8-21-2023 10:30 AM

Toward Building an Intelligent and Secure Network: An Internet Traffic Forecasting Perspective

Sajal Saha,

Supervisor: Haque, Anwar, *The University of Western Ontario*

A thesis submitted in partial fulfillment of the requirements for the Doctor of Philosophy degree in Computer Science

© Sajal Saha 2023

Abstract

Internet traffic forecast is a crucial component for the proactive management of self-organizing networks (SON) to ensure better Quality of Service (QoS) and Quality of Experience (QoE). Given the volatile and random nature of traffic data, this forecasting influences strategic development and investment decisions in the Internet Service Provider (ISP) industry. Modern machine learning algorithms have shown potential in dealing with complex Internet traffic prediction tasks, yet challenges persist.

This thesis systematically explores these issues over five empirical studies conducted in the past three years, focusing on four key research questions: How do outlier data samples impact prediction accuracy for both short-term and long-term forecasting? How can a denoising mechanism enhance prediction accuracy? How can robust machine learning models be built with limited data? How can out-of-distribution traffic data be used to improve the generalizability of prediction models? Based on extensive experiments, we propose a novel traffic forecast/prediction framework and associated models that integrate outlier management and noise reduction strategies, outperforming traditional machine learning models. Additionally, we suggest a transfer learning-based framework combined with a data augmentation technique to provide robust solutions with smaller datasets. Lastly, we propose a hybrid model with signal decomposition techniques to enhance model generalization for out-of-distribution data samples.

We also brought the issue of cyber threats as part of our forecast research, acknowledging their substantial influence on traffic unpredictability and forecasting challenges. Our thesis presents a detailed exploration of cyber-attack detection, employing methods that have been validated using multiple benchmark datasets. Initially, we incorporated ensemble feature selection with ensemble classification to improve DDoS (Distributed Denial-of-Service) attack detection accuracy with minimal false alarms. Our research further introduces a stacking ensemble framework for classifying diverse forms of cyber-attacks. Proceeding further, we proposed a weighted voting mechanism for Android malware detection to secure Mobile Cyber-Physical Systems, which integrates the mobility of various smart devices to exchange information between physical and cyber systems. Lastly, we employed Generative Adversarial Networks for generating flow-based DDoS attacks in Internet of Things environments.

By considering the impact of cyber-attacks on traffic volume and their challenges to traffic prediction, our research attempts to bridge the gap between traffic forecasting and cyber security, enhancing proactive management of networks and contributing to resilient and secure internet infrastructure.

Keywords: Proactive Networks, Internet Traffic Forecast, Machine Learning, Deep Learning, Transfer Learning, Outlier Data Samples, Traffic Denoising, Model Generalization, Out-of-Distribution Traffic, Data Augmentation, Cyber-attacks, Distributed Denial of Service (DDoS) Attacks, Ensemble Feature Selection (EnFS), Ensemble Classification (SupEnML), Android Malware Detection, IoT, Generative Adversarial Networks (GANs).

Lay Abstract

Predicting internet traffic is like forecasting the weather - it's crucial for planning, but it's also quite challenging due to the unpredictable nature of data flow. This process is important for companies that provide internet services because it helps them plan their resources and investments more effectively.

In our research, we have used modern computer algorithms, commonly known as machine learning, to predict this traffic. However, we faced certain challenges - for instance, sometimes we have limited data or outliers (data points that are significantly different from others), which can impact the prediction's accuracy.

Over the past three years, we conducted five studies to answer these challenges and others. The result was the creation of a new model that can predict traffic better than traditional ones by managing unusual data and reducing the noise in the information. We also developed methods to work with small data and improve predictions on data types not seen during the training of the model.

On top of traffic prediction, our work also focused on detecting cyber-attacks, specifically those causing a lot of internet traffic like DDoS attacks. These attacks often disguise themselves and cause unexpected traffic bursts, making it hard to predict traffic volumes accurately. To solve this, we combined different techniques to improve detection accuracy with fewer false alarms. We also created different models for detecting various types of attacks, malware on Android devices, and even developed a way to enhance data in Internet of Things (IoT) environments. We validated all these methods with multiple sets of data, proving the effectiveness of our solutions.

In simple terms, our work helps internet providers better predict traffic and secure networks from cyber-attacks. It's like providing them with a more accurate traffic forecast and a better security system, ensuring smoother internet experiences for all users.

Co-Authorship Statement

This thesis is written in an integrated article format. A total of 13 papers comprises this thesis. All of the papers related to this thesis are either published, in press, or to be submitted. The author of this dissertation is the primary author of 10 of the papers.

Chapters 2, 3, 4, and 5 have been co-authored by Dr. Anwar Haque and Dr. Greg Sidebottom. Sajal Saha, the author of this dissertation, conducted the literature survey, developed the approach, formulated the study's design, implemented the research, and drafted the manuscripts for publication. Dr. Anwar Haque, as the research supervisor, participated in the study's design, interpretation of results, and manuscript preparation. Additionally, as an industry collaborator, Dr. Greg Sidebottom provided real-world traffic data and reviewed our technical contributions.

Chapter 6 is co-authored by Saikat Das, Annita Tahsin Priyoti, Etee Kawna Roy, Frederick T Sheldon, Anwar Haque, and Sajjan Shiva. Sajal Saha, the author of this dissertation, performed data analysis and interpretation, carried out one out of three implementations, and drafted the manuscripts for publication. Saikat Das conducted two out of three experiments and made contributions to the manuscript drafting. Annita Tahsin and Etee Kawna Roy were responsible for the introduction and literature survey, respectively. Frederick T Sheldon, Anwar Haque, and Sajjan Shiva provided valuable insights, intuition, and assistance in designing the model, as well as in preparing the manuscripts for publication.

Chapter 7 is co-authored by Ibrahim Mohammod, Moinul Islam Sayed, and Anwar Haque. Ibrahim Mohammod performed data analysis and interpretation, carried out the implementation, and drafted the manuscripts for publication. Moinul Islam Sayed assisted in writing the introduction and literature review sections of the manuscript. Sajal Saha, the author of this dissertation, contributed to designing the model and participated in preparing the manuscript for publication. Dr. Anwar Haque provided valuable intuition, insights that aided in designing the model, and also assisted in preparing the manuscripts for publication.

Chapter 8 is co-authored by Moinul Islam Sayed and Anwar Haque. Moinul performed data analysis and interpretation, carried out the implementation, and drafted the manuscripts for publication. Sajal Saha, the author of this dissertation, contributed to designing the model and played a role in writing the introduction, literature review, methodology, and conclusion sections of the manuscript. Dr. Anwar Haque provided valuable intuition, insights that aided in designing the model, and also assisted in preparing the manuscripts for publication.

Chapter 9 is co-authored with Dr. Rashed Nekvi and Dr. Anwar Haque. Dr. Rashed Nekvi, a former postdoc in Dr. Anwar Haque's lab, made contributions to manuscript writing and preparation for publication. Sajal Saha, the author of this dissertation, performed the implementation and contributed to writing the manuscripts for publication. As the research supervisor, Dr. Anwar Haque participated in the study's design, interpretation of results, and preparation of the manuscripts for publication.

Acknowlegements

I would like to begin by expressing my deepest gratitude to God for His endless guidance and blessings that have been my pillar throughout this journey. His grace has made all things possible, and it is to Him that I owe my deepest thanks.

Following this, I would like to express my sincere appreciation and thanks to my supervisor, Dr. Anwar Haque. He is among the best mentors I have found so far. His guidance and belief in me during my Ph.D. journey have been invaluable. Dr. Haque has not only helped me with my research but also supported me when things were tough, both academically and personally. His advice and concern extended beyond our work, often guiding me in matters related to my career and family. His ability to understand and respect the balance between my studies and personal life has made this journey smoother. His support has been a driving force, always motivating me to push my boundaries. Dr. Haque has been more than a mentor - he has been a guiding light in my life, and for that, I am profoundly grateful.

Additionally, I would like to express my heartfelt gratitude to my lab members, Yaser Al Mtawa, Rashed Nekvi, Aakriti Sharma, Annita Tahsin Priyoti, Ibrahim Sayem, and Moinul Islam Sayed. Their presence and contribution have significantly enriched my Ph.D. journey. I am truly grateful for their support. The experiences we've shared and the knowledge we've built together have made this journey not only fruitful but also memorable. I extend my sincere gratitude to Ange for her indispensable administrative support at the university, an invaluable contribution to my research journey.

I would also like to extend my sincerest thanks to my colleagues at Juniper Network, Greg Sidebottom and Sabyasachi Mukhopadhyay. Their insights, professional guidance, and supportive partnership have been a considerable part of my journey. Despite their demanding professional schedules, they found time to contribute to my research, offering critical reviews and providing valuable suggestions that have shaped my study. I am incredibly grateful for their assistance and feel fortunate to have had the opportunity to work alongside such dedicated and talented individuals.

I also acknowledge the insightful thoughts and valuable feedback from my thesis examiners: Dr. Khalid Elgazzar, Dr. Xianbin Wang, Dr. Hanan Lutfiyya, and Dr. Apurva Narayan.

I wish to express my deepest gratitude to my family and my in-laws. Their unwavering support, understanding, and encouragement have been my stronghold during this journey. Their belief in my abilities has instilled in me the strength to persevere. I am forever grateful for their love and constant support. This achievement would not have been possible without them.

Finally, I am forever indebted to my partner, Susmita Saha, who has been a constant source of love, emotional support, and understanding. Raising our first child while supporting my studies has been tough, but she has been there every step of the way. Her belief in me and her constant encouragement have kept me going. I am incredibly grateful for everything she has done, and her support has made all of this possible. She is my biggest inspiration.

Contents

Abstract i

Lay Abstract ii

Co-Authorship Statement iii

Acknowlegements iv

List of Figures ix

List of Tables xii

List of Appendices xiv

1 Introduction 1

1.0.1 Thesis Contribution . . . 5
1.0.2 Chapter Mapping . . . 7
1.0.3 Thesis Organization . . . 8

2 Single and Multi-step Traffic Forecasting with Outlier Mitigation 11

2.1 Introduction . . . 11
2.2 Literature Review . . . 13
2.3 Proposed Methodology . . . 16
2.3.1 Dataset Description . . . 16
2.3.2 Handling Missing Value . . . 17
2.3.3 Autocorrelation Function (ACF) Analysis . . . 18
2.3.4 Anomaly Detection and Mitigation . . . 18
2.3.5 Time-Lagged Feature Extraction . . . 20
2.3.6 Time-series Cross-validation . . . 21
2.3.7 Boosting Algorithms for Traffic Prediction . . . 22
2.3.8 Deep Sequence Algorithms for Traffic Prediction . . . 24
2.3.9 Evaluation Metrics . . . 25
2.4 Analysis of Experimental Results . . . 26
2.4.1 Outlier Mitigation Impact Analysis . . . 27
2.4.2 Single-Step Prediction Result Analysis Using Boosting and Deep Sequence Algorithms . . . 28

2.4.3 Multi-Step Prediction Result Analysis Using Boosting Algorithm and Deep Sequence Model . . . 31
2.5 Conclusion . . . 44

3 EMD-KNN Integrated Efficient Traffic Prediction Model 45
3.1 Introduction . . . 46
3.2 Literature Review . . . 47
3.3 Proposed Methodology . . . 48
3.3.1 Empirical Mode Decomposition Based Noise Reduction . . . 49
3.3.2 Empirical Rule and KNN Based Outlier Management . . . 50
3.3.3 ARIMA Based Time-lagged Feature Extraction . . . 51
3.4 Analysis of Experimental Results . . . 51
3.4.1 Outlier Data Identification and Mitigation . . . 51
3.4.2 Internet Traffic Denoising . . . 53
3.4.3 Optimum Feature Selection for Prediction Model . . . 54
3.4.4 Proposed Model Performance Analysis . . . 55
3.5 Conclusion . . . 57

4 Traffic Prediction via Transfer Learning & Data Augmentation 58
4.1 Introduction . . . 58
4.2 Literature Review . . . 60
4.3 Proposed Methodology . . . 62
4.3.1 Dataset and Preprocessing Steps . . . 63
4.3.2 Data Augmentation . . . 65
4.3.3 Deep Transfer Learning . . . 65
4.3.4 Model Selection and Implementation . . . 67
LSTM_Seq2Seq (LSTM Encoder-Decoder) . . . 67
LSTM_Seq2Seq_ATN (LSTM Encoder-Decoder with Attention) . . . 68
4.4 Analysis of Experimental Results . . . 71
4.4.1 Prediction Model Performance in Source Domain . . . 71
4.4.2 Source Domain Model Performance on the Target Domain Before Data Augmentation . . . 72
4.4.3 Source Domain Model Performance on the Target Domain After Data Augmentation . . . 73
4.5 Conclusion . . . 75

5 Evaluating ML Models for Traffic Prediction: IID vs. OOD Data 77
5.1 Introduction . . . 77
5.2 Literature Review . . . 79
5.3 Proposed Methodology . . . 81
5.3.1 Dataset Description . . . 82
5.3.2 Data Preprocessing . . . 83
5.3.3 Discrete Wavelet Transform . . . 83
5.4 Analysis of Experimental Results . . . 86
5.4.1 Performance Analysis Using Boosting Algorithms . . . 86

5.4.2 Performance Analysis Using Deep Sequence Models 88
5.5 Conclusion 92

6 Ensemble ML & FS for Network Intrusion Detection 94
6.1 Introduction 94
6.2 Literature Review 96
6.3 Proposed Methodology 98
6.3.1 Data Collection 98
6.3.2 Data Preprocessing 98
6.3.3 Ensemble Feature Selection (EnFS) 100
6.3.4 Model Classification 100
6.3.5 Anomaly Detection 102
6.3.6 Proposed Algorithm 103
6.4 Analysis of Experimental Results 103
6.4.1 Software and Hardware Preliminaries 103
6.4.2 Datasets 103
6.4.3 Evaluation Metrics 104
6.4.4 Discussion of Results 104
6.5 Conclusion 111

7 Stacking Ensemble for Cyber-Attack Detection and Classification 112
7.1 Introduction 112
7.2 Literature Review 114
7.2.1 Machine Learning-Based Approach 114
7.2.2 Deep Learning-Based Approach 115
7.3 Proposed Methodology 117
7.3.1 Dataset Description and Preprocessing 119
Dataset Description 119
Data Pre-processing 120
7.3.2 Optimal Feature Selection 121
7.3.3 Data Resampling 123
7.3.4 Stacking Model Description 124
Base Model 125
Ensemble Model 126
7.3.5 Complexity Analysis of Proposed Model 127
7.4 Analysis of Experimental Results 128
7.4.1 Software and Hardware Requirements 128
7.4.2 Evaluation Metric 128
7.4.3 Analysis of Experimental Results 130
7.5 Conclusion 133

8 Dynamic Weighted Voting for Android Malware Detection in MCPS 135
8.1 Introduction 135
8.2 Literature Review 137
8.3 Proposed Methodology 138

8.3.1 Dataset and Data Preprocessing 138
8.3.2 Feature Selection 140
8.3.3 Malicious Application Detection Model 141
Majority Voting 141
Proposed Dynamic Weighted Voting 141
8.3.4 Evaluation Metrics 141
8.3.5 Software and Hardware Requirement 142
8.4 Analysis of Experimental Results 142
8.5 Conclusion 144

9 Exploring GANs for Smart Home DDoS Traffic Generation 146
9.1 Introduction 146
9.2 Literature Review 147
9.3 Proposed Methodology 149
9.3.1 Smart Home Emulated Network and IoT Dataset 149
Network Topology Model 149
Data Collection and Description 149
9.3.2 Synthetic Attack Data Generation and Validation Approach 150
GAN Based Data Generation in Feature Space 150
Evaluation of Synthetic IoT Traffic Dataset 152
9.4 Analysis of Experimental Results 152
9.4.1 Generated Synthetic IoT Traffic Dataset 152
9.4.2 Evaluation of Synthetic Traffic for Quality 153
9.5 Conclusion 154

10 Conclusion and Future Work 156
10.1 Conclusion 156
10.2 Future Work 157
10.2.1 Adaptation of External and Internal Factors 157
10.2.2 Internet Traffic Synthetic Dataset Generation using GAN 158
10.2.3 Real-time Anomaly Detection Based on Traffic Prediction 158
10.2.4 Anomaly Detection Based on Traffic Forecasting 158
10.2.5 Enhancing Attack Detection through Reinforcement Learning 159

Bibliography 160

A Copyright Forms of the Papers 178

Curriculum Vitae 188

List of Figures

1.1 Growth of internet user. 1
1.2 Thesis architecture. 9

2.1 High-level framework of proposed traffic prediction model integrated with anomaly detection and mitigation. 17
2.2 Outliers identified by Three-Sigma rule. 19
2.3 Anomaly detected by ISoF. 19
2.4 Anomaly detected by KNN. 20
2.5 Anomaly detected by CBLO. 20
2.6 Time-lagged feature extraction. 21
2.7 Rolling-based cross validation. 22
2.8 Chain multi-output regression model. 23
2.9 Original traffic vs. outlier mitigated traffic. 26
2.10 Empirical Cumulative Distribution Function (ECDF) plot of our traffic data. . . 26
2.11 Machine learning model accuracy comparison for different input length. 29
2.12 Actual vs. predicted traffic by proposed LSTM_Seq2Seq model. 30
2.13 Average execution time of our proposed models for different forecast length. . . 35
2.14 A comparative comparison among different feature subset performances regarding average prediction accuracy for six-steps forecasting. The average accuracy is calculated as the mean of each step's individual prediction accuracy. . 35
2.15 A comparative comparison among different feature subset performances regarding average prediction accuracy for nine-steps forecasting. The average accuracy is calculated as the mean of each step's individual prediction accuracy. 39
2.16 A comparative comparison among different feature subset performances regarding average prediction accuracy for twelve-steps forecasting. The average accuracy is calculated as the mean of each step's individual prediction accuracy. 39
2.17 Multi-step actual vs. predicted traffic by best-peforming deep sequence model. . 40
2.18 Model average accuracy comparison with the input length. 41
2.19 LSTM_Seq2Seq model average accuracy comparison with the input length. . . 42
2.20 LSTM_Seq2Seq_ATN model average accuracy comparison with the input length. 43

3.1 High-level framework of our proposed methodology integrated with EMD based noise reduction and KNN based outlier mitigation. 49
3.2 Outlier points identified using empirical rule. 52
3.3 Distribution of outlier points. 52
3.4 Denoised traffic data. 52

4.1 High-level overview of our proposed traffic prediction model integrated with transfer learning and data augmentation. 62
4.2 Data distribution of source and target domain datasets. 64

5.1 High-level framework of our proposed hybrid traffic prediction model for our-of-distribution generalization. 81
5.2 Data distribution of different datasets. 82
5.3 Best accuracy comparison between standalone and hybrid model. 86
5.4 Comparison of actual and predicted traffic using our proposed hybrid model for OOD dataset. 87
5.5 Comparison of actual and predicted traffic using our proposed hybrid model for OOD dataset. 90

6.1 High-level framework of our proposed ensemble machine learning model for intrusion detection. 99
6.2 Ensemble feature selection approach (EnFS) using majority voting. 100
6.3 Ensemble machine learning framework as the detection technique of an IDS. . . 102
6.4 a) Comparative performance analysis using nine feature sets, full feature sets and ensemble feature sets for NSL-KDD dataset, b) ROC curves for ensemble feature selection using NSL-KDD datasets. 106
6.5 a) Comparative performance analysis using nine feature sets, full feature sets and ensemble feature sets for UNSW-NB15 dataset, b) ROC curves for ensemble feature selection using UNSW-NB15 datasets. 108
6.6 a) Comparative performance analysis using nine feature sets, full feature sets and ensemble feature sets for CICIDS2017 dataset, b) ROC curves for ensemble feature selection using CICIDS2017 datasets. 109

7.1 The framework of the proposed NIDS. 118
7.2 UNSW-15 dataset attack distributions. 119
7.3 CICIDS-2017 dataset attack distributions. 120
7.4 Feature importance in UNSW-15 dataset. 122
7.5 Feature importance in CICIDS-2017 dataset. 122
7.6 Attack distribution of UNSW-15 after resampling. 123
7.7 Attack distribution of CICIDS-2017 after resampling. 124
7.8 The stacking ensemble architecture. 126
7.9 Configuration of the proposed stacking model. 127
7.10 Confusion matrix. 129
7.11 Confusion matrix of the proposed ENIDS model on the UNSW-15 dataset. . . . 131
7.12 Confusion matrix of the proposed ENIDS model on the CICIDS-2017 dataset. . 133

8.1 High-level framework of proposed methodology of Android malware detection. 138
8.2 Frequency of features over feature importance. 140
8.3 Performance comparison among detection models. 142
8.4 Confusion matrix for majority voting model. 143
8.5 Confusion Matrix for dynamic weighted voting model. 144

9.1 High-level overview of proposed framework for adversarial attack generation using GAN. 149
9.2 Accuracy of different classifiers trained by synthetic datasets. 153

List of Tables

2.1 Our Core Contribution Compared to Existing Works. 16
2.2 Single-Step Traffic Prediction Machine Learning Model Performance Summary. 26
2.3 Statistical Analysis Before and After Outlier Mitigation. 27
2.4 Single-Step Traffic Prediction Deep Learning Model Performance Summary. . . 30
2.5 Six-Step Forecasting MAPE (%) Value For The Standard And Proposed Machine Learning Models. Each Value Represents An Average Gap Between Actual and Predicted Traffic Volume For Each Individual Forecast Step. 32
2.6 Nine-Step Forecasting MAPE (%) Value For The Standard And Proposed Machine Learning Models. Each Value Represents An Average Gap Between Actual and Predicted Traffic Volume For Each Individual Forecast Step. 33
2.7 Twelve-Step Forecasting MAPE (%) Value For The Standard And Proposed Machine Learning Models. Each Value Represents An Average Gap Between Actual and Predicted Traffic Volume For Each Individual Forecast Step. 34
2.8 Six-Step Forecasting MAPE (%) Value For The Standard And Proposed Deep Learning Models. Each Value Represents An Average Gap Between Actual and Predicted Traffic Volume For Each Individual Forecast Step. 36
2.9 Nine-Step Forecasting MAPE (%) Value For The Standard And Proposed Deep Learning Models. Each Value Represents An Average Gap Between Actual and Predicted Traffic Volume For Each Individual Forecast Step. 37
2.10 Twelve-Step Forecasting MAPE (%) Value For The Standard And Proposed Deep Learning Models. Each Value Represents An Average Gap Between Actual and Predicted Traffic Volume For Each Individual Forecast Step. 38

3.1 KNN-Regressor Prediction Error (RMSE) For Different *K* Values. 53
3.2 Signal-to-Noise(SNR) Ratio Comparison. 54
3.3 Top Ten Best-Performing ARIMA Model Parameter Configuration. 54
3.4 Prediction Accuracy by Different Traffic Model. 56

4.1 Summary Statistics Comparison across Different Datasets. 64
4.2 Prediction Model Performance Summary in Source Domain. 71
4.3 Multi-step Model Performance Summary Before Data Augmentation in Target Domain. 72
4.4 Multi-step Model Performance Summary After Data Augmentation in Target Domain. 74

5.1 Model Performance: Train and Test on Dataset A Using Gradient Boosting Algorithm. 85

5.2 Model Performance: Trained on Dataset A, Tested on Dataset B Using Gradient Boosting Algorithm. 85
5.3 Model Performance: Trained on Dataset A, Tested on Dataset C Using Gradient Boosting Algorithm. 85
5.4 Model Performance: Trained on Dataset A, Tested on Dataset D Using Gradient Boosting Algorithm. 85
5.5 Model Performance: Train and Test on Dataset A Using Deep Sequence Algorithm. 88
5.6 Model Performance: Trained on Dataset A, Tested on Dataset B Using Deep Sequence Model. 89
5.7 Model Performance: Trained on Dataset A, Tested on Dataset C Using Deep Sequence Model. 89
5.8 Model Performance: Trained on Dataset A, Tested on Dataset D Using Deep Sequence Model. 89

6.1 A Comparative Analysis of the Study of Literature Review. 97
6.2 Data Instances for NSL-KDD, UNSW-NB15, and CICIDS2017 Datasets Used in Experiment. 101
6.3 Hyper-Parameter Values Used for Different Individual and Ensemble Classifiers. 101
6.4 Best Performing Results Using Nine Feature Sets, Full Feature Sets and Ensemble Feature Sets for NSL-KDD Dataset. 105
6.5 Best Performing Results Using Nine Feature Sets, Full Feature Sets and Ensemble Feature Sets for UNSW-NB15 Dataset. 107
6.6 Best Performing Results Using Nine Feature Sets, Full Feature Sets and Ensemble Feature Sets for CICIDS2017 Dataset. 107
6.7 A Performance Comparison of Our Method Compared to Existing Methods. . . 110
6.8 Verified the Best Performing Models' Performances Using Verification Set for Three Datasets. 110

7.1 Model Performance Comparison for the UNSW-15 Dataset. 131
7.2 Model Performance Comparison for the CICIDS-2017 Dataset. 132

8.1 Result Comparison with Existing Works. 144

9.1 Collected Features for Attack Classification 150
9.2 Proportion of Quality Synthetic Dataset Generated by Various Batch Sizes (BS) of GAN. 152

List of Appendices

Copyright Forms of the Papers 178

Chapter 1

Introduction

The internet continues changing how we connect with others, organize the flow of things, and communicate information worldwide. The demand for network traffic has risen significantly around the globe as network technology has advanced and digital activities such as video streaming, remote conferencing, online gaming, and e-commerce have increased. According to projections in the study of Cisco [1], the total number of internet users worldwide is expected to increase from 3.9 billion in 2018 to 5.3 billion by 2023, which corresponds to a compound annual growth rate (CAGR) of 6%. This means that in 2018, about 51% of the world's population was using the internet, while by 2023, it is estimated that about 66 percent of the global population will have access to the internet. Although the growth of internet users is a worldwide phenomenon, there are regional disparities, as indicated in Fig 1.1. While North America (and subsequently Western Europe) is projected to have the highest adoption rate throughout the forecast period, the Middle East and Africa are expected to experience the fastest growth, with a projected CAGR of 10% from 2018 to 2023.

