoder: Between the Encoder and Decoder layer, we added a RepeatVector Layer to create the copies of the 1×16 vector equal to the number of time steps, which we called Encoded Features. For example, the size of time steps in our model is 10, therefore this layer will create 10 copies of the encoded features as a two-dimensional vector 10×16. The output of this layer becomes the input of the LSTM Decoder at each time step, which means each copy of the encoded features at each time step will be the input of the next set of LSTM cells. As depicted, the difference of LSTM network between the Encoder and Decoder is that the output of each LSTM cell at each time step (y_t) in the Decoder cannot be discarded and are outputted as y_t. There are two reasons to get the output of each of the LSTM cells: firstly, for the added layer, TimeDistributed, the input can be a 3-dimensional vector, which means the output from the LSTM cells has to be a 3-dimensional vector; and second, to ensure the output of each time step is as close as possible to the input. Therefore, the LSTM Decoder representations obtained in low-dimensionality encoding are used as input in the decoder and there are utilised to reproduce the original input data using the LSTM network. This means the output from the last set of LSTM cells (copied t times) then becomes the input to the LSTM Decoder network.

Each 1×16 set is fed as an input to the decoder which creates a 3 Layer network with 10 LSTM cell units. Each LSTM cell unit processes each 1×16 encoded feature. These LSTM units produce an output that represents the result of the learning from the encoded feature where the output is multiplied with the 1×16 vector created by the additional TimeDistribution layer. At the same time, each LSTM cell unit produces a second output containing the state of the data processed by the current LSTM cell which is passed to the next LSTM - with the exception of the last LSTM unit. The matrix multiplication between the output of each LSTM layer (L) (10×16) and the TimeDistributed layer (16×1) results in a vector with of size 10×1 - the same as the size of the input.

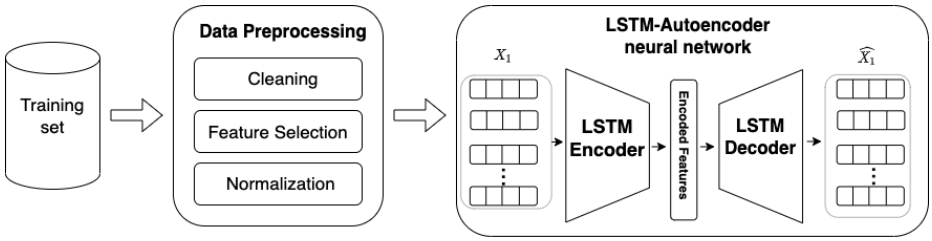
The LSTM Decoder (see Fig. 3.1: LSTM-Autoencoder Neural Network) architecture is defined as follows:

- The encoded feature in the bottleneck layer will be the input of the LSTM Decoder.
- An LSTM Decoder layer consists of 16 LSTM units with "tanh" activation functions.
- We added a dropout layer (0.2) to prevent overfitting in the Decoder section.

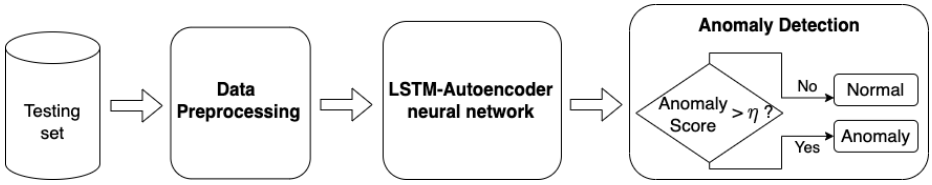
- Typically, the output of LSTM Decoder has two outputs, including the output data (traffic flows) (O_t) and the new hidden state (H_t). The output of H_t can be discarded. Thus, the output from the decoder part is the reconstructed feature of the same size and dimension as the input data.

The final aim of LSTM-AE is to reconstruct the input from the output, i.e. $\hat{X_1} \approx X_1$ where X_1 indicates the input while $\hat{X_1}$ indicates the output.

LSTM-AE Training: We divide the CICDDoS2019 dataset into a training set (70%), a validation set (10%), and a testing set (20%). Note that the training and validation set consists of benign samples only, in order to train the proposed LSTM-AE model. The testing set consists of both benign and attack samples for detecting anomalies. The architecture of the proposed LSTM-AE is designed to minimize the reconstruction error between the original input and the reconstructed output based on time series sequence traffic flows. This LSTM model aims to learn the patterns in the data. Our motivation for using the Encoder and Decoder scheme to detect anomalies is the fact that their working scheme is able to detect anomalies based on the low distribution of normal data. As such, our training data set consists of only normal sequence data and this is utilized for training the proposed LSTM-AE model. The LSTM-AE is taught normal traffic behavior, using the benign samples. Our proposed LSTM-AE is an Encoder-Decoder unsupervised learning model, and no label is provided in the training phase. The training process is depicted in Fig. 3.3(a). To address the problem of overfitting, dropout was added in both the encoder and decoder stages and set to 0.2. For the other settings in our model, "Adam" is the optimization method, "tanh" is used as the activation function, and "MAE" can be chosen as a loss function.



(a) LSTM



(b) AE

Figure 3.3: Training and testing phase

LSTM-AE Testing

Fig. 3.3(b) shows the details of how anomalies can be detected using the reconstruction-based anomaly detection method. Here, the maximum value of MAE from the training process can be designated as the threshold. A reconstruction error rate for each data point from the testing set is compared with this threshold. If the reconstruction loss value is greater than the threshold η, this data point is noted as an anomaly, otherwise, it is designated to be normal. This is shown in the following Equation 3.7.

$$X' = \begin{cases} X'_i \text{ is anomalies}, & \text{if } Score > \eta \\ X'_i \text{ is normal}, & \text{otherwise} \end{cases} \tag{3.7}$$

where X' indicates a reconstructed time-series, X'_i is a data point contained in the time-series, and a *score* is a result of a reconstruction loss function using MAE.

3.3.4 Reconstruction-based anomaly detection

In order to effectively learn time series DDoS traffic behavior using reconstruction-based anomaly detection, our LSTM-AE model is trained with a dataset that contains only benign traffic flows from CICDDoS2019.

3.3.4.1 LSTM-AE for anomaly detection

An anomaly can be defined as an observation diverging from the majority of the data. A threshold can be set as a decision point to determine how much an observation deviates. Any observations that exceed the threshold are defined as anomalies. To better demonstrate the functionality of reconstruction-based time series anomaly detection, the LSTM-AE model can be applied to detect anomalies for each input sequence. This allows us to obtain the reconstruction error rates associated with the benign samples of the DDoS dataset. A backpropagation methodology is applied to adjust the weights and parameters of the model. We use the Mean Absolute Error (MAE) algorithm, as shown in Equation 3.8, as the reconstruction error (loss) function.

$$Loss(MAE) = \frac{\sum_{i=1}^{n} |x_i - \hat{x_i}|}{n} \tag{3.8}$$

where n indicates the total number of samples, x_i is the representation of the original input being fed to the encoder, and $\hat{x_i}$ is the output produced by the decoder.

Once training is done and the reconstruction error is computed on all samples, the LSTM-AE model learns a low MAE and sets the maximum reconstruction error as a threshold. By contrast, if the testing set presents different behavior from the training process, the resulting MAE will be greater than the threshold and can be considered an anomaly. To the best of our knowledge of the CICDDoS2019 dataset, there are a few benign samples in each provided CSV file. In order to learn the normal behaviors of traffic

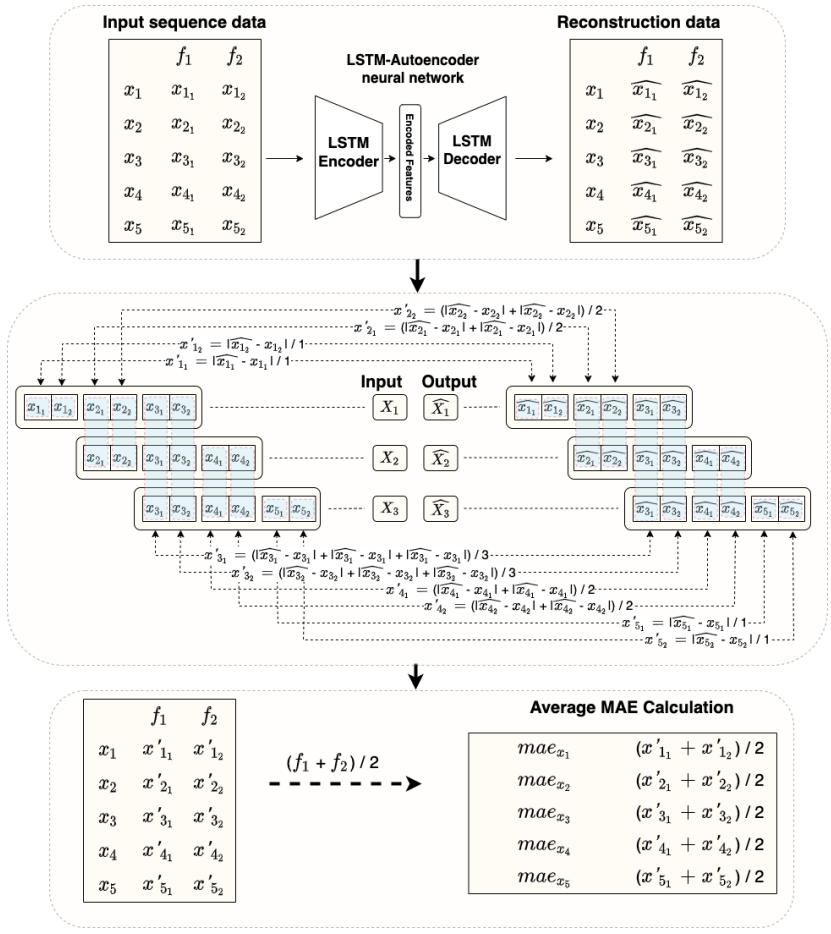


Figure 3.4: Computing reconstruction error on multivariate time series

flows, we extract all benign samples from the CICDDoS2019 dataset, and designate 80% of the benign traffic as the training set, while the remaining 20% of the benign traffic can be combined with different attack types to be used in the testing set.

3.3.4.2 Reconstruction Error threshold calculation

Our proposed reconstruction-based LSTM-AE model was trained in an unsupervised manner and aimed to minimize reconstruction error between input and output while using unlabeled data. In [90], the univariate time series LSTM-AE anomaly detection is performed using reconstruction error, shown in Fig. 3.4. The figure illustrates how to calculate reconstruction error for each sample contained in the different multivariate time series sequences. Suppose that there is a dataset containing 5 data samples with 2 features (shown as Input sequence data) which are 3 time-series sequences of $[X_1, X_2, X_3]$ where each sequence contains 3 samples with 2 features over 3 different timesteps. For example, the first sequence $X_1 \in \{[x_{1_1}, x_{1_2}], [x_{2_1}, x_{2_2}], [x_{3_1}, x_{3_2}]\}$, $X_2 \in \{[x_{2_1}, x_{2_2}], [x_{3_1}, x_{3_2}],$

$[x_{4_1}, x_{4_2}]\}$, and $X_3 \in \{[x_{3_1}, x_{3_2}], [x_{4_1}, x_{4_2}], [x_{5_1}, x_{5_2}]\}$. Our model trains these 3 time series of sequences as inputs and constructs the outputs that map to each sequence $\hat{X_1}$, $\hat{X_2}$, and $\hat{X_3}$.

Using Equation 3.8, we can obtain the MAE value of each data sample for each feature. In order to obtain the MAE value of each data sample over multivariate features, we use the average MAE of each data sample for each feature. For example, in 3 time-series sequences of $[X_1, X_2, X_3]$. As such, we obtain the reconstruction error of each data sample of each feature. As can be seen in Fig. 3.4, with the assumption that each data sample has two features, we can calculate each data sample for each feature using time steps. This figure shows the reconstruction error calculation for each feature when the time step is 3. As an example of one of the data samples of feature 1 (x_{3_1}), and the reconstruction error of x_{3_1}, the calculation is $(|\hat{x_{3_1}} - x_{3_1}| + |\hat{x_{3_1}} - x_{3_1}| + |\hat{x_{3_1}} - x_{3_1}|)/3$, and is defined as x'_{3_1}. Similarly, the reconstruction error of feature 2 (x_{3_2}) is $(|\hat{x_{3_2}} - x_{3_2}| + |\hat{x_{3_2}} - x_{3_2}| + |\hat{x_{3_2}} - x_{3_2}|)/3$, and is defined as x'_{3_2}. Note that the reconstructed data sample for each feature can be slightly different, which means each x_{3_1} in X_1, X_2, and X_3 are different. The x_{3_2} is presented with the same definition. After we obtain the reconstruction error of these two features, the total reconstruction error of x_3 is calculated by using the average value of features 1 and 2, illustrated as $MAE_{x_3} = (x'_{3_1} + x'_{3_2})/2$ where 2 is presented two features of x_3. When we obtain all the average reconstruction errors for each data sample on the training set, the maximum average reconstruction error is set as the threshold. During testing, any samples whose reconstruction error goes beyond this maximum average reconstruction error are therefore labeled as anomalies.

3.4 Data and Methodologies

3.4.1 CICDDoS2019 Dataset

In this study, we use the CICDDoS2019 [74] dataset which is widely used for DDoS attack detection and classification. The dataset contains a large number of up-to-date realistic DDoS attack samples as well as benign samples. The total number of records contained in the CICDDoS2019 dataset is depicted in Table 3.1.

Table 3.1: The number of records in CICDDoS2019

dataset	**total**	**benign**	**malicious**
Training day	50,063,112	56,863	50,006,249
Testing day	20,364,525	56,965	20,307,560

The CICDDoS2019 dataset contains different DDoS attack types that exploit a wide range of network and application protocols. In our study, we used DDoS attack types (i.e., DNS) and benign traffic samples to train and test our proposed LSTM-AE model. The dataset is broken down as follows:

- **Benign**: Benign traffic based on HTTP, HTTPS, FTP, SSH, and email protocols.
- **Attacks**: These DDoS attacks cover two different categories, Reflection-based and Exploitation-based. In terms of our CICDDoS2019 dataset, any traffic included in MSSQL, SSDP, NTP, TFTP, DNS, LDAP, NetBIOS, and SNMP is categorised as a reflection-based attack. Traffic labeled as SYN, UDP, and UDP-lag in CICDDoS2019 belongs to the exploitation-based category.

All CICDDoS2019 data samples from the training day set are depicted in Table 3.2. Note that all data samples can be counted from different "CSV" files, which are collected and saved based on their various attack types. Note that the "WebDDoS" attack was collected and saved together with the "UDPLag" attack file.

Table 3.2: Three reflection-based attack types on training day

Attack Type	Malicious	Benign	Total Flow count	Attack times
DNS	5,071,011	3,402	5,074,413	10:52 - 11:05
LDAP	2,179,930	1,612	2,181,542	11:22 - 11:32
SNMP	5,159,870	1,507	5,161,377	12:12 - 12:23

The high-level description of the nature of the DDoS attack used in our study is summarised in Table 3.3.

Table 3.3: Selected attack types in CICDDoS2019

Attack Type	**Attack Description**
DNS Attack	DNS attacks are a type of amplification DDoS attack that exploits Domain Name Servers and exhausts the bandwidth of the victims. This attack can overwhelm the victims and make them inaccessible.
LDAP Attack	LDAP attacks are a DDoS attack associated with exploiting Lightweight Directory Access Protocol (LDAP) protocol. The attacker sends a massive number of LDAP requests to the vulnerable LDAP servers by pretending to be a legitimate LDAP client using spoofed IP addresses. The LDAP server becomes too busy to create responses for attackers and becomes unable to respond to real LDAP clients.
SNMP Attack	SNMP attacks are a volumetric DDoS attack that stands for Simple Network Management Protocol (SNMP), and these attacks aim to generate a large number of SNMP attacks utilizing a spoofing IP address that directed to the victims from multiple networks.

In this dataset, there are 88 features in total, and the best top 5 features of each attack type and benign traffics have been used based on the RandomForestRegressor class of scikit-learn which can calculate the importance of each feature in the dataset [74]. Table 3.4 shows the top 5 important features which are employed in this research as well as a brief description.

Table 3.4: Selected features in CICDDoS2019

Feature Name	Description
Max Packet Length	The maximum length of a flow
Fwd Packet Length Max	The maximum size of packets in the forward direction
Fwd Packet Length Min	The minimum size of packets in the forward direction
Average Packet Size	The average size of packets
Min Packet Length	The minimum length of a flow

3.4.2 Data Pre-processing

In this section, we discuss the methodologies we used to process our dataset in order to feed it into our proposed LSTM-AE model.

3.4.2.1 Data Cleaning

The original dataset contained 88 features. As suggested by [74], they depicted the top 5 most important features of each attack type and benign samples based on weight and mean value calculated. According to these top 5 significant features which were selected, we also use these top-5 features as our features among different attack types. For example, we used DNS, LDAP, and SNMP as our analysis attack types, which provide three attack types (anomalies) as well as benign (normal) samples. We chose these three attack types because they shared the same top 5 important features based on their weight and the mean value calculation. There are five important features to be used in this research: "Max Packet Length", "Fwd Packet Length Max", "Fwd Packet Length Min", "Average Packet Size", and "Min Packet Length".

3.4.2.2 Label Encoding

We substituted the categorical labels for deep models as they only operate on float/numerical values. One categorical value which was converted was the attack label (benign or an attack type). For label encoding, "0" indicates a benign (normal) sample, while "1" indicates an attack type (anomalies).

3.4.2.3 Data Normalization

There are several widely used methods to perform feature scaling, including Z Score, standardization, and normalization. As proposed by [82], we utilize the MinMax-based normalization for our feature scaling. This method maps the original range of each feature into a new range using Equation (3.9), as follows:

$$Z_i = \frac{Z_i - min}{max - min} \tag{3.9}$$

where Z_i denotes all the normalized numeric values ranging between [0-1]; max and min denote the maximum and minimum values from all data points.

3.5 Experimental Results

Table 3.5: Implementation environment specification

Unit	**Description**
Processor	3.4GH$_z$ Inter Core i5
RAM	16GB
OS	MacOS Big Sur 11.4
Packages used	tensorflow 2.0.0, sklearn 0.24.1

3.5.1 Experiment Setup

Our experiments were carried out using the system setup shown in Table 3.5.

The hyperparameters used in the training phase are described (with the values for each parameter) complete with description in Table 3.6.

Table 3.6: LSTM-AE training parameters

Hyperparameters	Values	Descriptions
Learning rate	0.001	Learning speed (within range 0.0 and 1.0)
Dropout	0.2	No. of neurons ignored
Batch size	64	No. of samples in one fwd/bwd pass
Epoch	30	No. of one fwd/bwd pass of all samples

3.5.2 Performance Matrix

To evaluate the performance of our model, we used the following metrtics: classification accuracy, precision, recall, and F1 score. Table 3.7 illustrates the confusion matrix.

Table 3.7: Confusion Matrix

Total Population		Predicted Condition	
		Normal	Anomaly
Actual Condition	Normal	TN	FP
	Anomaly	FN	TP

where;

- True Positive (TP) indicates an anomalous data point correctly classified as anomalous.
- True Negative (TN) indicates a normal data point correctly classified as normal.
- False Positive (FP) indicates a normal data point incorrectly classified as anomalous.
- False Negative (FN) indicates an anomalous data point incorrectly classified as normal.

Based on the aforementioned terms, the evaluation metrics are calculated as follows:

$$TPR(TruePositiveRate/Recall) = \frac{TP}{TP+FN} \tag{3.10}$$

$$FPR(FalsePositiveRate) = \frac{FP}{FP+TN} \tag{3.11}$$

$$Precision = \frac{TP}{TP+FP} \tag{3.12}$$

$$F1-score = 2 \times \left(\frac{Precision \times Recall}{Precision + Recall}\right) \tag{3.13}$$

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{3.14}$$

The Area Under the Curve (AUC) computes the area under the Receiver Operating Characteristics (ROC) curve which is plotted using the true positive rate on the y-axis and the false positive rate on the x-axis over different thresholds. Mathematically, the AUC is computed as shown in Equation (3.15).

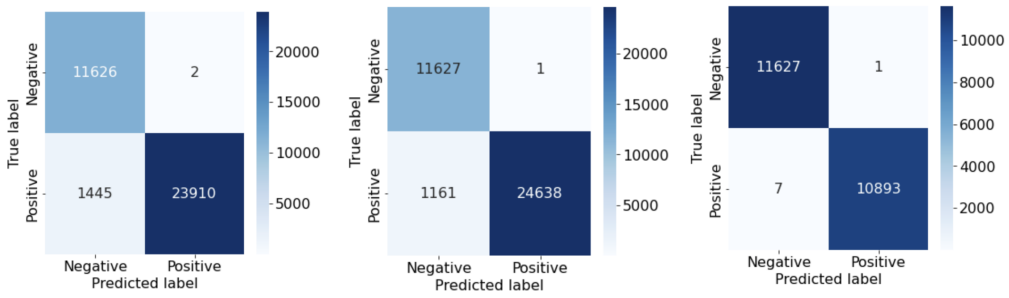
$$AUC_{ROC} = \int_0^1 \frac{TP}{TP+FN} d\frac{FP}{TN+FP} \tag{3.15}$$

Table 3.8: Three attack types' performance based on different time window lengths

Attacks	w = 10ms				w = 50ms				w = 100ms			
	Acc	Pre	Re	F1	Acc	Pre	Re	F1	Acc	Pre	Re	F1
DNS	96.08	99.99	94.30	97.06	95.78	99.99	93.85	96.82	95.83	99.99	93.92	96.86
LDAP	99.96	99.99	99.93	99.96	98.68	99.99	97.28	98.61	99.77	99.99	97.47	98.71
SNMP	96.89	99.99	95.49	97.69	96.86	99.99	95.45	97.67	96.82	99.99	95.40	97.64

Table 3.9: Performance comparison between the experimental results and three attack types on the CICDDoS2019 dataset

Algorithms		Confusion Matrix			
		Acc	Pre	Re	F1
ID3 [74]		-	0.78	0.65	0.69
RF [74]		-	0.77	0.56	0.62
Naive Bayes [74]		-	0.41	0.11	0.05
Logistic regression [74]		-	0.25	0.02	0.04
Our model	DNS	0.96	0.99	0.94	0.97
	LDAP	0.99	0.99	0.99	0.99
	SNMP	0.96	0.99	0.95	0.97



(a) DNS Attack (b) SNMP Attack (c) LDAP Attack

Figure 3.5: Performance of anomaly detection on different DDoS attack types using the Confusion Matrix

3.5.3 Results and Evaluations

3.5.3.1 Training and testing dataset

The CICDDoS2019 data collection was based on millisecond intervals. We first select 0.5% of the data based on the original millisecond time interval. However, after selection, there is a large number of malicious (anomalies) samples with only a few benign samples selected. As mentioned previously, our LSTM-AE model only uses benign samples in the training stage. Thus, to solve this issue, we first extracted all the benign samples from the training day collection and separated them into training (80%) and testing (20%) sets. For the model training, we only use 80% benign samples to train, while the rest of the benign samples can be used in the testing set. Moreover, We used 0.5% of the original CICDDoS2019 dataset for each attack type from the training day collection for testing as it was not feasible to use the full dataset due to performance considerations. Note that each testing set includes benign samples and different attack types.

We experimented with three attack types in the LSTM-AE model to get initial anomaly detection results based on the top 5 most important features for each attack type. Table 3.9 shows the initial anomaly detection results for the three attack types: DNS, LDAP, and SNMP. The results show that the anomaly detection for LDAP attacks performed the best in terms of accuracy, precision, recall, and F1-score, based on the mentioned top 5 features.

Figure 3.5 illustrates the number of records classified for anomaly detection performance using the three DDoS attack types.

3.5.3.2 Influence of time window length

Table 3.8 shows the performance of all three attack types based on different time window lengths. The results show that performance was best for detecting LDAP attacks over diverse time window lengths, with over 99% in accuracy and precision, recall, and F1-score, compared with other attack types as shown in Table 3.8. As can be seen in Table 3.8,

as the time window length increased, the accuracy decreased for all three attack types. The highest accuracy was achieved at over 96% for a time window length of 10. We also compared the results with the baseline of other machine learning techniques from [74], in which the same features are used for testing purposes. In the comparison table (Table 3.9), the evaluation results show our model performed impressively in terms of accuracy, precision, recall, and F1-score.

3.5.3.3 Results of batch size

The testing results were evaluated based on different batch sizes while dropout, learning rate, and epoch remained constant.

DNS attack: For DNS attacks (Table 3.10), the highest accuracy was 96.08% and required a relatively brief computational time (approximately 12s per epoch) with the batch size set to 64. We found that the smallest batch size value tested - size 10 - resulted in the lowest accuracy (88.83%) and required a relatively high computation time at approximately 42s per epoch.