The increased growth of internet users globally has led to a significant increase in internet traffic, which has put a strain on the capacity of existing networks. Key factors such as increased M2M (Machine-to-Machine) connections, IPv6 adaptation, and the rise in video conferencing have been instrumental in driving this change. However, another important yet often overlooked factor impacting internet traffic worldwide is the escalating frequency and scale of cyber attacks. Cyber attacks, specifically Distributed Denial of Service (DDoS) attacks, can significantly contribute to internet traffic. According to report [1], when a DDoS attack is happening, it can consume up to a quarter of a country's total Internet traffic. DDoS attacks function by flooding a target with unwanted data traffic in an attempt to overwhelm its

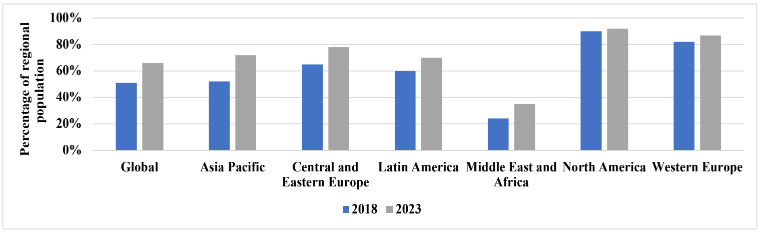


Figure 1.1: Growth of internet user.

resources and disrupt normal operations. This unwanted data represents a substantial portion of internet traffic when such an attack is underway. Given the rising intensity and prevalence of these attacks, their contribution to overall internet traffic is becoming increasingly significant. Moreover, as attackers leverage the growth in internet traffic to camouflage their activities, their methods are becoming more sophisticated. This leads to an increased volume of malicious traffic, contributing further to the overall traffic growth. The use of botnets, which involve a network of compromised devices that generate traffic to carry out large-scale attacks, further exacerbates the situation. In conclusion, the growth in global internet traffic is influenced not only by increasing user numbers, M2M connections, and the adoption of new technologies like IPv6 but is also significantly affected by the escalating trend of cyber attacks. As the digital world continues to expand, understanding and addressing these factors will be vital for maintaining network capacity and integrity.

Accurate forecasting of internet traffic patterns has become crucial to enable service providers to meet the demands of a larger customer base. Traffic forecasting helps service providers plan their network capacity and optimize the allocation of resources to ensure that customers receive reliable and consistent internet speeds. This can lead to fewer network outages, faster download and upload speeds, and better overall network performance. Accurate traffic forecasting allows service providers to identify potential bottlenecks in their networks and take corrective actions before they impact the quality of service experienced by customers. This can lead to a better customer experience, higher customer satisfaction, and increased customer loyalty. By accurately forecasting traffic patterns, service providers can optimize their network resources and reduce their costs. For example, they can avoid over-provisioning their network capacity, which can lead to unnecessary costs, or under-provisioning, which can result in network congestion and poor customer experience. Providing reliable and high-quality service is key to attracting and retaining customers. Accurate traffic forecasting helps service providers ensure that they can deliver a consistent and reliable service, which can increase customer satisfaction and loyalty, leading to a larger customer chunk. As a result, predicting network traffic based on historical trends is indispensable for dynamic bandwidth reservation and allocation, congestion control, admission control [2], and privacy-preserving routing [3] to ensure better Quality of Service (QoS) and Quality of Experience (QoE).

Overall, the benefits of wireless internet traffic forecasting based on real statistical data are significant for the telecommunications industry. By leveraging advanced analytics and machine learning techniques to analyze historical data and predict future traffic patterns, service providers can improve their network performance, reduce costs, and gain a competitive edge in the market. ISP industries are focusing on Artificial Intelligence (AI) to accommodate with global wireless network infrastructure ecosystem market is expected to surpass USD 51,716 Million by 2030 with a CAGR rate of 12.32% during the projected period [4]. For example, Verizon[5], a major wireless service provider in the United States, uses advanced analytics and machine learning to forecast wireless internet traffic. By doing so, the company is able to optimize its network performance and deliver a better quality of service to its customers. Huawei[6], a global leader in telecommunications equipment and services, has developed a wireless traffic forecasting solution that uses machine learning and big data analytics to predict network traffic trends. The solution has been successfully deployed by several major service providers around the world. Nokia[7], another leading telecommunications equipment and services provider, offers a wireless traffic forecasting solution that uses artificial intelligence to predict network traffic patterns. The solution has been shown to improve network performance and reduce costs for service providers. However, despite the progress made in machine learning techniques, accurately predicting real-world internet traffic remains a difficult task. The main reasons are as follows [8]:

- **Data Anomaly:** The traffic in real-world internet networks is affected by a multitude of external and internal factors that generate complex non-stationary traffic patterns. The internal factors that influence internet traffic are associated with Internet Service Provider (ISP) activities, such as the introduction of new services, traffic migration, and speed upgrades. Conversely, external factors are linked to events and circumstances that occur outside the ISP's control, such as the emergence of new internet applications, regional economic fluctuations, and seasonal effects. These factors often result in unexpected and suspicious variations in real IP traffic, commonly referred to as outliers or anomalies. These abnormal data points in traffic flow can have an adverse effect on the learning of the general trends in the data. As a result, prediction models may generate incorrect inferences, mistaking these anomalies for normal behavior. Consequently, it is crucial to identify and address any anomalies or outliers in internet traffic before applying any prediction model.

- **Data Scarcity:** To enhance the accuracy of predictions on the testing set, it is imperative to train the prediction model with a vast dataset, allowing the model to comprehend patterns effectively. However, in reality, managing a large historical dataset to train the prediction model can be challenging, resulting in poor performance. Despite this, numerous synthetic public datasets are available for research purposes, but they lack the random characteristics present in real-world traffic data. Consequently, developing an efficient traffic prediction tool that can provide accurate predictions in real-world scenarios is challenging.

- **Data Heterogeneity:** Internet traffic exhibits heterogeneity in the temporal dimension, with distinct geographical regions displaying different traffic patterns at different periods. Thus, proposing a generic prediction model that can accurately predict traffic patterns in diverse datasets is a challenging task. Additionally, the traffic prediction tool is likely to encounter deployment data that is slightly different or entirely unknown compared to the model's training and testing data distribution. Consequently, measuring the performance of the prediction tool using identically distributed data alone can be difficult. As a result, constructing a robust machine-learning model capable of handling a shift in data distribution is a non-trivial task.

- **Unpredictable Bursts of Traffic:** Cyber attacks, particularly DDoS attacks, create sudden, unexpected surges in traffic. These traffic bursts are unpredictable and can significantly skew the volume, making it difficult for traditional forecasting models to accurately predict overall internet traffic.

There are several existing works on wireless internet traffic prediction. Predicting internet traffic is commonly thought of as a time series forecasting problem that can be tackled by using traditional statistical techniques such as ARIMA [9], SARIMA, Holt-Winter [10], etc.

But these models cannot predict the non-linear component of the actual internet traffic. On the other hand, the non-linear element of the time-series data can be captured by Neural Network (NN) based model such as a multi-resolution Finite-Impulse-Response (FIR) model [11], Genetic Algorithm and Radial Based Function Network (GA-RBF) [12] etc. Also, there are some non-linear statistical models e.g. Threshold AutoRegressive (TAR) [13], and Exponential AutoRegressive (ExpAR) [14] for handling the non-linear part in time series data. However, recent studies noticed an extensive usage of machine learning and deep learning models for accurate and efficient prediction in different domains, e.g., predicting the stock market[15], finance[16], weather forecast[17], etc. Among various deep neural networks, Recurrent Neural Networks (RNN) and their variants, such as Long Short Time Memory (LSTM), LSTM Encode-Decoder, Gated Recurrent Unit (GRU), etc., received extra attention due to their capability of sequential data processing. The sequence model can store previous inputs and share hyper-parameters across time. Since the standard neural networks cannot remember previous state information, the sequence model architecture provides an extra benefit for effectively processing internet traffic in a time-series format. For example, in [18], they used a residual network (LAResNet) to model the spatial characteristics of the sequence data. Also, the combination of RNN and an attention structure is employed to understand the transformation mode of the wireless network in the temporal dimension and to extract the difference between different regional traffic patterns. The LSTM and Online-Sequential Extreme Learning Machine (OS-ELM) models are used for forecasting traffic load using a real ISP dataset [19]. In [20], an LSTM encoder-decoder model has been proposed to predict the statistical characteristics of the traffic data in a 6G environment. A multi-variate time-series dataset was collected from their experimental test bed, and the results show that their proposed novel framework provides accurate performance compared to the ground truths. The literature suggests that there are various approaches to predicting traffic, such as statistical models, machine learning, and deep learning. However, many of these solutions are highly dependent on the data, which creates substantial research gaps, as below, when it comes to accurately forecasting real-world internet traffic.

Firstly, outlier point detection is important for internet traffic forecasting because it helps identify unusual patterns or behavior in the traffic data. Outliers are data points that deviate significantly from the average or expected values of the dataset. These outliers can be caused by various factors such as network failures, cyber-attacks, unexpected spikes in demand, or other anomalies. If outliers are not detected and removed from the dataset, they can significantly skew the statistical analysis and forecasting models. This can lead to inaccurate predictions, which can have serious consequences for businesses that rely on internet traffic data for planning and decision-making. By detecting and removing outliers, internet traffic forecasting models can be improved in accuracy and reliability. This can help businesses make better decisions about network capacity planning, traffic management, and other related activities. In summary, outlier point detection is crucial for internet traffic forecasting because it helps identify abnormal patterns in the data and improves the accuracy and reliability of forecasting models. But current literature did not analyze the impact of outlier on traffic prediction. As most of the works considered synthetic dataset without having any abnormal characteristics for their experiment and it did not require any outlier detection and mitigation before developing the prediction model. Although, outlier analysis is one of the fundamental step of traffic analysis.

Secondly, noise reduction is useful in removing unwanted or irrelevant data from the signal, which can improve the accuracy of the predictive model. This is particularly important in wireless traffic forecasting because wireless networks can be subject to various sources of noise, such as interference from other networks, radio frequency interference, or environmental factors. By removing this noise, the predictive model can better identify the underlying patterns in the data, resulting in more accurate predictions. However, noise reduction can also lead to the loss of important information, particularly if the noise is not well understood or if the model is overly aggressive in removing it. There are various ways of noise reduction from time series data. But in case of wireless traffic forecasting, these techniques are not well analyzed in existing works.

Thirdly, the field of internet traffic prediction has seen significant advancements with the integration of artificial intelligence, machine learning, and deep learning-based models. However, these models require a vast amount of historical data to learn the general patterns in traffic, making it challenging to develop individual models for different geographical or network sectors. But transfer learning is being explored as a potential solution, allowing knowledge to be transferred from existing prediction models to devise new models for smaller datasets. By leveraging transfer learning techniques such as parameter transfer, domain adaptation, and multi-task learning, researchers aim to create more accurate and reliable traffic prediction models that can be efficiently deployed across different network segments. We found a limited exploration of transfer learning in wireless traffic prediction so that we can make better prediction model for smaller dataset using prior knowledge.

Finally, developing an accurate prediction model for real internet traffic is a challenging task due to its non-linear characteristics such as time-variability, long-term correlation, self-similarity, suddenness, and chaos[21]. Despite these complexities, machine learning and deep learning-based methods have shown impressive performance. However, most of the current approaches assume that the data is independent and identically distributed (IID) which is not always the case in real-world scenarios. The data distribution can vary due to heterogeneous and anomalous internet traffic, which can result in selection biases, confounding factors, and other biases in the datasets[22]. These biases can lead to models that overfit to the training data and fail to generalize to out-of-distribution data[23]. Therefore, OOD generalization is required to address the problem of overfitting and ensure that the predictive model can effectively generalize to new scenarios. This can be achieved by training the model on a diverse range of traffic scenarios, including those that are different from the ones encountered in the training data. The model can then learn to identify the underlying patterns in the traffic data, enabling it to accurately predict traffic in new scenarios. Current works on internet traffic forecasting did not explore this particular topic extensively.

1.0.1 Thesis Contribution

In our research, we address the aforementioned limitations in internet traffic forecasting and perform a comprehensive analysis to investigate each problem separately. Our core contributions in internet traffic volume prediction domain are as follows:

- We conducted a comprehensive analysis of the performance of deep sequence models and gradient-boosting algorithms for single-step and multi-step traffic prediction. We also compared the performance of these models with standard machine learning models commonly used for traffic prediction. To further enhance the accuracy of the prediction model, we integrated an outlier detection and mitigation module to identify and remove data points that deviate significantly from the general pattern in the data. We evaluated the impact of outlier data points on traffic prediction and compared the performance of the model with and without the outlier detection and mitigation module.

- In internet traffic forecasting, outlier data point analysis, and noise reduction are commonly used techniques to improve prediction accuracy. In this study, we proposed a unique traffic prediction model integrated with an Empirical Mode Decomposition (EMD)-based noise reduction technique. We applied this technique to remove unwanted or irrelevant data from the signal, such as random fluctuations or interference, to improve the accuracy of the prediction model.

- We devised a unique approach for predicting internet traffic with limited data. We integrated transfer learning and data augmentation ti utilize knowledge from a larger dataset and addressed overfitting concerns using Discrete Wavelet Transform (DWT) augmentation. We analyzed model performance under varying source-to-target data ratios.

- We addressed the challenges posed by the unpredictable and complex nature of internet traffic, specifically focusing on the accuracy of prediction models in scenarios where the distribution of test data deviates significantly from that of the training data. We assess the performance of various boosting and deep sequence models using both Independent and Identically Distributed (IID) and Out-of-Distribution (OOD) data samples. Through our analysis, we found a significant reduction in the predictive accuracy of these models when dealing with OOD samples, indicating a performance limitation of classical machine learning models in OOD scenarios. To overcome this problem, we proposed a novel solution that combines a hybrid deep learning model with Discrete Wavelet Transformation (DWT).

Following a thorough investigation into traffic prediction, we have identified several challenges intrinsic to real-world traffic forecasting, particularly those arising from the unpredictability of traffic bursts. Cyberattacks, notably Distributed Denial of Service (DDoS) attacks, significantly contribute to the spike in global network traffic. During their occurrence, DDoS attacks can account for as much as 25 percent of a country's total Internet traffic.

The escalating frequency and intensity of cyber attacks pose a significant hurdle for accurate internet traffic volume forecasting. This complexity is compounded by the fast-evolving techniques employed by cybercriminals, which create new forms of traffic that can outpace the adaptability of forecasting models. The increased noise from heightened attack activities, the insufficiency of historical data due to the rapidly changing cyber threat landscape, and the variable impact of attacks further obscure patterns essential for precise forecasts. Consequently, tackling this challenge necessitates the evolution of more advanced forecasting models capable of considering the unique traits and implications of cyber attacks. Therefore, we plan to initially employ supervised learning for attack detection before transitioning to forecasting-based models is founded on several strategic advantages. Firstly, supervised learning offers a basic understanding of the domain, laying the groundwork for more complex methodologies by demonstrating how labeled data can predict outcomes. Its relative simplicity compared to forecasting models makes it an ideal starting point. Furthermore, it facilitates the establishment of a performance baseline for comparison with more complex models. The abundant availability of labeled datasets for network intrusion detection and the interpretability of supervised learning models also support this choice. Moreover, understanding supervised learning is crucial as forecasting-based models may still require some form of supervised learning for labeling anomalies. Therefore, starting with supervised learning ensures a solid foundation and a thorough understanding of the data, which will be valuable during the planned future transition to forecasting-based attack detection models. Our core contributions in cyber-attack detection domain are as follows:

- Our method involves systematic steps for optimal network anomaly detection. We utilized grid search to find the best feature selection techniques, leading to our novel ensemble feature selection (EnFS) approach. This improved classification outcomes by optimizing feature sets. We extended the Supervised Ensemble Machine Learning framework for broader anomaly detection and incorporated EnFS, yielding a robust Intrusion Detection System validated on renowned datasets.

- We presented ENIDS, an innovative Ensemble Network Intrusion Detection System. ENIDS employs deep learning algorithms (CNN, LSTM, GRU) to detect known attacks and integrates them into an efficient stacked ensemble framework. Addressing class imbalance, ENIDS utilizes SMOTEENN and resampling to enhance detection performance. Rigorously evaluated on real-world datasets, ENIDS outperforms in network intrusion detection.

- Tackling Android malware in Mobile Cyber-Physical Systems, our approach introduced dynamic analysis for identifying malicious apps and their categories. The key innovation is ”Dynamic Weighted Voting,” an ensemble technique outperforming other deep learning models in detecting and categorizing Android malware.

- In IoT-based smart homes, we addressed neural network vulnerability to attacks by using GANs to create realistic synthetic datasets simulating DDoS attack traffic. Uniquely focusing on smart home-specific IoT traffic, our GAN model generates datasets for training classifiers, evaluated using Train-on-Synthetic, Test-on-Real approach against actual IoT traffic data.

1.0.2 Chapter Mapping

The chapters in this thesis are thoughtfully organized to furnish the reader with a comprehensive understanding of the progression of our ideas. We commenced our research primarily focused on addressing data anomalies within real-world internet traffic, examining both point outliers and noise. Initially, we conducted an investigation into outlier data points, analyzing the performance of various deep sequence models and boosting algorithms for both single and multi-step prediction. This exploration is reflected in Chapter 2. Subsequently, we ventured to design machine learning models adept at managing noise in internet traffic. To this end, we integrated an Empirical Mode Decomposition (EMD) based noise reduction technique into our traffic prediction framework. This methodology is elaborated in Chapter 3. Building on this, we proposed a transfer learning-based traffic prediction model to tackle data scarcity. Furthermore, we integrated a data augmentation technique to identify the optimal ratio between source and target domain data sizes. This concept is detailed in Chapter 4. In our final approach to enhance traffic prediction task, we addressed out-of-distribution data generalization in traffic prediction. We introduced a hybrid machine learning model, coupled with Discrete Wavelet Transformation (DWT), aiming to enhance the generalization capability of our prediction model. This architecture is discussed in Chapter 5.

Parallel to these efforts, our research also delved into cyber attack detection, specifically zeroing in on Distributed Denial of Service (DDoS) attacks, a significant contributor to the global network traffic surge. The escalating frequency and severity of these cyber attacks pose a considerable challenge to accurate internet traffic volume forecasting. Our initial foray into attack detection, based on supervised learning, is presented in Chapter 6. Here, we introduced an ensemble machine learning model to detect DDoS attacks, validating this approach with three benchmark datasets. In Chapter 7, we further enhanced our work with deep learning techniques, shifting our focus towards multi-classification rather than binary classification. Here, we expound on our stacking ensemble model designed to detect multiple types of attacks. Chapter 8 proposes a weighted voting mechanism for Android malware detection in mobile cyber-physical systems. Lastly, in Chapter 9, we turn our attention to the Internet of Things (IoT) environment. Here, we constructed smart home networks to collect DDoS attack data and integrated a Generative Adversarial Network (GAN) based approach to generate additional synthetic data within the smart home environment.

1.0.3 Thesis Organization

The organization of this thesis (Fig. 1.2) is as follows:

Chapter 1 sets the stage, presenting an introduction that encapsulates the problem at hand, the driving motivation behind this research, the objectives of the thesis, and the contributions it seeks to make to the wider research community.

In **Chapter 2**, we conduct a performance analysis of a variety of deep sequence models and boosting algorithms, focusing on their integration with outlier management for both single and multi-step predictions.

In **Chapter 3**, our focus shifts towards noise reduction in traffic data. We aim to enhance prediction accuracy by incorporating an Empirical Mode Decomposition (EMD)-based noise reduction technique into our traffic prediction framework. This chapter delves into the integration process and its potential impact on improving our predictions.

Chapter 4 introduces our proposal of a transfer learning-based traffic prediction model to combat data scarcity. Here, we also incorporate a data augmentation technique to discern the ideal ratio between source and target domain data sizes.

In **Chapter 5**, we tackle the issue of out-of-distribution data generalization in traffic prediction. We present a hybrid machine learning model paired with Discrete Wavelet Transformation (DWT) in our quest to boost the generalization potential of our prediction model.

Chapter 6 introduces our ensemble machine learning model, designed to detect DDoS attacks. We validate this approach using three benchmark datasets.

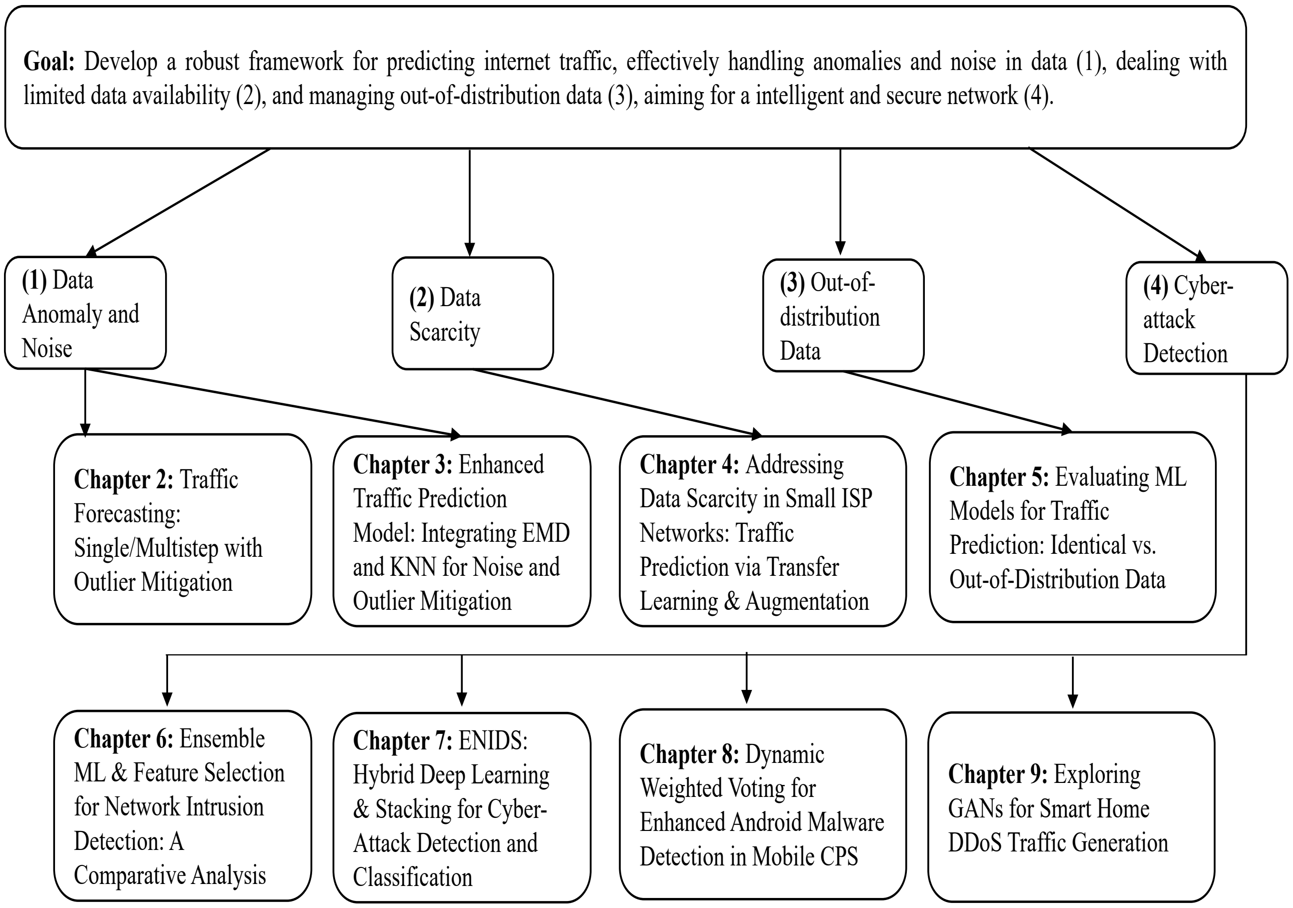

Goal: Develop a robust framework for predicting internet traffic, effectively handling anomalies and noise in data (1), dealing with limited data availability (2), and managing out-of-distribution data (3), aiming for a intelligent and secure network (4).
(1) Data Anomaly and Noise
(2) Data Scarcity
(3) Out-of-distribution Data
(4) Cyber-attack Detection
Chapter 2: Traffic Forecasting: Single/Multistep with Outlier Mitigation
Chapter 3: Enhanced Traffic Prediction Model: Integrating EMD and KNN for Noise and Outlier Mitigation
Chapter 4: Addressing Data Scarcity in Small ISP Networks: Traffic Prediction via Transfer Learning & Augmentation
Chapter 5: Evaluating ML Models for Traffic Prediction: Identical vs. Out-of-Distribution Data
Chapter 6: Ensemble ML & Feature Selection for Network Intrusion Detection: A Comparative Analysis
Chapter 7: ENIDS: Hybrid Deep Learning & Stacking for Cyber-Attack Detection and Classification
Chapter 8: Dynamic Weighted Voting for Enhanced Android Malware Detection in Mobile CPS
Chapter 9: Exploring GANs for Smart Home DDoS Traffic Generation

Figure 1.2: Thesis architecture.

Chapter 7 extends our work with the inclusion of deep learning techniques, moving our focus from binary to multi-classification. This chapter delves into our stacking ensemble model, engineered to identify multiple types of attacks.

Chapter 8 introduces a proposal for a weighted voting mechanism, designed for the detection of Android malware within mobile cyber-physical systems.

In **Chapter 9**, we turn our lens onto the Internet of Things (IoT) environment, constructing smart home networks to amass DDoS attack data. Here, we also incorporate a Generative Adversarial Network (GAN)-based approach to generate further synthetic data within the smart home environment.

Finally, **Chapter 10** provides a succinct summary of the thesis, evaluating its limitations and suggesting potential directions for future research.

Chapter 2

Single-step and Multi-step Internet Traffic Forecasting: Integrating Outlier Detection and Mitigation Techniques

Abstract: Internet traffic prediction (ITF), particularly multi-step forecasting, is complex due to data volatility and the presence of outliers. This issue is critical in the Internet Service Provider (ISP) industry for long-term planning and network management. We propose methodologies integrating outlier detection and mitigation (OTM) with deep sequence and gradient descent and boosting models. First, we contrast conventional deep sequence models (RNN, LSTM, LSTM_Seq2Seq, LSTM_Seq2Seq_ATN, and GRU) with our proposed outlier integrated model. Second, we introduce an ITF framework merging OTM with gradient descent and boosting algorithms. Applying real-world anomalous ISP traffic data, we explore five regression models (GBR, XGB, LGB, CBR, SGD) for both single and multi-step ITF across multiple forecast horizons. Our framework, pre-processing outliers, surpasses traditional models in prediction accuracy. These studies suggest a new direction in internet traffic prediction and offer insights for ISP industry's network management and strategic planning.

2.1 Introduction

Internet traffic volume forecasting is one of the most important tasks in the proactive management of modern telecommunication networks. Improving the accuracy and efficiency of traffic demand forecasting can help ISP companies develop reasonable business planning and enhance the industry's economic benefits. Moreover, the forecasting results with high accuracy can also be effective for better resource management, route scheduling, short and long-term business capacity planning, sophisticated network design, etc. In other words, an accurate traffic prediction framework can assist ISPs with preemptive network management and ensure better network Quality of Service (QoS) and Quality of Experience (QoE) [24]. Therefore, internet traffic demand management is vital for future requirements, capacity utilization, management, planning, and optimization. Because of these reasons, research on internet traffic forecasting has gained significant interest from researchers, the ISP industry, and operational planners. However, predicting wireless internet traffic remains challenging due to the high variability and unpredictability of network traffic. Traditional forecasting methods tend to be limited in their ability to handle the complex and dynamic nature of wireless network traffic, particularly in detecting and mitigating outliers or anomalous data points.

One significant research gap in this field is the need for a multi-step wireless internet traffic forecasting model integrated with outlier point detection and mitigation techniques. Real-world internet traffic is influenced by a multitude of internal and external factors, leading to unpredictable and random characteristics [25]. Internal factors such as service launches, traffic migration, and speed upgrades, and external factors such as new internet applications, regional economic variables, and seasonal effects can result in sudden changes in traffic patterns. As a result, accurately predicting internet traffic can be challenging. Moreover, anomalies or outliers are common in real-world internet traffic [26], which can further complicate traffic forecasting. These anomalies can occur due to issues with data collection sensors, leading to faulty data being included in the analysis. To improve the generalization capability of prediction models, it is essential to identify and address anomalies/outliers in internet traffic before using any prediction model. Thus, it is critical to develop robust methods for identifying and mitigating anomalies in internet traffic to improve the accuracy of traffic forecasts. By doing so, network operators can more effectively allocate network resources, improve the quality of service for end-users, and reduce the occurrence of network failures or congestion. Outlier data points can significantly impact the accuracy of traffic forecasts and lead to sub-optimal network performance. Therefore, the development of a multi-step traffic forecasting model that incorporates outlier detection and mitigation techniques can have several significant benefits.

The proposed model should be able to accurately predict traffic patterns while identifying and mitigating outlier data points in real-time. This approach will ensure that the forecasting model is robust and can provide accurate predictions, even when anomalous data points are present in the data. The use of outlier detection techniques can also provide network operators with insights into network behavior, enabling them to take proactive measures to maintain network performance. Moreover, the proposed model's integration with outlier detection and mitigation techniques can lead to better network performance and user experience. Network operators can use the model's predictions to optimize network resource allocation, which can lead to improved network reliability, reduced network congestion, and lower latency [27]. Therefore, the motivation for this research is to develop a multi-step wireless internet traffic forecasting model integrated with outlier point detection and mitigation techniques. The proposed model's effectiveness will be evaluated using real-world wireless network data, and the results will be compared to traditional forecasting methods to demonstrate its superiority. This research will contribute to the advancement of wireless network technologies and provide valuable insights for network operators to improve network performance and reliability.

Wireless internet traffic forecasting is a critical task in the field of telecommunications as it enables efficient network management and resource allocation. Multi-step forecasting, which involves predicting traffic patterns for multiple time steps into the future [28], is particularly important for proactive network planning and optimization. Gradient Boosting Algorithm (GBM) is a popular machine learning technique that has shown impressive results in various forecasting tasks, including time-series forecasting. Studies have demonstrated that GBM has been shown to outperform traditional forecasting methods such as ARIMA and neural networks, making it an attractive option for wireless internet traffic forecasting. However, there is a research gap in the comparative analysis of different gradient-boosting algorithms for wireless multi-step internet traffic forecasting. Understanding the performance of different GBM algorithms, such as XGBoost, LightGBM, and CatBoost, can help researchers and practitioners choose the most suitable algorithm for their specific needs. Additionally, exploring the impact of different feature subsets on the performance of GBM algorithms can provide valuable insights into optimizing these models for wireless multi-step internet traffic forecasting. We also extended this experiment using deep learning models as we found a lack of investigation in the performance comparison of deep sequences modeling techniques such as RNN and their variations which have the unique ability of sequence data analysis, e.g., time-series, audio data analysis, etc. The research motivation for this topic, therefore, is to conduct a comparative analysis of different GBM algorithms and deep sequence models for wireless multi-step internet traffic forecasting. The study aims to address the research gap in this area and provide insights into the most effective ways to use gradient boosting and deep sequence algorithms for traffic prediction task. Ultimately, this research can contribute to the development of more accurate and efficient wireless internet traffic forecasting models, which can have practical applications in telecommunications network management and resource allocation.

This chapter is organized as follows. Section 2.2 describes the literature review of current traffic prediction using machine learning models. Section 2.3 presents the methodology, including dataset description, machine learning models explanation, anomaly identification process, and experiment details. Section 2.4 summarizes different machine learning methods' performance for single-step and multi-step prediction and draws a comparative picture among prediction models with and without outliers in the dataset. Finally, section 2.5 concludes our paper and sheds light on future research directions.

2.2 Literature Review

Currently, machine learning models have been extensively using to predict complex real IP traffic. S. Fischer et al. [29] proposed a traffic prediction system called DEEPFLOW that processes the ingress traffic data and produces a prediction for all traffic flows using different machine learning techniques. The prediction model includes two different categories of models, such as statistical model and neural network-based deep learning model. They specifically focused on handling traffic volatility using a non-neural VAR (Vector Autoregression) model. According to their experimental results, the average model prediction error was around 10%, which could have been improved. D. Szosta et al. [30] extensively investigated machine learning classification and regression models for optical network traffic prediction. The proposed model considered four supervised machine learning models tested on real traffic patterns collected from Seattle Internet Exchange Point. They used a total of five different datasets in their experiment to evaluate the prediction performance based on their proposed evaluation metric called Traffic Level Prediction Quality (TLPQ). The prediction models were trained on the different feature sets arranged from various window sizes and other features such as minute, day, and traffic values. Their experiment claims the outperformance of the regression model over the classification model in traffic prediction. Y. Xu et al. [31] proposed a Gaussian Process (GP) based machine learning model for real-world traffic prediction. They compared the model performance against the state-of-the-art seasonal ARIMA (SARIMA) and sinusoid superposition. The computational complexity of extracting optimal hypermeters for the prediction model has also been reduced from $O(n^3)$ to $O(n^2)$. The GP-based machine learning model shows an average prediction of 3% when predicting one-hour-ahead traffic, and it's increased to 5% when the prediction length is extended to ten hours ahead. The performance of conventional statistical models such as ARIMA, SARIMA, SARIMAX, and Holt-Winter for real-world ISP traffic prediction has been studied in [32]. They also discussed a training technique called rolling prediction that significantly increases the performance of the traditional statistical prediction model compared with the standard training procedure. T. P. Oliveira et al. [33] experimented with two different machine learning models for traffic prediction: multilayer perceptron and stacked autoencoder. They used a dataset collected from a private ISP in European cities. They used two hidden layers of MLP (Multi-Layer Perceptron) with 60 and 40 neurons, respectively, while they found the best result for four hidden layers of SAE with 80, 60, 60, and 40 neurons. They trained their model for 1000 epochs in both MLP and SAE (Stacked Auto Encoder), although the SAE training was divided into two-stage the unsupervised pre-training for 900 epochs and 100 epochs supervised training. Different prediction length has been investigated using their prediction model, and the error increased with longer prediction length. Their experimental results show the better performance of the simpler MLP than the SAE deep neural network. Also, the MLP has taken lesser computational resources than SAE. Y. Zang et al. [34] author proposed a traffic prediction model by combining K-means clustering, Elman-NN, and wavelet decomposition. They cluster the adjacent and correlated base stations using K-means so it can improve the overall prediction accuracy. They reconstructed traffic into high-frequency and low-frequency components using wavelet decomposition to feed their traffic prediction model. Finally, a three-layer feed-forward neural network ENN is used to train the decomposed traffic data for making the prediction. P. Cortez et al.[35] proposed three different forecasting methods to predict the volume of internet traffic in TCP/IP-based networks. They investigated both the neural network model and statistical model in their experiment. The proposed novel neural ensemble method performs better in two different time scales, 5-minute and hourly forecasting, while the Holt-Winter outperforms daily forecasting. They used the linear interpolation technique to replace the missing data. L. Miguel et al.[24] compared the performance of the Artificial Neural Network (ANN) model and a statistical model, Holt-Winter, in traffic volume forecasting. The proposed ensemble of Time Lagged Feed-Forward Network (TLFN) explicitly handled the temporal data by incorporating a short-term memory in the input layer of the ANN model. R. Alfred et al. [36] identified a few drawbacks, such as slow convergence, a long training time, and easily falling into a local minimum of Back Propagation Neural Network (BPNN) in predicting time series data. They proposed a modified version of BPNN (GABPNN), where the initial model weights and threshold were optimized using a Genetic Algorithm (GA). The model has been trained on different configurations to determine the best-performing hyper-parameters. The overall performance of the GA-based BPNN was significantly better than the BPNN. C.W. Huang et al. [37] investigated three state-art-of deep learning models for predicting mobile traffic data. The geographical and temporal properties of the time series data have been extracted using CNN and RNN models, respectively. Their experiment proposes a hybrid model combining CNN and RNN, outperforming deep and non-deep learning models. Their investigation considered various parameter settings to identify the best-performing model. W. Wang et al. [38] proposed a novel traffic prediction model called SDAPM based on a stacked denoising autoencoder prediction model (SDA). The SDAPM can extract the generic attributes from the traffic flow. Their model is fine-tuned using a different combination of related hypermeters such as the number of hidden layers, number of neurons in the hidden layer, learning rate, etc. R. Madan and P.S. Mangipudi [39] proposed an ensemble traffic prediction model combining the statistical model ARIMA (Auto Regressive Integrated Moving Averages) and the deep learning model RNN (Recurrent Neural Network). The Discrete Wavelet Transform (DWT) transformation technique has been applied to separate the linear and non-linear components from the original time series data. The linear and non-linear parts were analyzed using two different prediction models, ARIMA and RNN, respectively, and combined to predict the final value. Their proposed ensemble model shows better prediction than the individual prediction model.

The modern machine learning model for predicting internet traffic is getting attention due to its ability to handle complex non-linear real-world traffic data, which are volatile and random. Several existing works on internet traffic prediction use machine learning and deep learning models, which are compared with a conventional statistical model for performance evaluation [31][35][24]. However, we found a lack of investigation in the performance comparison of deep sequences algorithms such as RNN and their variations which have the unique ability of sequence data analysis, e.g., time-series, audio data analysis, etc. Existing works experimented with sequence models compared to other machine learning models for either single-step [31] [38] or multi-step forecasting [40] [41]. But in this proposal, we investigated the performance among different deep sequence models for single and multi-step prediction.

We found another popular machine learning technique called boosting algorithm that has shown impressive results in various forecasting tasks, including time-series forecasting. Boosting algorithm outperform traditional forecasting methods such as ARIMA and neural networks, making it an attractive option for wireless internet traffic forecasting. However, there is a lack of investigation in the performance analysis of different gradient-boosting algorithms for internet traffic forecasting. Understanding the performance of different boosting algorithms, such as XGBoost, LightGBM, and CatBoost, can help researchers and practitioners choose the most suitable algorithm for their specific needs.

Real-world internet traffic is influenced by many internal and external factors, leading to unpredictable and random characteristics. Internal factors such as service launches, traffic migration, and speed upgrades, and external factors such as new internet applications, regional economic variables, and seasonal effects can result in sudden changes in traffic patterns. As a result, accurately predicting internet traffic can be challenging. Moreover, anomalies or outliers are common in real-world internet traffic, further complicating traffic forecasting. These anomalies can occur due to issues with data collection sensors, leading to erroneous data being included in the analysis. To improve the generalization capability of prediction models, it is essential to identify and address anomalies/outliers in internet traffic before using any prediction model. Thus, it is critical to develop robust methods for identifying and mitigating anomalies in Internet traffic to improve the accuracy of traffic forecasts. But in the current literature, we found a lack of investigation on analyzing the impact of outlier data points on traffic prediction model performance. Additionally, exploring the impact of different feature subsets on the performance of machine learning algorithms has not been done in the current literature. Finally, most existing works consider a static forecast length for their prediction model.

Therefore, the research motivation is to conduct a comparative analysis of different deep sequence and boosting algorithms for single and multi-step internet traffic forecasting. Also, we plan to integrate an outlier detection and mitigation module with our prediction model to

Table 2.1: Our Core Contribution Compared to Existing Works.

Contribution	Our proposed work	Existing works
Anomaly detection and mitigation	We have integrated statistical and unsupervised anomaly detection techniques with our prediction model. Also, the anomaly mitigation module has been combined to handle them before feeding into deep learning models.	Although anomaly detection and mitigation is a fundamental step of traffic analysis, according to our literature survey, we could not find any existing works considering this step in the traffic prediction tasks.
Forecast length	We have considered both single-step and multi-step forecasting in our work.	Most of the existing works focused on single-step forecasting. Multi-step forecasting works consider a limited forecast horizon, while we consider four different forecast horizons to investigate the pattern of prediction error with prediction length.
Time-lagged feature optimization	Our feature extraction module considers different window-width for identifying the optimized time-lagged feature set based on prediction accuracy.	To the best of our knowledge, feature optimization is not used in the current traffic prediction literature.

analyze the impact of abnormal data points on model accuracy. The study aims to address existing research gaps in traffic forecasting and provide insights into the most effective ways to use deep sequence and boosting algorithms. Ultimately, this research can contribute to developing more accurate and efficient wireless internet traffic forecasting models, which can have practical applications in telecommunications network management and resource allocation.

2.3 Proposed Methodology

In this section, we discuss our methodology depicted in Fig. 2.1. We begin by describing our data in subsection 2.3.1 and perform several data preprocessing steps in subsections 2.3.2 and 2.3.3. The next subsection, 2.3.4, focuses on anomaly detection and mitigation tasks. Following that, we delve into feature extraction and time-series cross-validation in subsections 2.3.5 and 2.3.6, respectively. Subsequently, we discuss our prediction model in subsections 2.3.7 and 2.3.8.

2.3.1 Dataset Description

Real internet traffic telemetry on several high-speed interfaces has been used for this experiment. Telemetry data were collected by sampling the value of the SNMP (Simple Network Management Protocol) interface MIB (Management Information Base) counter of a core-facing interface on a provider edge router. Samples are taken at 5-minute intervals, with the bps (bit per second) value for the interval being the difference between the samples at each end of the interval times 8. This was a 40 Gbs interface, so at no point in the sampling period were there any discards. There are 8563 data samples in our dataset consisting of 29 days of complete data (288 data instances per day), while the last one-day data is incomplete. We considered only the timestamp (GMT) and traffic data from the original data file in JSON (JavaScript Object Notation) format, and all other information is discarded. Ultimately, 29 days of data were considered for developing our prediction model.

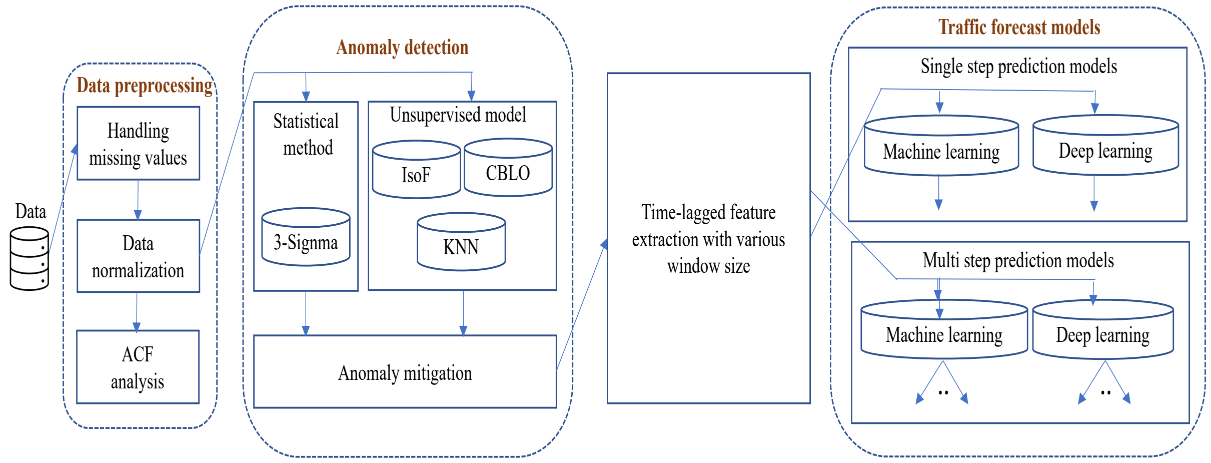


Figure 2.1: High-level framework of proposed traffic prediction model integrated with anomaly detection and mitigation.

2.3.2 Handling Missing Value

The last day data from our dataset was removed in our experiment as its network trace was collected partially on that day. There are 29 missing values in our dataset, which are replaced using the forward-filling technique. The previous valid data instance has replaced the missing value in our traffic data. There are other methods, such as linear interpolation, quadratic interpolation, replacement with mean value, etc., to handle the missing data in time series analysis. Linear interpolation assumes a linear relationship among data points and estimates a missing value connecting points in a straight line. Since our traffic data is non-linear, we found this method inappropriate for handling missing values. Also, the polynomial interpolation method seems unsuitable because we must specify the order before applying this method to replace the missing data. It replaces missing values with the smallest probable degree that passes through available data points. In the case of mean value replacement, the outliers must be treated first. The real-world traffic data may contain outlier or anomalous data points, so we do not consider the mean value to change the missing values. Finally, we found the forward-filling technique[42] useful for our experiment, mainly used for time-series data. This method implies that the generated traffic volume would have remained consistent from the moment of dropout through data collection completion rather than dropping or rising further and replaces a missing value after dropout with the most recent measurement. Additionally, it is predicated on the idea that missing data are random, i.e., the probability of missing values is not associated with factors that affect traffic volume at a particular time-stamp.

2.3.3 Autocorrelation Function (ACF) Analysis

The ACF is widely used to assess the data in time series analysis and prediction. The ACF plot visually displays the degree of a correlation between an observation in a time series and observations made at earlier time steps. The underlying time series must be stationary for ACF to operate. ACF plot identifies the correlation between a time series data point and previous data points, called lag (l), of the same time series. For a given set of lags, the ACF evaluates the correlations among samples in a time series. The ACF for time series x is given by:

$$ACF = Corr(x_t, x_{t-l}), \text{ where } l = 1, 2, 3.. \tag{2.1}$$

We can determine the stationarity and randomness of time-series data using an ACF plot. Also, the time-series seasonality and trend can be identified based on ACF plot information. Each bar in an ACF plot indicates the strength and direction of the correlation among data points. The bar value should be near zero for all lags for random data, which we can also consider as white noise. Non-random time-series data should have at least one lag value with a strong correlation. We can use time-lagged features in building a prediction model for non-random time-series data. Our traffic data has many lags with strong correlations which indicates data is non-random. Therefore, we considered previous timestamp features for our regression models to predict the following values. Based on our ACF plot, we now analyze two important time series characteristics: seasonality and trend. Smaller lags frequently exhibit strong associations when trends are prevalent in the time series because samples closer in the period tend to have comparable values. As the lags lengthen, the correlations gradually diminish. When periodic patterns are evident, multiples of the frequent recurrence have stronger autocorrelations than other lags. The ACF plot blends both characteristics when a time series contains a trend and seasonality. We can conclude that our traffic data has trend and seasonality based on the correlation value for different lags. Since the correlation is higher for smaller lag and decreases for larger lag, there is a trend in our traffic. Also, there is a repetition of higher correlation values for every 288 lag values for our traffic dataset in which samples are collected every five minutes interval. In other words, we find a daily periodic pattern in our traffic data. So, ACF plot analysis gives us several important pieces of information about our traffic data, such as our traffic data is non-random, has a trend, and daily seasonality. Also, it helps to decide that we can better model our regressor model based on the time-lagged feature. Hence, we investigated several lagged feature set for single and multi-step forecasting model for optimum input settings.

2.3.4 Anomaly Detection and Mitigation

In this subsection, we discuss our methods to detect those data points that are deviated from most of the data instances in the dataset. Those unexpected data points are called anomalies or outliers. Many external and internal factors make real-world IP internet traffic susceptible. These issues disrupt normal traffic flow, which must be discovered and corrected so that a machine-learning model may improve its generalization capability. There are different types of anomalies: point anomalies, contextual anomalies, and collective anomalies. In our research, we considered only the point anomalies, i.e., those data points which are far away from the

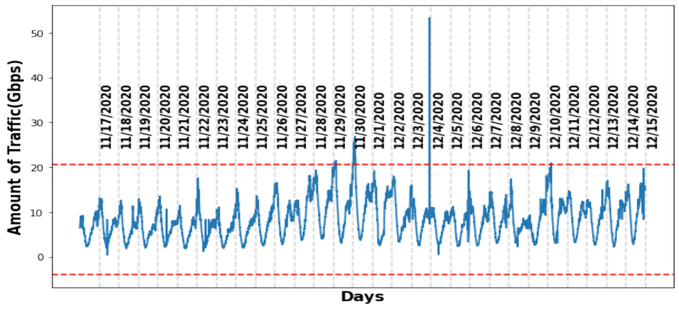


Figure 2.2: Outliers identified by Three-Sigma rule.