Table 3.10: Batch size performance metrics for detecting DNS attacks

Batch size	Acc	Pre	Re	F1	AUC-ROC	Time (s) $(\mu \pm \sigma)$/epoch
10	88.83	100	83.71	91.13	91.58	42 ± 1.24
32	93.49	100	90.51	95.02	95.26	12 ± 1.03
64	96.08	99.99	94.30	97.06	97.14	12 ± 0.94

SNMP attack: In the case of SNMP attacks as in Table 3.11, analyzing the performances for each batch size demonstrated that the best result was achieved when the batch size was increased to 64. Comparing this with a smaller batch size of 10, all performance metrics showed significant improvement at the larger batch size, particularly computational time. As can be seen in Table 3.11, a batch size of 10 (38s) takes approximately three times longer than a batch size of 64 (12s), per epoch.

Table 3.11: Batch size performance metrics for detecting SNMP attacks

Batch size	Acc	Pre	Re	F1	AUC-ROC	Time (s) $(\mu \pm \sigma)$/epoch
10	88.47	99.99	83.29	90.88	91.64	38 ± 1.85
32	96.84	99.98	95.44	97.66	97.71	14 ± 1.31
64	96.89	99.99	95.49	97.69	97.75	12 ± 0.96

LDAP attack: The performance of LDAP attacks for each hyperparameter of batch size is presented in Table 3.12. When analyzing the performance of each batch size, we can see that the LDAP attacks are detected with impressive performance across the different batch sizes, achieving over 98%. However, a batch size of 10 (42s) takes significantly longer (over three times greater) than a batch size of 10 (12s) per epoch.

Table 3.12: Batch size performance metrics for detecting LDAP attacks

Batch size	Acc	Pre	Re	F1	AUC-ROC	Time (s) $(\mu \pm \sigma)$/epoch
10	98.82	99.99	97.57	98.76	98.78	42 ± 1.93
32	98.55	99.99	97.02	98.48	98.51	14 ± 0.80
64	99.96	99.99	99.93	99.96	99.96	12 ± 0.82

3.5.3.4 Results of learning rate

The learning rate of the "Adam" optimizer is set at three candidate rates: 0.01, 0.001, and 0.00001. This was tested over the three different attack types, while dropout and epoch values remained constant.

DNS attack: The results of the DNS attacks are shown in Table 3.13, with different learning rates. The highest performance metrics for accuracy, recall, and F1-score was achieved at 96.08%, 94.30%, and 97.06% respectively. These results were all using the learning rate value of 0.001. On the other hand, the smallest learning value at 0.00001 resulted in the lowest accuracy (95.48%), recall (93.41%), and F1-score (96.70%). While selecting different learning rates has an impact on the results, the time taken is not much different per epoch.

Table 3.13: Learning rate performance metrics for detecting DNS attacks

Learning Rate	Acc	Pre	Re	F1	AUC-ROC	Time (s) $(\mu \pm \sigma)$/epoch
0.001	96.08	99.99	94.30	97.06	97.14	10 ± 0.99
0.0001	95.80	99.99	93.88	96.84	96.94	10 ± 0.84
0.00001	95.48	99.99	93.41	96.59	96.70	10 ± 1.05

SNMP attack: In examining the performance of detection on the SNMP attack in Table 3.14, we can see that the best performance for accuracy was achieved at 96.89% using the learning rate of 0.001. Moreover, the results of all performance metrics decrease slightly as the learning rate changed from 0.001 to 0.00001. The processing time did not vary significantly due to the changes in the learning rate.

Table 3.14: Learning rate performance metrics for detecting SNMP attacks

Learning Rate	Acc	Pre	Re	F1	AUC-ROC	Time (s) $(\mu \pm \sigma)$/epoch
0.001	96.89	99.99	95.49	97.69	97.75	12 ± 0.96
0.0001	96.82	99.99	95.39	97.63	97.69	12 ± 0.99
0.00001	96.88	99.99	95.48	97.68	97.74	12 ± 1.05

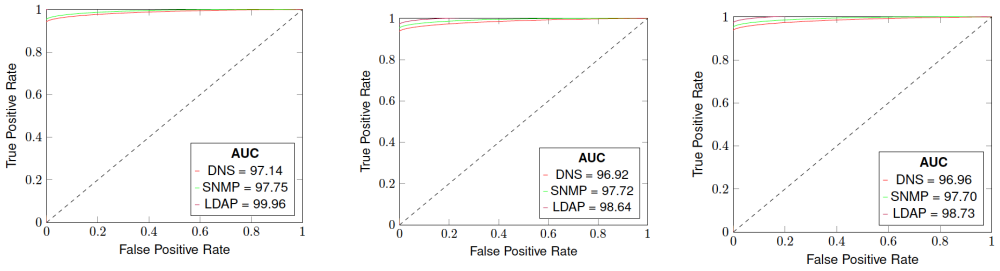
LDAP attack: As shown in Table 3.15, the LDAP attack performed the best over our experiments when compared to the other attack types. The results of detection overall metrics - accuracy, precision, recall, F1, and AUC-ROC - provided the smallest variation over LDAP attacks using the different learning rates, all scores being over 99%. The

processing times showed there was a slightly greater cost using the smallest learning rate of 0.00001.

The impact on the different learning rate configurations presents little variation, but a learning rate of 0.001 gave the best results for accuracy. Similarly, the lowest value for the learning rate took a long time to converge, which results in a significantly longer time for computation per epoch.

Table 3.15: Learning rate performance metrics for detecting LDAP attacks

Learning Rate	Acc	Pre	Re	F1	AUC-ROC	Time (s) $(\mu \pm \sigma)$/epoch
0.001	99.96	99.99	99.93	99.96	99.96	11 ± 0.91
0.0001	99.94	99.99	99.89	98.94	99.95	11 ± 0.90
0.00001	99.96	99.99	99.93	99.96	99.96	12 ± 0.94



(a) time window length: 10 (b) time window length: 50 (c) time window length: 100

Figure 3.6: AUC-ROC visualization

Figure 3.6 shows the AUC-ROC for anomaly detection over our three DDoS attack types using different time window length selections. Note that all three attack types achieved over 97% for the AUC, and that the highest value for the AUC-ROC was achieved for a time window length of 10. Our experimental evidence shows that the proposed model offers significant potential to detect DDoS attacks effectively.

Table 3.16 shows the performance comparison for our proposed LSTM-AE model with other similar methods from shallow machine learning and deep learning-based approaches. As the results demonstrate, our approach offered the best performance of LDAP in terms of all aspects of evaluation metrics, reaching an average of 99.96% accuracy while precision, recall, and F1-score all remain very competitive at 99.99%, 99.93%, and 99.96% respectively.

3.6 Conclusion

In this study, we demonstrate that DDoS attacks can be detected with high accuracy using a combination of multiple deep learning-based techniques. Our proposed reconstruction-based LSTM-AE anomaly detection model leverages the benefits of an LSTM model

Table 3.16: Comparison of CICDDoS2019 with different algorithms

Paper	Techniques	Accuracy	Precision	Recall	F1-score
[56]	LSTM-Fuzzy	-		93.13	
[14]	DDoSNet 24 features	-	91.12	72.91	74.00
	DDosNet 82 features + FS	-	91.16	79.41	79.39
[21]	DDosNet (RNN+AE)	99	99	99	99
[41]	K-means + Active Learning Method	90	-	-	95
[7]	CNN + BiLSTM	94.52	94.74	92.04	93.44
[31]	LSTM	98.9	99.47	99.31	99.35
[25]	LSTMED	89.7	44.8	88.3	59.4
[72]	LSTM(scenario II)	88.5	88.1	87.8	87.0
This study	LSTM-AE	99.96	99.99	99.93	99.96

and an Autoencoder in order to detect DDoS traffic anomalies. We use LSTM networks consisting of multiple LSTM units that work with each other to learn the long short-term correlation of DDoS traffic within a time series sequence. An Autoencoder is employed to identify the optimal threshold based on the reconstruction error rates. This can be used to identify anomalies in traffic. We have demonstrated the impact of different window lengths for classifying anomalies over different DDoS attack types. Our proposed model offers potential as an effective DDoS defense tool to assist in detecting a massively growing number of DDoS anomalies. Our model has been comprehensively and extensively tested against three different DDoS attack types. The evaluation results demonstrate high levels of performance on different time window lengths over many performance metrics including precision (99%), recall (99%), F1-score (99%), and accuracy (99%). Our model performed best for the LDAP attack detection case against all performance metrics, exceeding 99% and outperforming other state-of-the-art methods.

Chapter 4

LSTM-Autoencoder based Anomaly Detection for Indoor Air Quality Time Series Data

Abstract

Anomaly detection for indoor air quality (IAQ) data has become an important area of research as the quality of air is closely related to human health and well-being. However, traditional statistics and shallow machine learning-based approaches in anomaly detection in the IAQ area could not detect anomalies involving the observation of correlations across several data points (i.e., often referred to as long-term dependencies). We propose a hybrid deep learning model that combines LSTM with Autoencoder for anomaly detection tasks in IAQ to address this issue. In our approach, the LSTM network is comprised of multiple LSTM cells that work with each other to learn the long-term dependencies of the data in a time-series sequence. Autoencoder identifies the optimal threshold based on the reconstruction loss rates evaluated on every data across all time-series sequences. Our experimental results, based on the Dunedin CO_2 time-series dataset obtained through a real-world deployment of the schools in New Zealand, demonstrate a very high and robust accuracy rate (99.50%) that outperforms other similar models.

4.1 Introduction

indoor air quality (IAQ) is closely related to human health, productivity, and work efficiency [36]. Good air quality is even more important for children who spend the vast majority of their time at school. Providing children with fresh air in their classroom environment is of high importance for their health and well-being. However, the formulation of CO_2, which is considered to be the major constituent of indoor air pollutants, can be easily built up by children studying/playing inside classrooms and also the emission from floors and other surfaces [51]. Such accumulated CO_2 can become the basis of creating harmful mold and bacteria that could contribute to poor health and degrading academic performance.

Constant monitoring of indoor air quality including the measurement of the level of CO_2 in school environments has been problematic for many countries including the OECD as the large-scale monitoring of indoor air quality has proven to be too expensive for budget-strapped schools. To address this concern, a team of researchers at Massey University developed a low-cost monitoring suite called SKOol MOnitoring BOx (SKOMOBO) with the National Institute of Water and Atmospheric Research (NIWA) [86, 87, 92]. A SKOMOBO unit is a small box, approximately the size of 100 $\times$ 100 $\times$ 100 mm, designed to house a number of low-cost sensors that could capture a number of indoor air quality-related data such as particulate matter ($PM_{2.5}$ and PM_{10}), temperature, relative humidity, carbon dioxide (CO_2) and human occupancy in classrooms.

The monitoring of measurable indoor air quality is challenging due to the fluctuation of IAQ data reading and the question of data quality, for example, deployment in non-stationary or uncontrolled environments [100]. In addition, rare or consistent contamination events can also corrupt the data quality (i.e., natural disasters such as fire, flooding, and thunderstorms). As a result, numerous proposals for detecting anomalous events in IAQ has been attempted by utilizing the latest advancement in artificial intelligence-based techniques. For instance, many attempted to use statistical methods (e.g., using the means, standard deviation, Gaussian q-distribution) [88, 93] and other combinations of shallow machine learning techniques (e.g., KNN, k-means, regression) [40, 58, 101] to find the patterns of normal behaviors of IAQ and use that as a basis to detect anomalous events or for improving forecasting capabilities such as [62, 76]. However, these existing methods could not detect anomalies where the observation of several correlated data points are necessary.

In this research, we propose a hybrid deep learning model that combines the capabilities of long short-term memory (LSTM) and Autoencoder (AE) for detecting anomalous data points in IAQ datasets based on the understanding of long-term dependencies that exist in data samples.

The main contributions of our proposed model are the following.

Summary of Original Contributions

- We propose a hybrid deep learning model that utilizes the capacity of LSTM and Autoencoder (AE). The proposed hybrid model can detect anomalous air quality data points as a result of analyzing the time-series sequences of the dataset observed during a certain duration in an automated fashion. Our model uses LSTM networks comprised of multiple LSTM units each of which works with the other to learn the long-term dependencies of multiple data points within a time-series sequence. In addition, our model uses Autoencoder (AE) to reduce data dimension for more efficient training and also computes an optimal threshold associated with each time-series sequence. Together, our model is more suited to detect contextual anomalies.
- We apply our proposed model to the Dunedin CO_2 Dataset obtained from a real-world deployment of multiple primary/secondary schools in New Zealand.
- We compare the performance of the proposed model with other similar approaches that use different aspects of LSTM and/or AE. Our experimental results performed based on the comprehensive set of evaluation criteria, demonstrate that our proposed model can effectively detect anomalies reaching the detection accuracy that exceeds 99%.

The rest of this Chapter is structured as follows. Section 4.2 introduces related works in the field of indoor air quality. Section 4.3 introduces the preliminary of the techniques. Section 4.4 introduces the details of our proposed model. Section 4.5 and Section 4.6 illustrates the experimental setup and analysis of results evaluated on the Dunedin CO_2 dataset. Section 4.7 concludes the paper with the planned future works.

4.2 Related Work

We review the existing state-of-the-art techniques for detecting anomalies in indoor air quality datasets.

Ottosen et al. [58] introduced two anomaly detection techniques that use k-Nearest Neighbour (KNN) and AutoRegressive Integrated Moving Average (ARIMA) to detect both point and contextual anomalies separately in a low-cost air quality dataset. In their approach to detecting point anomalies, they simply utilized the average Euclidean distance to compute the similarity with the remaining points and assign an anomaly score to the individual point. ARIMA was used to detect contextual anomalies by calculating anomaly scores for each data point between the model and measurement based on the absolute value of the residual. Both point and contextual anomalies are classified into two clusters, normal and anomaly, by K-means clustering.

Wei et al. [88] proposed a hybrid model of MSD-Kmeans to detect anomalies with the indoor PM_{10} dataset. They first used the statistical method of Mean and Standard Deviation (MSD) which was used to eliminate noisy data to reduce the impact of clustering

from the noise. Then they applied the K-means algorithm to achieve better local optimal clustering. The performance of their proposal showed the detection accuracy (97.6%) and F1-score (91.9%).

Li et al. [40] proposed clustering-based Fuzzy C-means. In their approach, the authors used a reconstruction criterion to reconstruct the optimal cluster centers and partition matrix based on multivariate subsequences data. They also used a reconstruction error as the fitness function of the Particle Swarm Optimization (PSO) algorithm to define a level for detecting anomalies in multivariate data. However, the proposed algorithm was not able to reveal the structure of high dimensional multivariate time series due to the issue involved in the PSO algorithm trap in local optima.

Sharma et al. [76] proposed a low-cost framework named IndoAirSense to estimate and forecast indoor air quality in selected classrooms in the university. They first used Multi-Layer Perceptron (MLP) and eXtream Gradient Boosting Regression (XGBR) to estimate real-time indoor air quality. Then they used LSTM-wF (Long Short Term Memory without using the forget gate) to reduce the complexity of LSTM to forecast indoor air pollutants. Clearly not using the forget gate that keeps the long-term memory, this model could not detect anomalies in the time series dataset.

Mumtaz et al. [52] proposed an LSTM-based model for predicting the concentration of different air pollutants to examine the overall quality of an indoor environment. In their research, they collected the base data through IoT sensors that collect different air pollutants (e.g., NH_3, CO, NO_2, CH_4, CO_2, $PM_{2.5}$). Their proposed system had the capability of sending alerts after detecting anomalies in air quality.

Xu et al. [94] proposed an LSTM model with an added error correction model (ECM) for improving the prediction of indoor temperature in public buildings. In their approach, an ECM is built when the predicted and measured data are co-integrated in the same order, then utilized to revise the prediction of the testing dataset. Similarly, Jung et al. [32] utilized LSTM for predicting the conditions of indoor space for facility management based on the three IoT sensor datasets that measure the temperature, humidity, and brightness of a space. LSTM was used to detect anomalous indoor space conditions where the readings on the combination of three indoor air condition datasets deviating from a threshold were obtained during training.

Hossain et al. [28] proposed a combined prediction scheme using two variations of the RNN model (i.e., GRU and LSTM, respectively) to forecast the daily air quality index (AQI) for two of the biggest cities (i.e., Dhaka and Chattogram) in Bangladesh. In their proposal, they used GRU and LSTM as the first and second hidden layers respectively followed by two dense layers as a prediction model. Results reflected that their model followed the actual AQI trends for both cities and demonstrated that their two models improved the overall performance when compared to when only a single model of GRU or LSTM was used.

We argue that these existing works have several limitations. For example, more classic time-series approaches such as using ARIMA and regressive machine learning approaches (such as K-means, or KNN) often rely on manual extraction of features for better prediction of the behavior of the features. This can be time-consuming and require the intervention of human experts which can be expensive. These existing models are also often very sensitive to outliers and tend to perform poorly when the underlying dataset size is large and the probability distribution is unknown.

Using LSTM, the use of time-series forecasting models can predict future values based on the previous pattern of sequential data which provides greater accuracy for prediction and assists in better decision making such as detecting anomalies. Unlike existing methods, through the use of Autoencoder, our proposed model can provide much better results when dealing with complex autocorrelation sequences with large datasets even in the presence of unpredictable data distribution.

4.3 Preliminaries

4.3.1 LSTM

LSTM stands for long short term memory which is often regarded as an extension of Recurrent Neural Networks (RNN). RNN provided the capability of "short-term memory" which allowed the use of the previous information (at a certain point only) to be used for the present task. Extending from RNN, LSTM architecture provides the capability of "long-term memory" where a list of all of the previous information (as opposed to a point of time) is available for the current neural node.

A common LSTM unit, depicted in Fig. 4.1, is composed of a cell, an input gate, an output gate, and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information in and out of the cell.

Note that:

- The *Cell State* refers to the current long-term memory of the network that stores the list of previous information.
- The previous *Hidden State* refers to the output at the previous point in time which can be seen as short-term memory.
- The input data contains the input value at the current time step.

Step1: Forget Gate - The main purpose of the forget gate is to decide which bits of the cell state are useful given both the previous hidden state and new input data. If relevant, it outputs the number closer to 1 otherwise 0 using a sigmoid activation function. Mathematically, the results (f_t) from the forget gate can be presented as:

$$f_t = \sigma(w_f[H_{t-1}, X_t] + b_f) \tag{4.1}$$

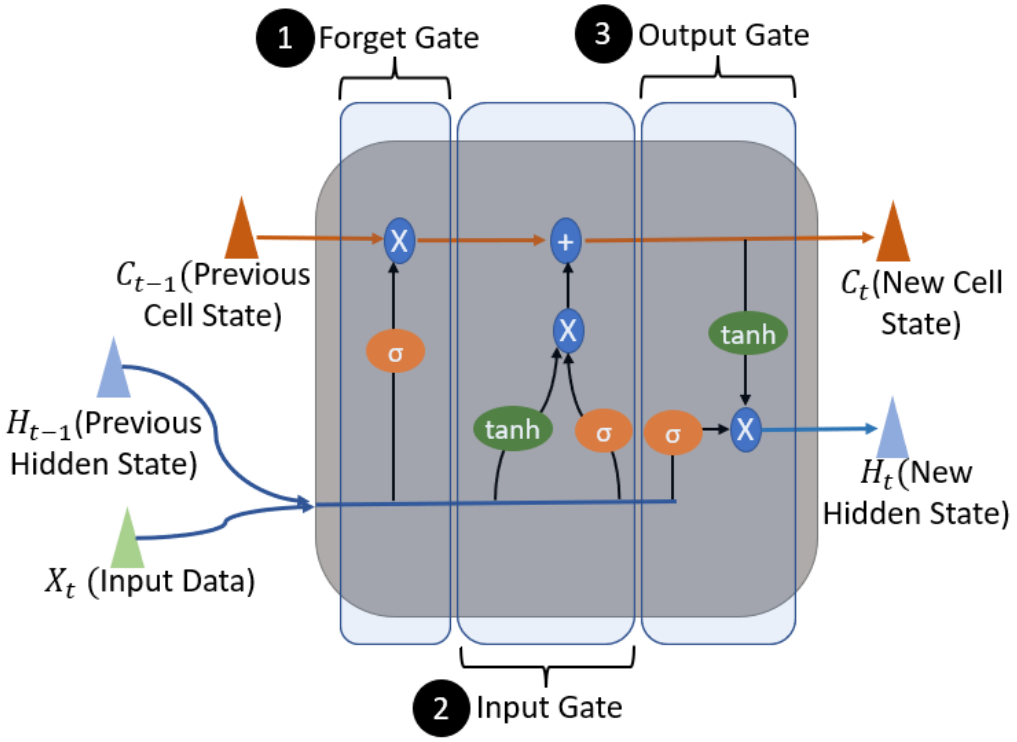


Figure 4.1: How LSTM Unit Works

where σ is the activation function, w_f and b_f are the weights and bias of the forget gate. H_{t-1} and X_t present the concatenation of the hidden state and the current input respectively.

Step2: Input Gate - The main purpose of the input gate is two folds. The first is to check if the new information (i.e., the previous hidden state and new input data) is worth keeping in the cell state. If there are, secondly, it decides what new information should be added to the cell state. Towards this, the input gate goes through two processes. The process of deciding how much of the new information, now presented as $\tilde{C}_t$, is depicted in Equation 4.2 as follows.

$$\tilde{C}_t = tanh(w_c[H_{t-1}, X_t] + b_c) \tag{4.2}$$

where w_c and b_c are the weight matrices and the bias of the input gate respectively while a tanh is used as an activation function in the range of [-1, 1] where the negative values are used to reduce the impact. Identifying which components of the new input are depicted in Equation 4.3 as follows.

$$i_t = \sigma(w_i[H_{t-1}, X_t] + b_i) \tag{4.3}$$

where w_i and b_i are the weight matrices and the bias of the input gate.

These two processes are pointwise multiplied to regulate the magnitude of the new information. This resulting combined vector is then added to the cell state, resulting in

the long-term memory of the network being updated, as shown in Equation 4.4.

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{4.4}$$

Step3: Output Gate - The main purpose of the output gate is to decide the new hidden state.

It first applies the previous hidden state and current input data through the sigmoid activated network to obtain the filter vector o_t as shown in Equation 4.5.

$$o_t = \sigma(w_o[H_{t-1}, X_t] + b_o) \tag{4.5}$$

where w_o and b_o are the weight matrices and the bias of the output gate.

The cell state is passed through a tanh activation function to force the values into the interval [-1, 1] to create a squished cell state which is applied to the filter vector by pointwise multiplication. A new hidden state H_t is created and outputted, along with the new cell state C_t, as shown in Equation 4.6.

$$H_t = o_t \odot tanh(C_t) \tag{4.6}$$

The new cell state C_t becomes a previous cell state $C_{t-1}$to the next LSTM unit while the new hidden state H_t becomes a previous hidden state H_{t-1} to the next LSTM unit. These are repeated until the input data from all time-series sequences are processed by all LSTM cells involved.

4.3.2 Autoencoder (AE)

An autoencoder is a type of unsupervised neural network that is used to learn efficient codings of unlabelled data. It learns a representation for a set of input data by training the neural network to ignore insignificant data (i.e., often termed as "noise"). A typical autoencoder is composed of an input layer, an output layer, and several hidden layers. The operations of an autoencoder can be divided into Encoding, Decoding, and Reconstruction Loss as illustrated in Fig. 4.2.

Step1: Encoding - In the encoding operation, input data x is a m high-dimensional vector ($x \in \mathbb{R}^m$) that is mapped to a low-dimensional bottleneck layer representation (h) after removing any insignificant features, as shown in Equation (4.7).

$$h = f_1(w_i x + b_i) \tag{4.7}$$

where w_i is the weight matrix, b_i is a bias and f_1 is an activation function.