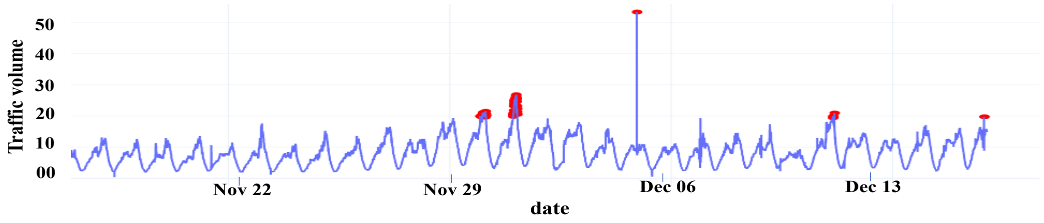


Figure 2.3: Anomaly detected by ISoF.

general distribution of the data. There are mainly three main categories of anomaly detection methods: statistical profiling, supervised learning, and unsupervised modeling.

In statistics, the three-sigma rule is a statistical calculation that defines the upper limit and lower limit for point anomaly detection based on three standard deviations from the mean value of the dataset. The data points outside the boundary defined by the three-sigma rule are considered outliers in the dataset. Therefore, it is necessary to calculate the absolute difference for each data point and their average; if the difference is within the three standard deviations of the dataset, it is considered a statistically valid data point. According to the three-sigma rule, the probability of a data point X lying within three standard deviations of the mean is 99.73%. In our experiment, we applied this empirical method for identifying point anomalies from our dataset before providing them to our machine learning models. Using the Three-sigma rule assumes that the probability distribution of the data follows a normal distribution. It is true that data that follows a normal distribution responds to the three-sigma rule the best. However, even with non-normally distributed variables, the Bienaymé-Chebyshev inequality, sometimes known as Chebyshev's inequality, states that at least 88.8% of cases should fall inside correctly computed three-sigma ranges. For a wide range of distinct probabilistic, Chebyshev's inequality states that a minimum of just 75 percent of observations must reside within two standard deviations of the mean and 88.89 percent between three standard deviations[43, 44]. The valid data points are surrounded by the upper and lower horizontal red lines in Fig. 2.2 according to the three-sigma rule.

We applied three different unsupervised machine learning models from the clustering category, although there are other categories: model-based, graphical, distance-based, and supervised learning. Isolation Forest (ISoF), K-Nearest Neighbors (KNN), and Clustering-Based Local Outliers (CBLO) are selected for our experiment. The clustering-based approach grouped all data points into several clusters, and the data instances that do not belong to these clusters

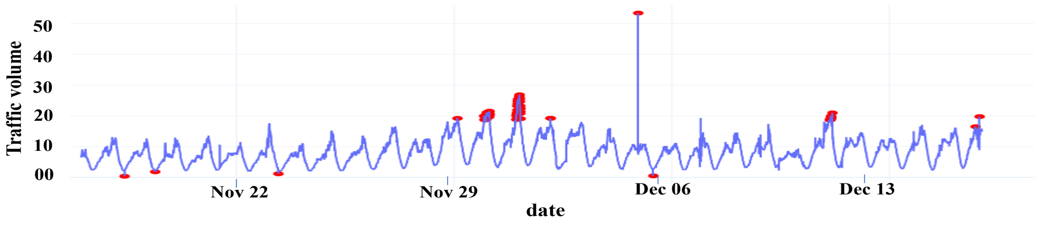


Figure 2.4: Anomaly detected by KNN.

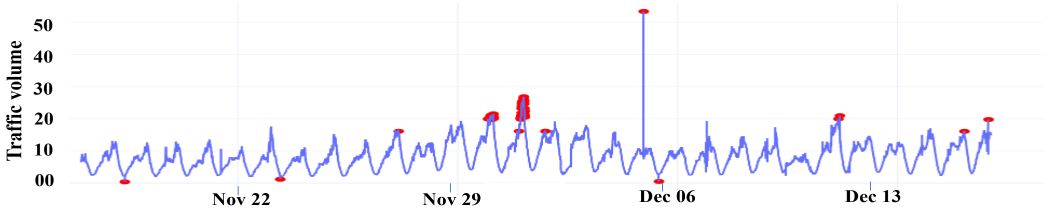


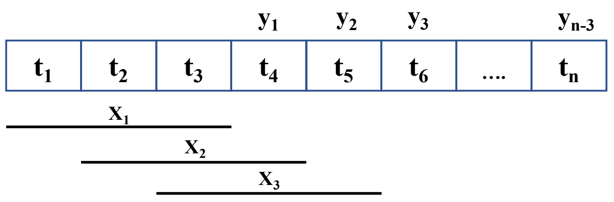
Figure 2.5: Anomaly detected by CBLO.

are called outliers. Generally, it is challenging to define and identify the outliers from the datasets, and that is why we compared the performance of different models to evaluate their corresponding results. For the sake of the analysis, the contamination percentage in the data is set at 1%. Fig. 2.3, Fig. 2.4, and Fig. 2.5 depicted the anomalies or outliers identified by ISoF, KNN, and CBLO respectively. After studying the associated temporal information, we discovered that the data points recommended by the Three-Sigma rule are more likely to constitute the anomaly. And backward filling has been applied for correcting outliers in our experiment in which the next valid data point replaces the outlier.

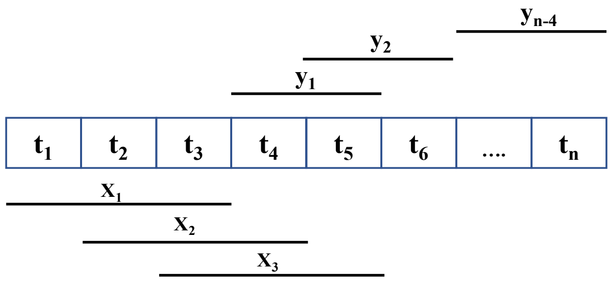
2.3.5 Time-Lagged Feature Extraction

Time series data need to be expressed in the proper format for supervised learning. Generally, the time-series data consists of several tuples (time, value), which is inappropriate for feeding them into the machine-learning model. So, we restructured our original time series data using the sliding window technique. The sliding window technique is illustrated in Fig. 2.6. Every time series data instance in this figure is represented by t_i where i denotes the index of the data. For example, in Fig. 2.6(a), we considered the first three data points as a feature set X_1 to predict the fourth data point denoted by y_1 as the target. The number of previous time steps used to predict the next time-step is called window width or lag size. This process continues until we consider the last value in our training set for the prediction. It represents the sliding window method for single-step prediction, while for multi-step prediction, the target values must be more than one.

We consider different window widths for single-step prediction to predict our next step. We performed a grid search to find out the best window width for single-step data conversion. A set of five different window widths as {6, 9, 12, 15, 18} is considered for our experiment, which indicates five different data conversion configurations for single-step prediction. The initial data configuration consists of six features to predict the target, and it continues to eighteen



(a) Feature extraction for single-step prediction



(b) Feature extraction for multi-step prediction

Figure 2.6: Time-lagged feature extraction.

features for identifying the optimized window width for the sliding technique.

In the case of multi-step prediction, previous time steps are used to predict the next two or more steps, known as forecast horizon. Fig. 2.6(b) illustrates the process of feature extraction for multi-step forecasting. We considered three different: six steps, nine steps, and twelve steps ahead forecast in our experiment. In addition, we explored various feature sets based on forecast length to find optimum inputs for the model. For example, we searched through a window width set of {6, 9, 12, 15, 18} for six-step prediction to identify the optimum number of inputs for six-step forecast models. Therefore, our proposed prediction model's performance has been evaluated based on multiple combinations of (features and targets). We provided five different varieties of feature and target variables such as {(6, 6), (9, 6), (12, 6), (15, 6), (18, 6)} to our prediction model to find the best input set for six step prediction. The exact process has been followed for other forecasting lengths in our experiment. For example, a total of four and three different (features, input) combinations have been considered for nine and twelve steps forecasting models. We analyze the performance of each combination for every particular forecast horizon and identify the best input settings for the corresponding model.

2.3.6 Time-series Cross-validation

A cross-validation method was used in our experiment to evaluate the performance of our traffic prediction models. Different ways exist to split the dataset into several folds to train and test the classification/prediction models. K-fold-cross validation splits the dataset into K almost similar size folds and all folds except one are used to train the model while the remaining fold is kept for testing[45]. The process continues until all the model is tested on every fold, and the final performance of the model is measured as the average performance on each fold. Since most of the cross-validation techniques in machine learning select folds randomly, we need to follow a different approach in splitting and selecting folds from time-series data to keep the

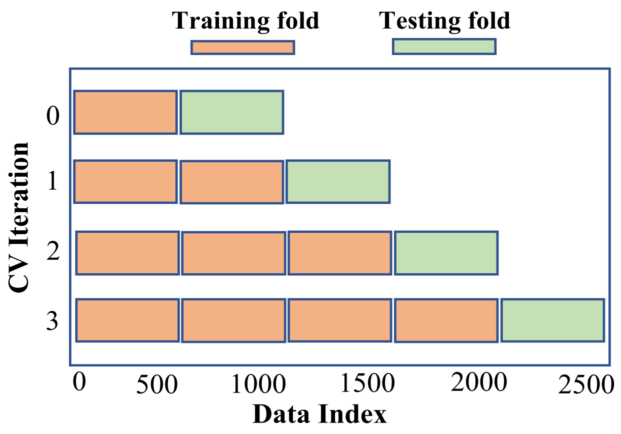


Figure 2.7: Rolling-based cross validation.

temporal relation among folds. Our experiment used a rolling basis cross-validation technique where training starts with one-fold and finishes by predicting the next fold. In the next step, the test fold from the previous step is included in the training process, and the subsequent fold is for the testing. The final performance of the model is the average of the prediction on each fold. We split our 29 days of original time series data into two smaller datasets: a training dataset of 21 days and a holdout dataset of 8 days, which is 70% and 30% of the dataset, respectively. The training dataset was cross-validated using the TimeSeriesSplit method from scikit-learn [46] for model training, while the holdout dataset was used for testing.

2.3.7 Boosting Algorithms for Traffic Prediction

This subsection briefly explained machine learning models used for our traffic task. We also discussed the multi-output strategy for multi-step prediction using boosting algorithms. A total of five different boosting algorithms have considered for our experiment as below:

- Gradient Boosting Regressor (GBR): Gradient Boosting permits for the optimization of random differentiable loss functions and constructs an additive model in a forward stage-wise process [46]. A regression tree in each stage fits the non-positive gradient of the provided loss function.

- Extreme Gradient Boosting (XGB): It is an implementation of the gradient boosting algorithm initially developed in [47]. This model can be used for both classification and regression problems, and it is comparatively fast in computation in comparison with the other implementation of gradient boosting.

- Light Gradient Boosting Machine (LGB): LightGBM is another implementation of the gradient boosting algorithm proposed in [48]. This algorithm minimizes the limitation of the histogram-based algorithm by introducing two novel techniques: Gradient-based One Side Sampling and Exclusive Feature Bundling (EFB).

- CatBoost Regressor (CBR): This is a binary-tree based implementation of gradient boosting. The catBoost technique addresses a very general implementation problem of gradient boosting and tries to solve the issue by proposing an ordering principle. All of the gradient boosting implementation relies on the targets of all training samples after several steps [49].

- Stochastic Gradient Descent (SGD): It is a simple but efficient machine learning model used for both classification and regression [50]. The basic difference between gradient descent and SGD is the number of samples taken to compute the derivatives. SGD randomly select one data sample in each iteration for calculating the gradient, which significantly reduces the number of computations in comparison with the gradient descent.

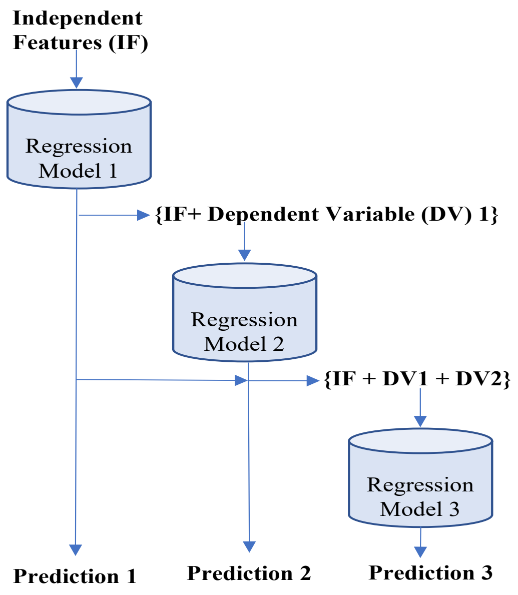


Figure 2.8: Chain multi-output regression model.

Several machine learning models, such as linear regression, decision tree regressor, random forest regressor, etc., can predict multiple outputs. But not all machine learning regression models, such as the support vector regression model, directly allow various outputs. However, there are different ways of modifying these models for multi-output regression. One popular method is to divide the multi-output regression problem into multiple individual regression sub-problem. In that case, the machine learning model predicts each particular step separately based on the same input data. It is sometimes called a direct multi-output regression model where outputs are assumed independent of each other. There is an extension of this approach where individual model outputs are connected to each other. A sequence of regressors can be employed to solve multi-output regression problems, where each regressor in the sequence learns the relationship between the input variables and a specific output variable. The first regressor learns the relation between the inputs and the first output, and the subsequent regressors use the inputs and the outputs predicted by the previous regressors to learn the remaining output variables. The final regression model then uses all the input variables and the predicted outputs from the previous regressors to predict the final output. This approach is known as chained multi-output regression[51], as depicted in Fig. 2.8. We applied the second strategy for our multi-step prediction problem as there is a correlation between traffic volume for two consecutive timestamps. However, one major drawback of this approach is that the order in which the output variables are arranged in the sequence can have a significant impact on the accuracy of the predicted outputs.

2.3.8 Deep Sequence Algorithms for Traffic Prediction

Five deep sequence models have been chosen for our traffic prediction task. First, we started our experiment with a Recurrent Neural Network (RNN)[52] as it is the basic deep learning model capable of processing sequence data such as time series. After that, the experiment was extended by adding two more models, Long Short-Term Memory (LSTM)[53] and Gated Recurrent Unit (GRU)[54] because they can retain the information from a longer sequence compared to the RNN. Finally, we included another model architecture capable of predicting more than one step at a time, called the sequence-to-sequence model. LSTM Encoder-Decoder[55] model has the limitation of extracting strong contextual information from long sequential data, which we tried to solve by incorporating an attention layer in the encoder-decoder model. This subsection briefly explained deep learning techniques for single- and multi-step prediction.

- Recurrent Neural Network (RNN): RNN model is specifically designed to handle sequential data such as text mining, audio classification, language modeling, time series analysis, etc. The RNN uses the current and previous sequence information to produce the current output at every step. Thus, the model learns about all previous data points in the series at the last step. However, there is a short-term memory problem in the RNN model training process which is caused due to the vanishing gradient issues.

- Gated Recurrent Unit (GRU): The GRU has been proposed to solve the short-time memory problem in the RNN model. The gate concept is used in this model to control the flow of information between two consecutive cells. The GRU model has an update gate that decides whether to transfer the previous cell output to the next cell. A gate is a mathematical unit that can measure the importance of the information and determine whether it should be stored. There are two gates called the update gate and reset gate GRU model, which works on the update of cell state.

- Long Short-Term Memory (LSTM): The purpose of the LSTM is similar to the GRU model. There are two additional gates called forget and output gates, along with the update and reset gates. LSTM has more control in transferring information among cells of the network. LSTM network is popular for processing time-series data to classify and make predictions. It alleviates the inherent vanishing gradient problem of the traditional RNN model and performs comparatively better.

- LSTM Encoder-Decoder (LSTM_Seq2Seq): This model can predict an output sequence from an input sequence known as the sequence-to-sequence model. It consists of two recurrent neural networks; one is called an encoder, and the other is a decoder. The encoder converts the input sequence into a fixed-length context vector and passes it to the decoder. The decoder uses the context vector and the final state of the encoder as the input and returns a sequence of output. We used this model for both single-step and multi-step prediction.

- LSTM Encoder-Decoder with Attention Layer (LSTM_Seq2Seq_ATN): This model can also predict an output sequence from an input sequence, known as the sequence-to-sequence model. However, the conventional encoder-decoder model has a drawback, which is solved by adding an extra layer called the attention layer, first proposed in [56]. The encoder-decoder model cannot extract the strong contextual relationship from long sequence data, which affects the model performance and decreases accuracy. On the other hand, the extra attention layer in the encoder-decoder model can identify the significance of sequence data.

2.3.9 Evaluation Metrics

The performance of our traffic forecasting models was estimated using Mean Absolute Percentage Error (MAPE). The performance metric identifies the variation of the anticipated result from the original data. MAPE error, for example, is the average percentage of the variance between the actual and predicted values. As a result, we can formally define our performance metric MAPE and Mean Accuracy (MA) for single-step forecasting as follow where p_i and o_i are predicted and original values respectively, and n is the total number of test instance:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{p_i - o_i}{o_i}\right| \times 100 \tag{2.2}$$

$$MA = (100 - MAPE)\% \tag{2.3}$$

The MAPE formula for multi-step prediction is similar to the formula for single-step prediction, but it considers all the predicted values over the forecast horizon. Assuming actual traffic volume values o_t and predicted values p_{t+h} for h steps ahead, the formula for $MAPE_h$ for multi-step prediction is:

$$MAPE_h = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{o_t - \hat{p}_{t+h}}{o_t}\right| \times 100 \tag{2.4}$$

We can define MA_h for multi-step forecasting as the average MA_i of each individual step prediction, i, in forecast horizon h.

$$MAPE_{avg} = \frac{1}{h}\sum_{i=1}^{h} MAPE_i \tag{2.5}$$

$$MA_h = (100 - MAPE_{avg})\% \tag{2.6}$$

Table 2.2: Single-Step Traffic Prediction Machine Learning Model Performance Summary.

Model	With Outlier					Without Outlier				
	6	9	12	15	18	6	9	12	15	18
GBR	**7.47**	7.74	7.57	7.60	7.76	**5.20**	5.31	5.24	5.26	5.25
XGB	7.65	7.69	**7.47**	7.59	7.60	5.16	5.20	5.19	**5.15**	5.17
LGB	8.51	8.47	**8.47**	8.47	8.53	5.11	**5.09**	5.13	5.14	5.10
CBR	**7.56**	7.58	7.64	7.78	8.12	5.10	**5.08**	5.20	5.32	5.44
SGD	12.80	10.51	**10.44**	11.13	12.16	**6.10**	7.27	8.01	8.50	8.23

2.4 Analysis of Experimental Results

The first 21 days of IP traffic, that is 70% of our total data were utilized for training our machine learning models, while the rest 30% data , i.e., eight days were used for testing. Our investigation is divided into two phases: I) machine learning model performance evaluation for single-step traffic prediction, and II) machine learning model performance evaluation for multi-step traffic prediction, as described in subsections in subsection 2.4.2, and 2.4.3, respectively. Before that, we discussed the impact of outlier mitigation on data variability in subsection 2.4.1.

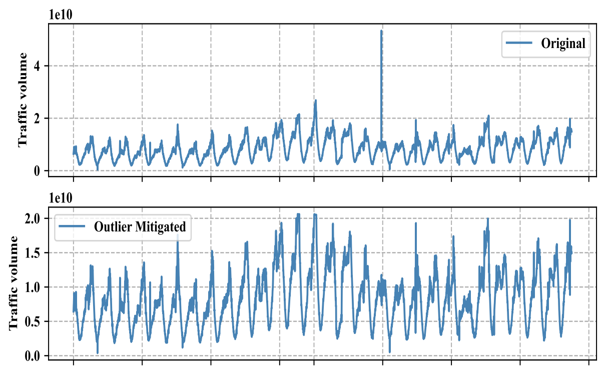


Figure 2.9: Original traffic vs. outlier mitigated traffic.

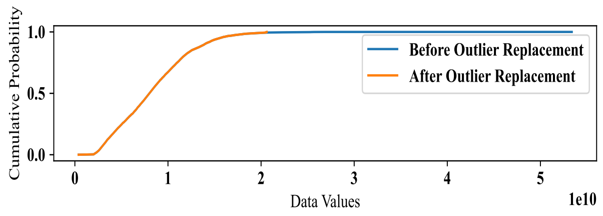


Figure 2.10: Empirical Cumulative Distribution Function (ECDF) plot of our traffic data.

Table 2.3: Statistical Analysis Before and After Outlier Mitigation.

Data Samples Statistics

Statistics	***Before***	***After***
Mean	8364742790	8344127355
SD	4096690529	4013117118

Statistical Test Results Summary

Test	***Test statistic***	***p-value***
Two-sample t-test	0.328505869	0.742533327
Wilcoxon Rank-sum test	0.015409873	0.987705187
Kolmogorov-Smirnov test	0.005148467	0.999891457

2.4.1 Outlier Mitigation Impact Analysis

The outlier replacement on our dataset, resulting in a change in the mean and standard deviation of the data as shown in Table 2.3. In addition, we depict the actual traffic and outlier mitigated traffic to show their variability in Fig. 2.9. To assess whether this change is statistically significant, we conducted three hypothesis tests: a two-sample t-test, a Wilcoxon rank-sum test, and a Kolmogorov-Smirnov test and the result has been summarized in Table 2.3. Before performing these tests, we compared the distribution of our dataset before and after outlier treatment depicted in Fig. 2.10.

The ECDF plot capture a clear visual difference in the range of the two datasets before and after outlier replacement. The ECDF plot suggests that the range of values is narrower after outlier replacement, indicating a reduction in variability. While the removal of the outlier reduced the range of values, it may not have had a substantial impact on the overall variability of the data. To determine this fact, we performed three different statistical tests to examine the overall shape of the distributions, rather than just the range of values. It is possible for the standard deviation of the two groups to be similar, but for the range of values to differ due to the presence or absence of an outlier.

The two-sample t-test is used to compare the means of two independent samples. In this case, the two samples are the data before outlier replacement and the data after outlier replacement. The test statistic is the t-statistic, which measures the difference between the means relative to the variability within the samples. The p-value is the probability of obtaining a test statistic as extreme as the one observed, assuming that the null hypothesis (no difference in means) is true. The p-value for the two-sample t-test is 0.7425, which is greater than the conventional threshold of 0.05 for statistical significance. This suggests that there is insufficient evidence to reject the null hypothesis of no difference in means, and that the difference between the means is likely due to chance. The Wilcoxon rank-sum test, also known as the Mann-Whitney U test, is a non-parametric test that compares the medians of two independent samples. Like the t-test, it can be used to test for a difference in location (i.e., central tendency) between two groups. The p-value for the Wilcoxon rank-sum test is 0.9877, which is much greater than 0.05. This also suggests that there is insufficient evidence to reject the null hypothesis of no difference in medians. The Kolmogorov-Smirnov test is a non-parametric test that compares the distributions of two samples. It is based on the maximum difference between the cumulative distribution functions of the two samples. The p-value for the Kolmogorov-Smirnov test is 0.9999, which is very close to 1. This also suggests that there is insufficient evidence to reject the null hypothesis of no difference in distributions. In summary, based on the results of these three tests, there is no significant evidence to suggest that the outlier replacement has had a significant effect on the central tendency or distribution of the data. Based on the statistical tests we performed, it appears that the variability of our traffic data has decreased after the outlier replacement. However, the difference in standard deviation is not statistically significant, as indicated by the p-values of the tests. The fact that these tests did not find a statistically significant difference in the standard deviation of the two groups suggests that the reduction in variability after outlier replacement is not likely to be due to chance. Therefore, it may be appropriate to interpret the results as follows:

- The ECDF plot suggests that the range of values is narrower after outlier replacement, indicating a reduction in variability.
- The statistical tests indicate that there is no significant difference in the standard deviation of the two groups, suggesting that the difference in range is not likely to be due to chance.
- Taken together, these results suggest that the removal of the outlier have resulted in a reduction in the range of values, even though the standard deviation of the two groups is similar.

Based on the results, it appears that outlier replacement has some impact on reducing the variability of in data, but the magnitude of this reduction may not be large enough to be statistically significant.

2.4.2 Single-Step Prediction Result Analysis Using Boosting and Deep Sequence Algorithms

We applied total of five machine learning models such as XGB, LGB, CBR, SGD, GBR for our traffic prediction task. We used five-fold cross-validation to train our models. The performance evaluation metrics of our experimental model are summarized in Table 2.2. Two different versions of each model were investigated based on the data with and without outliers. We used a total of five different feature sets of equivalent traffic lengths of 30 minutes, 45 minutes, 60 minutes, 75 minutes, and 90 minutes for training our model to identify the optimum number of inputs for our prediction model. The best results for the individual model are marked bold and underlined in the table.