Step2: Decoding - In the decoding operation, the bottleneck layer representation of (h) is used to generate the output $\hat{x}$ that maps back into the reconstruction of x, as shown in Equation (4.8):

$$\hat{x} = f_2(w_j h + b_j) \tag{4.8}$$

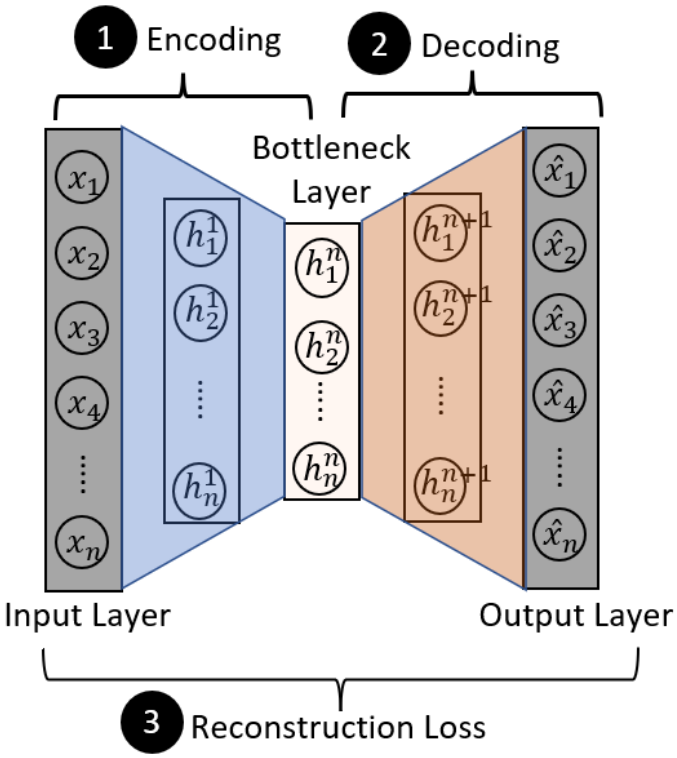


Figure 4.2: How Autoencoder Works

where f_2 is an activation function for the decoder. w_j is the weight matrix, b_j represents a bias and $\hat{x}$ represent reconstructed input sample. Note that w_j and b_j may be unrelated to the corresponding w_i and b_i for the encoder.

Step3: Reconstruction Loss - In a standard autoencoder model, a reconstruction loss (L) is calculated to minimize the difference between the output and the input, as shown in Equation 4.9. It is often this reconstruction loss that is used for anomaly detection task [89, 95].

$$L(x - \hat{x}) = \frac{1}{n} \sum_{n=1}^{n} |\hat{x}_t - x_t| \tag{4.9}$$

where x represents the input data, $\hat{x}$ indicates the output data and n is the number of samples in the training dataset.

However, this is extended to compute a reconstruction loss of a sample in our model as follows:

$$\begin{aligned} x_i &= \frac{1}{n} \sum_{n=1}^{n} |\hat{x}_i - x_i| \\ s.t. \quad n &= \begin{cases} i & \leq \frac{N+1}{2} \\ N - i + 1 & n > \frac{N+1}{2} \end{cases} \end{aligned} \tag{4.10}$$

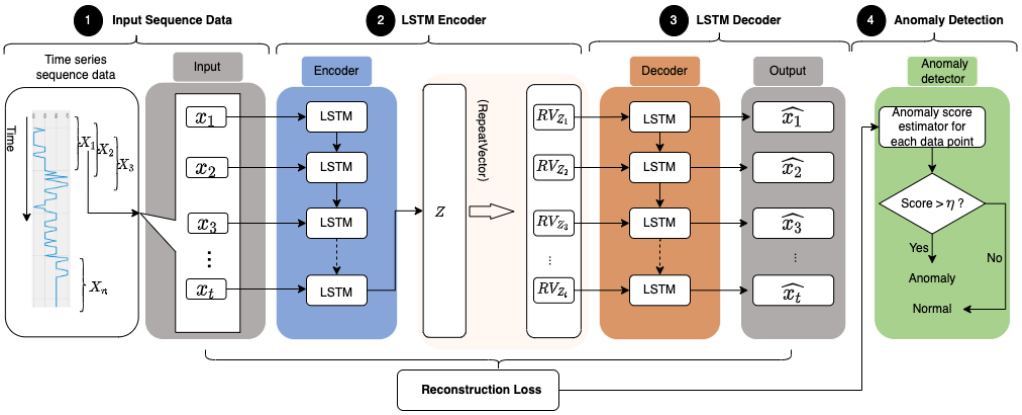


Figure 4.3: Overview of our proposed model

where N is the total number of samples, and n dedicates the nth sample, $X_i = \{x_1, ..., x_i\}$.

Then, we compute the reconstruction loss for all samples in time-series as follows.

$$loss = \frac{1}{N} \sum_{N=i}^{1} x_i \tag{4.11}$$

where N is the total number of samples, and x indicates a reconstruction loss computed for a sample.

4.4 Methodology

In this section, we introduce our proposed model that uses the combination of LSTM and Autoencoder to detect anomalies based on the analysis of time-series data. We first provide the overview of our proposed model based on the four steps of creating the input sequence, LSTM Encoder, LSTM Decoder, and anomaly detection. We also provide a detailed description of the algorithm our model uses in terms of training and testing phases.

4.4.1 LSTM-Autoencoder

The overview of our proposed model is illustrated in Fig. 4.3. Our LSTM-Autoencoder utilizes the capabilities of both the LSTM neural network and Autoencoder. In the first step, the original dataset is converted as the sequence of the high-dimensional input data as a fixed-size vector. In the second step, the LSTM Encoder creates multiple LSTM units each of which works together to retain all data points in a time series sequence to analyze the pattern of features (i.e., the pattern of CO_2 values) while keeps reducing the high dimensional input vector representation into low-dimensional representation until it reaches the latent space. In the third step, LSTM Decoder reproduces the fixed-size input sequence from the reduced representation of the input data in the latent space. In the

last stage of anomaly detection, the reconstruction error rate across the output to the input is calculated to find an optimal threshold. The threshold is then used in the test phase to detect a contextual anomaly across different time sequences.

Step1: Input Sequence Data - The original dataset is made as a series of time sequence $[X_1, X_2, X_3, ..., X_n]$. Each sequence X with a fixed T-length time window data $[x_1, x_2, x_3, ..., x_t]$ is created where $x_t \in R^m$ represents an m-features input at time-instance t. This is then again reshaped into a 2d (2-dimensional) array, representing samples and timesteps. For example, a sequence of our CO_2 data is converted into the 2d array where each dimension indicates the list of samples at 10 timesteps.

Step2: LSTM Encoder - The main purpose of the LSTM encoder is to act like a sequence folding layer that converts features to a batch of time-based feature sequences. It is like convolution operations on timesteps of feature sequences independently. Fig. 4.4 describes the details of how the (AE) encoder interacts with the series of LSTM unit cells trained to recognize the most relevant features in the input sequence.

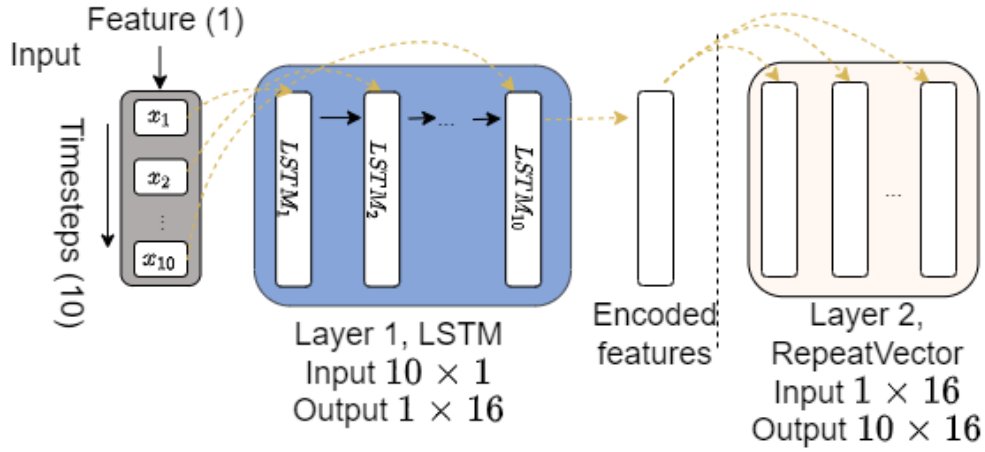


Figure 4.4: Details of LSTM encoder

Each time series of X_i contains 10 samples collected on 10 timesteps (of a 1-minute interval). This one-dimensional dataset is reshaped into a two-dimensional dataset to feed to the encoder. For example, to unroll the input dataset based on timesteps, the number of input is coveted as a 2d vector where one dimension contains the 10 timesteps and another dimension contains the feature (i.e., samples of CO_2 reading), presenting as a vector of 10×1. This is now fed to the encoder. The encoder creates Layer 1 which contains an LSTM network with 10 LSTM cells. Each LSTM cell unit processes a sample. 10 LSTM cells work in a sequential manner where the 1st LSTM unit passes the result of the sample to the 2nd LSTM. The 2nd LSTM unit decides whether to keep the previous sample from the 1st LSTM to keep or forget it. If the 2nd LSTM decides to keep it, it writes it in the long-term memory and passes the information of the sample from the 1st LSTM along with the feature information processed from the sample it is processing to the 3rd LSTM

and so on. The last LSTM, the 10th in our model has all samples worth keeping processed by the 9 previous LSTM cells. The information about all relevant samples is outputted by the last LSTM cell. This output is now coveted as the 1×16 vector as encoded features.

Note that we added a RepeatVector as Layer 2 to create the copies of the 1×16 vector as many as equal to the number of timesteps. For example, the size of time steps in our model is 10 therefore Layer 2 creates 10 copies of encoded features as a two-dimensional vector that equals 10×16.

Step3: LSTM Decoder - The main purpose of the LSTM decoder is to act like a sequence unfolding layer that restores the sequence structure of the input data after the sequence folding on timesteps. Fig. 4.5 describes the details of how the decoder interacts with LSTM cells to reconstruct the outputs.

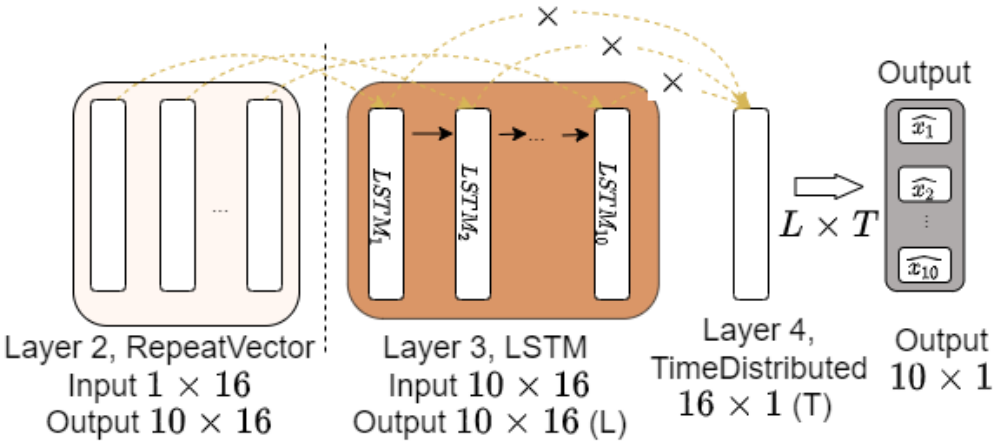


Figure 4.5: Details of LSTM decoder

Each 1×16 set is now fed as an input to the decoder which creates a Layer 3 network with 10 LSTM cell units. Each LSTM cell unit processes each 1×16 encoded feature. Each LSTM unit produces an output that represents the result of the learning from the encoded feature where the output is multiplied with the 1×16 vector created by the additional TimeDistribution layer. At the same time, each LSTM cell unit produces another 2nd output containing the state of what has been processed by the current LSTM cell passing to the next LSTM, except the last LSTM unit. Note that matrix multiplication between the output of each LSTM layer (L) (10×16) and the TimeDistribution layer (16×1) is calculated which results in a vector with the size of 10×1 which is the same as the size of the input.

Step4: Anomaly Detection -

An anomaly can be defined as an observation diverging from the majority of the data. A threshold can be set as a decision point to decide how much an observation deviates. Any observations that go beyond the threshold are defined as anomalies.

Applying this threshold-based anomaly detection technique, our model is trained with the dataset that contains the CO_2 values within a normal range. This is to obtain the

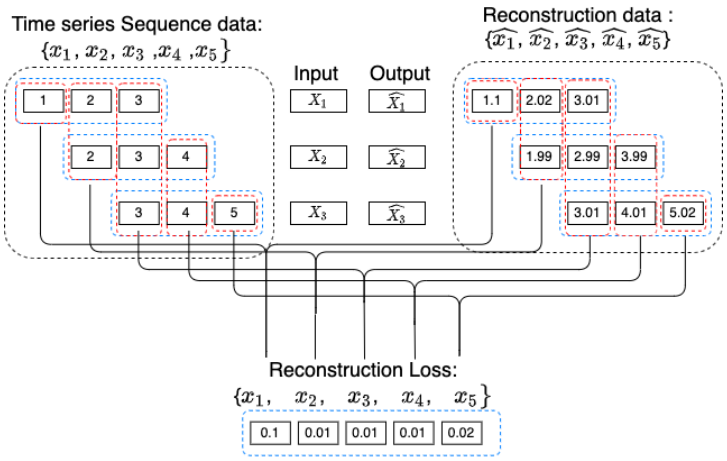


Figure 4.6: Computing reconstruction loss on time series

reconstruction error rates associated with the normal CO_2 data points. Once training is done and all different reconstruction error is computed on all samples, the max reconstruction error rate is set as a threshold. Once the threshold is decided, we input the testing CO_2 dataset which now contains all ranges of CO_2 readings. A reconstruction error rate of each CO_2 value is computed for each sample in the testing set. If the reconstruction error rate goes beyond the threshold, this sample is considered an anomaly.

Fig. 4.6 illustrates how we calculate reconstruction loss for each sample contained different time-series sequences. Let's presume that there are 5 samples $[x_1, x_2, x_3, x_4, x_5]$ which are made as 3 time-series sequences of $[X_1, X_2, X_3]$ where each sequence containing 3 samples on 3 different timesteps where $X_1 \in [x_1, x_2, x_3]$, $X_2 \in [x_2, x_3, x_4]$, and $X_3 \in [x_3, x_4, x_5]$ - as shown in blue dotted blocks. Our model trains this 3 time-series of sequences as inputs and construct the outputs that map to each sequence

$\hat{X}_1 \in [\hat{x_1}, \hat{x_2}, \hat{x_3}]$, $\hat{X}_2 \in [\hat{x_2}, \hat{x_3}, \hat{x_4}]$, *and* $\hat{X}_3 \in [\hat{x_3}, \hat{x_4}, \hat{x_5}]$.

Let's assume that the original value for the 3 sequences is: $X_1 \in [x_1 = 1, x_2 = 2, x_3 = 3]$, $X_2 \in [x_2 = 2, x_3 = 3, x_4 = 4]$, and $X_3 \in [x_3 = 3, x_4 = 4, x_5 = 5]$ where the mapping outputs for each time sequence came out as

$\hat{X}_1 \in [\hat{x_1} = 1.1, \hat{x_2} = 2.02, \hat{x_3} = 3.01]$, $\hat{X}_2 \in [\hat{x_2} = 1.99, \hat{x_3} = 2.99, \hat{x_4} = 3.99]$, *and* $\hat{X}_3 \in [\hat{x_3} = 3.01, \hat{x_4} = 4.02, \hat{x_5} = 5.02]$.

The reconstruction loss for each sample can be calculated as:

$$x_1 = |1.1 - 1| \,/\, 1 = 0.1$$
$$x_2 = (|2.02 - 2| + |1.99 - 2|) \,/\, 2 = 0.01$$
$$x_3 = (|3.01 - 3| + |2.99 - 3| + |3.01 - 3|) \,/\, 3 = 0.01$$
$$x_4 = (|3.99 - 4| + |4.01 - 4|) \,/\, 2 = 0.01$$
$$x_5 = |5.02 - 5| \,/\, 1 = 0.02$$

The max reconstruction loss is set as a threshold which in our case is set to be 0.1. During testing, any samples whose reconstruction loss goes beyond 0.1 is now labeled as an anomaly.

4.4.2 Algorithm

The algorithm for the proposed model is shown in Algorithm in Fig. 4.7. The main goal of the training phase in our proposed model is two folds. Firstly, the focus of the training is to minimize the reconstruction error so that the outputs reconstructed from the reduced representation of the input resemble the input as much as possible. Secondly, our model obtains the typical reconstruction error rate associated with the normal range CO_2 data points to find an optimal threshold to use for detection during the test phase. The main goal of the testing phase is to use the threshold to detect anomalies in the test dataset.

Training Phase - The first step in the training phase is to reshape the original dataset into time-series sequences, as shown in Algorithm in Fig. 4.7 `phase 1: To sequence`. Dataset X_i represents a sequence in the training dataset. In our model, each sequence contains 10 CO_2 samples on 10 timesteps. As depicted in the `Phase 2: LSTM-AE training` within Algorithm in Fig. 4.7, the training of the model starts where each sequence is fed to the encoder one at a time where a sample in the sequence is trained by a single LSTM in a sequential manner. Once the training of each sequence completes, the latent space of the encoder rearranges the concatenation of the (relevant) data points as a 1-dimensional encoded feature representation. The RepeatVector layer makes multiple copies of the encoded feature.

The decoder creates an LSTM network with the number of LSTM cells according to the timesteps (i.e., also matching the copies of the encoded features). Each encoded feature is processed by a single LSTM cell. The results of the processing by all LSTM cells are made as a single-dimensional vector at the TimeDistributed Dense Layer which produces the output.

A reconstruction loss between the output and input is calculated, as in steps 8 to 13. A backpropagation strategy is applied to adjust the weights and parameters of the model. We use the Mean Absolute Error (MAE) algorithm, as shown in Equation 4.12, as the reconstruction error loss function.

$$Loss(MAE) = \frac{\sum_{i=1}^{n} |x_i - \hat{x_i}|}{n} \tag{4.12}$$

where n indicates the total number of samples, x_i is the representation of the original input being fed to the encoder, and $\hat{x_i}$ is the output produced by the decoder.

The model trains on all time-series sequences until the reconstruction loss is minimized for all samples. Note that we use 16 neurons in the latent space of the encoder to capture the output from the 10th LSTM. We used "tanh" as the activation function in our proposed model. We also use two Dropout layers (0.2) in the encoder and decoder respectively. We use a RepeatVector layer between the encoder and decoder. An additional

Algorithm 1 LSTM-AE Anomaly Detection

Input:
Training set $\{x_0, x_1, x_2, \ldots, x_{n-1}\}$,
Test set $\{x'_0, x'_1, x'_2, \ldots, x'_{m-1}\}$,
Timesteps t
Output: A Set of anomalies(A_t) or normal (N_t)

begin
  /* Phase 1: To sequence */
  X_i, X'_i: sets of training and testing data based on timesteps (t=10)
  for $i \in [0, n-t)$ **do**
    $X_i = [x_i :: x_{i+t}]$
  end
  for $i \in [0, m-t)$ **do**
    $X'_i = [x'_i :: x'_{i+t}]$
  end
  /* Phase 2: LSTM-AE training */
  Initialize the parameter of LSTM-AE model (M)
  for $X_i \in [X_0, X_1, ..., X_{n-t})$ **do**
    $\hat{X}_i = M(X_i)$
    L_{err} = $\sum |X_i - \hat{X}_i|$
    Update LSTM-AE to minimize L_{err} by eq. (9)
  end
  /* Phase 3: Threshold setting */

  function RLOSS(X)
  /* X_i reconstruction error calculation */
  for $i \in (0, n-t)$ **do**
    $\hat{X}_i = M(X_i)$
    $Err_{arr}[i, i:i+t] = |\hat{X}_i - X_i|$
  end
  /* All data reconstruction error calculation */
  for $i \in (0, n)$ **do**
    $l_{arr}[i] = \sum Err_{arr}[:, i] / \sum (Err_{arr}[:, i]! = 0)$
  end
  return l_{arr}
  /* Max RLoss from training set */
  $threshold(\eta) = \max(RLOSS([X_0, X_1, ..., X_{n-t}))$
  /* Phase 4: Anomaly detection on testing set */
  $ltest_{arr} = RLOSS([X'_0, X'_1, ..., X'_{m-t})$
  for $i \in (0, m)$ **do**
    if $ltest_{arr}[i] > \eta$ **then**
      $x'_i \longrightarrow \mathrm{A}_t$
    else
      $x'_i \longrightarrow \mathrm{N}_t$
    end
  end
end

Figure 4.7: LSTM-AE Algorithem

TimeDistrutedDense layer is used before the output layer. Once training is complete, the max reconstruction error is obtained as a threshold as shown in `Phase 3: Threshold setting`.

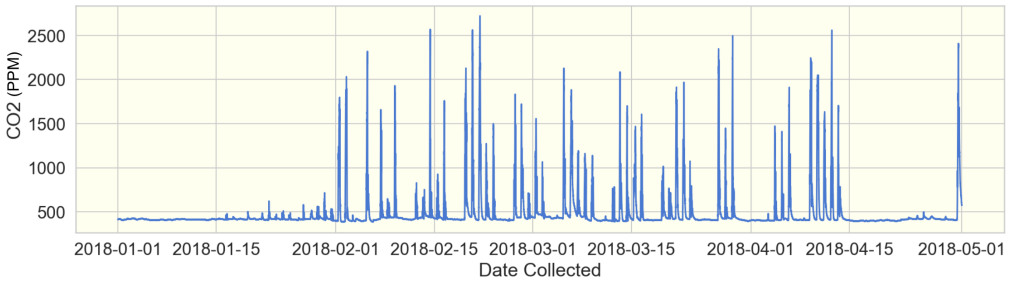


Figure 4.8: The projection of the original raw dataset

Testing Phase - The details of the testing phase of our model are shown in `Phase 4: Anomaly detection on testing set`. A sequence of time series containing 10 data points of 10 timesteps is fed to the trained LSTM encoder. Note that this time the 10 data points contain all ranges of CO_2 values. The LSTM decoder produces a single time series also containing 10 data points of 10 timesteps using the encoded (and reduced) feature representation of the input sample. A reconstruction error rate of each data point is compared with the threshold. The calculation of the reconstruction loss strategy is the same as we mentioned in the training phase in Equation 3.8. If the reconstruction loss value is bigger than the threshold η, this data point is labeled as an anomaly otherwise labeled as normal. This is shown in the following Equation 4.13.

$$X' = \begin{cases} X'_i \text{ is anomalies}, & \text{if } ltest_{arr}[i] > \eta \\ X'_i \text{ is normal}, & \text{otherwise} \end{cases} \tag{4.13}$$

where X' indicates a reconstructed time series, X'_i is a data point contained in the time series, and $ltest_{arr}[i]$ is a result of the reconstruction loss function using MAE.

4.5 Data and Data processing

We discuss the details of the dataset we use in our study along with the description of the data preprocessing strategies we adopted.

4.5.1 Dunedin CO_2 Dataset

With the focus on understanding the relationship between the level of CO_2, weather conditions, and student performance, 74 units of SKOMOBO boxes were deployed in multiple primary/secondary schools in Dunedin, South Island, New Zealand. Records containing the reading of CO_2 were collected over a period of four months between 01/01/2018 and 30/04/2018, at a 1-minute interval. The projection of the CO_2 reading is shown in Fig. 4.8. As expected, any changes in the CO_2 were not observed when there was a school break (e.g., during January 2018, and the 1st term break in the last two weeks of April 2018. As students occupy classrooms, we observe the changes of CO_2 fluctuating a lot

of which some could be anomalous. The total number of CO_2 readings presented in the dataset was 247,263.

4.5.2 Data Preprocessing

We first clean up the original records. We first removed all duplicate records. For example, we removed the records containing the CO_2 readings with identical timestamps. We also removed the records where it contained both CO_2 and timestamp with NaN values. We kept the records where the CO_2 reading was either an empty value or NaN value but a timestamp was legit. In this case, we replaced either empty or NaN value with the numeric 0. After this clean-up, we had a total of 171,067 records.

4.5.3 Training and Test dataset

Two separate datasets for the training and testing phase of our model were prepared as follows.

Training Dataset - According to [44], the typical CO_2 value accepted as the normal range is between 0 and 968 (PPM).

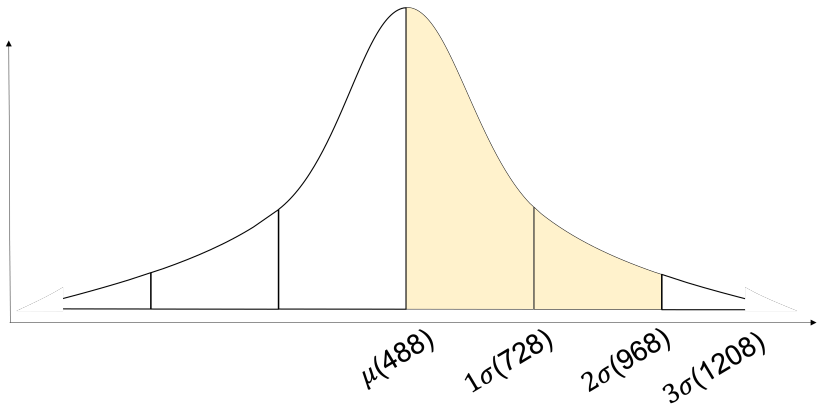


Figure 4.9: The distribution of CO_2 reading according to 3-sigma

Based on our analysis of the dataset, we find that the majority of the CO_2 readings sit if we use the 2-sigma rule of the normal distribution (i.e, around CO_2 readings < 968 when the mean of the CO_2 readings is around 488) which can be considered as the acceptable normal range, as shown in Fig. 4.9.