For the XGB model, we achieved the best prediction using 12 and 15 inputs respectively for models with and without outlier data. The best average deviation between actual and predicted traffic is 7.47% and 5.15% with and without outliers, respectively, which indicated more than 30% improvement in traffic prediction after adjusting outliers. In the LGB model, the minimum prediction error between actual and predicted traffic is 5.09% for 9 outlier-adjusted input variables. The prediction improvement is around 40% when compared with the best forecasting error of 8.51% for the prediction model having outliers in the training data. For other input variables 9, 12, 15, and 18, we have seen a similar amount, around 40% (8.51% to 5.11%, 8.47% to 5.13%, 8.47% to 5.14%, and 8.53% to 5.10% respectively) of prediction enhancements after adjusting the anomalies in the data. For SGD, we found the best prediction result with MAPE of 6.10% using an input length of 6, and the result improved by more than 41% after removing the outliers compared with 10.44% error in prediction with outlier data. The other input variables show a lower gap between actual and predicted traffic for outlier-adjusted data. The GBR model showed a minimum deviation of 5.20% when trained using thirty minutes of traffic data without the outlier. After mitigating abnormal data points, our experimental results depict more than 30% better predictions for all input variables. In the CBR model, the minimum prediction error of 5.08% is achieved using 9 input variables. The traffic prediction improvement for the CBR model is more than 30% compared with the minimum error of 7.56% in the case of outlier data. Our experimental results show overall performance improvement for all considering machine learning models after adjusting the abnormal traffic. From our experimental result, we can conclude, the outlier mitigation in the dataset before training the prediction model can improve the prediction by an average of 30% more accuracy. Our experimental result shows that outlier mitigation in the dataset before training the prediction model can improve the prediction by an average of 30% more accuracy.

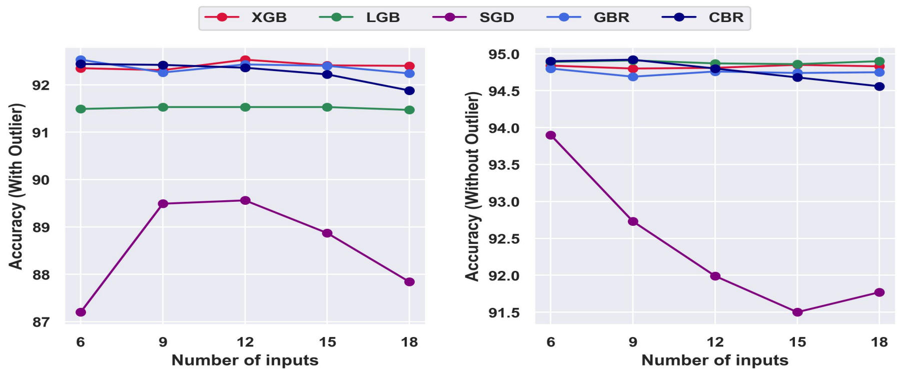


Figure 2.11: Machine learning model accuracy comparison for different input length.

The comparative analysis of the model mean accuracy based on different input lengths is depicted in Fig. 2.11. Our experimental results showed that the performance of SGD is more sensitive to the input lengths. The SGD model accuracy rises from input 6 to 9 then plummets from 15 to 18 when trained with outlier data. But in the other case, the accuracy decreases with the increased input length. There is a variation in the model accuracy with the input length for other models, but the magnitude is much lower than the SGD. However, many internal and external factors can affect the regular traffic pattern in the real world. Since machine learning-based traffic prediction algorithms learn the general pattern in the dataset and predict accordingly, it is essential to handle the outliers before providing them to learn. Otherwise, there is a chance of learning from abnormal traffic patterns, affecting the prediction result. Our experimental results also showed that outliers in the data make the model performance poor than the clean data.

After experimenting with boosting algorithms for single-step traffic prediction, we applied several deep sequences models such as RNN and their variants, including LSTM, LSTM_Seq2Seq, LSTM_Seq2Seq_ATN, and GRU to evaluate the comparative performance in traffic prediction.

Table 2.4: Single-Step Traffic Prediction Deep Learning Model Performance Summary.

Model	With Outlier	Without Outlier
RNN	7.51%	5.28%
LSTM	5.03 %	3.80 %
GRU	6.41 %	5.28%
LSTM_Seq2Seq	**3.94**%	**3.51**%
LSTM_Seq2Seq_ATN	3.95%	3.55%

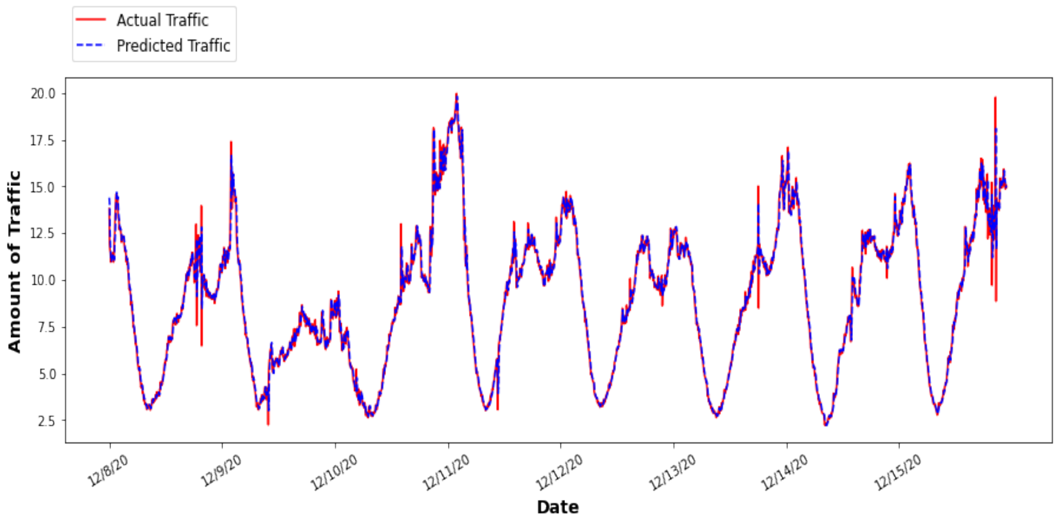


Figure 2.12: Actual vs. predicted traffic by proposed LSTM_Seq2Seq model.

All our model training continued for 100 epochs with a batch size of 16. The performance evaluation metrics of our experimental model are summarized in Table 2.4. Two different versions of each model were investigated to identify the impact of anomaly or outlier detection in traffic prediction. Our results show improved performance for prediction models trained on outlier-adjusted data. For example, the average deviation between actual and predicted traffic by the RNN model is 7.51% and 5.28% with and without outliers, respectively, which improved traffic prediction by more than 29% after adjusting outliers. In the LSTM model, the average prediction error between actual and predicted traffic is reduced from 5.03% to 3.80%, i.e., more than 24% after handling the outlier. Similarly, we noticed an error reduction of more than 11% (3.94% to 3.51%) and 10% (3.95% to 3.55%) due to the outlier adjustment in LSTM_Seq2Seq and LSTM_Seq2Seq_ATN, respectively. The deviation between actual and predicted traffic is reduced by more than 19% from 6.41% to 5.28% in the GRU model. According to our experimental result, LSTM_Seq2Seq is the best prediction model with a minimum prediction error of 3.94% with outliers in the data and 3.51% without outliers. A graphical representation of real traffic and predicted traffic using the LSTM_Seq2Seq model after adjusting the outliers are shown in Fig. 2.12. Our experimental results show overall performance improvement for all considering deep sequence models after adjusting the abnormal traffic. However, in the real world, many internal and external factors can affect the regular traffic pattern. Since machine learning-based traffic prediction algorithms learn the general pattern in the dataset and predict accordingly, it is essential to handle the outliers before providing them to learn. Otherwise, there is a chance of learning from abnormal traffic patterns, which can affect the prediction result. Our experimental results also showed that outliers in the data make the model performance poor than the clean data.

2.4.3 Multi-Step Prediction Result Analysis Using Boosting Algorithm and Deep Sequence Model

Our multi-step forecasting experiment considered different forecast lengths of six, nine, and twelve steps. We evaluated and compared the performance of several models from the gradient boosting category, such as Gradient Boosting Regressor (GBR), XGBoost (XGB), LightGBM (LGB), CatBoost (CBR), and the gradient descent category, such as Stochastic Gradient Descent (SGD). Tables 2.5, 2.6, and 2.7 summarize the performance for six-step, nine-step, and twelve-step forecasting, respectively. We report the mean absolute percentage error between actual and predicted traffic for each step in multi-step forecasting, for different input lengths. The best performance for the corresponding model is highlighted in bold and underlined.

Now, we analyze the detailed prediction results for multi-step forecasting. In the case of six-step forecasting, our proposed GBR model performs better for each step. The minimum prediction error we achieved with eighteen input data where the last prediction accuracy was around 93% with an average error of 7.03%. The individual step prediction error increases with larger forecast steps as the prediction error in the current steps accumulates on the next step prediction. As a result, multi-step forecasting is more challenging than single-step forecasting. In the case of nine and twelve steps forecasting, our proposed GBR model performs better with a minimum average gap of 8.88% and 10.60%, respectively, between actual and predicted traffic for the last step. Next, we investigated the performance of the XGB model and noticed a general performance improvement compared to GBR. For example, in the case of six-step forecasting, our proposed XGB model provided the lowest average gap of 6.99% with eighteen features, which is better than the GBR model's performance and standard XGB performance. XGB and GBR both follow the same principle of gradient boosting. But XGB can control overfitting by formalizing a more regularized model, which results in better prediction accuracy. As a result, we noticed an outperformance of the XGB model over GBR. Compared to nine and twelve steps forecasting, our proposed XGB performs better than GBR for twelve-step prediction only. But GBR took approximately 1.5 times more execution time than XGB for nine-step forecasting, as depicted in Fig. 2.13. Moreover, the execution time for the six and twelve steps forecasting model, GBR execution time, is 1.8 and 1.7 times greater than XGB. XGB takes lesser time for execution than GBR because XGB reduces the size of the search space for possible splitting by considering the distribution of the feature across all data samples in the tree leaf. Overall, XGB's performance is better than GBR evaluating prediction accuracy and execution time.

Now, we analyze the performance of another model of gradient boosting category called LightGBM (LGB). Compared with XGB and GBR prediction performance, our proposed LGB gave the minimum average prediction error of 6.30% and 7.18% for nine and twelve-step prediction, respectively. In contrast, for nine and twelve steps, the best performing GBR and XGB

Table 2.5: Six-Step Forecasting MAPE (%) Value For The Standard And Proposed Machine Learning Models. Each Value Represents An Average Gap Between Actual and Predicted Traffic Volume For Each Individual Forecast Step.

Model		**GBR**		**XGB**		**LGB**		**CBR**		**SGD**	
		With Outlier	Without Outlier	With Outlier	Without Outlier	With Outlier	Without Outlier	With Outlier	Without Outlier	With Outlier	Without Outlier
6	Step1	3.80	3.76	3.78	3.75	3.68	3.63	3.65	3.61	**6.41**	5.23
	Step2	4.95	4.84	4.85	4.85	4.71	4.63	4.55	4.54	**7.45**	6.46
	Step3	5.93	5.85	5.88	5.84	5.53	5.44	5.44	5.38	**8.62**	7.57
	Step4	6.85	6.78	6.84	6.80	6.26	6.22	6.21	6.16	**9.72**	8.71
	Step5	7.74	7.58	7.76	7.63	7.02	6.96	6.90	6.91	**10.85**	9.81
	Step6	8.66	8.46	8.68	8.62	7.77	7.76	7.70	7.68	**11.80**	10.92
9	Step1	3.79	3.70	3.75	3.71	3.67	3.60	3.64	3.60	7.36	5.61
	Step2	4.82	4.73	4.75	4.71	4.61	4.52	4.55	4.46	8.54	6.71
	Step3	5.70	5.54	5.71	5.63	5.44	5.31	5.27	5.24	9.76	7.77
	Step4	6.60	6.42	6.59	6.51	6.08	5.98	5.99	5.94	10.75	8.83
	Step5	7.31	7.22	7.34	7.21	6.69	6.55	6.56	6.51	11.79	9.92
	Step6	8.15	7.99	8.14	8.10	7.31	7.25	7.17	7.19	12.99	10.96
12	Step1	3.92	3.68	3.78	3.69	3.72	3.58	3.60	3.55	8.40	5.38
	Step2	4.73	4.58	4.62	4.55	4.59	4.43	4.43	4.43	9.49	6.39
	Step3	5.51	5.36	5.57	5.37	5.31	5.21	5.18	5.12	10.67	7.35
	Step4	6.29	6.13	6.24	6.13	5.94	5.89	5.74	5.74	11.65	8.31
	Step5	6.85	6.75	7.00	6.79	6.52	6.45	6.35	6.33	12.86	9.22
	Step6	7.63	7.45	7.60	7.50	7.08	7.01	6.91	6.92	13.92	10.10
15	Step1	3.69	3.59	3.71	3.60	3.68	3.60	**3.66**	**3.62**	9.15	4.87
	Step2	4.53	4.46	4.53	4.43	4.57	4.43	**4.44**	**4.39**	10.82	5.73
	Step3	5.26	5.19	5.26	5.16	5.26	5.14	**5.14**	**5.09**	11.39	6.59
	Step4	5.89	5.81	5.91	5.76	5.87	5.79	**5.66**	**5.68**	12.24	7.36
	Step5	6.57	6.38	6.59	6.46	6.45	6.41	**6.27**	**6.21**	13.31	8.15
	Step6	7.23	7.06	7.16	7.12	7.03	7.01	**6.76**	**6.85**	14.71	8.90
18	Step1	**3.68**	**3.55**	**3.67**	**3.55**	**3.60**	**3.53**	3.64	3.62	9.48	**4.36**
	Step2	**4.45**	**4.40**	**4.46**	**4.39**	**4.54**	**4.43**	4.48	4.42	10.50	**5.12**
	Step3	**5.19**	**5.08**	**5.17**	**5.08**	**5.21**	**5.09**	5.12	5.07	11.50	**5.87**
	Step4	**5.81**	**5.73**	**5.82**	**5.72**	**5.81**	**5.76**	5.71	5.67	12.72	**6.59**
	Step5	**6.51**	**6.38**	**6.51**	**6.39**	**6.40**	**6.35**	6.35	6.31	13.34	**7.25**
	Step6	**7.08**	**7.03**	**7.10**	**6.99**	**7.04**	**7.02**	6.93	6.96	14.31	**7.94**

Table 2.6: Nine-Step Forecasting MAPE (%) Value For The Standard And Proposed Machine Learning Models. Each Value Represents An Average Gap Between Actual and Predicted Traffic Volume For Each Individual Forecast Step.

Model		**GBR**		**XGB**		**LGB**		**CAB**		**SGD**	
		With Outlier	Without Outlier	With Outlier	Without Outlier	With Outlier	Without Outlier	With Outlier	Without Outlier	With Outlier	Without Outlier
	Step1	3.78	3.71	3.76	3.69	3.66	3.60	3.67	3.60	**7.40**	5.64
	Step2	4.83	4.67	4.76	4.74	4.61	4.54	4.60	4.52	**8.56**	6.73
	Step3	5.73	5.55	5.69	5.59	5.42	5.33	5.28	5.25	**9.66**	7.80
	Step4	6.69	6.40	6.58	6.42	6.07	6.03	5.98	5.91	**10.74**	8.90
9	Step5	7.35	7.20	7.37	7.15	6.65	6.56	6.58	6.53	**11.90**	9.90
	Step6	8.12	7.94	8.10	8.03	7.29	7.24	7.23	7.20	**13.22**	10.96
	Step7	8.97	8.77	9.01	8.79	7.98	7.92	7.96	7.88	**14.15**	11.98
	Step8	9.66	9.43	9.65	9.48	8.55	8.51	8.53	8.50	**15.49**	12.91
	Step9	10.41	10.15	10.49	10.17	9.17	9.17	9.08	9.02	**16.26**	13.84
	Step1	3.84	3.66	3.82	3.64	3.70	3.58	3.59	3.56	8.47	5.37
	Step2	4.68	4.58	4.65	4.56	4.57	4.44	4.43	4.41	9.51	6.42
	Step3	5.53	5.40	5.51	5.39	5.31	5.22	5.18	5.14	10.56	7.36
	Step4	6.26	6.10	6.20	6.16	5.97	5.84	5.76	5.77	11.74	8.31
12	Step5	6.91	6.77	7.05	6.80	6.57	6.41	6.43	6.36	12.77	9.22
	Step6	7.63	7.56	7.60	7.52	7.09	6.99	6.96	6.95	13.97	10.12
	Step7	8.28	8.16	8.34	8.18	7.73	7.69	7.57	7.53	14.92	10.97
	Step8	8.98	8.84	8.96	8.78	8.24	8.25	8.08	8.06	16.03	11.74
	Step9	9.64	9.50	9.56	9.40	8.78	8.77	8.60	8.69	17.62	12.59
	Step1	3.73	3.59	3.68	3.57	3.64	3.58	**3.67**	**3.64**	9.12	4.88
	Step2	4.57	4.42	4.52	4.43	4.54	4.42	**4.51**	**4.41**	10.37	5.74
	Step3	5.26	5.16	5.23	5.15	5.27	5.15	**5.10**	**5.08**	11.28	6.55
	Step4	5.89	5.86	5.90	5.84	5.86	5.77	**5.69**	**5.67**	12.28	7.36
15	Step5	6.60	6.46	6.63	6.48	6.46	6.41	**6.28**	**6.23**	13.36	8.19
	Step6	7.22	7.15	7.19	7.10	7.09	6.99	**6.76**	**6.90**	14.46	8.90
	Step7	7.89	7.77	7.91	7.77	7.71	7.63	**7.40**	**7.42**	15.44	9.37
	Step8	8.54	8.41	8.52	8.39	8.09	8.13	**7.93**	**8.04**	16.33	10.05
	Step9	9.14	9.07	9.19	8.99	8.73	8.65	**8.53**	**8.61**	17.24	10.72
	Step1	**3.66**	**3.54**	**3.68**	**3.54**	**3.61**	**3.55**	3.61	3.63	9.68	**4.36**
	Step2	**4.45**	**4.43**	**4.46**	**4.39**	**4.52**	**4.43**	4.49	4.44	10.60	**5.12**
	Step3	**5.19**	**5.06**	**5.19**	**5.10**	**5.21**	**5.11**	5.13	5.09	11.39	**5.87**
	Step4	**5.79**	**5.76**	**5.84**	**5.74**	**5.85**	**5.75**	5.73	5.66	12.42	**6.59**
18	Step5	**6.59**	**6.40**	**6.52**	**6.37**	**6.47**	**6.35**	6.33	6.28	13.34	**7.27**
	Step6	**7.12**	**7.00**	**7.10**	**6.97**	**6.98**	**7.03**	6.95	6.94	14.40	**7.91**
	Step7	**7.80**	**7.66**	**7.88**	**7.67**	**7.63**	**7.57**	7.52	7.52	15.24	**8.43**
	Step8	**8.33**	**8.27**	**8.34**	**8.24**	**8.15**	**8.16**	8.11	8.16	16.12	**9.06**
	Step9	**8.98**	**8.88**	**8.97**	**8.90**	**8.74**	**8.72**	8.70	8.75	17.10	**9.69**

Table 2.7: Twelve-Step Forecasting MAPE (%) Value For The Standard And Proposed Machine Learning Models. Each Value Represents An Average Gap Between Actual and Predicted Traffic Volume For Each Individual Forecast Step.

Model		**GBR**		**XGB**		**LGB**		**CBR**		**SGD**	
		With Outlier	Without Outlier	With Outlier	Without Outlier	With Outlier	Without Outlier	With Outlier	Without Outlier	With Outlier	Without Outlier
12	Step1	3.92	3.67	3.82	3.65	3.70	3.58	3.59	3.56	**8.42**	5.36
	Step2	4.75	4.57	4.69	4.54	4.56	4.46	4.44	4.42	**9.47**	6.40
	Step3	5.52	5.38	5.52	5.42	5.30	5.23	5.17	5.12	**10.59**	7.36
	Step4	6.24	6.11	6.26	6.11	5.92	5.85	5.77	5.76	**11.75**	8.35
	Step5	6.91	6.80	7.00	6.84	6.55	6.47	6.42	6.34	**12.85**	9.23
	Step6	7.62	7.48	7.61	7.41	7.11	6.95	6.94	6.99	**13.82**	10.15
	Step7	8.26	8.16	8.34	8.17	7.73	7.66	7.50	7.55	**15.05**	11.02
	Step8	8.94	8.80	8.94	8.71	8.31	8.21	8.05	8.11	**15.91**	11.78
	Step9	9.56	9.50	9.56	9.40	8.84	8.84	8.63	8.74	**17.03**	12.58
	Step10	10.22	10.11	10.20	10.03	9.35	9.31	9.18	9.25	**18.15**	13.41
	Step11	10.81	10.68	10.88	10.63	9.96	9.90	9.79	9.87	**19.16**	13.86
	Step12	11.46	11.39	11.43	11.27	10.52	10.52	10.50	10.48	**20.11**	14.69
15	Step1	3.68	3.61	3.70	3.60	3.66	3.57	**3.68**	**3.63**	9.12	4.84
	Step2	4.57	4.48	4.52	4.44	4.54	4.43	**4.51**	**4.42**	10.23	5.74
	Step3	5.28	5.23	5.29	5.15	5.22	5.16	**5.11**	**5.10**	11.32	6.61
	Step4	5.99	5.81	5.95	5.83	5.84	5.75	**5.73**	**5.70**	12.50	7.44
	Step5	6.55	6.50	6.62	6.48	6.45	6.39	**6.28**	**6.19**	13.43	8.19
	Step6	7.23	7.12	7.25	7.03	7.09	7.02	**6.84**	**6.82**	14.30	8.91
	Step7	7.86	7.73	7.91	7.81	7.74	7.66	**7.48**	**7.45**	15.37	9.61
	Step8	8.51	8.41	8.50	8.39	8.08	8.14	**7.99**	**7.99**	16.46	10.03
	Step9	9.16	9.03	9.14	9.01	8.73	8.72	**8.53**	**8.62**	17.36	10.71
	Step10	9.65	9.66	9.72	9.64	9.25	9.26	**9.16**	**9.17**	18.32	11.43
	Step11	10.26	10.19	10.30	10.14	9.80	9.86	**9.82**	**9.80**	19.24	12.13
	Step12	10.90	10.79	10.94	10.78	10.43	10.40	**10.36**	**10.47**	20.27	12.80
18	Step1	**3.68**	**3.55**	**3.65**	**3.53**	**3.63**	**3.53**	3.68	3.61	9.51	**4.35**
	Step2	**4.46**	**4.41**	**4.46**	**4.38**	**4.58**	**4.42**	4.51	4.40	10.43	**5.15**
	Step3	**5.20**	**5.10**	**5.20**	**5.09**	**5.21**	**5.12**	5.16	5.10	11.48	**5.87**
	Step4	**5.84**	**5.74**	**5.83**	**5.74**	**5.91**	**5.73**	5.76	5.69	12.43	**6.57**
	Step5	**6.49**	**6.40**	**6.57**	**6.40**	**6.47**	**6.36**	6.36	6.28	13.66	**7.27**
	Step6	**7.12**	**6.97**	**7.15**	**6.99**	**7.06**	**7.01**	6.93	6.95	14.47	**7.96**
	Step7	**7.84**	**7.69**	**7.80**	**7.66**	**7.76**	**7.67**	7.58	7.52	15.32	**8.47**
	Step8	**8.31**	**8.27**	**8.35**	**8.21**	**8.19**	**8.10**	8.14	8.15	16.14	**9.04**
	Step9	**8.96**	**8.97**	**9.02**	**8.92**	**8.70**	**8.74**	8.68	8.73	17.18	**9.72**
	Step10	**9.49**	**9.42**	**9.45**	**9.43**	**9.20**	**9.28**	9.18	9.19	18.20	**10.31**
	Step11	**10.03**	**10.02**	**10.06**	**10.06**	**9.78**	**9.81**	9.74	9.77	19.82	**10.89**
	Step12	**10.71**	**10.60**	**10.72**	**10.57**	**10.35**	**10.36**	10.42	10.38	20.28	**11.58**

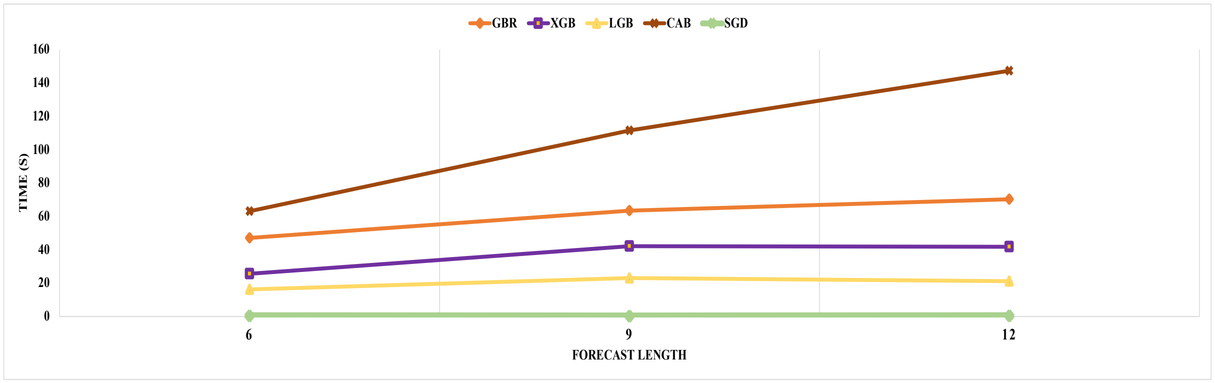


Figure 2.13: Average execution time of our proposed models for different forecast length.