As the training of the model learns the typical reconstruction error associated with normal data points, we create a training dataset only to contain CO_2 data points within a normal range. Towards this, we first set aside 3 months of the original dataset (from 01/01/2018 - 31/03/2018). Then we calculate the 2-sigma rule to check each sample if it sits in the normal range or not. If there are any CO_2 readings beyond the 2-sigma rule, we remove them. The illustration of creating the training dataset is shown in Fig. 4.10.

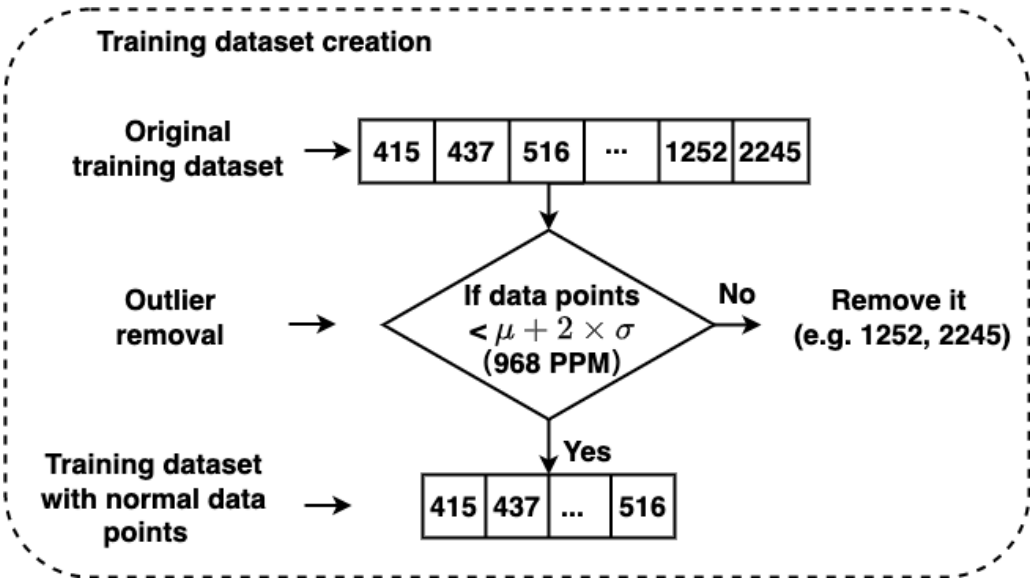


Figure 4.10: Creating training dataset

Test Dataset - We used 1 month of the original data (from 01/04/2018 - 30/04/2018) as a testing dataset. Note that this dataset contains all different ranges of CO_2 readings. We add the numeric 0 as a label if the CO_2 reading is within the 2 sigma-rule range, otherwise, we add the numeric 1. These labels are only used to evaluate the performance of our model - whether our model is good at detecting anomalous data points to normal data points. The illustration of adding labels in the test dataset is shown in Fig. 4.11.

4.5.4 Data Normalization

We apply a data normalization technique to eliminate the impacts of different scales across CO_2 readings thus reducing the execution time and computational complexity of the model training. We use a standard scalar normalization as depicted in Equation (4.14).

$$Z_i = \frac{X_i - \mu}{S} \tag{4.14}$$

where Z_i denotes all the normalized numeric values ranging between [0-1]; X_i indicates a data point while μ and S refer to the mean and standard deviation.

4.6 Evaluations

In this section, we provide the details of the experiment including the environment setup, performance metrics we use, analysis of results, and discussion.

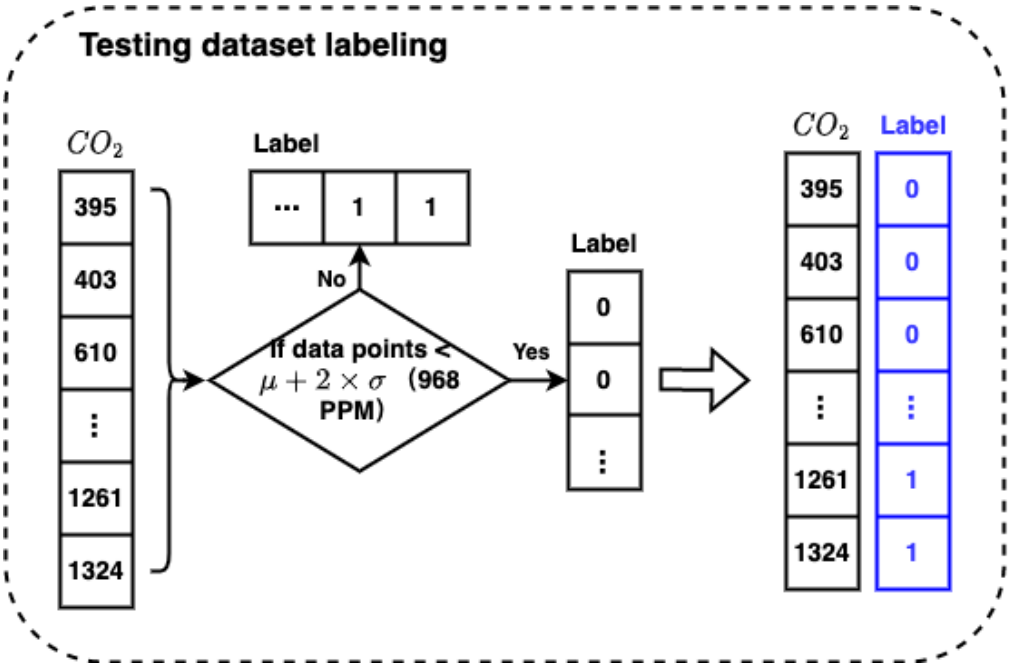


Figure 4.11: Creating test dataset

Table 4.1: Implementation environment specification

Unit	**Description**
Processor	3.4GH$_z$ Inter Core i5
RAM	16GB
OS	MacOS Big Sur 11.4
Packages used	tensorflow 2.0.0, sklearn 0.24.1

4.6.1 Experiment Setup

Our experiments were carried out using the following system setup shown in Table 4.1.

The hyperparameters used in the training phase are illustrated with the values for each parameter along with the description in Table 4.2.

Table 4.2: LSTM-AE training parameters

Hyperparameters	Values	Descriptions
Learning rate	0.001	Learning speed (within range 0.0 and 1.0)
Dropout	0.2	No. of neurons ignored
Batch size	64	No. of samples in one fwd/bwd pass
Epoch	30	No. of one fwd/bwd pass of all samples

4.6.2 Performance Metrics

To evaluate the performance of our model, we used classification accuracy, precision, recall, and F1 score as performance metrics. Table 4.3 illustrates the confusion matrix.

Table 4.3: Confusion Matrix

Total Population		Predicted Condition	
		Normal	Anomaly
Actual Condition	Normal	TN	FP
	Anomaly	FN	TP

where;

- True Positive (TP) indicates anomalous data point correctly classified as anomalous.
- True Negative (TN) indicates normal data point correctly classified as normal.
- False Positive (FP) indicates normal data point incorrectly classified as anomalous.
- False Negative (FN) indicates anomalous data point incorrectly classified as normal.

Based on the aforementioned terms, the evaluation metrics are calculated as follows:

$$TPR(TruePositiveRate/Recall) = \frac{TP}{TP + FN} \tag{4.15}$$

$$FPR(FalsePositiveRate) = \frac{FP}{FP + TN} \tag{4.16}$$

$$Precision = \frac{TP}{TP + FP} \tag{4.17}$$

$$F1 - score = 2 \times \left(\frac{Precision \times Recall}{Precision + Recall} \right) \tag{4.18}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.19}$$

The area under the curve (AUC) computes the area under the receiver operating characteristics (ROC) curve which is plotted based on the trade-off between the true positive rate on the y-axis and the false positive rate on the x-axis across different thresholds. Mathematically, AUC is computed as shown in Equation (4.20).

$$AUC_{ROC} = \int_0^1 \frac{TP}{TP + FN} d \frac{FP}{TN + FP} \tag{4.20}$$

4.6.3 Results

We provide the results of the performance of our proposed model observed from a number of different evaluation aspects.

4.6.3.1 Training

Fig. 4.12 shows the trends of the loss at different epoch intervals. The training loss (the blue line) assesses the error rate of the model during training. We can see that the training loss stabilizes pretty quickly approximately around 8 epochs. We set aside 10% of the validation set from the training dataset to assess the performance of our model during training. As expected, the validation loss is not stabilized before 8 epochs. However, after 8 epochs, the validation loss presents a similar loss rate to the training loss (i.e., the average of approximately 0.07%). This can be regarded as a good fit and our proposed model works well (i.e., our model does not overfit or underfit).

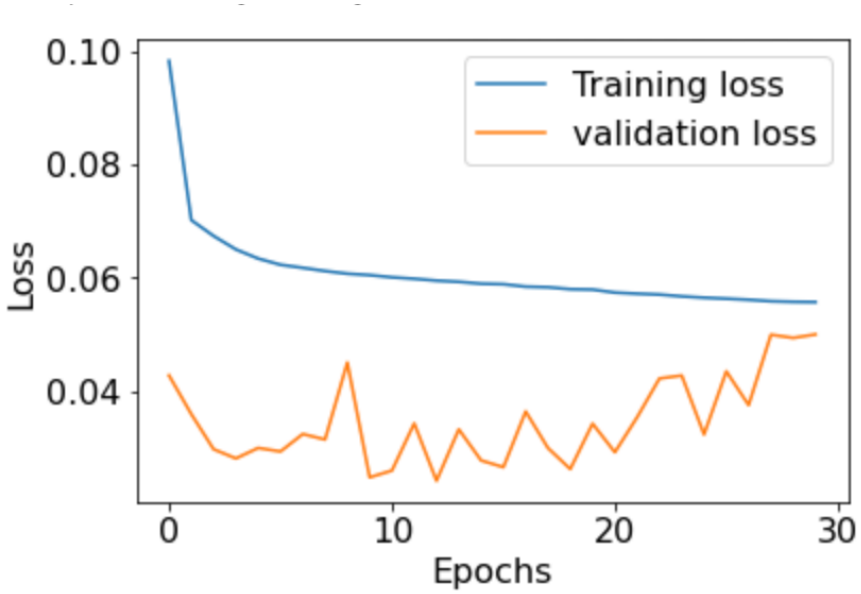


Figure 4.12: Training/Validation loss

4.6.3.2 Impact of Model Architecture

We also tested the sensitivity of our model in terms of the model architecture that differs in the number of the hidden layer (s) and the number of LSTM cell units used. Three different types of model architects, consisting of 1 hidden layer, 2 hidden layers, and n hidden layers, at the encoder and decoder were evaluated. The number of hidden layers at the encoder and the decoder vary while the number of LSTM units used at different model architectures is the same. The details of the three model architectures we evaluated are shown in Fig. 4.13.

There were slight differences in terms of the model performance depending on the number of hidden layers. For example, the model architecture with 1 hidden layer worked best with the F1-score above 94.68% while the F1-score of 93.48% and 93.31% were

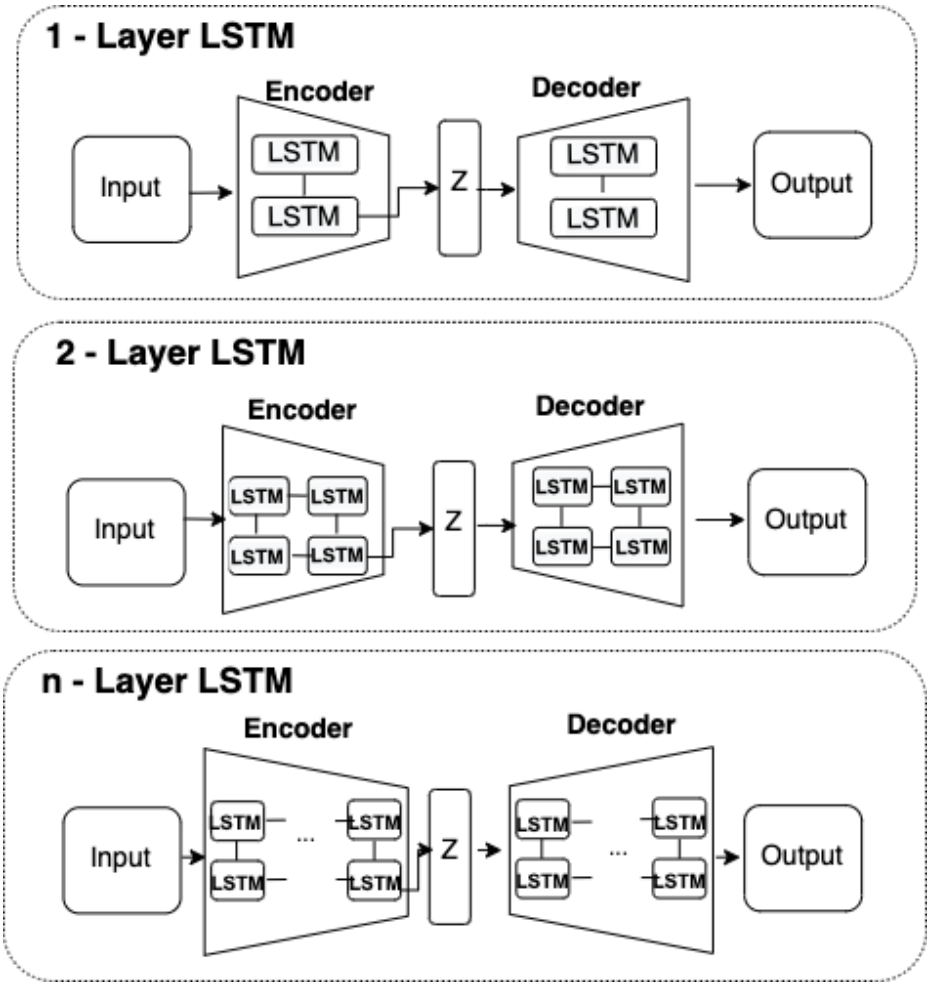


Figure 4.13: Different model architecture

reached by 2-layer and 3-layer models, respectively. The size of the output vector used by different architectures had a higher impact on the model performance through the smallest size of the output vector used by the 1-layer model worked the best in reaching a 94.68% F1-score, as shown in Table 4.4. Note that the first and the second line both use a 1-layer model but with a different number of units.

Table 4.4: Performance of different model architectures

No. (Layers)	No. (Units)	Accuracy	Precision	Recall	F1-score
1	128	99.29	100	85.71	92.31
1	**16**	**99.50**	**100**	**89.90**	**94.68**
2	64,16	99.39	100	87.76	93.48
3	128,64,16	99.38	100	87.47	93.31

4.6.3.3 Impact of Model Parameters

Table 4.5 shows the performance of different model parameters such as different sizes of learning rate and batch size while dropout and epoch values remain constant.

In the observation of the performance of the batch size, the batch size value of 64 resulted in the best performance with the highest accuracy, recall, and f1-score. Because there are more data in the bigger batch size, it also contributed to the faster response time where the time it took to run the proposed model took three times less compared to when the batch size of 10 was used. In the performance of different learning rates, the learning rate value of 0.001 showed the best performance while taking the smallest time to run the model.

Table 4.5: Parameter comparison

Parameters		Performances					
	Batch size	Acc	Pre	Re	F1	AUC	Time (s) ($\mu \pm \sigma$) /epoch
Learning rate: 0.001	10	99.41	100	87.90	93.56	94.0	99.72± 1.48
	32	99.45	100	88.52	94.15	94.5	44.37 ± 3.39
	64	**99.50**	**100**	**89.90**	**94.68**	**95.0**	35.70 ± 1.28
	Learning rate	Acc	Pre	Re	F1	AUC	Time (s) ($\mu \pm \sigma$) /epoch
Batch size: 64	**0.001**	**99.50**	**100**	**89.90**	**94.68**	**95.0**	35.70 ± 1.28
	0.0001	99.43	100	88.48	93.89	94.2	35.82 ± 1.28
	0.00001	99.39	100	87.62	93.40	93.8	35.83 ± 1.14

4.6.3.4 Impact of The Size of Time Sliding Window

The size of the time sliding window that decides the number of timesteps to contain in a sequence can impact the overall performance as it can affect the way the reconstruction error rate is computed. Thus we also tested the sensitivity of our model in terms of the size of the time sliding window and the model performance. We tested our model in terms of using the size of the time sliding window at 10, 15, 20, 25, 30, 35, and 40. As shown in Fig. 4.14, the time sliding window at 10 performed best with the highest TPR rate which contributed to the best F1-score and accuracy. The size of the time window at 15 and 20 worked worst with the lowest TPR rate just above 80%. Above the time sliding window size 20, we observed that TPR and accordingly F1-score started decreasing as the number of time sliding windows increased.

4.6.3.5 Model Performance

Fig. 4.15 illustrates the performance of our model based on the confusion matrix. The total number of test samples = 42,787 containing the normal samples = 40,697 and the abnormal samples = 2,100 according to our label. Our model was able to detect a total of 1,888 abnorml data points correctly out of the 2,100 abnormal data samples (i.e., 89.90%

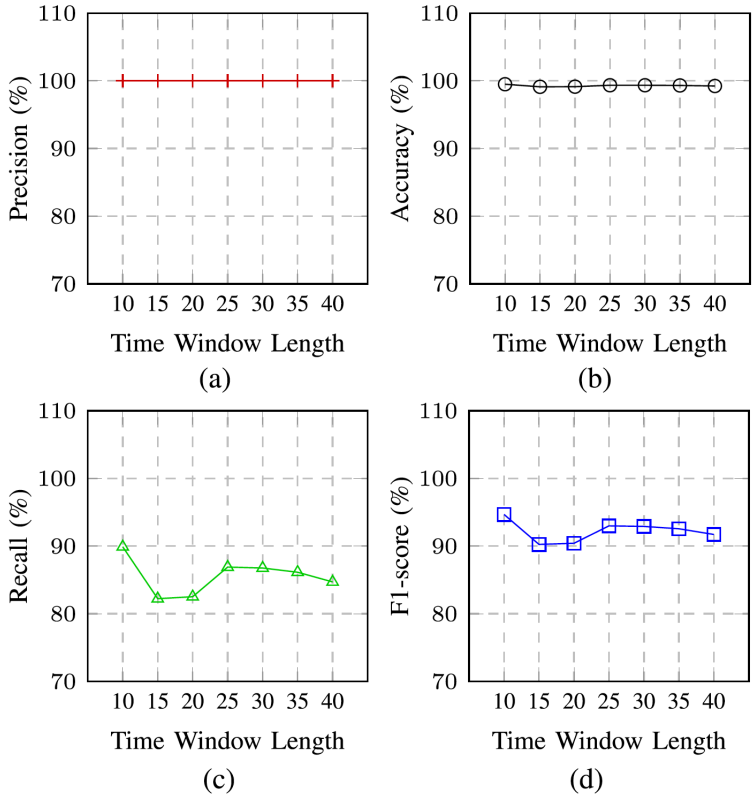


Figure 4.14: Performance comparison of LSTM-AE under different time window length

of accuracy). Our model detected the total number of 40,697 normal data points correctly out of the 40,697 normal data samples (i.e., 100% of accuracy). Our model had none of FP by incorrectly classifying the normal samples as abnormal while it had 212 of FN by incorrectly classifying the abnormal samples as normal. By accounting for all of these, our model resulted in an accuracy of 99.50%, a precision of 100%, a recall of 89.90%, and an F1-score of 94.68%.

Fig. 4.16 shows our model performance in terms of the AUC-ROC graph that clearly demonstrates the trade-off between true positive rate and false-positive rate. The curve confirms that our proposed model is highly effective in accurately detecting anomalies by achieving an AUC-ROC score of 95.0%. This result is calculated based on the whole time series testing dataset. To detect the model efficiency, we also compared different time window lengths in 1 minutes intervals. We observed that the AUC-ROC curve starts decreasing as the size of the time window length increases. The best performance was shown at 95.0% when the size of the time window length = 10 while the worst performance was shown when the size of the time window length was = 10 and 20. Similar to the performance of the impact of the time sliding window, the AUC-ROC score decreased slightly as the size of the time window increased from the size of over 25.

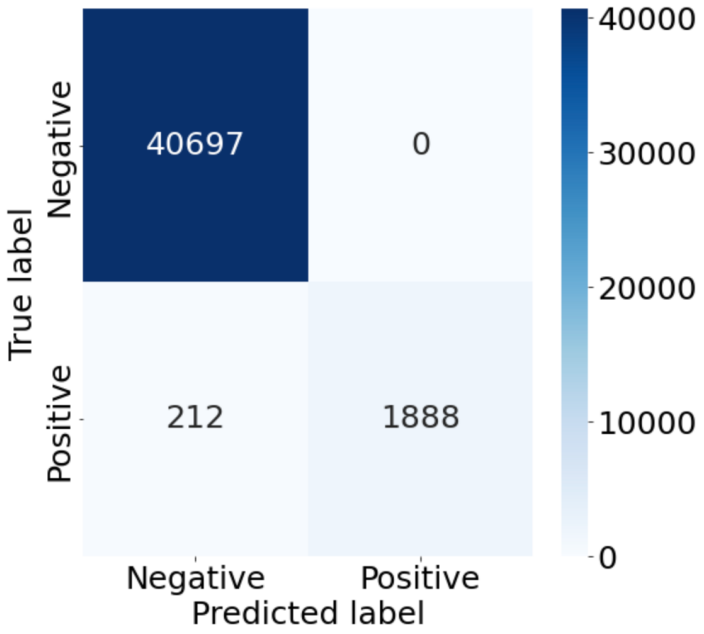


Figure 4.15: Detection results based on Confusion Matrix

Fig. 4.17 shows where anomalous data points were detected (i.e., see by the red dots) based on the test dataset. In our observation, the threshold was set at 1.742 when the training was done. During the test, we also observed that any CO_2 readings greater than 1,000 usually had a reconstruction error rate greater than 1.742 and therefore were considered anomalous.

4.6.3.6 Comparison to Other Similar Models

Table 4.6 shows the performance comparison of our model evaluated on the Dunedin CO_2 dataset with other similar models that use different variations of LSTM Autoencoder. As the result shows, our approach shows the best performance in terms of accuracy (at 99.50%) and precision (at 100%). The similar model proposed by Yin et al. [97] shows the most competitive performance compared to ours with similar accuracy and F1-score. The model proposed by Nguyen et al. [54] shows a higher F1-score (at 96.98%) though the accuracy of their model is lower than our method. In our further investigation, they used a One-Class SVM as an additional classifier to reduce the false positives.

4.7 Conclusions

We proposed an LSTM-Autoencoder-based deep-learning technique for detecting anomalies in indoor air quality datasets. Using the latest advancement in time-series data analysis using an LSTM model, our proposed model provides a great capability to learn a

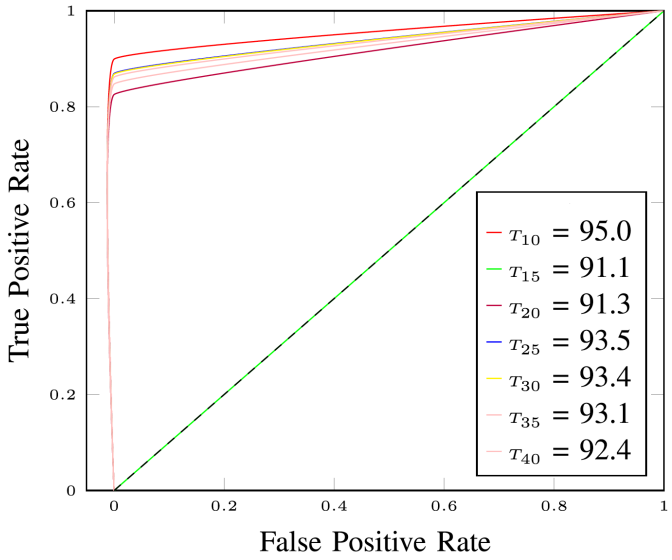


Figure 4.16: AUC-ROC visualization

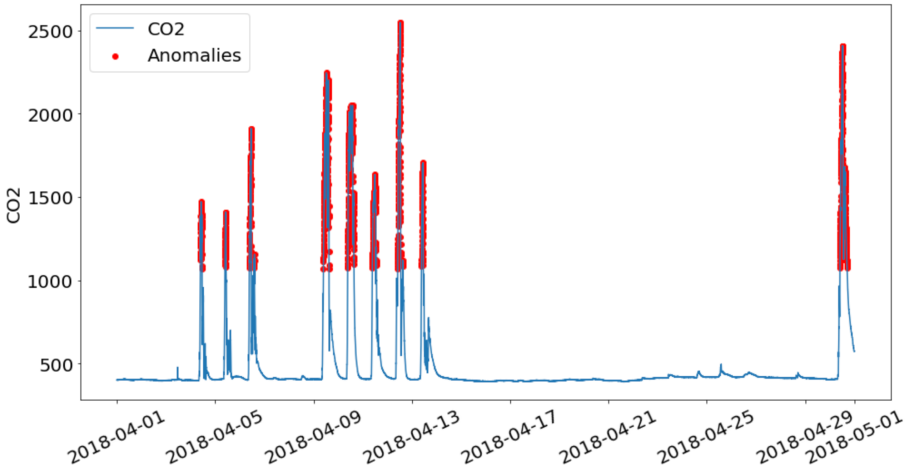


Figure 4.17: Normal and anomalies distribution on the testing set

pattern that exists in the CO_2 time sequence data. Together with Autoencoder, our proposed model computes a typical reconstruction error associated with each time sequence pattern and uses this information for anomaly detection while providing more efficient model training with data dimension reduced. Our hybrid approach effectively removes the limitations associated with the existing state-of-the-art by removing the requirement of having to often manually extract relevant features and be sensitive to the presence of outliers often contributed to poor performance.

Our proposed model was applied to the CO_2 time-series dataset obtained from a real-world deployment. The experimental results showed that our proposed model is highly

Table 4.6: Comparison to other similar models

Paper	Techniques	Datasets	Acc	Pre	Re	F1
1	LSTM-AE	Yahoo Webscope S5	99.25	97.84	94.16	95.97
2	LSTM-AE	ECG	98.57	97.55	97.55	-
3	LSTM-AE	WSNs position at IBRL	-	89.1	86.9	85.24
4	VAE-LSTM	Ambient temperature	-	80.6	1.0	89.2
		CPU utilization AWS	-	69.4	1.0	81.9
		Machine temperature	-	55.9	1.0	71.7
5	LSTM-AE	CERT insider threat dataset	90.17	-	91.03	-
6	LSTM-AE	Numenta Anomaly Benchmark (NBA)	-	90.8	98.8	94.6
7	LSTM-AE adversarial learning	e-VDS	94.16	90.31	88.45	89.37
		CCV	83.03	75.08	73.26	74.16
8	LSTM-AE	Break Operating Unit (BOU) data	94.44	97.94	85.77	91.45
9	LSTM-AE-OCSVM	Generated dataset	98.36	98.45	99.59	96.98
10	LSTM-AE iforest	simulated data in Fashion industry	95	100	94	87
Our study	LSTM-AE	Dunedin CO2 Dataset	99.50	100	89.90	94.68

efficient for detecting anomalous CO_2 values with high accuracy of 99.50% and outperforms other similar models.

Chapter 5

Classification and Explanation of Distributed Denial-of-Service (DDoS) Attack Detection using Machine Learning and Shapley Additive Explanation (SHAP) Methods

Abstract

DDoS attacks involve overwhelming a target system with a large number of requests or traffic from multiple sources, disrupting the normal traffic of a targeted server, service, or network. Distinguishing between legitimate traffic and malicious traffic is a challenging task. It is possible to classify legitimate traffic and malicious traffic and analysis the network traffic by using machine learning (ML) and deep learning (DL) techniques. Moreover, ML/DL techniques are faster to identify DDoS attacks and allow for efficient response and mitigation. However, due to the model's complexity, especially deep learning, it was difficult to gain the explanation that produced the reasons behind the model's decisions. To this end, Explainable Artificial Intelligence (XAI) can explain the decision-making of the ML/DL models that can be classified and identify DDoS traffic. In this context, we proposed a framework that can not only classify legitimate traffic and malicious traffic of DDoS attacks but also use SHAP to explain the decision-making of the classifier model. To address this concern, we first adopt feature selection techniques to select the top 20 important features based on a combination of feature importance techniques (e.g., XGB-based feature importance, Permutation feature importance, and SHAP feature importance). Following that, the Multi-layer Perceptron Network (MLP) part of our proposed model uses the optimized features of the DDoS attack dataset as inputs to

classify legitimate traffic and malicious traffic. We perform extensive experiments with all features and selected features in one-to-one (benign vs. one attack type) and one-to-all (benign vs. multi-attack types) scenarios. The evaluation results show that the model performance with selected features achieves above 99% accuracy in both one-to-one and one-to-all classifications. Finally, to provide interpretability, XAI can be adopted to explain the model performance between the prediction results and features based on global and local explanations by SHAP, which can better explain the results achieved by our proposed framework.

5.1 Introduction

Cybersecurity refers to the aspect of protecting the individual or organization's network and information systems to defend against cyber attacks, including protecting data availability, confidentiality, and integrity [71]. In the era of rapid information technology development, network security is becoming increasingly important to networks and computer users. However, the rapid proliferation of innovative technologies and communication infrastructure brings the potential for cyberattacks and other threats to Internet users [27, 64, 99]. One of the significant cyber attacks is the distributed denial-of-service (DDoS) attack. A DDoS attack is a cyberattack that poses serious risks to the network system service. A DDoS attack is a kind of malicious traffic attempt to disrupt the normal traffic of the targeted server or network service by sending a flood of malicious traffic, making the target services unavailable to legitimate users [12, 20, 37, 67]. Cyber-attacks, especially DDoS attacks can be used to maliciously disable computers, prevent access to them, steal data, or use a compromised computer as a launching pad for other attacks [12]. As a result, preventing and mitigating DDoS attacks is critical to protecting the network from significant financial loss [21, 68].

In recent years, machine learning (ML) has been proposed and developed by many researchers for the detection of DDoS attacks. For example, K-Nearest Neghbors (KNN) [19], Random Forest (RF) [15], Extreme Gradient Boosting (XGBoost) [16], and Decision Tree (DT) [34, 38]. In addition, Deep Learning (DL) techniques have also been used to detect DDoS attacks. These include long short-term memory (LSTM) [12], Autoencoder (AE) [68], AE-MLP [89], Convolutional Neural Network (CNN) [27], etc. As aforementioned, many researchers achieve power in terms of performance and prediction by using AI techniques, especially ML learning and DL learning in terms of application in the cybersecurity domain, for example, DDoS attack detection, intrusion detection, and malicious detection based on benchmark datasets such as NSL-KDD, CICIDS 2017, CICDDoS2019 datasets [4, 29, 35, 57, 61]. This AI-based ML and DL techniques for DDoS detection can be divided into two categories: white box and black box models. The white-box decision-making of some of the ML models can be easily explained and trusted by domain experts based on the interpretable output like linear models. [45, 75]. However, despite the good performance of the AI-based model, the domain experts were still unable to understand the inner workings of some AI-based models and fully trust them, resulting in a potential loss of security and trust. These models can be referred to as black-box models in ML and DL models. With the complexity of the AI models being developed, understanding the good performance of the decisions made by AI-based models has become an important issue so that domain experts can gain more confidence in the black-box models. Therefore, it is important to make a correct and transparent interpretative prediction. To address this issue, Explainable Artificial Intelligence (XAI) is proposed to investigate the interpretability of the inner logic of the model and to explain

the decision-making behind the model, thereby making the model transparent to user understanding.

In this study, we propose a framework using the Shapley Additive Explanations (SHAP) technique to address cybersecurity issues such as misclassification and provide a transparent explanation for DDoS detection. The goal of this paper is twofold. First, a combination of XGB-based feature importance (e.g. SHAP feature importance) uses as a feature extraction to select the top n contributing features that feed into the classifier model in order to obtain the accuracy and misclassifications of the predictions. Second, the XAI-based technique, such as SHAP, provides global and local interpretations that can explain the contribution of the features for the decision-making of the model.

The main contributions of our proposed model are the following.

- We propose a novel framework that consists of two components: DDoS classification using MLP and XAI-based technique of SHAP for transparency to explain the most contributing features of the classification of the benign and attack traffics. This work helps the users trust and have a better understanding of the prediction results of the proposed model.
- We propose the XAI-based SHAP interpretation framework for the global and local explanation of the classification result in DDoS attack detection. The global explanation we conduct is based on summary plots and dependence plots. For the local explanation, we focus the analysis on a single feature in four circumstances, which are benign and malicious traffic classified correctly, and misclassify traffic, while other studies only provide the analysis on misclassified traffic.
- We conducted an extensive evaluation using the CICDDoS2019 dataset to find the most corresponding features, with the explanation attached as the best exploration of the most contributing features. The evaluation results show that using the feature selected by the XAI feature importance technique, the classification results show higher performance as well as without feature selection.

The rest of this Chapter is structured as follows: Section 5.2 introduces related works in the field of DDoS attack detection and explanation. Section 5.3 introduces our methodology. Section 5.4 illustrates the experimental setup and Section 5.5 details the analysis of our results and evaluated the global and local explanations. Section 5.6 concludes the paper with the planned future works.

5.2 Related Work

In recent years, ML and DL techniques have been widely used to detect DDoS attacks, analyze network traffic patterns, and detect normal or abnormal behavior that may detect DDoS traffic efficiently. However, these techniques are commonly referred to as black-box

models, which can make it difficult to understand the inner workings of the decisions behind them. XAI is an area of research that aims to make the results of AI-driven models more transparent and easier for domain experts to understand. This section discusses some existing work that addresses DDoS attack detection and explainability.

5.2.1 Machine learning for DDoS

Wang et al. [85] used a dynamic multilayer perceptron (MLP) with 31 optimized sequence features and feedback mechanism to detect DDoS attacks based on the NSL-KDD dataset, achieving 97.66% accuracy with the SBS-MLP classifier. Wei et al. [89] proposed a hybrid deep learning Autoencoder-MLP (AE-MLP) for DDoS detection and classification. They first used the autoencoder to extract 5 optimal features, which were fed into MLP classifiers to perform multi-class classification of different attack types on the CICDDoS2019 dataset. The evaluation results achieved over 98% accuracy in classifying all attack types. In addition, Can et al. [14] proposed Distributed Denial of Service attacks based on neural networks (DDoSNet) to detect and classify DDoS attacks based on a fully-connected MLP classifier with 24 selected features from the CICDDoS2019 dataset. The proposed method achieved 99% accuracy for binary classification. Samom et al. [78] used machine learning models (i.e., Logistic Regression, Random Forest, Multi-Layer Perceptron, etc) to classify four different attack types (i.e., SYN, NET, Portmap, and UDPLag) with 20 selected features from CICDDoS2019 dataset. The results showed that Random Forest demonstrated the best performance in classification results. The aforementioned machine learning techniques achieved higher performance accuracy for DDoS detection and classification with selected features, but lower performance as they used the entire feature set for classification.

5.2.2 XAI for models

Kalutharage et al. [33] proposed XAI-based techniques for detecting DDoS attack anomalies on the USBIDS (University of Sannio, Benevento Instrution Detection System) dataset. The research focuses on instance-by-instance, local and global explanations, and feature correlations, explaining anomalies by providing Autoencoder and Kernel SHAP techniques. The proposed method achieves a better accuracy (98%) on HULK attacks (Hulk No Defense) compared to other detection methods, such as Decision Tree (97%), and Deep Neural Network (67%). However, this research was implemented in a static dataset. Antwarg et al. [11] proposed Kenal SHAP for explaining anomalies on the NSL-KDD dataset by an unsupervised model of Autoencodr. The main focus was on explaining to the experts the relationship between the impact of reconstruction error features and high reconstruction error features. The explanation presented the main contribution and offset the important features of the anomalies. The evaluation of the explanation also showed that utilizing the SHAP explanation technique to explain the generated subset

of explanatory features is more robust than the other explanation techniques of LIME. Šarčević et al. [71] provided both SHAP's XAI technique and If-then decision tree rules to extract information from a network attack dataset CIC-IDS2017 to mitigate network security. The challenges of these two techniques for information extraction were compared and their use in different situations was pointed out. The limitations of both techniques were also highlighted. If-then decision tree rules face the challenge of increasing the depth of the trees and cannot provide transparency in the decision-making of the results, while SHAP provides an adequate explanation of the model but is less comprehensive.

Tabassun et al. [80] extended their work by employing XAI techniques of SHAP, LIME, and ELI5 to explain the classification of IoT network attacks (DDoS attacks) based on machine learning and deep learning models (e.g., DT, RF, Adaboost, ANN, etc). XAI explanation techniques are based on the decision-making of the binary classification results of 96% accuracy of all models. Using SHAP for a both local and global explanation of the prediction results based on SHAP values, while LIME provided only a local explanation, and ELI5 pointed out the most important features both locally and globally. Houda et al. [3] proposed an XAI framework to explain the decision of deep learning for Internet of Things (IoT)-related Intrusion Detection Systems (IDSs). They also provide three XAI techniques, including SHAP, LIME, and RuleFit, to optimize the interpretability of deep learning decisions through global and local explanations.

In this study, we propose a framework for classifying legitimate traffic (benign) and malicious (attack) traffic based on a machine learning technique of MLP classifier and give an interpretable to the trained model using the XAI-based SHAP method. Compared to the aforementioned methods [14, 85, 89]uses a different feature selection technique to select the top n features, we proposed a feature selection framework by selecting the top 20 important features based on the combination of XGB-based feature importance, Permutation, and SHAP feature importance (based on the counting frequency of occurrence of each feature).

5.3 Methodologies

Machine learning models are also regarded as "black box" models because of the difficulty to explain and interpret. However, modern interpretability in machine learning has been improved and implemented from complex ML models, which can be categorized into two kinds of interpretability: global and local interpretability. Global interpretability aims to understand the model structure based on its features, while local interpretability wants to find out the reason for making the decision. SHAP is a method that can be offered global and local interpretability of the model.In this paper, SHAP interpretability can be used to extract the most contributed features as a method of feature selection to improve the classification efficiency and also explained the model. SHAP can be introduced in the following selection. SHAP commonly connects to the Shapley value, thus, the Shapley value also can be depicted in this section.

5.3.1 SHAPLEY value

The Shapley value introduced by Shapely [73], is a method from cooperative game theory to determine individual contributions among the features. The method can be contributed to machine learning prediction, weighted, and encapsulated through the contribution of each feature value. The Shapley value of a feature value is its average contribution to the prediction in different coalitions:

$$\phi_j(\nu) = \sum_{S \subseteq \{x_1,\ldots,x_p\} \setminus \{x_j\}} \frac{|S|!(p-|S|-1)!}{p!}(\nu(S \cup \{x_j\}) - \nu(S)) \tag{5.1}$$

where:

- $S \subseteq \{x_1, ..., x_p\} \setminus \{x_j\}$ represents that S is a subset of p features in the model, x is the vector of feature value of the instance to be explained, and p is the number of features.
- $\nu(S)$ is the prediction for feature values in set S.

The Shapley value is the sole attribute technique that satisfies four properties: Efficiency, Dummy, Symmetry, and Additivity. These four properties are required for a definition of a fair prediction (payout).

Efficiency: The feature contributions have to add up to the difference of prediction for the average predicted value.

Dummy: The feature j does not modify the predicted value regardless of which irrespective of feature values it is added to, this should have a 0 Shapley value.

$$\nu(S \cup \{x_j\}) = \nu(S), for\ all\ S \subseteq \{x_1, ...x_p\},\ then\ \phi_j = 0 \tag{5.2}$$

Symmetry: If the two features j and i contribute equally to all possible coalition, then the contributions of these two features should be the same.

$$\nu(S \cup \{x_j\}) = \nu(S \cup x_i),\ for\ all\ S \subseteq \{x_1, ..., x_p\} \setminus \{x_j\}, \quad then\ \nu_j = \nu_i \tag{5.3}$$

Additivity: The respective Shapley values for a game with combined payouts $\nu + \nu^+$, which can be show as follows:

$$\phi_j + \phi_j^+ \tag{5.4}$$

5.3.2 Shapley Additive Explanations (SHAP)

SHAP proposed by Lundberg LEE [45], is a unified framework for the interpretation prediction of models. SHAP aims to explain the prediction of an instance x by computing the contribution of each feature to the final predictions of the model. This contribution of each feature can be either positive or negative. One innovation of the SHAP explanation

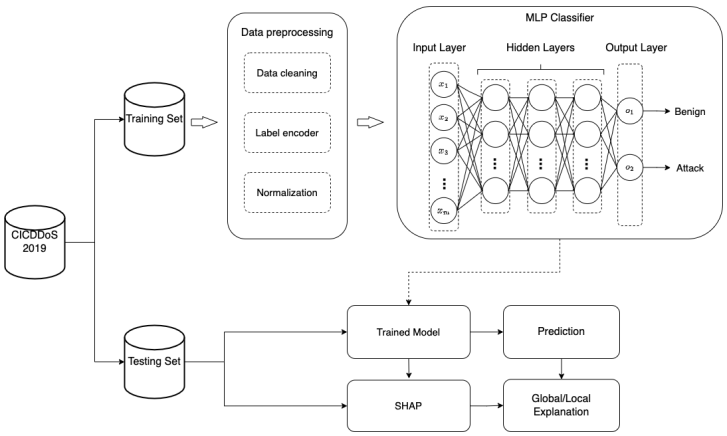


Figure 5.1: Overview of our proposed framework

method brings to the table is that the shapley value explanation is represented as a linear model - an additive feature attribution method. The strength of SHAP can be computed for any model rather than only a linear model. Moreover, instead of only explaining local interpretations, the SHAP interprets global interpretation by summing the attributed value of each input feature and averaging features individually. SHAP defines the explanation of an instance x can be expressed as follows:

$$g(z') = \phi_0 + \sum_{j=1}^{M} \phi_j {z'}_j \tag{5.5}$$

where

- g represents the explanation model.
- z' represents simplified features, and $z' \in \{0,1\}^M$.
- M represents the maximum coalition size.
- $\phi_j \in R$ is the shapley value, representing the feature attribution for a feature j.

The Shapley values have two major advantages over other methods. First, the Shapley values are a method from coalitional game theory. Shapley values are the solutions that satisfy four properties: Efficiency, Symmetry, Dummy, and additivity. It helps to unify the field of interpretable machine learning and deep learning models.

5.3.3 Proposed Methodology

The proposed framework is divided into the classification, and model explanation (see Fig. 5.1).

Before feeding the data into the classifier, selecting the most significant important, and relevant subset of features is an important step in the optimization process that can

improve the performance efficiency of the model. Feature selection is an important phase to select the most important features, which can 1)reduce feature space or dimensional; 2) reduce the computational time; 3) improve the model efficiency. Several feature selection techniques are depicted in [24] to gain the most relevant features, including Random Forest with SHAP-based, and XGB with Permutation-based, SHAP-based, etc. Therefore, in this work, we adopt a combination method of XBGoost-based feature importance, Permutation importance, and SHAP feature importance to select the top 20 important features to feed into the classifier to classify the benign traffic and attacks. Table 5.1 shows the top 20 important features based on XGB-based feature importance methods. According to the occurrence frequency of each feature, we select the top 20 features based on the order of most occurrence frequency from high to low. As a result, the top 20 most important features in terms of single feature importance (FI) and multiple feature importance are shown in Table 5.1. Note that in Table 5.1, three of the top 20 XGB-based feature importance methods are based on counting single features. The top 20 important features of each XGB-based technique shown in the figure are based on counting single features. Similarly, for the single-multiple importance features, the top 20 most important features of each XGB-based technique of each attack were counted separately, and then the top 20 were selected from the frequency of occurrence of each feature. For example, we counted the top 20 important features of each DNS, SNMP, LDAP, and NetBIOS attack separately, and then selected the top 20 based on the frequency of occurrence of each feature of each attack. As a consequence, the top 20 important features of one-multiple feature importance are shown in Table 5.1.

After feature selection, we use the MLP classifier to classify the benign traffic and the attacks. Note that the benign traffic is classified as 1, while the attack is classified as 0. The Multi-Layer Perceptron (MLP) is a feed-forward network, it has an input layer, multiple hidden layers, and an output layer (equal to attacks and benign traffic). The setting of hyperparameters in this proposed framework is employed to better classify benign traffic and attacks, which is based on the MLP classifier from [89]. We have experimented with the best optimized MLP architecture the one that uses 5 layers – 1 input layer, 3 hidden layers, and 1 output layer. The Hyperparameters of hidden layer size are [23, 15, 10]. Note that the output layer mostly uses a sigmoid function for binary class classification problems.

After training the classifier model of MLP, then we obtain a trained classifier model (also refer a black box model), and prediction results. Finally, we use the SHAP explanation technique to explain the model based on the global and local explanation. The overview of our the proposed framework of classification is depicted in Fig. 5.1. To explain the trained black box model, we use Kernel SHAP to obtain the global and local explanation (Fig. 5.1). Kernel SHAP is suitable for the classification model with tabular data, which can be applied to any model.

Table 5.1: Top 20 important features based on three XGB-based feature importance methods

	XGBoost Feature Importance	Permutation Importance	SHAP Feature Importance	one-one-FI	one-multiple-FI
1	Bwd Packets/s	Inbound	Min Packet Length	Protocol	Protocol
2	Protocol	Protocol	Fwd Packet Length Mean	URG Flag Count	URG Flag Count
3	Inbound	Init Win bytes forward	Fwd Packet Length Min	Flow Duration	Init Win bytes forward
4	Init Win bytes backward	URG Flag Count	Average Packet Size	Init Win bytes forward	Min Packet Length
5	URG Flag Count	Min Packet Length	Fwd Packet Length Max	Fwd Packet Length Min	Fwd Packet Length Max
6	ACK Flag Count	Fwd Packet Length Max	Max Packet Length	Min Packet Length	Inbound
7	Total Fwd Packets	Bwd Packets/s	Protocol	Fwd Packets/s	Total Backward Packets
8	Flow Duration	Max Packet Length	Flow IAT Mean	Max Packet Length	Flow IAT Min
9	Init Win bytes forward	ACK Flag Count	Init Win bytes forward	Flow IAT Min	Flow Duration
10	Fwd Packet Length Min	Fwd Packet Length Min	URG Flag Count	Fwd Packet Length Max	Bwd Packets/s
11	Bwd IAT Total	Flow Duration	Flow Duration	Average Packet Size	Fwd Packets/s
12	Total Backward Packets	Average Packet Size	Flow IAT Std	Bwd Packets/s	Max Packet Length
13	Packet Length Std	Init Win bytes backward	Flow IAT Min	Inbound	Init Win bytes backward
14	Min Packet Length	Fwd Packets/s	Fwd Packets/s	Init Win bytes backward	Fwd Packet Length Min
15	Active Min	Fwd IAT Min	Total Length of Fwd Packets	ACK Flag Count	Fwd Packet Length Mean
16	Fwd Packets/s	Packet Length Mean	Fwd IAT Total	Total Fwd Packets	Average Packet Size
17	Max Packet Length	Flow IAT Min	Total Fwd Packets	Total Backward Packets	Packet Length Std
18	Flow IAT Min	Flow IAT Mean	Total Backward Packets	Flow IAT Mean	Packet Length Mean
19	Fwd Packet Length Max	SYN Flag Count	act data pkt fwd	Packet Length Std	SYN Flag Count
20	Average Packet Size	CWE Flag Count	Fwd IAT Mean	Bwd IAT Total	Bwd IAT Total

5.4 Data and Experiment Setup

Our experiments were carried out using the system setup shown in Table 5.2.

Table 5.2: Implementation environment specification

Unit	**Description**
Processor	3.4GH$_z$ Inter Core i5
RAM	16GB
OS	MacOS Big Sur 11.4
Packages used	tensorflow 2.0.0, sklearn 0.24.1

5.4.1 Data Pre-prosessing

In this section, we discuss the methodologies we used to process our dataset in order to feed it into our proposed model.

5.4.1.1 CICDDOS2019 Dataset

In this study, we use CICDDoS2019 [74] dataset that has been widely used for DDoS attack detection and classification. The dataset contains a large amount of up-to-date realistic DDoS attack samples as well as benign samples. The total number of records contained in CICDDoS2019 is depicted in Table 5.3. Table 5.3 illustrated that all data was captured in two days: training day and Testing day. Furthermore, all attack types can be collected from the application layer by using TCP/UDP-based protocols, which can be separated as reflection and exploitation-based DDoS attacks. The taxonomy of the CICDDoS2019 dataset is depicted in Table 5.4, including 13 different attack types, in terms of MSSQL, SSDP, DNS, LDAP, NetBIOS, SNMP, NTP, TFTP, UDP, UDP-Lag, SYN, PORTMAP, WebDDoS, and its attack time. Note that the "WebDDoS" attack was collected and saved together with the "UDPLag" attack file. In this research, In this research, we use four reflection-based attack types: DNS, LDAP, NetBIOS, and SNMP for classification and explanation, which is captured from 10:52 to 12:23 for the training day on January 12th.