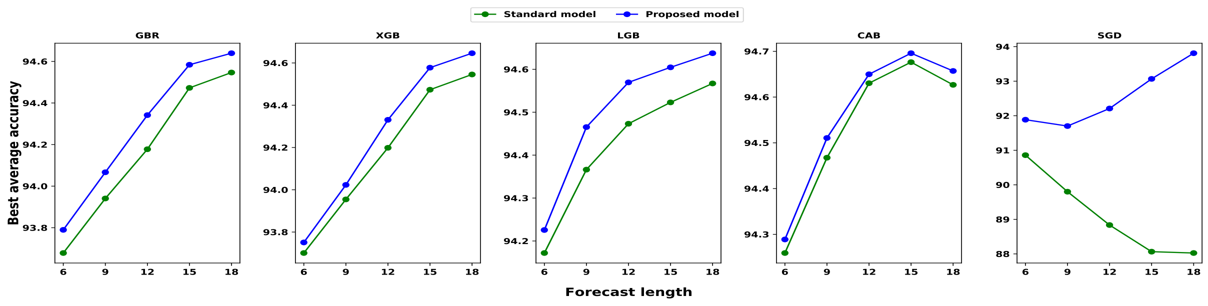


Figure 2.14: A comparative comparison among different feature subset performances regarding average prediction accuracy for six-steps forecasting. The average accuracy is calculated as the mean of each step's individual prediction accuracy.

prediction errors are 6.33% and 7.26%, and 6.32% and 7.25%, respectively. For six-step prediction, the LGB average prediction accuracy was 94.64%, the same as GBR, while the XGB provided the highest average accuracy of 94.65%. Overall, LGB performs prediction accuracy better compared to GBR and XGB. Since LGB performs leaf-wise growth as opposed to level-wise expansion in XGB, which produces larger loss reduction and much more complex trees and, as a result, improved accuracy while also being faster.

Also, we noticed a slow error-propagation rate between consecutive forecast steps in LGB compared to GBR and XGB. For example, in nine-step forecasting, our proposed XGB model started with a better first-step prediction with an error of 3.54%, which is lesser than the LGB first-step error of 3.55%, but for the last step, the XGB error was 8.90%, which is higher than the LGB's last-step error of 8.72%. According to Fig.3, LGB execution time is lesser than GBR and XGB. After analyzing the execution time for each forecast length, we concluded that LGB execution time is a minimum of 2 and 1.5 times shorter than GBR and XGB. LGB is a histogram-based approach that accelerates the training process by bucketing continuous attribute values into discrete bins. Finally, our last model from gradient boosting is CatBoost (CBR). CBR is the best-performing model with and without outlier detection among all considering models in our single-step and multi-step forecasting experiment. The best average prediction accuracy using our proposed CBR is 94.70%, 93.78%, and 92.89%, respectively, for six, nine, and twelve-step forecasting. All best-performing CBR models provided the high-

Table 2.8: Six-Step Forecasting MAPE (%) Value For The Standard And Proposed Deep Learning Models. Each Value Represents An Average Gap Between Actual and Predicted Traffic Volume For Each Individual Forecast Step.

Model		LSTM_Seq2Seq		LSTM_Seq2Seq_ATN	
		With Outlier	Without Outlier	With Outlier	Without Outlier
6	Step 1	4.76	3.75	4.93	3.78
	Step 2	5.46	4.71	5.58	4.69
	Step 3	6.34	5.64	6.40	5.61
	Step 4	7.31	6.56	7.30	6.53
	Step 5	8.36	7.61	8.30	7.59
	Step 6	9.42	8.66	9.31	8.67
9	Step 1	4.72	3.74	5.00	3.83
	Step 2	5.20	4.53	5.35	4.60
	Step 3	5.91	5.32	5.96	5.41
	Step 4	6.75	6.14	6.70	6.19
	Step 5	7.70	7.07	7.56	7.05
	Step 6	8.67	8.05	8.43	7.95
12	Step 1	4.62	3.65	4.95	3.72
	Step 2	5.07	4.34	5.15	4.41
	Step 3	5.70	5.07	5.73	5.17
	Step 4	6.48	5.78	6.49	5.89
	Step 5	7.41	6.50	7.37	6.60
	Step 6	8.38	7.21	8.28	7.28
15	Step 1	**4.58**	**3.47**	**4.83**	**3.51**
	Step 2	**5.02**	**4.18**	**5.05**	**4.21**
	Step 3	**5.65**	**4.85**	**5.61**	**4.91**
	Step 4	**6.40**	**5.48**	**6.32**	**5.58**
	Step 5	**7.28**	**6.10**	**7.11**	**6.27**
	Step 6	**8.20**	**6.71**	**7.89**	**6.96**
18	Step 1	4.49	3.48	4.63	**3.48**
	Step 2	4.90	4.21	5.00	**4.13**
	Step 3	5.75	4.86	5.80	**4.77**
	Step 4	6.65	5.47	6.67	**5.37**
	Step 5	7.47	6.15	7.49	**5.99**
	Step 6	8.19	6.85	8.22	**6.59**

Table 2.9: Nine-Step Forecasting MAPE (%) Value For The Standard And Proposed Deep Learning Models. Each Value Represents An Average Gap Between Actual and Predicted Traffic Volume For Each Individual Forecast Step.

Model		LSTM_Seq2Seq		LSTM_Seq2Seq_ATN	
		With Outlier	Without Outlier	With Outlier	Without Outlier
9	Step 1	5.19	3.90	5.34	3.92
	Step 2	5.66	4.53	5.74	4.57
	Step 3	6.26	5.33	6.21	5.35
	Step 4	7.02	6.13	6.90	6.10
	Step 5	7.94	6.97	7.77	6.90
	Step 6	8.87	7.81	8.65	7.69
	Step 7	9.83	8.60	9.56	8.42
	Step 8	10.76	9.30	10.41	9.05
	Step 9	11.69	10.01	11.27	9.69
12	Step 1	5.01	3.75	5.25	3.73
	Step 2	5.24	4.37	5.35	4.34
	Step 3	5.81	5.07	5.86	5.04
	Step 4	6.52	5.77	6.55	5.73
	Step 5	7.34	6.45	7.34	6.39
	Step 6	8.18	7.11	8.13	7.00
	Step 7	9.06	7.72	8.96	7.55
	Step 8	9.91	8.21	9.75	8.00
	Step 9	10.77	8.72	10.54	8.49
15	Step 1	4.88	3.75	5.18	3.70
	Step 2	5.37	4.42	5.45	4.21
	Step 3	6.13	5.11	6.28	4.82
	Step 4	6.96	5.77	7.10	5.43
	Step 5	7.76	6.46	7.87	6.04
	Step 6	8.54	7.11	8.60	6.62
	Step 7	9.30	7.78	9.32	7.15
	Step 8	10.00	8.38	10.03	7.60
	Step 9	10.76	9.00	10.78	8.13
18	Step 1	**4.85**	**3.54**	**4.66**	**3.56**
	Step 2	**4.86**	**4.17**	**4.81**	**4.14**
	Step 3	**5.34**	**4.79**	**5.33**	**4.84**
	Step 4	**5.96**	**5.39**	**5.96**	**5.49**
	Step 5	**6.64**	**6.01**	**6.63**	**6.08**
	Step 6	**7.34**	**6.63**	**7.30**	**6.63**
	Step 7	**8.08**	**7.22**	**7.99**	**7.16**
	Step 8	**8.82**	**7.72**	**8.64**	**7.62**
	Step 9	**9.61**	**8.28**	**9.33**	**8.16**

Table 2.10: Twelve-Step Forecasting MAPE (%) Value For The Standard And Proposed Deep Learning Models. Each Value Represents An Average Gap Between Actual and Predicted Traffic Volume For Each Individual Forecast Step.

Model		LSTM_Seq2Seq		LSTM_Seq2Seq_ATN	
		With Outlier	Without Outlier	With Outlier	Without Outlier
12	Step 1	5.25	4.07	5.71	3.95
	Step 2	5.42	4.63	5.64	4.47
	Step 3	6.06	5.27	6.17	5.16
	Step 4	6.81	5.92	6.88	5.86
	Step 5	7.59	6.62	7.62	6.56
	Step 6	8.32	7.26	8.32	7.20
	Step 7	9.08	7.94	9.02	7.80
	Step 8	9.81	8.52	9.69	8.34
	Step 9	10.59	9.12	10.40	8.89
	Step 10	11.37	9.69	11.08	9.46
	Step 11	12.22	10.33	11.81	10.13
	Step 12	13.08	11.01	12.56	10.82
15	Step 1	4.99	3.75	5.54	3.84
	Step 2	5.23	4.23	5.57	4.25
	Step 3	5.66	4.88	5.92	4.86
	Step 4	6.23	5.55	6.43	5.50
	Step 5	6.91	6.19	7.04	6.12
	Step 6	7.61	6.80	7.66	6.73
	Step 7	8.34	7.37	8.30	7.26
	Step 8	9.06	7.83	8.91	7.74
	Step 9	9.81	8.35	9.53	8.25
	Step 10	10.51	8.85	10.11	8.76
	Step 11	11.27	9.43	10.73	9.34
	Step 12	12.06	10.06	11.38	9.98
18	Step 1	**5.18**	**3.75**	**5.57**	**3.66**
	Step 2	**5.03**	**4.41**	**5.26**	**4.19**
	Step 3	**5.46**	**4.97**	**5.58**	**4.77**
	Step 4	**6.10**	**5.55**	**6.16**	**5.35**
	Step 5	**6.78**	**6.21**	**6.78**	**5.95**
	Step 6	**7.44**	**6.87**	**7.39**	**6.52**
	Step 7	**8.08**	**7.53**	**8.00**	**7.05**
	Step 8	**8.71**	**8.08**	**8.60**	**7.52**
	Step 9	**9.39**	**8.65**	**9.27**	**8.05**
	Step 10	**10.09**	**9.16**	**9.96**	**8.53**
	Step 11	**10.87**	**9.72**	**10.72**	**9.10**
	Step 12	**11.67**	**10.32**	**11.51**	**9.69**

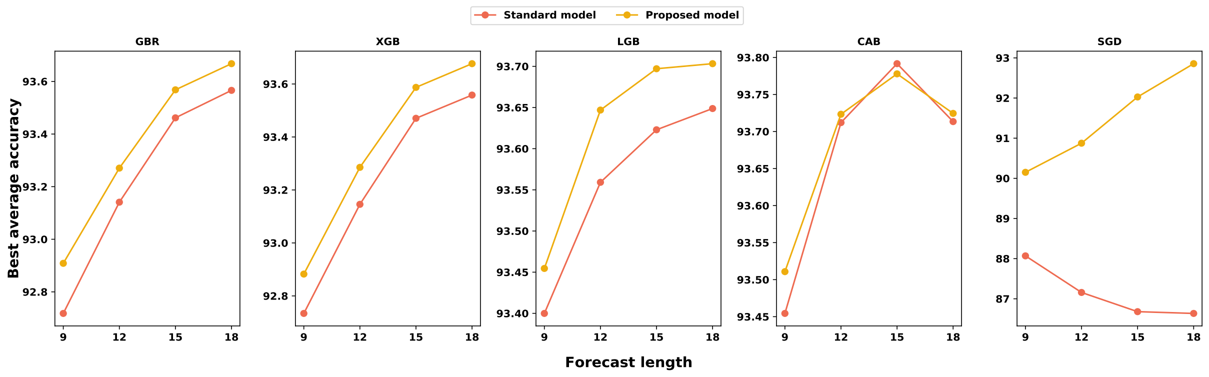


Figure 2.15: A comparative comparison among different feature subset performances regarding average prediction accuracy for nine-steps forecasting. The average accuracy is calculated as the mean of each step's individual prediction accuracy.

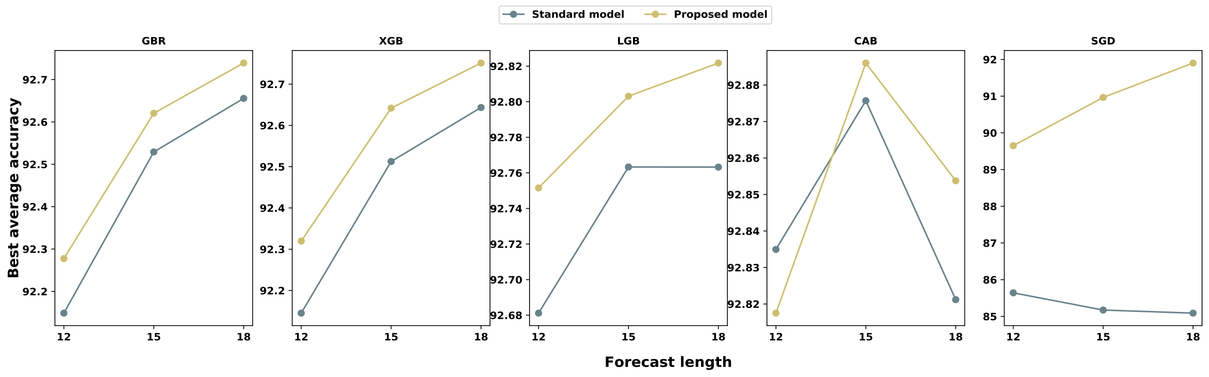
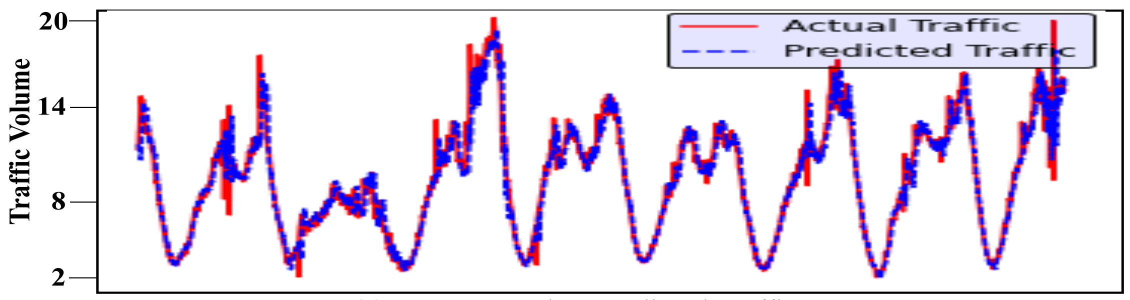
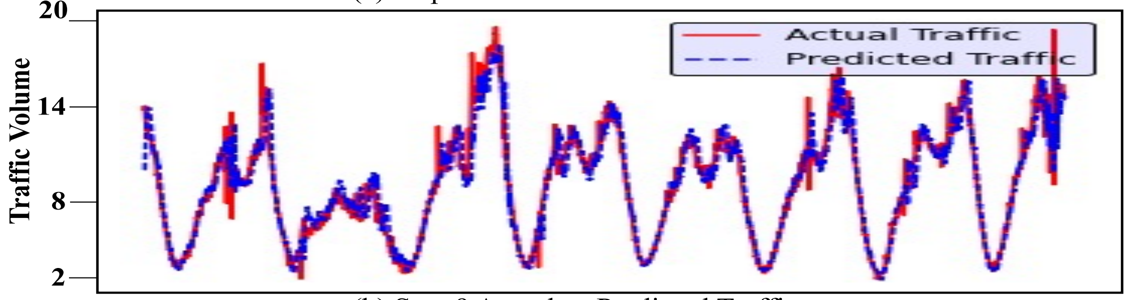


Figure 2.16: A comparative comparison among different feature subset performances regarding average prediction accuracy for twelve-steps forecasting. The average accuracy is calculated as the mean of each step's individual prediction accuracy.

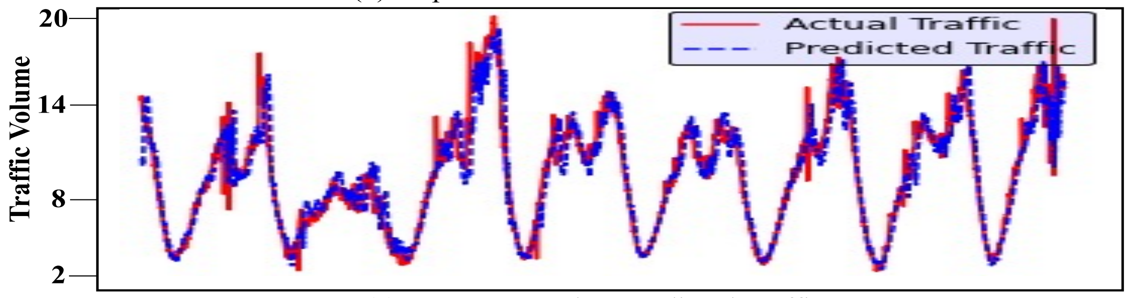
est accuracy with fifteen features. A weighted sample variation of SGD, known as minimal variance sampling (MVS), is provided by CBR. This method uses weighted sampling at the tree layer rather than the split level. To increase the precision of split grading, the data for each boosting tree are chosen in a certain way, resulting in better accuracy. However, CBR takes the highest execution time among all prediction models. For example, LGB is the fastest algorithm from boosting category, and CBR execution time is more than three, four, and six times more than LGB, respectively, for six, nine, and twelve-step forecasting. Lastly, we consider a simple machine learning model called stochastic gradient descent (SGD) for our traffic prediction task. Since gradient boosting and gradient descent both work similarly and descend the slope of the loss function, we consider SGD for our experiment to show a comparative analysis with the gradient boosting algorithm. According to our investigation, the SGD prediction is lower among all prediction models for single-step and multi-step. For example, the best standard SGD model performance for twelve-step forecasting is 85.64% without outlier detection, which is more than 6% less than the gradient boosting algorithm. But our proposed SGD model performance improved significantly compared to classical SGD with an average accu-



(a) Step 6 Actual vs Predicted Traffic



(b) Step 9 Actual vs Predicted Traffic



(c) Step 12 Actual vs Predicted Traffic

Figure 2.17: Multi-step actual vs. predicted traffic by best-peforming deep sequence model.

racy of 91.90% for twelve-step prediction; but still, it is smaller than the gradient boosting best performance. Gradient boosts descent gradient by adding new models, as opposed to gradient descent, which descends the gradient by introducing modifications to hyper-parameters. As a result, the model architecture gradient boosting changed dynamically as opposed to the fixed model in SGD, which gave us a better prediction using gradient boosting than SGD. However, the SGD is the fastest algorithm among all prediction models from boosting category.

We also investigated the average accuracy for each input set. Then, we calculated all individual step prediction accuracy to determine the average accuracy for particular input and model settings. Fig. 2.14, Fig. 2.15, and Fig. 2.16 illustrate the average accuracy comparison between the standard model and our proposed model for each feature set. From these figures, we noticed our proposed model outperforms traditional models for each input configuration, and the average accuracy is increasing with the larger feature set except the CBR model. In the case of the CBR model, the model performance dropped for eighteen time-lagged features, which is common for each multi-step forecast window. However, SGD showed a completely different behavior compared to gradient boosting algorithms. Furthermore, the standard SGD model average prediction accuracy decreases with an increase in the input size, while our pro-

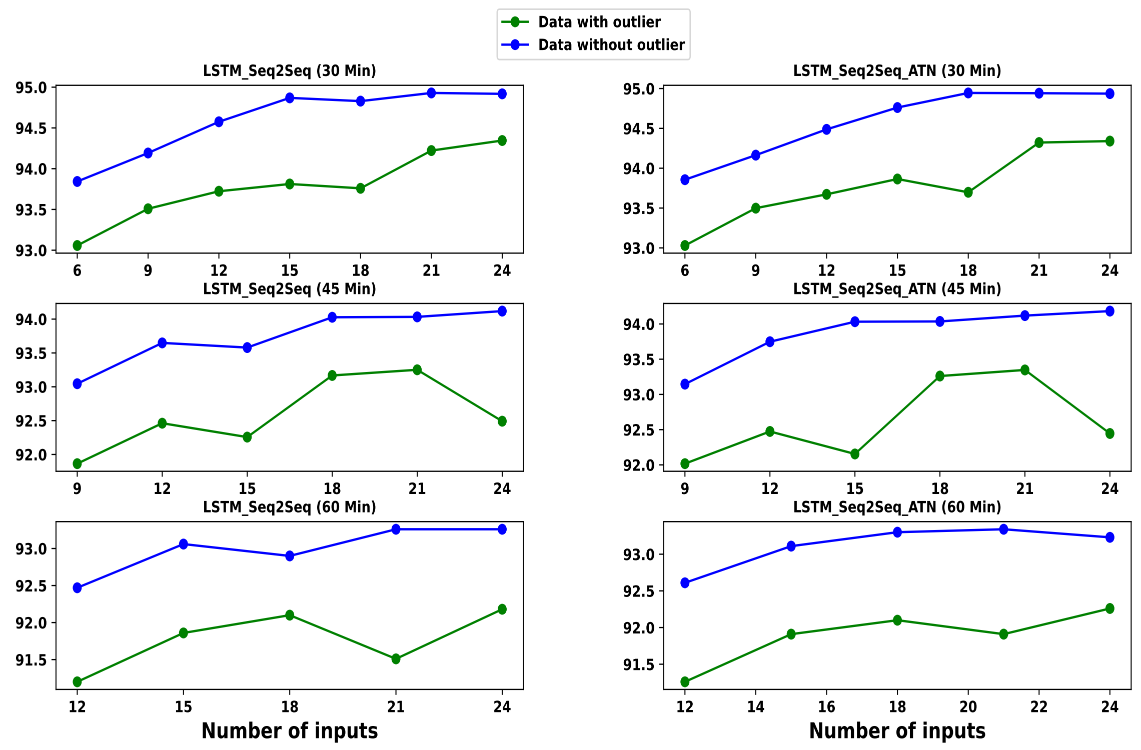


Figure 2.18: Model average accuracy comparison with the input length.

posed SGD model average accuracy increases with the input size. Therefore, we can conclude that the anomaly mitigation module helps the model to learn better the traffic pattern resulting in better accuracy.

After performing a comprehensive analysis of multi-step forecasting using boosting algorithms, we have noticed that best-performing single-step boosting algorithms also perform better in multi-step forecasting. Therefore, we chose two best-performing deep sequence models from our single-step experiment: LSTM_Seq2Seq and LSTM_Seq2Seq_ATN for the multi-step forecasting tasks. Multi-step prediction is more challenging than single-step prediction as the prediction error accumulates forward with the prediction length. So, the prediction error is more likely to increase for more extended step forecasting. In this experiment, we considered different prediction lengths of six steps (30 minutes), nine steps (45 minutes), and twelve steps (60 minutes). All multi-step forecasting models are trained using data with and without outliers to analyze the anomaly's impact on traffic prediction. Also, we considered varying input lengths for data windowing to extract the optimum feature length for the predictive model. Next, we explained our multi-step experimental results based on different prediction lengths.

A total of seven different input sets are used to train our predictive model to identify the optimum feature length for model training. The input set starts from length six to length 24 with a gap of break of 15 minutes of traffic. The experimental results are represented in Table 2.8. The LSTM_Seq2Seq_ATN model with an input length of 18 produced the best prediction result for six steps with an average MAPE of 5.06 %, where the data was outlier free. Our best prediction model's MAPE for individual steps are 3.48%, 4.13%, 4.77%, 5.37%, 5.99%, and 6.59%. Although the input lengths 21 and 24 gave us a minimum average MAPE

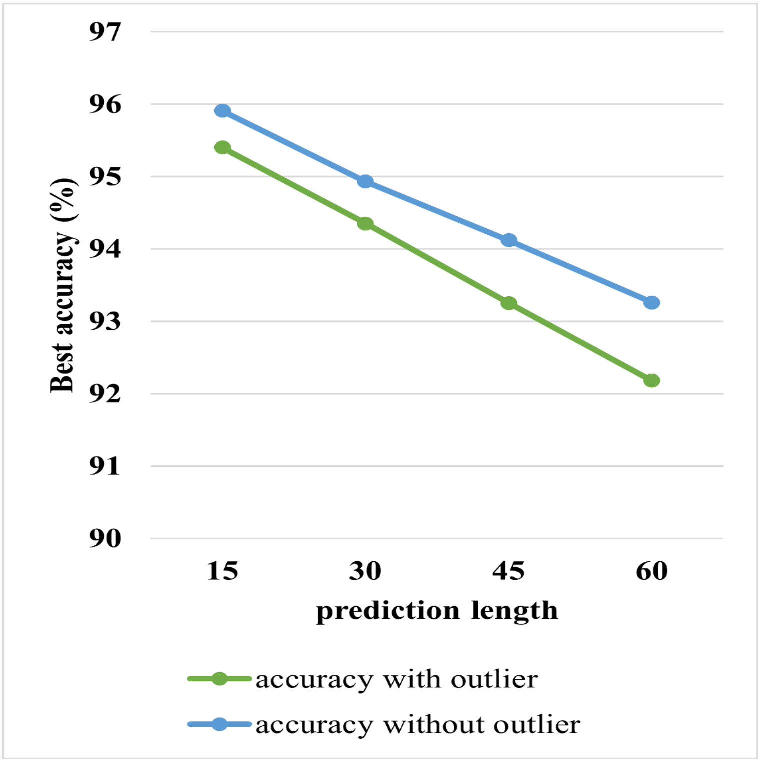


Figure 2.19: LSTM_Seq2Seq model average accuracy comparison with the input length.

of 5.06%, we considered the minimum feature length to reduce model training time. The LSTM_Seq2Seq_ATN also performed better with outliers in the data, and the average MAPE was 5.66% for input length 24. We noticed an increasing pattern in the gap between actual and predicted traffic for multi-step forecasting as the error in the previous step is propagated to the next steps. Fig. 2.17(b)compares the actual and predicted traffic by the best-performing model for the sixth step. Our results again indicate a better prediction when we mitigate the outliers before training the models.

According to the experiment in Table 2.9, the best prediction model for nine steps ahead forecasting is the LSTM_Seq2Seq_ATN with an input layer of 24-time steps. The minimum average MAPE between actual and predicted traffic is 5.82% without outliers in the data. We used six sets of input variables to determine the optimum number of inputs for the model. Firstly, we tried 45 minutes of traffic to predict the next 45 minutes. Then we increased the input length by 15 minutes and extended it to 120 minutes or 24 inputs variables. However, The LSTM_Seq2Seq model also showed better performance of average MAPE of 5.88% for an input length of 24 compared with data having anomalies (6.75% for an input length of 21). Our experimental result showed an outperformance of the LSTM_Seq2Seq_ATN (average MAPE of 6.65% and 5.82% for input lengths 21 and 24) model compared to the LSTM_Seq2Seq (average MAPE of 6.75% and 5.88% for input length 21 and 24) for both data type scenarios. The step-wise comparison between actual and predicted traffic by the best-performing model is shown in Fig. 2.17(c). Again, the anomalies in data affect the model performance, and it improves as outliers are mitigated.

Finally, we trained our model for 12 steps ahead prediction for 60 minutes of traffic.

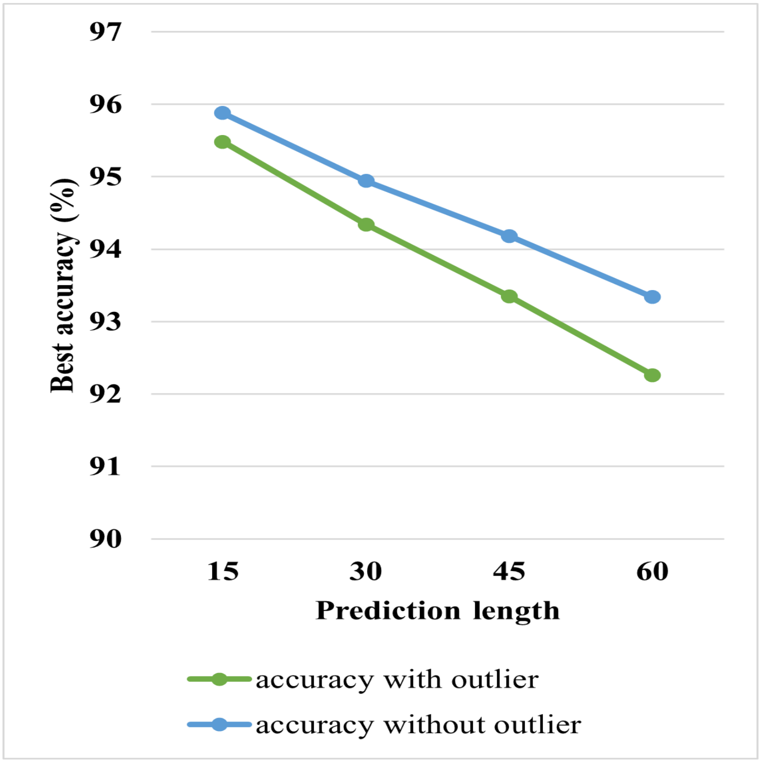


Figure 2.20: LSTM_Seq2Seq_ATN model average accuracy comparison with the input length.

The input lengths are varied from 12 to 24 with a gap of 3 steps. We showed our experimental results in Table 2.10. The experimental results showed better performance of the LSTM_Seq2Seq_ATN (average MAPE of 6.66% for input length 21) model compared with the LSTM_Seq2Seq (average MAPE of 6.74% for input length 21) when they trained with anomalous data. LSTM_Seq2Seq_ATN model also performs better than LSTM_Seq2Seq when trained using data with outliers. We illustrated the actual and predicted for the last step in Fig. 2.17(c). Ultimately, the prediction model performed better when trained with cleaned data.

The model accuracy is calculated as the mean of each step for the corresponding model. Fig. 2.18 illustrates a comparison of the average model accuracy with the input length. Overall, all graphs show an accuracy improvement with the larger input length. In the outlier-handled dataset, the accuracy for the LSTM_Seq2Seq_ATN model rises quickly as we increase the input, and model performance seems stable for the last couple of inputs. For the LSTM_Seq2Seq model, the average accuracy increases soon for the first few input lengths and drops again, although it shows an upward trend. In the case of model training with outliers in the dataset, the graphs are shown an increasing pattern with a higher number of inputs. However, at some input points, the average accuracy drops sharply. According to our experiment, optimizing the input lengths for the prediction model significantly impacts the model's accuracy.

Fig. 2.19 and Fig. 2.20 depict a comparative analysis among best accuracy for different prediction lengths with and without outliers in the data. We have found a similar pattern of accuracy improvement after adjusting outlier for both LSTM_Seq2Seq, and LSTM_Seq2Seq_ATN model illustrates in Fig. 2.19 and Fig. 2.20. With the increasing prediction length, the gap between the best accuracy for two different data settings is also increasing. We notice a decreasing pattern in accuracy with the increased size of the prediction window, as the prediction error is cumulative for the next steps. The impact of outlier points in the traffic is apparent in both model performances, and it is essential to mitigate them for long-term prediction.

2.5 Conclusion

This study embarked on a comprehensive investigation of single-step and multi-step traffic prediction tasks using various machine learning models, which included gradient boosting models, gradient descent models, and deep sequence models. A range of algorithms were utilized, such as XGBoost (XGB), LightGBM (LGB), CatBoost (CBR), Stochastic Gradient Descent (SGD), Gradient Boosting Regressor (GBR), and Recurrent Neural Networks (RNN) and their variants, including Long Short-Term Memory (LSTM) models with distinct configurations.

Our findings clearly indicated that rigorous data preprocessing, particularly outlier mitigation, was instrumental in enhancing model accuracy, providing an average improvement of 30% in traffic prediction accuracy. In the single-step prediction tasks, XGB performed exceptionally well among boosting algorithms while LSTM_Seq2Seq achieved the lowest prediction error among deep sequence models. Interestingly, SGD was identified as the most sensitive model to variations in input length. In multi-step forecasting scenarios, different dynamics came into play. CBR emerged as the top performer across all forecast lengths (six, nine, and twelve-step predictions), exhibiting the highest prediction accuracy, although it had the highest execution time. Among the deep sequence models, LSTM_Seq2Seq and LSTM_Seq2Seq_ATN demonstrated promising multi-step forecasting capabilities, with accuracy improving as the input length increased. Furthermore, the importance of efficient outlier mitigation became increasingly pronounced with lengthier prediction horizons, accentuating the value of data quality management in predictive modeling. Across both single-step and multi-step prediction tasks, the importance of optimized input length and careful handling of outliers were the recurring themes. The models' sensitivities to these factors underline the necessity of thorough data preprocessing and model-specific parameter optimization for achieving high-quality traffic prediction.

Summarizing, this research provides valuable insights into the effectiveness of advanced machine learning algorithms for traffic prediction tasks, given proper training on outlier-adjusted datasets. The study's outcomes highlight the strengths and limitations of each model in different prediction scenarios and emphasize the necessity of robust outlier detection and removal. With ongoing advancements in machine learning, the accuracy and efficiency of traffic prediction systems can significantly improve, aiding in the development of smarter and more effective traffic management systems.

Chapter 3

Empirical Mode Decomposition and K-Nearest Neighbour Integrated Traffic Prediction Model: An Approach to Improve Noise Reduction and Outlier Mitigation

Abstract: Internet traffic volume estimation has a significant impact on the business policies of the ISP (Internet Service Provider) industry and business successions. Forecasting the internet traffic demand helps to shed light on the future traffic trend, which is often helpful for ISPs' decision-making in network planning activities and investments. Besides, the capability to understand future trend contributes to managing regular and long-term operations. This study aims to predict the network traffic volume demand using deep sequence methods that incorporate Empirical Mode Decomposition (EMD) based noise reduction, Empirical rule based outlier detection, and K-Nearest Neighbour (KNN) based outlier mitigation. In contrast to the former studies, the proposed model does not rely on a particular EMD decomposed component called Intrinsic Mode Function (IMF) for signal denoising. In our proposed traffic prediction model, we used an average of all IMFs components for signal denoising. Moreover, the abnormal data points are replaced by K nearest data point's average, and the value for K has been optimized based on the KNN regressor prediction error measured in Root Mean Squared Error (RMSE). Finally, we selected the best time-lagged feature subset for our prediction model based on AutoRegressive Integrated Moving Average (ARIMA) and Akaike Information Criterion (AIC) value. Our experiments are conducted on real-world internet traffic datasets from industry, and the proposed method is compared with various traditional deep sequence baseline models. Our results show that the proposed EMD-KNN integrated prediction models outperform comparative models.

3.1 Introduction

In the previous chapter, we discussed the impact of outlier data points in single-step and multi-step traffic forecasting using both machine learning and deep learning. According to our experimental results, the outlier detection and mitigation module helps us to improve overall prediction accuracy. This section proposes a traffic prediction model integrated with a noise reduction module. We consider only deep learning models for this experiment as they outperform machine learning models in the case of the previous experiment.

Outlier detection is the process of identifying data points that are significantly different from the rest of the data. These outliers may be caused by measurement errors, faulty sensors, or unusual events. Outliers can have a significant impact on statistical analysis, so detecting and removing them is important to ensure accurate results. In time series analysis, outlier detection typically involves identifying individual data points that are far from the expected value or pattern of the time series. Noise reduction, on the other hand, is the process of reducing random fluctuations in the data that are not related to the underlying trend or pattern. Noise can make it difficult to identify trends and patterns in the data and can obscure important information. In time series analysis, noise reduction typically involves applying smoothing techniques or filters to remove high-frequency noise. In summary, outlier detection focuses on identifying and removing individual data points that are significantly different from the rest of the data, while noise reduction focuses on reducing random fluctuations in the data to make it easier to identify underlying trends and patterns.

As a unique time-series, network traffic reflects the interaction and influence between network services through complex features like nonlinearity, fractality, bursts, disorder, and heterogeneity. In this work, we proposed a traffic prediction methodology integrating empirical mode decomposition (EMD) based denoising and Empirical rule-based outlier detection. There are several works where EMD-based hybrid models have been proposed for traffic prediction. Most of them used EMD to decompose a signal into several components, where each component was modeled separately using either a linear or non-linear model. This multiple-model prediction strategy is time-consuming, and selecting a suitable model for a particular component is non-trivial. In this study, we aim to predict network traffic volume demand using single deep sequence methods that incorporate Empirical Mode Decomposition (EMD) based noise reduction, Empirical rule based outlier detection, and K-Nearest Neighbour (KNN) based outlier mitigation. We used real-world internet traffic datasets from industry for our experiments and compared the proposed EMD-KNN integrated prediction models with various statistical and traditional deep sequence baseline models.

Our methodology involved several steps. First, we performed EMD-based noise reduction on the dataset to remove any high-frequency noise that might affect the accuracy of our predictions. We then applied Empirical rule based outlier detection to identify any abnormal data points in the dataset. These data points were then replaced by the K nearest data point's average using KNN-based outlier mitigation. To identify the best value for K, we performed a grid search algorithm based on the KNN regressor. We also used AutoRegressive Integrated Moving Average (ARIMA) and Akaike Information Criterion (AIC) values to select the best time-lagged feature subset for our prediction model. Finally, we used deep learning techniques to develop our EMD-KNN Traffic Forecaster model, which combines EMD-KNN methods with a deep neural network to predict network traffic volume demand accurately. This paper is organized as follows. Section 3.2 describes the literature review of current traffic prediction using machine learning models. Section 3.3 presents our proposed methodology. Section 3.4 summarizes the experimentation configuration and discusses results for comparative analysis. Finally, section 3.5 concludes our paper and sheds light on future research directions.

3.2 Literature Review

By making accurate predictions about how much traffic will be on the network in the future, network administrators can make plans for how to meet service delivery goals in the areas of resource allocation, congestion control, routing decisions, capacity planning, quality of service, and anomaly detection [57]. As a result, significant research works have been conducted to explore effective techniques that can be used to predict network traffic with the minimum deviation between actual and predicted traffic.

Since real-world traffic data is a nonlinear time series with noise, breaking down the time series into its hierarchical components would provide more accurate predicting results [58, 59]. It has been observed that splitting up traffic time series into finite subsequences using the Discrete Wavelet Transform (DWT), Stationary Wavelet Transform (SWT), or Empirical Mode Decomposition (EMD) is an efficient way to capture both the general trend and certain variations in traffic flow [58, 60]. For predicting computer network traffic, Rishabh and Partha [39] introduced the DWT, ARIMA model, and RNN-based approach. The traffic data is initially divided into non-linear (approximate) and linear (detailed) components using a discrete wavelet transform. Afterward, predictions are performed using ARIMA and RNN, respectively, and detailed and approximate components are rebuilt using inverse DWT. The approach, according to the authors, is simple to use and computationally less costly. Thus it may be simply deployed in data centers to increase QoS (quality of service) while lowering costs. The recurrent wavelet neural network (RWNN) based on the Elman network was developed. The dynamic gradient descent technique of RWNN was also provided, and it could be utilized to anticipate network traffic [61]. The network traffic prediction model based on RWNN is practical and efficient, according to experimental data. In [62], an artificial neural network model integrated with a multi-fractal DWT is proposed. With the input of the original traffic data, the network traffic is divided into low-frequency and high-frequency components using a mother wavelet called haar. When compared to the two current approaches, their model performs better. The authors of [63] conducted a comparative investigation of several DWT and spline-extrapolation techniques in order to forecast the features of IoT multimedia internet traffic. The best spline-extrapolation used B-splines, which had the lowest forecast error of 5%, while quadratic and Haar-wavelet splines had prediction errors of 10% and 7–10%, respectively.

In [64], authors proposed a novel short-term traffic flow forecast method based on combination model fusion and empirical mode decomposition. They begin by looking at the amplitude-frequency properties of short-term traffic flow series, then apply empirical mode decomposition to break the traffic flow up into numerous components with various frequencies. Second, improved extreme learning machines, seasonal autoregressive integrated moving averages, and autoregressive moving averages are chosen to predict various components based on the findings of the self-similarity analysis of each component. To anticipate traffic flow data at various yet typical time scales, authors suggested an ensemble framework with ensemble empirical mode decomposition (EEMD) and an Artificial Neural Network (ANN) model [65]. The suggested EEMD model divided the raw traffic flow data into several IMFs, suppressed the noisy IMFs, then combined the remaining IMFs to provide the noise-free traffic flow data. Following this, the ANN model was introduced to forecast traffic flow at various time scales. According to the experimental findings, hybrid models, specifically, EEMD+ANN and EMD+ANN significantly outperform the traditional ANN model when it comes to predicting traffic flow.

As a unique timeseries, network traffic reflects the interaction and influence between network services through complex features like nonlinearity, fractality, bursts, disorder, and heterogeneity [66]. In this work, we proposed a traffic prediction methodology integrating empirical mode decomposition (EMD) based denoising and Empirical rule-based outlier detection with various state-of-the-art deep sequence models. There are several works where EMD-based hybrid models have been proposed for traffic prediction. Most of them used EMD to decompose a signal into several components, where each component was modeled separately using either a linear or non-linear model. This multiple-model prediction strategy is time-consuming, and selecting a suitable model for a particular component is non-trivial. But in this work, we consider a single model strategy for traffic forecasting while EMD has been used to denoise our original traffic signal. After extracting all Intrinsic Mode Function (IMF) components by EMD, we calculate the average of all IMF elements to subtract them from the original signal to make it noise-free. Moreover, we integrate an outlier detection and mitigation module in our proposed methodology. In the real world, internet traffic is very susceptible to various internal and external factors resulting in many outliers in the original data. The Empirical Rule has been applied in our traffic data to identify the point outlier, which is mitigated using K nearest neighbor observation's average traffic. To identify the best value for K, we performed a grid search algorithm based on the KNN regressor. We consider the best K value based on minimum prediction error, which is later used as a parameter in KNN imputation method. Finally, we identify the optimum lagged features based on the ARIMA (AutoRegressive Integrated Moving Average) model and Akaike Information Criterion (AIC) value. ARIMA model uses the time-lagged feature to predict the future, which we fine-tuned by applying a grid search approach. The lower AIC value presents the better model performance, and that model gives us the corresponding best time-lagged feature parameter. Hence, we consider the optimum lagged feature number as an input parameter for our prediction models. In addition to EMD-based noise reduction, we proposed another novel traffic prediction model based on a single algorithm where the decomposed components from EMD have been used to prepare our training data for the prediction model. Finally, we compare two EMD-integrated traffic prediction methodologies to find best-possible way of modeling real-world internet traffic.

3.3 Proposed Methodology

In this section, we delve into the details of our proposed model, as illustrated in Fig. 3.1. We initiate the discussion by describing our noise reduction module in subsection 3.3.1. Following that, we explore our outlier detection module and feature extraction module in subsections 3.3.2 and 3.3.3, respectively.

3.3.1 Empirical Mode Decomposition Based Noise Reduction

Empirical Mode Decomposition (EMD) is a technique to extract several components from a signal assuming every signal comprises of sub-components. This approach is also known as Hilbert–Huang transforms (HHT) and is extensively used for time-frequency analysis of non-stationary and non-linear time series data. EMD decomposed an original signal into several zero mean and quasi-periodic components called Intrinsic Mode Functions (IMF) alongside a residue element representing the trend as shown in Eq.3.1 where each $h_i(t)$ stands for the ith IMF, $r(t)$ is the residual component, and $y(t)$ is the original value of the data.

$$y(t) = \sum_{i}^{n} h_i(t) + r(t) \tag{3.1}$$

Real-world internet traffic has random, non-stationary characteristics influenced by various external and internal factors related to ISP companies. These factors can be categorized as geographic factors, economic factors, ISP new service, service decommission factors, weather, time, day, season, special event, etc. Due to these factors, the ISP traffic is composed of many individual components, and

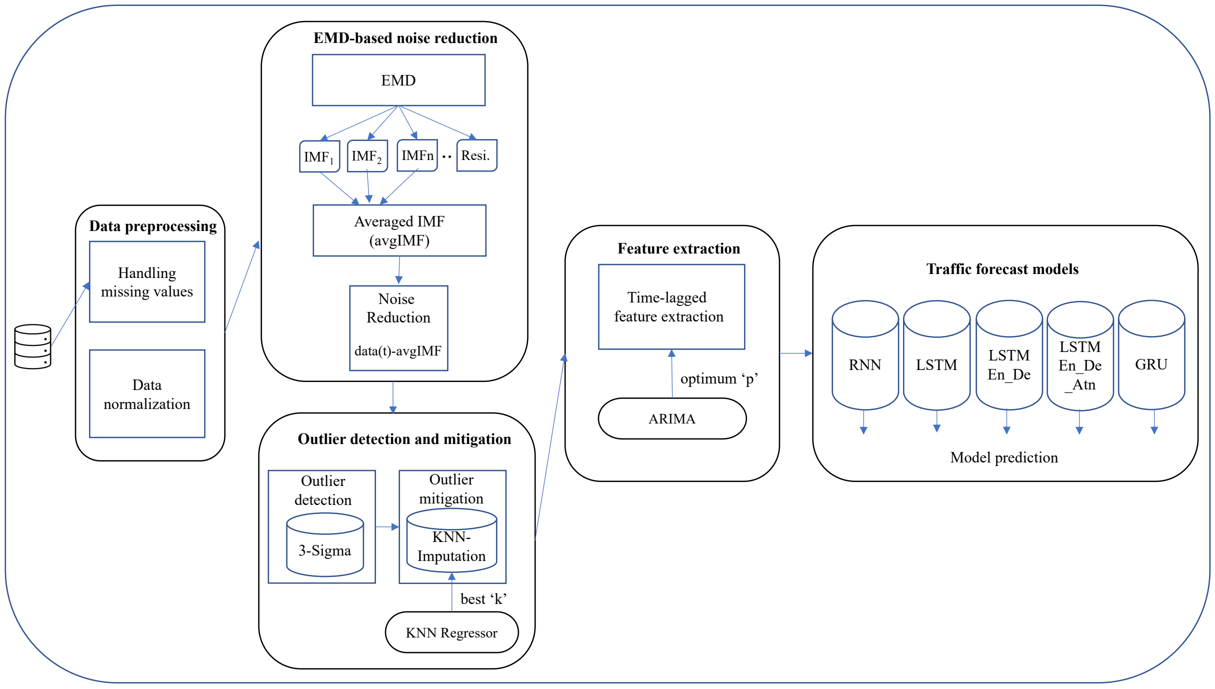


Figure 3.1: High-level framework of our proposed methodology integrated with EMD based noise reduction and KNN based outlier mitigation.

EMD can be helpful for better analysis and forecasting of internet traffic. Noise filtering or reduction or signal denoising is a process of removing noise from time-series data. Any time series may consist of three systematic elements: level, trend, and seasonality, and one non-systematic element, noise. The noise reduction approach for better learning and forecasting by the machine learning model should minimize noise elements in time series. Among different noise filtering approaches, EMD based denoising technique has been applied extensively in different areas. Classic EMD-based denoising techniques choose a particular IMF component to eliminate noise elements from the signal, but there is no formal logic for selecting an IMF from decomposed components. Moreover, choosing one IMF for denoising is difficult as the number of IMF depends on the original signal. Therefore, a new EMD-based noise filtering method has been proposed in [67]. They showed that the average of all IMF (avgIMF) components is normally distributed. The avgIMF corresponds to the maximum signal noise level and has the most white Gaussian noise features of any IMF element.

The EMD technique is used in this study to remove noise from the internet traffic time series data, which is considered one-dimensional data. The abrupt changes in our traffic forecasting data have been smoothed by removing noise based on steps 3 and 4 in Algorithm 1. We extracted the average of all IMF elements, avgIMF, from our original signal, $y(t)$, to obtain a noise-free traffic data $y_n(t)$.

3.3.2 Empirical Rule and KNN Based Outlier Management

Outlier detection is a necessary preprocessing step for real-world internet traffic analysis. The data points significantly different from most of the values are considered outliers. Outliers are characteristically different than noise in the time series. Noise is a random error in the data and needs to be removed entirely from the original signal for a better prediction model. On the contrary, the outliers are the part of the time series that impacts different statistical parameters, such as mean, standard deviation, correlation, etc., of the original signal. Outliers can lead to incorrect future predictions of internet traffic.

A statistical principle known as the empirical rule, the three-sigma rule or 68-95-99.7 rule, holds that almost all observed data will lie within three standard deviations of the mean with a normal distribution. However, this rule is also applicable for non-normally distributed data where 88.8% of data fall within the three-sigma interval as opposed to 99.97% for normal distribution. According to Chebyshev's inequality, 75% of the data lie inside two standard deviations for a wide range of various probability distributions, while the empirical rule claim 95% data points within the second standard deviation for normal distribution[43]. In this work, we set an upper and lower limit for most of the data instances in our original signal. The individual data point is outside the three-standard deviation considered as point outliers. Those point outliers in our dataset have been mitigated by K nearest data points based on the standard KNN-Imputation algorithm. Each outlier point is imputed using the mean value from K nearest neighbors in the training set. The training dataset's members' distances are calculated using a Euclidean distance measure that is NaN aware, which excludes NaN values from the calculation. For optimum K value, we apply KNN-regressor in our traffic dataset where past observations are used to predict the following data points, and this experiment has been conducted for different K values ranging from 2 to 24. The minimum prediction error measured in terms of RMSE (Root Mean Squared Error) is the criteria for best choosing the best K for imputing outlier data points.