Table 5.3: The number of records in CICDDoS2019

dataset	**total**	**benign**	**Attacks**
Training day	50,063,112	56,863	50,006,249
Testing day	20,364,525	56,965	20,307,560

5.4.1.2 data imbalance

In classification scenarios, Solving data imbalance is a common issue in a kind of machine learning problem. Table 5.3 shows the number of records in the CICDDoS2019 dataset,

Table 5.4: The taxonomy of CICDDOS2019 dataset

Collected Day	DDOS Attack	Protocol	Attack Types	Attack Time
Training day	Reflection-based	TCP	MSSQL	11:36-11:45
			SSDP	12:27-12:37
		TCP/UDP	**DNS**	10:52-11:05
			LDAP	11:22-11:32
			NetBIOS	11:50-12:00
			SNMP	12:12-12:23
		UDP	NTP	10:35-10:45
			TFTP	13:35-17:15
	Exploitation-based	UDP	UDP	12:45-13:09
			UDP-Lag	13:11-13:15
		TCP	SYN	13:29-13:34
Testing day	Reflection-based	TCP/UDP	PortMap	09:43-09:51
			NetBIOS	10:00-10:09
			LDAP	10:21-10:30
		TCP	MSSQL	10:33-10:42
	Exploitation-based	UDP	UDP	10:53-11:03
			UDP-Lag	11:14-11:24
		TCP	SYN	11:28-17:35

which can be seen as having an uneven distribution of the number of records. For example, benign samples have a lower number of records (56,863), while attacks have a large number of records (50,006,249) on training day collection. In this research, we use a DSN attack as a one-to-one classification, including benign traffic (labeled as 1) and a DNS attack (labeled as 0), and a one-to-multiple classification, including benign traffic (labeled as 1) and four attack types (malicious): DNS, LDAP, NetBIOS, SNMP attacks (labeled as 0). In both two scenarios, all benign traffic has been extracted first, then combined with 0.1% four attacks separately as the classification and explanation dataset.

5.4.1.3 Data Cleaning

The original CICDDoS2019 dataset contained 88 features. As suggested by [8], we also removed the features not contributing to detecting DDoS attacks. These include the

feature such as "Unnamed", "Flow ID", "Source IP", "Destination IP", "Source Port", "Destination Port", "Timestamp", "Flow Bytes", "Flow Packets", and "SimilarHTTP". After the exclusion of these 10 features, we have 78 features to work with. Following the recommendation of the work by [13, 69], we cleaned up the values containing NaN (not a number), redundant, and infinity values, which can significantly affect the data modeling efficiency and data knowledge discovery.

5.4.1.4 Label Encoding

We substituted the categorical labels with deep models as they only operate on float/numeric values. LabelEncoding is one approach to achieve this, where each unique category is assigned a numeric value. After employing LabelEncoding, we have obtained the label of the DDoS traffic flow as either benign traffic (1) or malicious traffic (0). Note that a one-to-one scenario represents benign and malicious (DNS attack traffic), while a one-to-all scenario represents benign and malicious (including DNS, LDAP, NetBIOS, and SNMP four attack types and labeled as malicious).

5.4.1.5 Data Normalization

The CICDDoS2019 datasets contain some features with very high variance in terms of value between the minimum and the maximum (e.g., "Min Packet Length", "Flow Duration", "URG Flag Count", "Fwd Packet Length Mean", etc.). We applied a normalization strategy to eliminate the impacts of big variance of the values across the features thus reducing the execution time for model training and improving accuracy. There are several widely used methods to perform feature scaling, including Z Score, standardization, and normalization. As proposed by [82], we use MinMax-based normalization for our feature scaling. This method maps the original range of each feature into a new range with Equation (5.6)

$$Z_i = \frac{Z_i - min}{max - min} \tag{5.6}$$

where Z_i donates all the normalized numeric values ranging between [0-1]; max and min donate the maximum and minimum values from all data points.

5.4.2 Performance Matrix

To evaluate the performance of our model, we used the following metrics: classification accuracy, precision, recall, and F1 score. Table 5.5 illustrates the confusion matrix.

where;

- True Positive (TP) indicates an anomalous data point correctly classified as anomalous.

Table 5.5: Confusion Matrix

Total Population		Predicted Condition	
		Normal	Anomaly
Actual Condition	Normal	TN	FP
	Anomaly	FN	TP

- True Negative (TN) indicates a normal data point correctly classified as normal.
- False Positive (FP) indicates a normal data point incorrectly classified as anomalous.
- False Negative (FN) indicates an anomalous data point incorrectly classified as normal.

Based on the aforementioned terms, the evaluation metrics are calculated as follows:

- True Positive Rate (also known as Recall) estimates the ratio of the correctly predicted samples of the class to the overall number of instances of the same class.

$$TPR(Recall) = \frac{TP}{TP + FN} \tag{5.7}$$

- False Positive Rate (FPR) presents the proportion of data points correctly classified as anomalous.

$$FPR = \frac{FP}{FP + TN} \tag{5.8}$$

- Precision (Pre) measures the quality of the correct predictions.

$$Precision = \frac{TP}{TP + FP} \tag{5.9}$$

- F1-Score computes the trade-off between precision and recall.

$$F1 - score = 2 \times \left(\frac{Precision \times Recall}{Precision + Recall} \right) \tag{5.10}$$

- Accuracy (Acc) measures the total number of data samples correctly classified.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.11}$$

The Area Under the Curve (AUC) computes the area under the Receiver Operating Characteristics (ROC) curve which is plotted using the true positive rate on the y-axis and the false positive rate on the x-axis over different thresholds. Mathematically, the AUC is computed as shown in Equation (5.12).

$$AUC_{ROC} = \int_0^1 \frac{TP}{TP + FN} d\frac{FP}{TN + FP} \tag{5.12}$$

Table 5.6: Performance matrix based on benign and attack

Attack Types	One-to-one			**One-to-one-FS**		
	Precision	Recall	F1-score	Precision	Recall	F1-score
Benign	99.91	99.83	99.87	100	99.91	99.95
Attack	99.80	99.90	99.85	99.90	100	99.95
	One-to-all			**One-to-all-FS**		
	Precision	Recall	F1-score	Precision	Recall	F1-score
Benign	99.93	99.81	99.87	100	99.91	99.95
Attack	99.90	99.96	99.93	99.90	100	99.95

5.5 Experimental Results

The experiment results are evaluated based on the performance matrix, in terms of accuracy, precision, recall, and F1-score. The confusion matrix of accuracy refers to the degree of closeness of a given set of traffic samples and its true samples, while F1-score is evaluated the classifier performance. Furthermore, we also give an explanation of the decision-making of the classifier model based on global and local explanations.

5.5.1 DDoS Binary Classification

Table 5.6 shows the experimental results of the proposed model, which achieved over 99% of all performance metrics in terms of over 99.91% of all precision, 99.83% recall, and 99.87% F1 score for benign traffic, while attacks achieved over 99.80% of all precision, 99.90% recall, and 99.84% F1 score. Furthermore, the feature selection results achieved better performance in terms of precision, recall, and F1 score in both one-to-one and one-to-all scenarios than the performance of using all features. Moreover, Table 5.7 shows that the total performance of accuracy of feature selection performed better than the accuracy results of using all features. For example, one-to-one and one-to-all scenarios with feature selection achieved 99.95% accuracy, while one-to-one and one-to-all scenarios with all features achieved 99.86% and 99.91% accuracy separately.

Table 5.7: Total performance matrix

Performance	Accuracy	Precision	Recall	F1-score	AUC
One-to-one	99.86	99.91	99.83	99.87	99.99
One-to-one-FS	99.95	100	99.91	99.95	100
One-to-all	99.91	99.93	99.81	99.87	99.95
One-to-all-FS	99.95	100	99.91	99.95	99.99

5.5.2 Model Explanation

In order to establish the trustworthiness of the prediction results of the proposed model, we use SHAP to explain the decision-making of the prediction results and to explain why DDoS traffic flows are classified as benign traffic or attacks. As mentioned above, SHAP is a method to generate an interpretation of the classifier's prediction based on Shapley values and the most important features. The explanation of SHAP can: 1) represent the importance of the features based on the SHAP values and presented in decreasing order from high to low; 2) provide insight into the contribution of high or low feature values to the prediction outcomes. We employ SHAP to investigate how the classifier correctly makes the classification decision between benign and malicious traffic by global and local explanations.

5.5.2.1 Global Explanation

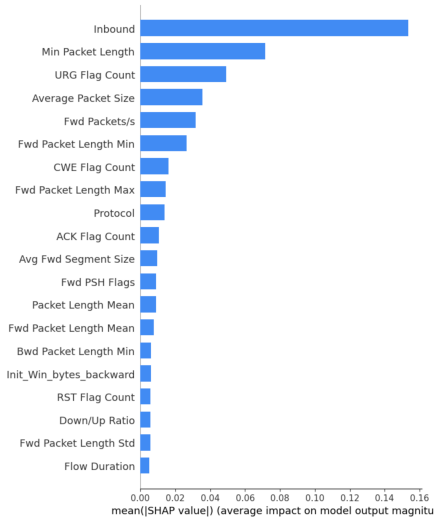
Global explanation aims to be explained the feature contribution of the prediction results. In this study, we use the test set, which consists of a subset of approximately 0.001% of the DDoS dataset, to generate explanations in the form of Shapley values. The global explanation of the prediction outcomes contains two scenarios between one-to-one and one-to-all explanations accompanied by all features and selected features separately. Note that the top 20 important features shown in the figure (e.g. Fig. 5.2a and Fig. 5.3a) are plotted using all features. Thus, we select the top 20 important features for feature selection based on the proposed feature selection method shown in Fig. 5.2c and Fig. 5.3c).

Global SHAP explanation is commonly presented in two visualization plots. One of the most important plots in the explanation is the summary plot (Fig. 5.2b and Fig. 5.3b). A dependence plot is another common global interpretation visualization plot (Fig. 5.4 and Fig. 5.5).

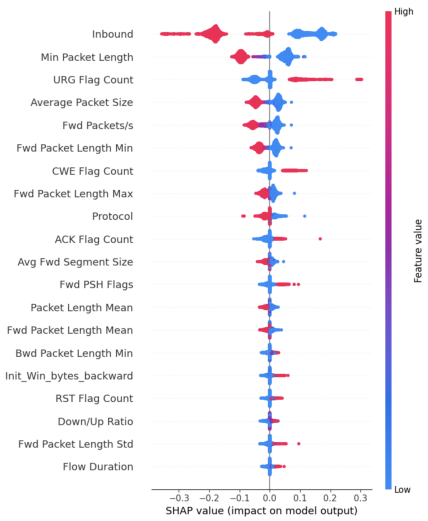
Summary Plot

Fig. 5.2a and 5.2c show SHAP values in bar plots visualizing the one-to-one scenario based on the global importance of each feature with or without feature selection, where the global importance of each feature takes the mean absolute value for that feature across all given samples. Fig. 5.2b shows the visualization of the summary plot of the SHAP values for the top 20 contributing features by using all features from the testing DDoS dataset, while Fig. 5.2d depicts the summer plot by using the top 20 selected features. The most contribution of the top 20 features is shown on the left in descending order from high to low.

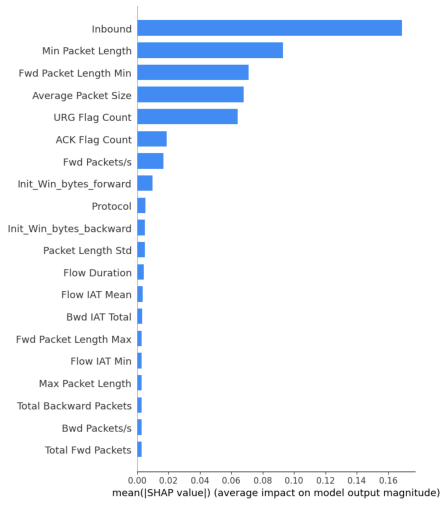
In the Fig. 5.2b and Fig. 5.2d of the summary plots, a single point is plotted for each traffic flow (a dot represents a traffic flow). The SHAP values that fall on the left side of the x-axis have a negative impact on the prediction results, lowering the predicted values down and increasing the chances of being benign traffic. The values shown on the right side of the x-axis have a positive impact, increasing the predicted values and



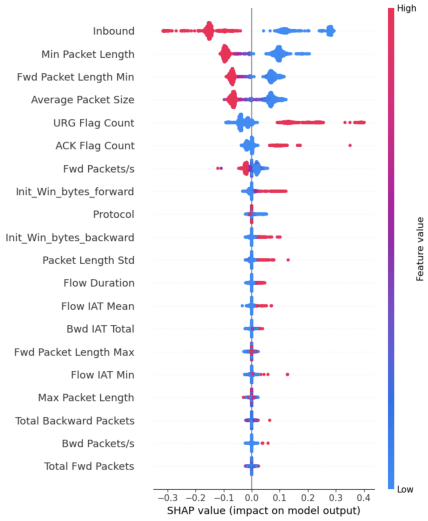
(a) Mean plot with all features



(b) Summary plot with all features



(c) Mean plot with selected features



(d) Summary plot with selected features

Figure 5.2: One-to-one (Benign vs. DNS): the explanation of SHAP values

bringing them closer to the malicious traffic. In addition, the left vertical axis represents the feature names, ranked in descending order of all feature importance, while the right vertical axis represents the original value as it appears in the dataset and is colored in dots to represent high (red) or low (blue). Note that the horizontal values represent the SHAP values of the predictions that are associated with high or low predictions, while the vertical axis above point zero (0.0) represents no impact on the prediction results. For example, a SHAP value of zero (0.0) represents no impact on the prediction results,

or close to zero represents low-quality predictions, while high-quality predictions, where SHAP values are far away from zero, also indicate either positive or negative correlations.

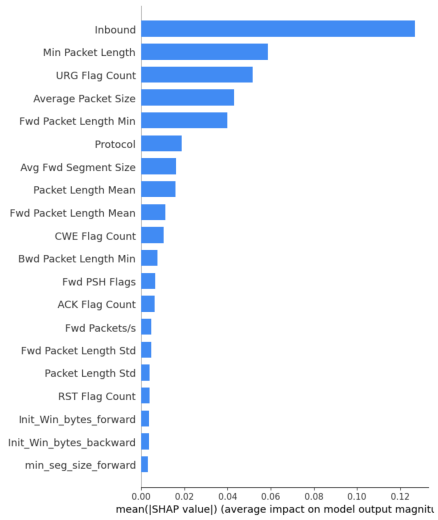
One-to-one classification explanation: Fig. 5.2a and 5.2c show the mean SHAP value of all traffic flows, which represents the importance of features in decreasing order from high to low. The largest mean SHAP value had the highest importance among the others, which is a significant predictive feature for the prediction results of the proposed model. For example, the *Inbount* feature is the most globally important feature in classifying DDoS traffic when using all features or the top 20 selected features. Although the *URG Flag Count* is the fifth most important feature in the classification of the prediction results when using the top 20 selected features (Fig. 5.2c), it is the third most important feature in classifying benign and malicious traffic when using all features (Fig. 5.2a). However, it is less important with the SHAP value of 0.05 in Fig. 5.2a than with the SHAP value of 0.07 in Fig. 5.2c. In addition, the feature *Flow Duration* is the least important feature in Fig. 5.2a. However, in Fig. 5.2c, *Flow Duration* is more important than among other features that fall under the *Flow Duration*. Finally, In Fig. 5.2a and Fig. 5.2c, the value of the feature *Inbound* is more important than about twice of the feature *Min Packet Length* on the prediction result. Overall, the importance of each feature has a smooth decreasing order in Fig. 5.2a, while the influence of feature importance decreases sharply from the sixth feature in Fig. 5.2c.

The *Inbound* feature has the largest impact on both using all features and the selected top 20 feature classifications in Fig. 5.2b and 5.2d. Higher values (red) have a negative impact on the prediction results, pushing them toward benign, while lower values (blue) have a positive impact on the prediction results, pushing them toward malicious. The *Min Packet Length*, *Average Packet Size*, *Fwd Packets/s* and *Fwd Packet Length Min* features have a similar impact tendency as *Inbound*, but these features have a different contribution impact order in Fig. 5.2b and 5.2d with or without feature selection.

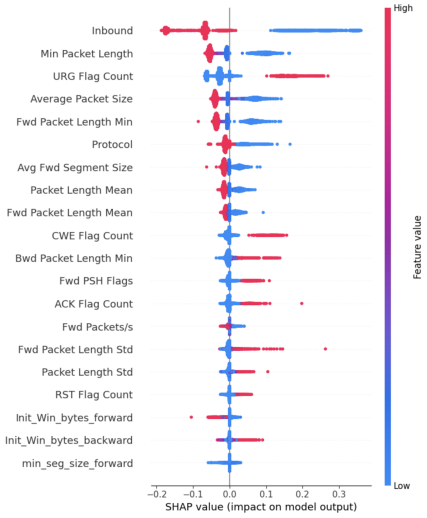
The higher SHAP values of the *URG Flag Count* feature have a higher positive impact on the prediction results, pushing it towards malicious, while the lower SHAP values have a negative impact, pushing them towards benign, as shown in both Fig. 5.2b and Fig. 5.2d. Furthermore, in the feature selection scenario, the *URG Flag Count* feature has a smaller impact than the use of all features.

In Fig. 5.2b, the smallest contribution of *Flow Duration* has the least impact on the classification prediction results because the SHAP values are closer to zero and have a less significant effect. However, the least impact feature *Total Fwd Packets* shown in Fig. 5.2d has no impact or less impact than most of the top 20 features in Fig. 5.2b because this feature is outside the range of the largest contribution of the top 20 features (this feature only shown in Fig. 5.2d, and not shown in figure Fig. 5.2b).

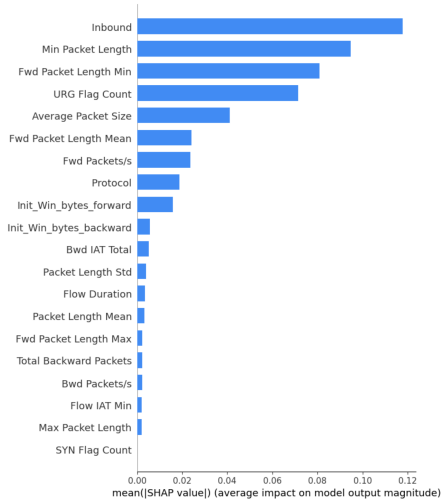
One-to-all classification explanation: Fig. 5.3 shows the mean SHAP value plot and the summary plot of the traffic flows in the one-to-all scenario, which also represents



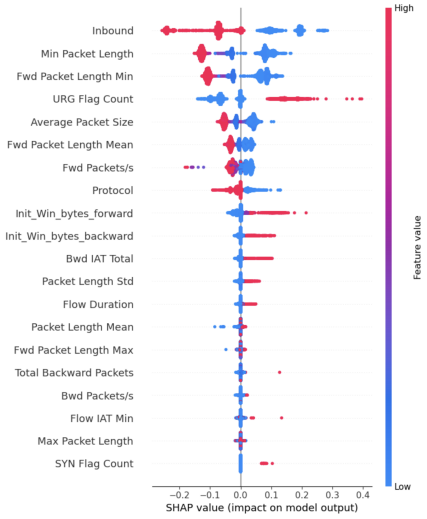
(a) Mean plot with all features



(b) Summary plot with all features



(c) Mean plot with selected features



(d) Summary plot with selected features

Figure 5.3: One-to-all (Benign vs. Four Malicious): the explanation of SHAP values

the feature importance and the most contributing features separately in decreasing order from high to low.

Fig. 5.3a and Fig. 5.3c show the global feature importance in bar plots, representing the importance of the features in decreasing order from high to low by using all features and selected features respectively. A large mean SHAP value on the x-axis illustrates the most predictive features for the classification. The features *Inbound*, *Min Packet Length* are the top 2 important predictive features for the classification predictions in

both Fig. 5.3a and Fig. 5.3c. The features *Inbound* is the largest important feature, and the mean SHAP value is greater than 0.12. However, the importance of the feature *Min Packet Length* with all features plotted is not as significant importance as with the selected features. This is because the mean SHAP value in the plot of all features is 0.07, compared to 0.09 in the plot of the selected feature. Similarly, the importance of each feature has a smooth decreasing order from the feature *Min Packet Length* in Fig. 5.3a, while the influence of feature importance decreases sharply from the fifth feature in Fig. 5.3c. The least important feature is *Min_seg_size_forward* in Fig. 5.3a, while the *SYN Flag Count* feature is the least important feature in Fig. 5.3c.

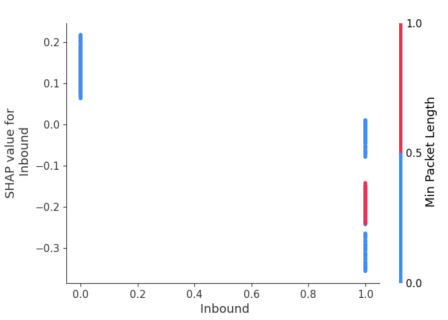
The summary plot of the one-to-all scenario shows in Fig. 5.3b and 5.3d. The top 5 most contributing features (except feature *URG Flag Count*) in Fig. 5.3b and 5.3d have a greater impact on the prediction results. Higher values have a negative impact on the prediction results, pushing them toward benign, while lower values have a positive impact on the prediction results, pushing them toward malicious. In contrast, the feature with a higher impact on the prediction result is *URG Flag Count*. The higher values have a positive impact, pushing it toward malicious, while lower values have a negative impact, pushing it toward benign.

The higher value of the *Packet Length Mean* feature has a negative impact in Fig. 5.3b, whereas it has a small positive impact in Fig. 5.3d. The higher values of the *Init_win_bytes_forward* and *Init_win_bytes_backward* have a higher positive impact on the prediction results in Fig. 5.3d, pushing them towards to malicious on the prediction decision. However, the feature *Init_win_bytes_forward* in Fig. 5.3b have a negative impact on the prediction results, pushing them towards benign, while *Init_win_bytes_backward* have a positive impact, and push them towards malicious. Overall, those two features contribute more impact on the prediction results in selected features shown in Fig.5.3d, compared to the contribution in Fig.5.3b.

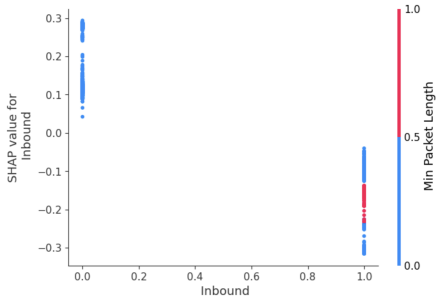
Dependence plot

The dependence plot is an important plot that can provide more information about the relationship between the feature values and the corresponding SHAP values for each traffic flow. A SHAP dependence plot is a scatter plot that illustrates the impact of a single feature on the predictions made by the proposed model. Each point represents a single prediction from the entire dataset. The horizontal axis represents the value of the features, while the left vertical axis illustrates the SHAP value of the corresponding feature, representing how much of the features impact the prediction results. The right vertical axis represents the effect of the interaction with the horizontal plot feature, highlighted in red color (Note that this interaction feature is chosen automatically).

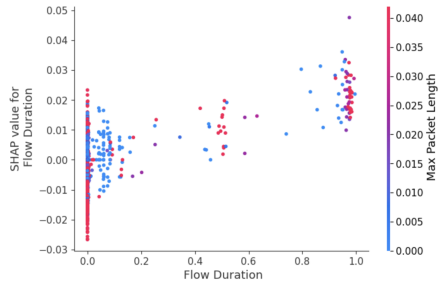
Fig. 5.4 shows the most and least important features before and after feature selection in a one-to-one scenario. The SHAP dependence plot with and without feature selection for the top 1 importance feature of *Inbound* are shown in Fig. 5.4a and 5.4b. In these two figures, the negative value of SHAP values (e.g. -0.1) with the *Inbound* value (0)



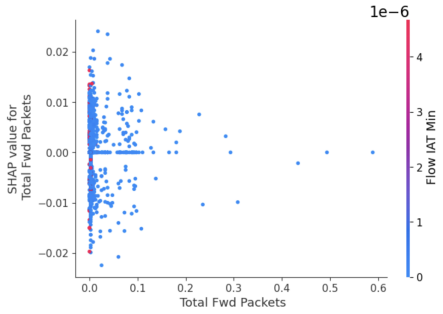
(a) All features: top 1 important feature



(b) Selected features: top 1 important feature



(c) All features: bottom 1 feature

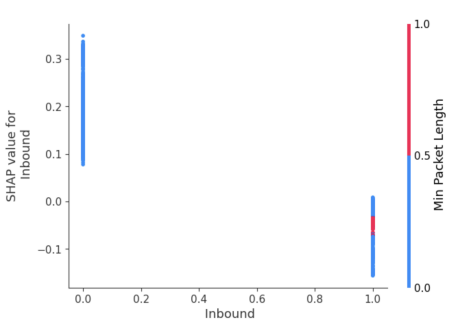


(d) Selected features: bottom 1 feature

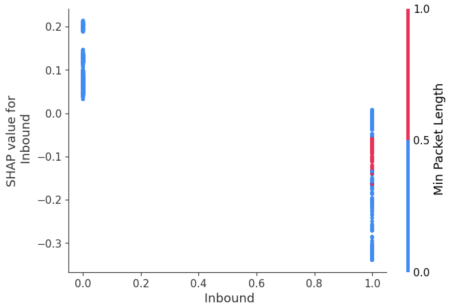
Figure 5.4: Global explanation: One-to-one (Benign vs. DNS) dependence plot

has a great impact on the malicious. Meanwhile, there are a large number of positive SHAP values (e.g. 0.1) localized with the *Inbound*feature value (1), as shown in Fig. 5.4a and 5.4b, which has a large impact on the benign traffic. In contrast to the top one feature impact of the prediction results, Fig. 5.4c and 5.4d show the lowest botten (bottom 1) influence of the prediction results. The dots are distributed around the SHAP values of 0.0, which has a small influence on the prediction results.

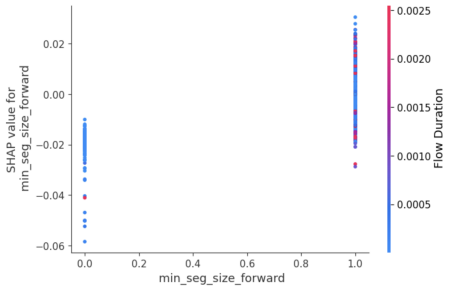
In a one-to-all scenario, the dependence plots of the top 1 important feature with and without feature selection are shown in Fig. 5.5a and 5.5b separately. The most contributing feature *Inbond* has both negative and positive SHAP values, corresponding to *Inbond* values of 1.0 and 0.0, which have influenced and pushed toward benign and malicious traffic respectively. Similarly, the interaction feature *Min Packet Length* takes the value 1 attached as the *Inbound* value of 1.0, which is likely to be benign traffic. In contrast to the top 1 contributing features, the bottom one impacted features shown in Fig. 5.5c and 5.5d. there is a large number of dots distributed closer to the SHAP value of 0.0, which has a smaller influence on the prediction results.



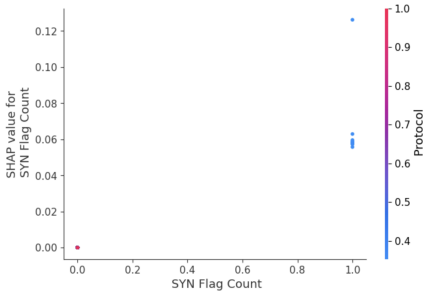
(a) All features: top 1 important feature



(b) Selected features: top 1 important feature



(c) All features: bottom 1 feature



(d) Selected features: bottom 1 feature

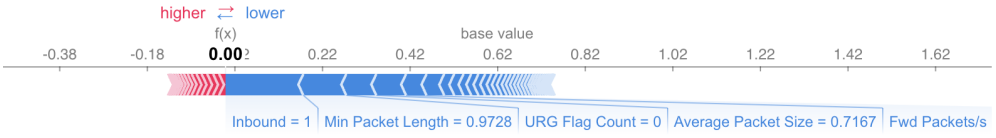
Figure 5.5: Global explanation: One-to-all (Benign vs. Four Malicious Types) dependence plot

5.5.2.2 Local Explanation

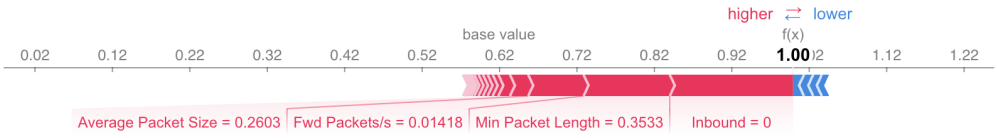
The local explanation can explain the local interpretability, provide an explanation for each individual sample and point out the contribution of features in explaining how a model makes decisions - based on Shapley values computed by SHAP, which can make the proposed model more convincing for the prediction results. Each feature value represents a force that either increases or decreases the prediction results.

Force Plot

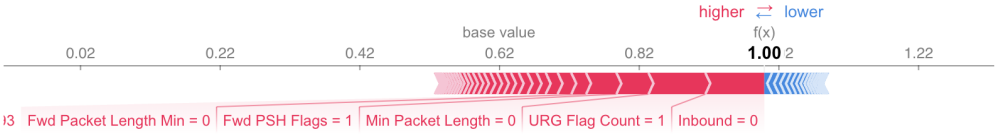
One of the most important plots in the local explanation is the force plot. The force plot allows us to visualize the contribution of the features to the classification prediction results for a particular traffic flow. In the force plots, the base value is the average of all the prediction result values, starting from the baseline (the boundary between red and blue). The red color represents features that push the prediction results toward higher, while the blue color represents features that push the prediction results toward lower. The length of the feature under the horizontal line determines the magnitude of the impact (e.g. the larger arrow block of the color, the greater impact of the feature on the prediction results).



(a) DNS: Predicted DNS



(b) DNS: Predicted Benign



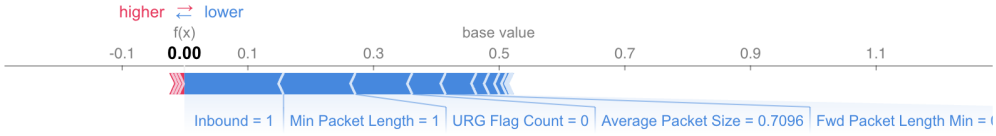
(c) Benign: Predicted Benign

(d) Benign: Predicted DNS

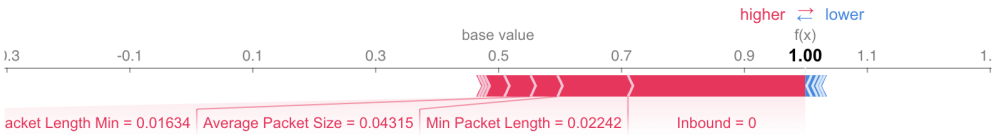
Figure 5.6: Local explanation: One-to-one (Benign vs. DNS) with all features

In this study, we use the SHAP explanation for interpreting the prediction of a specific traffic flow. The explanation can be divided into two scenarios to explain.

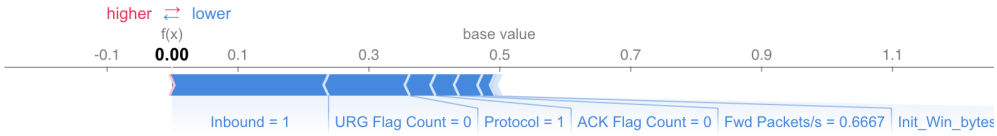
One-to-one scenario: Fig. 5.6 shows the force plot on the one-to-one scenario using all the features from the test set of the DDoS dataset, which can explain the contribution of each feature to the prediction results. The base value here is 0.62. Fig. 5.6a shows a malicious (DNS) sample, where the traffic is correctly predicted to be malicious (DNS), while Fig. 5.6c is a benign sample, which is correctly classified as benign. Furthermore, in Fig. 5.6a, the top 3 features (in order of their values) *Inbound* (1), *Min Packet Length* (0.9728), *URG Flag Count* (0) have the most negative impact (blue) on the prediction results, pushing it towards to malicious (0), while the top 3 features *Inbound* (0), *URG Flag Count* (1), *Min Packet Length* (0) have the most greater positive impact (red), pushing it towards benign (1). However, Fig. 5.6b and Fig. 5.6d show that the traffic flow cannot be classified correctly. For example, in Fig. 5.6b, the traffic flow is malicious, but it has been misclassified as benign, with the largest positive impact features (*Inbound*



(a) DNS: Predicted DNS



(b) Benign: Predicted Benign



(c) Benign: Predicted DNS

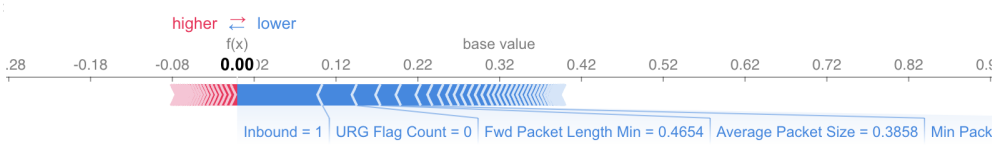
Figure 5.7: Local explanation: One-to-one (Benign vs. DNS) with feature selection

(0), *Min Packet Length* (0.3533), etc, pushing it towards benign (1). Similarly, a benign sample has been misclassified as malicious, because the features, including *Inbound* (1), *URG Flag Count* (0), etc have a negative impact and push it towards malicious(0).

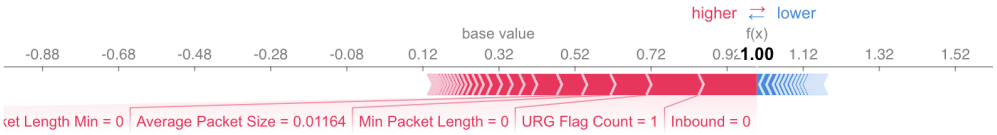
In the feature selection scenario, Fig. 5.7a and 5.7b show that the traffic flows are correctly classified as malicious (DNS) and benign traffic separately. In Fig. 5.7a, the most significant impact features are negative impacts (blue), including *Inbound* (1), *Min Packet Length* (1), etc., which increase the prediction result and push it towards malicious (0.00). Similarly, Fig. 5.7b shows the positive impact (red) of the features, including the features *Inbound* with the value of 0, *Min Packet Length* with the value of 0.02242, etc, which push it toward benign (1.00). Furthermore, Fig. 5.7c is one of the benign traffic misclassified as malicious with the positive impact features, including the feature *Inbound* with the value of 1, *URG Flag Count* with the value of 0, etc, which pushes it toward malicious (0.00). Note that all malicious traffic can be correctly classified as malicious.

One-to-all scenario: Fig. 5.8 shows force plots of a one-to-all scenario based on local explanation with all features, explaining benign and malicious traffic separately.

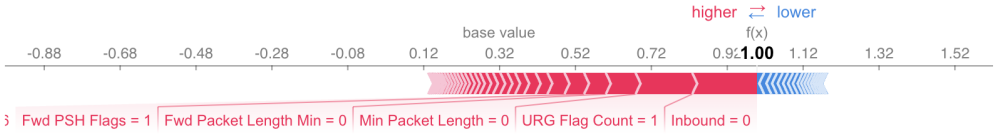
Fig. 5.8a and 5.8c show that the traffic flows are correctly classified as malicious and benign separately using all features. The negative impact is shown in blue color in Fig. 5.8a which pushes the prediction result towards malicious (0.00). The feature that contributes the most to increasing the prediction value is *Inbound* with a value of 1. The other higher contribution comes from the following features, including *URG Flag Count* (0),



(a) Malicious: Predicted Malicious



(b) Malicious: Predicted Benign



(c) Benign: Predicted Benign

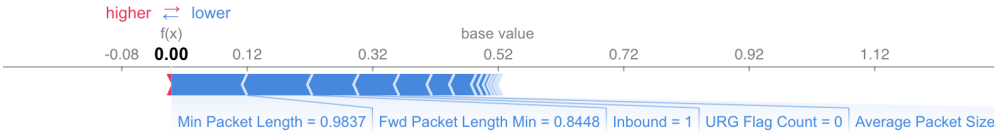
(d) Benign: Predicted Malicious

Figure 5.8: Local explanation: One-to-all (Benign vs. Four Malicious Types) with all fearues

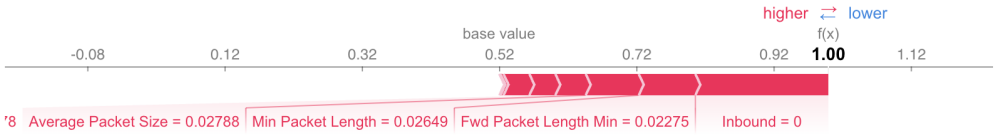
Fwd Packet Length Min (0.4654), etc. Similarly, the positive impact is shown in red color (Fig. 5.8c), which pushes the prediction towards benign (1.00) with the features *Inbound* (0), *URG Flag Count* (1) and *Min Packet Length* (0), etc. In contrast, Fig. 5.8b and 5.8d show that the traffic flows are misclassified. For example, the largest contribution of the *Inbound* in Fig. 5.8b with a value of 0 can cause this malicious traffic to be misclassified as benign. Fig. 5.8c shows a similar impact tendency (the largest impact feature *Inbound* with a value of 1) to cause the malicious traffic to be misclassified as benign.

Fig. 5.9a and 5.9b show that in the feature selection scenario, the traffic flows are correctly classified as malicious and benign separately. The features *Min Packet Length* (0.9837), *Fwd Packet Length Min* (0.8448), and *Inbound* (1), etc have a negative impact (blue), and push the prediction results toward malicious (0.00). Whereas, in 5.9b, the features *Inbound* (0), *Fwd Packet Length Min* (0.02275), and *Min Packet Length* (0.02649) have a greater positive impact (red) on the prediction results, pushing it toward benign (1.00). Furthermore, in Fig. 5.9c benign traffic is misclassified as malicious traffic because

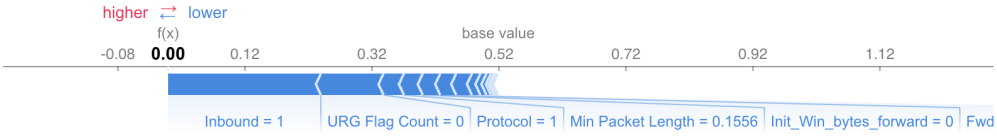
the larger contributing features have values that push it towards malicious (e.g. the features *Inbound* (1), and *URG Flag Count* (0), etc push it toward malicious).



(a) Malicious: Predicted Malicious



(b) Benign: Predicted Benign



(c) Benign: Predicted Malicious

Figure 5.9: Local explanation: One-to-all (Benign vs. Four Malicious Types) with feature selection

5.6 Conclusion

In this study, we propose a framework that can efficiently classify legitimate traffic and malicious traffic of DDoS attacks and use Kernel SHAP to provide an interpretation of the decision-making of the prediction results of the MLP classifier. In our proposed framework, we first select the top 20 important features based on three XGB-based feature importance techniques and create a subset of optimized features from the CICDDoS2019 dataset. The MLP classifier can be applied to classify legitimate traffic (benign) and malicious traffic by using the subset data of optimized features. We evaluated two binary classification scenarios: 1) one-to-one classification, which classifies legitimate traffic and a type of malicious traffic; and 2) one-to-all classification, where legitimate traffic and malicious traffic (including multiple types of DNS, LDAP, SNMP, and NetBIOS traffic labeled as malicious) of DDoS attacks are classified. The evaluation results show that the classification model achieves a high performance of over 99% accuracy in both two scenarios with the top 20 selected important features. To provide interpretability, we use SHAP values to explain the prediction results of the classification, which is important to

understand how the MLP classifier model classifies and identifies DDoS traffic. Furthermore, the explanation method aims to: 1) bring the feature contribution in decreasing order from high to low for the prediction results in the global explanation, while the local explanation provides explanations for a specific instance that the feature contribution in both legitimate traffic and malicious traffic separately, 2) present a visual explanation of the most influential features, 3) analyze the feature contribution in both legitimate traffic and malicious traffic separately, and 4) depicts a visualization of the influence of individual features on the prediction results.

This study provides an efficient model for binary classification and a visual explanation of the decision-making process of the classification model. In future work, we plan to provide an explanation for the multi-class classification [89].

Chapter 6

Conclusions and Future Directions

The research aimed to detect DDoS attacks using AI-based machine learning and deep learning techniques to perform classification and anomaly detection, which is used to identify and mitigate DDoS attacks on computer networks and online services. We first investigate DDoS classification detection approaches that can classify DDoS network traffic as either benign or different types of malicious. We then conduct research to detect unusual traffic patterns also known as malicious or anomalous traffic detection, such as one or more sudden increases in requests from a particular IP address or botnet. Lastly, we employ Explainable Artificial Intelligence (XIA) techniques to explain the decision-making behind the machine learning or deep learning classification models. In this thesis, we use a state-of-the-art CICDDoS2019 dataset in the detection of classification and anomaly detection.

In Chapter 2, we propose a novel approach namely AE-MLP, which combines two deep learning approaches AE and MLP for effective detection and classification of DDoS attacks. AE provides an effective feature extraction that finds the most relevant features, while MLP classifier provides better classification based on the optimal features - the compressed and reduced features produced by AE to detect different types of DDoS attacks. Our proposed model can better extract the relevant features that improve the classification performance. Our proposed model, comprehensively and extensively tested against many subsets of large DDoS attack samples, demonstrates high performance against many performance metrics.

In Chapter 3, we propose a hybrid time-series anomaly detection architecture that leverages reconstruction-based LSTM-AE for efficient detection of DDoS anomaly attacks. The LSTM networks are comprised of multiple LSTM units to learn the long short-term correlation of DDoS traffic within a time-series sequence. The AE technique is applied to identify the optimal anomaly threshold based on the reconstruction error rates evaluated across all time-series sequences. Our proposed combination model LSTM-AE can not only learn delicate sub-pattern differences in attacks and benign traffic flows but also minimize reconstructed benign traffic to obtain a lower range reconstruction error, while attacks

present a larger reconstruction error. The evaluation results demonstrate high levels of performance on different time window lengths over many performance metrics.

In Chapter 4, to explore whether the proposed hybrid deep learning-based anomaly detection is more promising, we apply the aforementioned hybrid reconstruction-based LSTM-AE anomaly detection technique based on time-series sequence analysis, to the real-world IoT sensor data - SKOMOBO dataset deployed in the classroom in New Zealand for indoor air quality detection. Our experimental results performed based on the comprehensive set of evaluation criteria, demonstrate that our proposed model can effectively detect anomalies, reaching a high detection rate. We demonstrate the proposed hybrid deep learning-based techniques that can effectively detect anomalies in the large-scale IoT dataset.

In Chapter 5, we introduced the XAI-based SHAP explanation framework to explain the model's classification decision. This framework consists of model classification and explanation. We conduct the classification evaluation with and without feature selection using the MLP classifier. Among the feature selection, The combination of XGB-based feature importance is applied to select the most important features for the classification. For model explanation, we employed the SHAP approach to explain model decisions based on global and local explanations. The explanation aims to: 1) bring the feature contribution in decreasing order from high to low for the prediction results in the global explanation, while the local explanation provides explanations for a specific instance that the feature contribution in both benign traffic and malicious traffic separately, 2)present a visual explanation of the most influential features, 3) analyze the feature contribution in both benign and malicious traffic separately, and 4) depicts a visualization of the influence of individual features on the prediction results. Finally, the evaluation results show that using the feature selected by the XAI feature importance technique, the classification results show higher performance than using all features. This work helps the users to trust and have a better understanding of the prediction results of the proposed model.

Currently, this research focuses on multivariate time series anomaly detection in three types of attacks and explained in a binary classification model. Furthermore, the SHAP explanation is computationally intensive and slow to run. Looking ahead, the current research can be extended to the future directions:

- Apply the proposed classification and anomaly detection models to additional cybersecurity datasets for the detection of unseen or unknown attacks.
- Explore the extension of SHAP to multivariate time-series classification for the purpose of detecting malicious attacks.
- Investigate an extension where combinations of SHAP not only reduce computational time but also provide valuable insights into the detection and explanation of malicious attacks, enhancing robustness against DDoS attacks.

References

[1] What is a ddos attack?, 2023. Available online at: `https://aws.amazon.com/shield/ddos-attack-protection`.

[2] Abdallah, M., An Le Khac, N., Jahromi, H., and Delia Jurcut, A. A hybrid cnn-lstm based approach for anomaly detection systems in sdns. In *The 16th International Conference on Availability, Reliability and Security* (2021), pp. 1–7.

[3] Abou El Houda, Z., Brik, B., and Khoukhi, L. "why should i trust your ids?": An explainable deep learning framework for intrusion detection systems in internet of things networks. *IEEE Open Journal of the Communications Society 3* (2022), 1164–1176.

[4] Akgun, D., Hizal, S., and Cavusoglu, U. A new ddos attacks intrusion detection model based on deep learning for cybersecurity. *Computers & Security 118* (2022), 102748.

[5] Alamri, H. A., and Thayananthan, V. Bandwidth control mechanism and extreme gradient boosting algorithm for protecting software-defined networks against ddos attacks. *IEEE Access 8* (2020), 194269–194288.

[6] Alethea Toh, Anupam Vij, P. P. M., and Pasha, S. Azure ddos protection—2020 year in review, 2021.

[7] Alghazzawi, D., Bamasag, O., Ullah, H., and Asghar, M. Z. Efficient detection of ddos attacks using a hybrid deep learning model with improved feature selection. *Applied Sciences 11*, 24 (2021), 11634.

[8] Almiani, M., AbuGhazleh, A., Jararweh, Y., and Razaque, A. Ddos detection in 5g-enabled iot networks using deep kalman backpropagation neural network. *International Journal of Machine Learning and Cybernetics* (2021), 1–13.

[9] Alzahrani, S., and Hong, L. Detection of distributed denial of service (ddos) attacks using artificial intelligence on cloud. In *2018 IEEE World Congress on Services (SERVICES)* (2018), IEEE, pp. 35–36.

[10] Alzaylaee, M. K., Yerima, S. Y., and Sezer, S. Dl-droid: Deep learning based android malware detection using real devices. *Computers & Security 89* (2020), 101663.

[11] Antwarg, L., Miller, R. M., Shapira, B., and Rokach, L. Explaining anomalies detected by autoencoders using shapley additive explanations. *Expert Systems with Applications 186* (2021), 115736.

[12] Aydın, H., Orman, Z., and Aydın, M. A. A long short-term memory (lstm)-based distributed denial of service (ddos) detection and defense system design in public cloud network environment. *Computers & Security 118* (2022), 102725.

[13] Batchu, R. K., and Seetha, H. An integrated approach explaining the detection of distributed denial of service attacks. *Computer Networks 216* (2022), 109269.

[14] Can, D. C., Le, H. Q., and Ha, Q. T. Detection of distributed denial of service attacks using automatic feature selection with enhancement for imbalance dataset. *ACIIDS 2021* (2021).

[15] Chen, Y., Hou, J., Li, Q., and Long, H. Ddos attack detection based on random forest. In *2020 IEEE International Conference on Progress in Informatics and Computing (PIC)* (2020), IEEE, pp. 328–334.

[16] Chen, Z., Jiang, F., Cheng, Y., Gu, X., Liu, W., and Peng, J. Xgboost classifier for ddos attack detection and analysis in sdn-based cloud. In *2018 IEEE international conference on big data and smart computing (bigcomp)* (2018), IEEE, pp. 251–256.

[17] Cisco, U. Cisco annual internet report (2018–2023) white paper. *Cisco: San Jose, CA, USA* (2020).

[18] De Assis, M. V., Carvalho, L. F., Rodrigues, J. J., Lloret, J., and Proença Jr, M. L. Near real-time security system applied to sdn environments in iot networks using convolutional neural network. *Computers & Electrical Engineering 86* (2020), 106738.

[19] Dong, S., and Sarem, M. Ddos attack detection method based on improved knn with the degree of ddos attack in software-defined networks. *IEEE Access 8* (2019), 5039–5048.

[20] Doriguzzi-Corin, R., Millar, S., Scott-Hayward, S., Martinez-del Rincon, J., and Siracusa, D. Lucid: A practical, lightweight deep learning solution for ddos attack detection. *IEEE Transactions on Network and Service Management 17*, 2 (2020), 876–889.

[21] Elsayed, M. S., Le-Khac, N.-A., Dev, S., and Jurcut, A. D. Ddosnet: A deep-learning model for detecting network attacks. In *2020 IEEE 21st International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)* (2020), IEEE, pp. 391–396.

[22] Ergen, T., and Kozat, S. S. Unsupervised anomaly detection with lstm neural networks. *IEEE transactions on neural networks and learning systems 31*, 8 (2019), 3127–3141.

[23] Fouladi, R. F., Ermiş, O., and Anarim, E. Anomaly-based ddos attack detection by using sparse coding and frequency domain. In *2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)* (2019), IEEE, pp. 1–6.

[24] Gebreyesus, Y., Dalton, D., Nixon, S., De Chiara, D., and Chinnici, M. Machine learning for data center optimizations: Feature selection using shapley additive explanation (shap). *Future Internet 15*, 3 (2023), 88.

[25] Gniewkowski, M., Maciejewski, H., and Surmacz, T. Anomaly detection techniques for different ddos attack types. In *International Conference on Dependability and Complex Systems* (2022), Springer, pp. 63–78.

[26] Gohil, M., and Kumar, S. Evaluation of classification algorithms for distributed denial of service attack detection. In *2020 IEEE Third International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)* (2020), IEEE, pp. 138–141.

[27] Haider, S., Akhunzada, A., Mustafa, I., Patel, T. B., Fernandez, A., Choo, K.-K. R., and Iqbal, J. A deep cnn ensemble framework for efficient ddos attack detection in software defined networks. *Ieee Access 8* (2020), 53972–53983.

[28] Hossain, E., Shariff, M. A. U., Hossain, M. S., and Andersson, K. A novel deep learning approach to predict air quality index. In *Proceedings of International Conference on Trends in Computational and Cognitive Engineering* (2021), Springer, pp. 367–381.

[29] Idhammad, M., Afdel, K., and Belouch, M. Semi-supervised machine learning approach for ddos detection. *Applied Intelligence 48* (2018), 3193–3208.

[30] Javaid, A., Niyaz, Q., Sun, W., and Alam, M. A deep learning approach for network intrusion detection system. *EAI Endorsed Transactions on Security and Safety 3*, 9 (5 2016).

[31] Jia, Y., Zhong, F., Alrawais, A., Gong, B., and Cheng, X. Flowguard: an intelligent edge defense mechanism against iot ddos attacks. *IEEE Internet of Things Journal 7*, 10 (2020), 9552–9562.

[32] Jung, Y., Kang, T., and Chun, C. Anomaly analysis on indoor office spaces for facility management using deep learning methods. *Journal of Building Engineering 43* (2021), 103139.

[33] Kalutharage, C. S., Liu, X., Chrysoulas, C., Pitropakis, N., and Papadopoulos, P. Explainable ai-based ddos attack identification method for iot networks. *Computers 12*, 2 (2023), 32.

[34] Khare, M., and Oak, R. Real-time distributed denial-of-service (ddos) attack detection using decision trees for server performance maintenance. *Performance Management of Integrated Systems and its Applications in Software Engineering* (2020), 1–9.

[35] Khoei, T. T., Aissou, G., Hu, W. C., and Kaabouch, N. Ensemble learning methods for anomaly intrusion detection system in smart grid. In *2021 IEEE International Conference on Electro Information Technology (EIT)* (2021), IEEE, pp. 129–135.

[36] Kim, J.-Y., Chu, C.-H., and Shin, S.-M. Issaq: An integrated sensing systems for real-time indoor air quality monitoring. *IEEE Sensors Journal 14*, 12 (2014), 4230–4244.

[37] Koay, A., Chen, A., Welch, I., and Seah, W. K. A new multi-classifier system using entropy-based features in ddos attack detection. In *2018 International conference on information networking (ICOIN)* (2018), IEEE, pp. 162–167.

[38] Kousar, H., Mulla, M. M., Shettar, P., and Narayan, D. Detection of ddos attacks in software defined network using decision tree. In *2021 10th IEEE International Conference on Communication Systems and Network Technologies (CSNT)* (2021), IEEE, pp. 783–788.

[39] Kunang, Y. N., Nurmaini, S., Stiawan, D., Zarkasi, A., et al. Automatic features extraction using autoencoder in intrusion detection system. In *2018 International Conference on Electrical Engineering and Computer Science (ICECOS)* (2018), IEEE, pp. 219–224.

[40] Li, J., Izakian, H., Pedrycz, W., and Jamal, I. Clustering-based anomaly detection in multivariate time series data. *Applied Soft Computing 100* (2021), 106919.

[41] Liao, N., and Li, X. Traffic anomaly detection model using k-means and active learning method. *International Journal of Fuzzy Systems* (2022), 1–19.

[42] Liu, H., and Lang, B. Machine learning and deep learning methods for intrusion detection systems: A survey. *applied sciences 9*, 20 (2019), 4396.

[43] Liu, J., Lai, Y., and Zhang, S. Fl-guard: A detection and defense system for ddos attack in sdn. In *Proceedings of the 2017 international conference on cryptography, security and privacy* (2017), pp. 107–111.

[44] Liu, Y., Pang, Z., Karlsson, M., and Gong, S. Anomaly detection based on machine learning in iot-based vertical plant wall for indoor climate control. *Building and Environment 183* (2020), 107212.

[45] Lundberg, S. M., and Lee, S.-I. A unified approach to interpreting model predictions. *Advances in neural information processing systems 30* (2017).

[46] Maseer, Z. K., Yusof, R., Bahaman, N., Mostafa, S. A., and Foozy, C. F. M. Benchmarking of machine learning for anomaly based intrusion detection systems in the cicids2017 dataset. *IEEE Access 9* (2021), 22351–22370.

[47] Meng, T., Jing, X., Yan, Z., and Pedrycz, W. A survey on machine learning for data fusion. *Information Fusion 57* (2020), 115–129.

[48] Microsoft. Microsoft digital defense report, 2021.

[49] Mirkovic, J., Prier, G., and Reiher, P. Attacking ddos at the source. In *10th IEEE International Conference on Network Protocols, 2002. Proceedings.* (2002), IEEE, pp. 312–321.

[50] Mirkovic, J., and Reiher, P. A taxonomy of ddos attack and ddos defense mechanisms. *ACM SIGCOMM Computer Communication Review 34*, 2 (2004), 39–53.

[51] Motlagh, N. H., Zaidan, M. A., Lagerspetz, E., Varjonen, S., Toivonen, J., Mineraud, J., Rebeiro-Hargrave, A., Siekkinen, M., Hussein, T., Nurmi, P., et al. Indoor air quality monitoring using infrastructure-based motion detectors. In *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)* (2019), vol. 1, IEEE, pp. 902–907.

[52] Mumtaz, R., Zaidi, S. M. H., Shakir, M. Z., Shafi, U., Malik, M. M., Haque, A., Mumtaz, S., and Zaidi, S. A. R. Internet of things (iot) based indoor air quality sensing and predictive analytic—a covid-19 perspective. *Electronics 10*, 2 (2021), 184.

[53] Nezhad, S. M. T., Nazari, M., and Gharavol, E. A. A novel dos and ddos attacks detection algorithm using arima time series model and chaotic system in computer networks. *IEEE Communications Letters 20*, 4 (2016), 700–703.

[54] Nguyen, H., Tran, K. P., Thomassey, S., and Hamad, M. Forecasting and anomaly detection approaches using lstm and lstm autoencoder techniques with the applications in supply chain management. *International Journal of Information Management 57* (2021), 102282.

[55] Nguyen, T. T., and Reddi, V. J. Deep reinforcement learning for cyber security. *arXiv preprint arXiv:1906.05799* (2019).

[56] Novaes, M. P., Carvalho, L. F., Lloret, J., and Proenca, M. L. Long short-term memory and fuzzy logic for anomaly detection and mitigation in software-defined network environment. *IEEE Access 8* (2020), 83765–83781.

[57] Novaes, M. P., Carvalho, L. F., Lloret, J., and Proença Jr, M. L. Adversarial deep learning approach detection and defense against ddos attacks in sdn environments. *Future Generation Computer Systems 125* (2021), 156–167.

[58] Ottosen, T.-B., and Kumar, P. Outlier detection and gap filling methodologies for low-cost air quality measurements. *Environmental Science: Processes & Impacts 21*, 4 (2019), 701–713.

[59] Parfenov, D., Kuznetsova, L., Yanishevskaya, N., Bolodurina, I., Zhigalov, A., and Legashev, L. Research application of ensemble machine learning methods to the problem of multiclass classification of ddos attacks identification. In *2020 International Conference Engineering and Telecommunication (En&T)* (2020), IEEE, pp. 1–7.

[60] Pontes, C., Souza, M., Gondim, J., Bishop, M., and Marotta, M. A new method for flow-based network intrusion detection using the inverse potts model. *IEEE Transactions on Network and Service Management* (2021).

[61] Rajagopal, S., Kundapur, P. P., and Hareesha, K. Towards effective network intrusion detection: from concept to creation on azure cloud. *IEEE Access 9* (2021), 19723–19742.

[62] Rastogi, K., and Lohani, D. An internet of things framework to forecast indoor air quality using machine learning. In *Symposium on Machine Learning and Metaheuristics Algorithms, and Applications* (2019), Springer, pp. 90–104.

[63] Sadaf, K., and Sultana, J. Intrusion detection based on autoencoder and isolation forest in fog computing. *IEEE Access 8* (2020), 167059–167068.

[64] Sahoo, K. S., Panda, S. K., Sahoo, S., Sahoo, B., and Dash, R. Toward secure software-defined networks against distributed denial of service attack. *The Journal of Supercomputing 75*, 8 (2019), 4829–4874.

[65] Said Elsayed, M., Le-Khac, N.-A., Dev, S., and Jurcut, A. D. Network anomaly detection using lstm based autoencoder. In *Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks* (2020), pp. 37–45.

[66] Saied, A., Overill, R. E., and Radzik, T. Detection of known and unknown ddos attacks using artificial neural networks. *Neurocomputing 172* (2016), 385–393.

[67] Salahuddin, M. A., Bari, M. F., Alameddine, H. A., Pourahmadi, V., and Boutaba, R. Time-based anomaly detection using autoencoder. In *2020 16th International Conference on Network and Service Management (CNSM)* (2020), IEEE, pp. 1–9.

[68] Salahuddin, M. A., Pourahmadi, V., Alameddine, H. A., Bari, M. F., and Boutaba, R. Chronos: Ddos attack detection using time-based autoencoder. *IEEE Transactions on Network and Service Management* (2021).

[69] Samom, P. S., and Taggu, A. Distributed denial of service (ddos) attacks detection: A machine learning approach. In *Applied Soft Computing and Communication Networks*. Springer, 2021, pp. 75–87.

[70] Sanchez, O. R., Repello, M., Carrega, A., and Bolla, R. Evaluating ml-based ddos detection with grid search hyperparameter optimization. In *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)* (2021), IEEE, pp. 402–408.

[71] Šarčević, A., Pintar, D., Vranić, M., and Krajna, A. Cybersecurity knowledge extraction using xai. *Applied Sciences 12*, 17 (2022), 8669.

[72] Sayed, M. I., Sayem, I. M., Saha, S., and Haque, A. A multi-classifier for ddos attacks using stacking ensemble deep neural network. In *2022 International Wireless Communications and Mobile Computing (IWCMC)* (2022), IEEE, pp. 1125–1130.

[73] Shapley, L. S. 17. a value for n-person games. In *Contributions to the Theory of Games (AM-28), Volume II*. Princeton University Press, 2016, pp. 307–318.

[74] Sharafaldin, I., Lashkari, A. H., Hakak, S., and Ghorbani, A. A. Developing realistic distributed denial of service (ddos) attack dataset and taxonomy. In *2019 International Carnahan Conference on Security Technology (ICCST)* (2019), IEEE, pp. 1–8.

[75] Sharma, D. K., Mishra, J., Singh, A., Govil, R., Srivastava, G., and Lin, J. C.-W. Explainable artificial intelligence for cybersecurity. *Computers and Electrical Engineering 103* (2022), 108356.

[76] Sharma, P. K., Mondal, A., Jaiswal, S., Saha, M., Nandi, S., De, T., and Saha, S. Indoairsense: A framework for indoor air quality estimation and forecasting. *Atmospheric Pollution Research 12*, 1 (2021), 10–22.

[77] Shieh, C.-S., Lin, W.-W., Nguyen, T.-T., Chen, C.-H., Horng, M.-F., and Miu, D. Detection of unknown ddos attacks with deep learning and gaussian mixture model. *Applied Sciences 11*, 11 (2021), 5213.

[78] Singh Samom, P., and Taggu, A. Distributed denial of service (ddos) attacks detection: A machine learning approach. In *Applied Soft Computing and Communication Networks: Proceedings of ACN 2020* (2021), Springer, pp. 75–87.

[79] Sriram, S., Vinayakumar, R., Alazab, M., and Soman, K. Network flow based iot botnet attack detection using deep learning. In *IEEE INFOCOM 2020-IEEE conference on computer communications workshops (INFOCOM WKSHPS)* (2020), IEEE, pp. 189–194.

[80] Tabassum, S., Parvin, N., Hossain, N., Tasnim, A., Rahman, R., and Hossain, M. I. Iot network attack detection using xai and reliability analysis. In *2022 25th International Conference on Computer and Information Technology (ICCIT)* (2022), IEEE, pp. 176–181.

[81] Ullah, I., and Mahmoud, Q. H. A technique for generating a botnet dataset for anomalous activity detection in iot networks. In *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* (2020), IEEE, pp. 134–140.

[82] Ur Rehman, S., Khaliq, M., Imtiaz, S. I., Rasool, A., Shafiq, M., Javed, A. R., Jalil, Z., and Bashir, A. K. Diddos: An approach for detection and identification of distributed denial of service (ddos) cyberattacks using gated recurrent units (gru). *Future Generation Computer Systems 118* (2021), 453–466.

[83] Varghese, J. E., and Muniyal, B. An efficient ids framework for ddos attacks in sdn environment. *IEEE Access 9* (2021), 69680–69699.

[84] Vinayakumar, R., Alazab, M., Soman, K., Poornachandran, P., Al-Nemrat, A., and Venkatraman, S. Deep learning approach for intelligent intrusion detection system. *Ieee Access 7* (2019), 41525–41550.

[85] Wang, M., Lu, Y., and Qin, J. A dynamic mlp-based ddos attack detection method using feature selection and feedback. *Computers & Security 88* (2020), 101645.

[86] Wang, Y., Boulic, M., Phipps, R., Chitty, C., Moses, A., Weyers, R., Jang-Jaccard, J., Olivares, G., Ponder-Sutton, A., and Cunningham, C. Integrating open-source technologies to build a school indoor air quality monitoring box (skomobo). In *2017 4th Asia-Pacific World Congress on Computer Science and Engineering (APWC on CSE)* (2017), IEEE, pp. 216–223.

[87] Wang, Y., Jang-Jaccard, J., Boulic, M., Phipps, R., Chitty, C., Weyers, R., Moses, A., Olivares, G., Ponder-Sutton, A., and Cunningham, C. Deployment issues for integrated open-source—based indoor air quality school monitoring box (skomobo). In *2018 IEEE Sensors Applications Symposium (SAS)* (2018), IEEE, pp. 1–4.

[88] Wei, Y., Jang-Jaccard, J., Sabrina, F., and Alavizadeh, H. Large-scale outlier detection for low-cost pm10 sensors. *IEEE Access 8* (2020), 229033–229042.

[89] Wei, Y., Jang-Jaccard, J., Sabrina, F., Singh, A., Xu, W., and Camtepe, S. Ae-mlp: A hybrid deep learning approach for ddos detection and classification. *IEEE Access 9* (2021), 146810–146821.

[90] Wei, Y., Jang-Jaccard, J., Xu, W., Sabrina, F., Camtepe, S., and Boulic, M. Lstm-autoencoder based anomaly detection for indoor air quality time series data. *arXiv preprint arXiv:2204.06701* (2022).

[91] Wei, Y., Jang-Jaccard, J., Xu, W., Sabrina, F., Camtepe, S., and Boulic, M. Lstm-autoencoder based anomaly detection for indoor air quality time series data. *IEEE Sensors Journal* (2023).

[92] Weyers, R., Jang-Jaccard, J., Moses, A., Wang, Y., Boulic, M., Chitty, C., Phipps, R., and Cunningham, C. Low-cost indoor air quality (iaq) platform for healthier classrooms in new zealand: Engineering issues. In *2017 4th Asia-Pacific World Congress on Computer Science and Engineering (APWC on CSE)* (2017), IEEE, pp. 208–215.

[93] Wu, D., Jiang, Z., Xie, X., Wei, X., Yu, W., and Li, R. Lstm learning with bayesian and gaussian processing for anomaly detection in industrial iot. *IEEE Transactions on Industrial Informatics 16*, 8 (2019), 5244–5253.

[94] Xu, C., Chen, H., Wang, J., Guo, Y., and Yuan, Y. Improving prediction performance for indoor temperature in public buildings based on a novel deep learning method. *Building and Environment 148* (2019), 128–135.

[95] Xu, W., Jang-Jaccard, J., Singh, A., Wei, Y., and Sabrina, F. Improving performance of autoencoder-based network anomaly detection on nsl-kdd dataset. *IEEE Access 9* (2021), 140136–140146.

[96] Yeom, S., Choi, C., and Kim, K. Lstm-based collaborative source-side ddos attack detection. *IEEE Access 10* (2022), 44033–44045.

[97] Yin, C., Zhang, S., Wang, J., and Xiong, N. N. Anomaly detection based on convolutional recurrent autoencoder for iot time series. *IEEE Transactions on Systems, Man, and Cybernetics: Systems 52*, 1 (2020), 112–122.

[98] Yin, C., Zhu, Y., Fei, J., and He, X. A deep learning approach for intrusion detection using recurrent neural networks. *Ieee Access 5* (2017), 21954–21961.

[99] Yuan, X., Li, C., and Li, X. Deepdefense: identifying ddos attack via deep learning. In *2017 IEEE International Conference on Smart Computing (SMARTCOMP)* (2017), IEEE, pp. 1–8.

[100] Zeng, Y., Chen, J., Jin, N., Jin, X., and Du, Y. Air quality forecasting with hybrid lstm and extended stationary wavelet transform. *Building and Environment* (2022), 108822.

[101] Zhou, X., Hu, Y., Liang, W., Ma, J., and Jin, Q. Variational lstm enhanced anomaly detection for industrial big data. *IEEE Transactions on Industrial Informatics 17*, 5 (2020), 3469–3477.

[102] Zhu, J., Jang-Jaccard, J., Liu, T., and Zhou, J. Joint spectral clustering based on optimal graph and feature selection. *Neural Processing Letters 53*, 1 (2021), 257–273.

[103] Zikria, Y. B., Ali, R., Afzal, M. K., and Kim, S. W. Next-generation internet of things (iot): Opportunities, challenges, and solutions. *Sensors 21*, 4 (2021).

MASSEY UNIVERSITY
TE KUNENGA KI PŪREHUROA
UNIVERSITY OF NEW ZEALAND

GRADUATE RESEARCH SCHOOL

STATEMENT OF CONTRIBUTION
DOCTORATE WITH PUBLICATIONS/MANUSCRIPTS

We, the student and the student's main supervisor, certify that all co-authors have consented to their work being included in the thesis and they have accepted the student's contribution as indicated below in the Statement of Originality.

Student name:	Yuanyuan Wei
Name and title of main supervisor:	Associate Professor Julian Jang-Jaccard
In which chapter is the manuscript/published work?	Chapter 2
What percentage of the manuscript/published work was contributed by the student?	70

Describe the contribution that the student has made to the manuscript/published work:

Study design, methodology, implementation of results, preparation of the original draft, visualization , response to reviewers' comments

Please select one of the following three options:

(●) **The manuscript/published work is published or in press**

Please provide the full reference of the research output:

Wei, Y., Jang-Jaccard, J., Sabrina, F., Singh, A., Xu, W., & Camtepe, S. (2021). Ae-mlp: A hybrid deep learning approach for ddos detection and classification. IEEE Access, 9, 146810-146821.

( ) **The manuscript is currently under review for publication**

Please provide the name of the journal:

( ) **It is intended that the manuscript will be published, but it has not yet been submitted to a journal**

Student's signature:	yuanyuan wei Digitally signed by yuanyuan wei Date: 2023.08.14 12:28:19 +08'00'	Main supervisor's signature:	Julian Jang-Jaccard Digitally signed by Julian Jang-Jaccard Date: 2023.08.15 03:25:05 +12'00'

This form should be placed at the beginning of each relevant thesis chapter.

MASSEY UNIVERSITY
TE KUNENGA KI PŪREHUROA
UNIVERSITY OF NEW ZEALAND

GRADUATE RESEARCH SCHOOL

STATEMENT OF CONTRIBUTION
DOCTORATE WITH PUBLICATIONS/MANUSCRIPTS

We, the student and the student's main supervisor, certify that all co-authors have consented to their work being included in the thesis and they have accepted the student's contribution as indicated below in the Statement of Originality.

Student name:	Yuanyuan Wei
Name and title of main supervisor:	Associate Professor Julian Jang-Jaccard
In which chapter is the manuscript/published work?	Chapter 3
What percentage of the manuscript/published work was contributed by the student?	85

Describe the contribution that the student has made to the manuscript/published work:

Study design, methodology, implementation of results, visualization, writing manuscript

Please select one of the following three options:

- ○ **The manuscript/published work is published or in press**
  Please provide the full reference of the research output:
- ◉ **The manuscript is currently under review for publication**
  Please provide the name of the journal:
  IEEE Transactions on Network and Service Management
- ○ **It is intended that the manuscript will be published, but it has not yet been submitted to a journal**

Student's signature:	yuanyuan wei Digitally signed by yuanyuan wei Date: 2023.08.14 12:19:44 +08'00'	Main supervisor's signature:	Julian Jang-Jaccard Digitally signed by Julian Jang-Jaccard Date: 2023.08.15 03:25:33 +12'00'

This form should be placed at the beginning of each relevant thesis chapter.

MASSEY UNIVERSITY
TE KUNENGA KI PŪREHUROA
UNIVERSITY OF NEW ZEALAND

GRADUATE RESEARCH SCHOOL

STATEMENT OF CONTRIBUTION
DOCTORATE WITH PUBLICATIONS/MANUSCRIPTS

We, the student and the student's main supervisor, certify that all co-authors have consented to their work being included in the thesis and they have accepted the student's contribution as indicated below in the Statement of Originality.

Student name:	Yuanyuan Wei
Name and title of main supervisor:	Associate Professor Julian Jang-Jaccard
In which chapter is the manuscript/published work?	Chapter 4
What percentage of the manuscript/published work was contributed by the student?	70

Describe the contribution that the student has made to the manuscript/published work:

Study design, methodology, implementation of results, preparation of the original draft, visualization, response to reviewers' comments

Please select one of the following three options:

- (●) **The manuscript/published work is published or in press**

  Please provide the full reference of the research output:

  Wei, Y., Jang-Jaccard, J., Xu, W., Sabrina, F., Camtepe, S., & Boulic, M. (2023). LSTM-autoencoder-based anomaly detection for indoor air quality time-series data. IEEE Sensors Journal, 23(4), 3787-3800.

- ( ) **The manuscript is currently under review for publication**

  Please provide the name of the journal:

- ( ) **It is intended that the manuscript will be published, but it has not yet been submitted to a journal**

Student's signature:	yuanyuan wei — Digitally signed by yuanyuan wei Date: 2023.08.14 12:46:53 +08'00'	Main supervisor's signature:	Julian Jang-Jaccard — Digitally signed by Julian Jang-Jaccard Date: 2023.08.15 03:25:49 +12'00'

This form should be placed at the beginning of each relevant thesis chapter.

MASSEY UNIVERSITY
TE KUNENGA KI PŪREHUROA
UNIVERSITY OF NEW ZEALAND

GRADUATE RESEARCH SCHOOL

STATEMENT OF CONTRIBUTION
DOCTORATE WITH PUBLICATIONS/MANUSCRIPTS

We, the student and the student's main supervisor, certify that all co-authors have consented to their work being included in the thesis and they have accepted the student's contribution as indicated below in the Statement of Originality.

Student name:	Yuanyuan Wei
Name and title of main supervisor:	Associate Professor Julian Jang-Jaccard
In which chapter is the manuscript/published work?	Chapter 5
What percentage of the manuscript/published work was contributed by the student?	90

Describe the contribution that the student has made to the manuscript/published work:

Study design, methodology, implementation of results, visualization, writing manuscript

Please select one of the following three options:

- ○ **The manuscript/published work is published or in press**
  Please provide the full reference of the research output:
- ◉ **The manuscript is currently under review for publication**
  Please provide the name of the journal:
  IEEE Transactions on Dependable and Secure Computing
- ○ **It is intended that the manuscript will be published, but it has not yet been submitted to a journal**

Student's signature:	yuanyuan wei Digitally signed by yuanyuan wei Date: 2023.08.14 13:05:16 +08'00'	Main supervisor's signature:	Julian Jang-Jaccard Digitally signed by Julian Jang-Jaccard Date: 2023.08.15 03:26:18 +12'00'

This form should be placed at the beginning of each relevant thesis chapter.