3.3.3 ARIMA Based Time-lagged Feature Extraction

Generally, the time series prediction task uses previous data samples to predict the following values. We extract the time-lagged feature from our original dataset for training and testing our prediction model in this work. Based on ACF analysis in subsection 2.3.3, we concluded that our traffic data is non-random. Hence, we considered previous timestamps features for our deep-learning models to predict the following values. We performed a grid search based on the Akaike Information Criterion (AIC) value and a statistical prediction model called ARIMA to determine the optimum number of lagged features. The AIC measures the relative quality of statistical models for a given set of data and predicts prediction error. AIC calculates each model's efficiency in relation to the other predictions given a set of models for the data. As a result, AIC offers a model ranking method. However, we compare ARIMA model prediction performance with various settings of hyperparameters and rank them based on the AIC value. ARIMA model predicts the future value based on the past values of the time series, that is, its own lagged values. The model requires three parameters such as $AR(p)$, $MA(q)$, and $I(d)$, which represent the Autoregressive, Moving Average and differencing order. Among these parameters, the AR term defines the number of lagged features used to forecast the next value. Therefore, we performed a grid search using a different combination of p, q, and d to perform single-step prediction using the ARIMA model and select the best model by comparing their prediction performance based on our selection criteria. We consider AR term, p, from best performing model based on minimum AIC and p indicates the time-lagged feature which gave us better prediction. So, the prediction task in this study is performed as in Eq.3.2. where $y(t)$ is the traffic volume for the current time step, $y(t-1)$ to $y(t-p)$ represents the previous p data points, and h is the prediction function.

$$y(t+1) = h(y(t-1), y(t-2), ..., y(t-p)) \tag{3.2}$$

3.4 Analysis of Experimental Results

In this section, we begin by analyzing the output of the outlier detection and mitigation module in our proposed model in subsection 3.4.1. We then discuss the results of traffic denoising, demonstrating its positive impact on improving data quality in subsection 3.4.2. Subsection 3.4.3 focuses on the performance of our ARIMA-based feature selection module. Finally, we present the performance of our prediction model in Subsection 3.4.4.

3.4.1 Outlier Data Identification and Mitigation

According to our outlier detection module based on empirical rule, there are total of 43 outlier data points. In Fig. 3.2, we depicted the data samples which are lies outside three standard deviation. Moreover, the distribution of the data with highlighting outlier points has been illustrated in Fig. 3.3. It is a right-skewed histogram where the data is clustered towards the left side of the histogram and extends further to the right. This type of distribution is also called also called a positive skew. In this diagram, we identified the outlier points on the right-side of the right-skewed histogram. A right-skewed histogram with an outlier on the right side indicates that at least one data point has an extremely high value relative to the rest of the data and Fig. 3.2 depicted several data points which are relatively high. These outliers can have a significant impact on the overall distribution of the data, as they can affect the mean and median of the dataset. In general, it is important to identify and handle outliers in a dataset, especially if they are affecting the distribution of the data. Therefore, we applied KNN-Imputation to handle these outlier data points in our traffic dataset.

We used the KNN algorithm to handle outlier values in your dataset, and determined the optimal value of K by using a KNN regressor and comparing the root mean squared error (RMSE) for different values of K. Based on this, we replaced the outlier point with the average of the K nearest neighbors. The choice of K can have a significant impact on the quality of the imputed values. A small K value may result in overfitting, where the imputed values are too similar to the original data and may not be accurate. A large K value may result in underfitting, where the imputed values are too different from the original data and may not be representative of the underlying distribution. Therefore, a range of K values has been used and reported their corresponding RMSE in Table in 3.1. The K value 11 gave us the lowest RMSE of 1796783992 bps. Based on this, we considered average of 11 previous data points to replace corresponding outlier data.

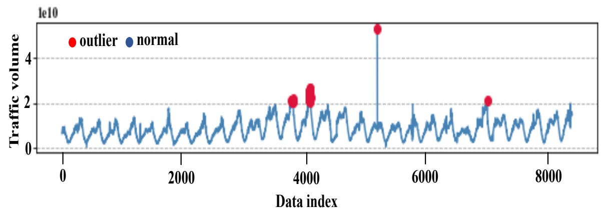


Figure 3.2: Outlier points identified using empirical rule.

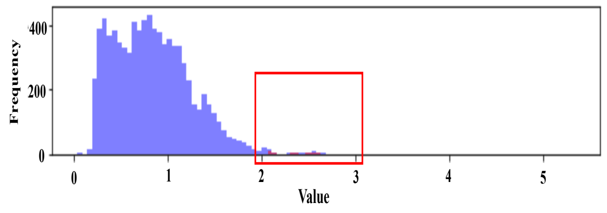


Figure 3.3: Distribution of outlier points.

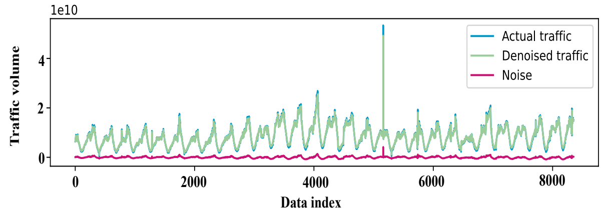


Figure 3.4: Denoised traffic data.

Table 3.1: KNN-Regressor Prediction Error (RMSE) For Different *K* Values.

K-value	**RMSE**
2	1856644563
3	1828291277
4	1816389491
5	1826662669
6	1811743008
7	1815585493
8	1817635025
9	1810554111
10	1797064368
11	1796783992
12	1804298868
13	1809809913
14	1814868475
15	1818749549
16	1831109596
17	1841230237
18	1856738081
19	1866925893
20	1881460309
21	1892611532
22	1902113203
23	1914844430
24	1922448158

3.4.2 Internet Traffic Denoising

After removing outliers from a dataset using techniques such as KNN imputation, it is often the case that the remaining data still contains some level of noise or unwanted variability. This noise can obscure the underlying patterns and relationships in the data, making it more difficult to analyze and interpret. Therefore, it can be useful to apply additional noise reduction techniques to further improve the quality of the data. EMD-based noise reduction is one such technique we used to remove unwanted noise and variability from the data, resulting in a smoother and more interpretable signal. We summarized Noise-to-Signal(SNR) ratio in the Table 3.2 for analyzing the signal quality after denoising. The signal-to-noise ratio (SNR) values that we obtained indicate that the EMD-based noise reduction method has significantly improved the quality of the signal. A negative SNR value for the noisy signal (-7.05 dB) indicates that the noise in the signal is actually stronger than the signal itself. This can make it difficult to accurately analyze and interpret the data. However, after applying EMD-based noise reduction, the denoised signal has a much higher SNR value (21.47 dB), indicating that the quality of the signal has been significantly improved relative to the noise. This means that the denoised signal is now much easier to analyze and interpret, and can provide more accurate

Table 3.2: Signal-to-Noise(SNR) Ratio Comparison.

	Signal-to-Noise (SNR) Ratio
Noisy signal SNR	-7.05 dB
Denoised signal SNR	21.47 dB

and reliable insights into the underlying patterns and relationships in the data. In summary, the significant improvement in SNR value for the denoised signal compared to the noisy signal suggests that the EMD-based noise reduction method has been effective in reducing unwanted noise and variability in the data, resulting in a higher quality and more interpretable signal. In Fig. 3.4, we depicted actual traffic, denoised traffic, and the noise in in the dataset.

3.4.3 Optimum Feature Selection for Prediction Model

We identify optimal time-lagged feature set for our deep learning prediction model based on ARIMA model performance in single-step prediction. The AR term in ARIMA refers to the number of lagged values of the dependent variable (i.e., the time series data) that are used to predict the future values. The AR term is denoted by p and it indicates the order of autoregression. For example, ARIMA(p,d,q), where p represents the order of the AR term. Table 3.3 summarize five best-performing ARIMA model

Table 3.3: Top Ten Best-Performing ARIMA Model Parameter Configuration.

SL.	(p, d, q)	AIC
1	(13, 1, 16)	3782.588307
2	(21, 1, 2)	3782.751381
3	(14,1,17)	3783.706900
4	(13, 1, 18)	3785.061129
5	(21, 1, 3)	3785.332434
6	(22, 1, 3)	3786.325772
7	(16, 1, 17)	3786.547150
8	(17,1,17)	3786.761965
9	(22, 1, 4)	3786.964070
10	(20, 1, 2)	3787.326614

hyper-parameters and their corresponding AIC value. We considered AR and MA term ranges from 2 to 24 for finding best combination of model parameters so that we can select optimum AR term represents the most relevant time lag for capturing the temporal dependencies in the traffic data. This time-lagged feature was then incorporated into our deep learning prediction model to enhance its performance. According to our experimental result, the ARIMA model with the lowest AIC value, which was ARIMA (13, 1, 16) with an AIC of 3782.588307, as the best-performing model for our analysis. Therefore, we chosen 13 time-lagged features to train our deep sequence model for traffic prediction.

3.4.4 Proposed Model Performance Analysis

As shown in Table 3.4, the best prediction performance from our baseline models is given by Multiple Seasonal-Trend decompositions (MSTL) with an average gap between actual and predicted traffic of 19.57%. Compared to other baseline models, the MSTL prediction accuracy is higher by 10%-15%. MSTL has the capability of handling multiple seasonality in the time-series data as opposed to ARIMA. The MSTL model uses a Local Polynomial Regression (LOESS) to divide the time series into different seasonalities. As our traffic data set has a daily seasonality, MSTL performs significantly better than other baseline models. ARIMA model requires additional parameters for handling seasonality in the time-series data, and hence it gave a very low prediction accuracy of 68.21% compared to MSTL.

Our proposed deep learning model integrated with KNN based outlier detection outperform conventional model. According to Table 3.4, each prediction model error has been reduced significantly compared to the standard deep learning model. For example, in case of RNN, the prediction error has been reduced from 7.51% to 4.27% when outliers were handled and it is approximately 43% less error compared to traditional RNN. Similarly, LSTM_KNN, LSTM_Seq2Seq_KNN, LSTM_Seq2Seq_ATN_KNN, and GRU_KNN gave better prediction accuracy compared to their corresponding traditional model. From our experimental data in Table 3.4, we noticed approximately 25%, 8%, 9%, and 40% less error we achieved respectively for LSTM_KNN, LSTM_Seq2Seq_KNN, LSTM_Seq2Seq_ATN_KNN, and GRU_KNN by managing outliers before using them to train our model. Overall, it seems that the addition of outlier detection techniques helps in reducing the MAPE and improving the accuracy of the models across different types of recurrent neural networks.

To further improve model performance, we analyze noise in our traffic data. Noise refers to random variations or errors in the data that can affect the accuracy of a model's predictions. Noise can arise from various sources, such as measurement errors, data recording issues, or other random fluctuations. By applying noise reduction techniques, such as smoothing, filtering, or denoising algorithms, the noisy data can be cleaned up, resulting in a more accurate representation of the underlying pattern in the data. This can lead to improved model performance by reducing the impact of noise on the model's predictions.

The conventional RNN model gave us a prediction error of 7.51%, which is almost 12% and 24% lesser than MSTL and ARIMA, respectively. This indicates that the RNN has certain adaptability when dealing with complex temporal sequences with seasonality. Moreover, our proposed RNN_KNN_EMD model prediction error is smaller by more than 3% with an average prediction error of 4.02% with noise reduction and outlier mitigation. This states the effectiveness of our proposed traffic prediction method in handling real-world internet traffic, which might have noise and outliers due to various external and internal factors. The experiment was expanded by including two extensions of RNN called Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) because these models had a greater capacity for knowledge retention from longer sequences than the RNN. Traditional LSTM and GRU outperformed RNN by 2.48% and 1.1% more prediction accuracy, respectively. Since RNN has an inherent problem of vanishing gradient problem in handling longer sequence data, its prediction accuracy was smaller than LSTM and GRU, specifically designed to address RNN limitations. Our experimental results indicate that LSTM and GRU have more power to retain information from sequential data compared to RNN. We further improved LSTM and GRU

Table 3.4: Prediction Accuracy by Different Traffic Model.

Model	MAPE	Mean Accuracy
Baseline model		
AutoARIMA	31.79	68.21
SeasonalNaive	35.53	64.47
ETS	31.80	68.20
MSTL	19.57	80.43
Conventional Deep Learning Model		
RNN	7.51	92.49
LSTM	5.03	94.97
LSTM_Seq2Seq	3.94	96.06
LSTM_Seq2Seq_ATN	3.95	96.05
GRU	6.41	93.59
Proposed Model Integrated with Outlier Management		
RNN_KNN	4.27	95.73
LSTM_KNN	3.77	96.23
LSTM_Seq2Seq_KNN	3.63	96.37
LSTM_Seq2Seq_ATN_KNN	3.60	96.40
GRU_KNN	3.88	96.12
Proposed Model Integrated with Outlier Management and Noise Reduction		
RNN_KNN_EMD	4.02	95.98
LSTM_KNN_EMD	3.30	96.70
LSTM_Seq2Seq_KNN_EMD	3.24	96.76
LSTM_Seq2Seq_ATN_KNN_EMD	3.22	96.78
GRU_KNN_EMD	3.52	96.48

performance by integrating our proposed denoising and outlier detection module. Our proposed LSTM_KNN_EMD and GRU_KNN_EMD perform better than conventional LSTM and GRU. The LSTM_KNN_EMD prediction accuracy is increased by 1.73% compared to LSTM, while for GRU_KNN_EMD, the accuracy improvement was 2.89% than GRU. Since deep sequence models considered only the temporal information for learning, the noise and outlier in the training data ultimately affect its generalization capability resulting in lower accuracy. Our EMD-based noise reduction and empirical rule based outlier mitigation provide traffic with random abrupt changes for model training, which eventually increase the prediction accuracy and decrease the average prediction error between actual and predicted traffic. Finally, we got our best prediction accuracy from our proposed LSTM_Seq2Seq_ATN_KNN_EMD model, where we integrated an attention layer. The extra layer helps our Seq2Seq architecture to extract strong contextual information from the traffic data. The conventional LSTM_Seq2Seq_ATN model without the proposed module provided the lowest prediction error of 3.95% compared to other deep learning models. Moreover, Our proposed LSTM_Seq2Seq_ATN_KNN_EMD prediction accuracy is the highest among all prediction models with 96.48% accurate forecast. Our proposed model performs better than the traditional Seq2Seq model with nearly 1% more prediction accuracy.

For all model, we achieved better prediction accuracy by combining handling both outlier and noise in the traffic data. For example, LSTM_Seq2Seq_KNN_EMD gave approximately 18% less error compared to LSTM_Seq2Seq while LSTM_Seq2Seq_KNN provide approximately 8% less prediction error. So, it is evident that the combination of outlier management and noise reduction would be better approach to deal with real-world internet traffic. For example, our best performing LSTM_Seq2Seq_KNN_EMD model provided us approximately 11% less error compared to LSTM_Seq2Seq_KNN. Therefore, based on all model performance, we can conclude that, traffic analysis is very crucial before using them to develop model. Specially, the outliers and noise in the traffic might affect the model prediction accuracy and it is very important to deal with them first before using traffic data to training and evaluating model.

3.5 Conclusion

Traffic volume forecasting is an essential tool for the ISP industry to assist them in their network capacity planning activities and network investment decisions. Assessing the network traffic trend accurately helps ISPs to define, develop, and adjust their current and new infrastructure and services. Therefore, it is worthwhile to improve the accuracy of internet traffic volume predictions. This study proposes a deep learning methodology that integrates an EMD-based noise reduction and an empirical rule-based outlier detection module. Most of the previous hybrid models use EMD to obtain ensemble prediction models. Unlike the earlier studies, the proposed algorithm is not an ensemble model and does not depend on a specific Intrinsic Mode Function (IMF) for model learning. The proposed algorithm applies EMD method for denoising the original signal to grasp the general tendency of the data. After EMD denoising, the deep learning model is trained on the noise-free dataset. However, the EMD process requires the selection of a stopping criterion to determine the number of IMFs to be extracted. The choice of this criterion can significantly affect the quality of the decomposition and the effectiveness of noise reduction. In future, we plan to explore other methods of noise reduction such as Singular value decomposition (SVD)-based methods, Non-local means (NLM)-based methods, and deep learning-based methods. We identified the point outlier based on the empirical rule, and these points are mitigated with near K values, optimized based on KNN regressor. There are few limitations of using of KNN for parameter optimization. For example, the KNN algorithm can be computationally expensive when the dataset has a large number of features or dimensions. As the number of features increases, the distance between the nearest neighbors can become more similar, which can make it difficult to identify the K nearest neighbors. Also, the choice of distance metric can have a significant impact on the quality of the imputed values, and different distance metrics may be more appropriate for different types of data. Results are evaluated with widely used MAPE and mean accuracy measures to perform a favorable comparison. The proposed method is also compared with traditional statistical and deep sequence models and is trained on the original signal. According to the results, the proposed method outperforms all baseline prediction models. The performance of our proposed algorithm clearly shows its potential in accurately forecasting internet traffic demand compared to the other approaches.

Chapter 4

Overcoming Data Scarcity Challenges: Predicting Internet Traffic in Small ISP Networks with Transfer Learning and Data Augmentation

Abstract: Predicting internet traffic is vital for efficient network management, especially in tasks like anomaly detection, traffic engineering, and capacity planning. For smaller Internet Service Providers (ISPs), limited data availability impedes the creation of reliable prediction models. This paper tackles this challenge by applying transfer learning and data augmentation techniques, specifically for smaller ISP networks. Two models, LSTM_Seq2Seq and LSTM_Seq2Seq_ATN, initially trained on a larger dataset, were fine-tuned on smaller ones. Performance assessment on single-step and multi-step predictions revealed that while both models performed well in single-step predictions, they struggled with multi-step ones, underscoring the inherent difficulties in long-term forecasts. On smaller datasets, LSTM_Seq2Seq generally outperformed LSTM_Seq2Seq_ATN, indicating that greater model complexity does not necessarily yield better results. Moreover, model performance varied across domains, suggesting unique characteristics per domain affecting prediction accuracy. To improve these results, Discrete Wavelet Transform (DWT) was employed to augment the target dataset size, which led to notable performance enhancement. This study offers a practical solution to data scarcity in smaller ISPs, showing that transfer learning and data augmentation can significantly improve prediction model performance.

4.1 Introduction

The rapid growth of internet traffic has led to an increasing demand for accurate and efficient prediction models to manage and optimize network resources. Deep learning techniques have emerged as a powerful tool for time-series prediction, demonstrating promising results in various applications, such as finance, healthcare, and transportation [68]. However, the performance of deep learning models is heavily reliant on the availability of large amounts of training data, which may not always be possible in real-world scenarios [69] due to various reasons. Consequently, this paradigm underlines the necessity for the development of innovative strategies, capable of addressing the challenges of limited training data whilst preserving the robust performance of the models.

Transfer learning has surfaced as a viable solution to the problem posed by limited availability of training data. This strategy is premised on utilizing pre-existing knowledge from a model trained on a related task or domain, effectively addressing the scarcity of data in a new context [70]. This concept of applying learned information from one scenario to another has proven to be particularly beneficial in enhancing the performance of deep learning models, even when operating with restricted training data [71]. Despite these promising attributes, the implementation of transfer learning in the field of time-series prediction, especially in the context of internet traffic prediction, has not been thoroughly investigated. This is a domain where prediction tasks are inherently complex due to unique characteristics of the data such as seasonality, trends, and auto-correlation. It stands to reason, therefore, that transfer learning could potentially be harnessed to improve the accuracy and efficiency of prediction models in this context. In essence, transfer learning could bridge the gap between the necessity of large datasets and the constraints of practical data availability, thus presenting a valuable tool for internet traffic prediction. Its potential effectiveness in this context warrants further exploration, as it could pave the way towards more robust network management and optimization strategies in scenarios with limited training data.

In this study, we turn our attention towards tackling the challenge of constrained training data for internet traffic time-series prediction. We propose the use of a transfer learning framework, driven by the capacity of this approach to boost the predictive accuracy of models operating with limited training data. This enhancement is pivotal to the goal of advancing network resource management and optimization strategies. Our initial step involves the creation of a predictive model using a comparatively large dataset of 8,000 samples. This model caters to both single-step and multi-step predictions, functioning as our source domain. Following this, we use this developed model as a foundational reference to construct additional models, each utilizing smaller datasets of approximately 350 samples, serving as our target domains. By exploiting the potential of transfer learning, our objective is to shed light on its efficacy in amplifying the performance of prediction models specifically designed for internet traffic. Consequently, we aim to demonstrate how this approach can contribute significantly towards augmenting network management and optimization strategies, even when confronted with the challenge of limited training data.

Transfer learning involves leveraging knowledge from a pre-trained model on a related task or domain. While this approach can be very effective, especially when dealing with smaller datasets, the success of transfer learning can still be somewhat dependent on the amount and diversity of the available data. If the target task's dataset is small or lacks diversity, the model might not generalize well, leading to overfitting. This is where data augmentation comes in. Data augmentation techniques generate synthetic data by applying transformations to the original data samples. For time-series data, these transformations can include methods like adding noise, scaling, shifting, or applying wavelet transforms. In image data, it could involve rotation, scaling, translation, flipping, or adding filters. This process increases the size of the dataset and introduces more variability, thereby helping the model generalize better to unseen data.

Given the marked disparity in the sizes of the source and target datasets in our study, there exists a potential hurdle to the effective generalization of the prediction model to the target domains. To thoroughly investigate this issue, we undertook an expansion of our target datasets from a base of approximately 350 samples to a sizeable 2,000 samples. This increase was made possible through the implementation of the Discrete Wavelet Transform (DWT) data augmentation technique. The DWT technique offers a valuable approach to augmenting data by extracting high and low-frequency components, allowing us to effectively increase the size of our target dataset without losing the inherent patterns and correlations present in the original internet traffic data. This augmentation is crucial in bridging the size discrepancy between the source and target datasets and improving the model's generalizability. Subsequently, we carried out a comparative analysis of the model's performance, specifically examining the implications of differing ratios of source to target data. This examination enables us to assess the efficacy of transfer learning under varied conditions and ultimately identify the optimal ratio for model performance in internet traffic time-series prediction. Through this comprehensive analysis, we aim to better understand how data augmentation and transfer learning interplay to enhance the robustness and performance of internet traffic prediction models, especially in scenarios where training data may be limited. This chapter is organized as follows. Section 4.2 describes the literature review of current traffic prediction using deep learning models. Section 4.3 presents the methodology, including dataset preprocessing, deep transfer learning, deep learning models explanation, and experiment details. Section 4.4 summarizes different deep learning methods' performance with smaller datasets by applying both standard learning and transfer learning and draws a comparative picture among them. Finally, section 4.5 concludes our paper and sheds light on future research directions.

4.2 Literature Review

Wu, Qiong, et al. [72] proposed a novel mobile traffic prediction framework that combines the parameter-transfer [73] and domain adaption [74] approaches from deep transfer learning to enhance the model performance with a smaller dataset. The framework functionality is divided into two main parts: build the target prediction model with a massive dataset and then use the pre-trained model knowledge from the source domain, which faces the data-scarcity problem. Furthermore, they applied a GAN-based approach to solving the domain shift problem due to different data distribution between source and target domain. According to their experiment, the GAN-based domain adaption helps their model leverage the knowledge from the source domain to the target domain, giving a better prediction for a smaller dataset.

However, the effectiveness of the adapter may be affected by the quality and representativeness of the synthetic data generated by the GAN. If the GAN fails to generate samples that accurately capture the nuances and complexities of the source data distribution, the adapter may not be able to effectively align the data distributions of the two cities and may even harm the performance of the target model. Also the proposed approach requires a significant amount of computational resources and time to train the GAN and the adapter, especially if the source and target domains are very different. This can be a significant challenge in practice, especially if the data sets are large and complex. Finally, the proposed approach assumes that the source domain and the target domain have similar enough features that can be aligned through the GAN. If the domains are too dissimilar, the proposed approach may not be effective in aligning the data distributions.

Li, Ning, et al. [27] proposed a satellite traffic prediction model based on Gated Recurrent Unit (GRU) architecture that uses the transfer learning and particle filter algorithm for better prediction with a smaller dataset and lower training time. According to their experimental environment, they used similar distribution for the source and target domain in transfer learning. It is unclear how their prediction model will perform if the source and target distribution are asymmetric. The distribution is unlikely to be similar for the source and target domain in the real world, and that's why it is crucial to validate the model performance with data coming from a different distribution than that source domain. They evaluated the proposed approach on a single dataset, which may limit the generalizability of the approach to other datasets and scenarios. Further evaluation on diverse datasets would be necessary to demonstrate the effectiveness and generalizability of the approach.

A wireless cellular traffic prediction model has been proposed by Zeng, Qingtian, et al. [25], which is trained based on a cross-domain dataset. They also used the already trained model's parameters for the target domain by adjusting the parameter's values or transferring the learned features to improve the model accuracy. The experimental results showed the outperformance of the model with the transfer learning capability than the model having no transfer learning. In [75], Dridi, Aicha, et al. proposed a transfer-learning based deep learning model for time series classification and prediction. The transfer-learning technique is adapted in their model mainly for two reasons: better prediction with a smaller dataset and re-adaption of the already trained model for another domain. Their experimental results showed an outperformance of transfer learning in time-series prediction. However, their source and target domain data are drawn from the same distribution, which is unlikely to happen in the real world.

The current works show the usefulness of the transfer learning method in traffic prediction. But we found a lack of investigation in the performance comparison of the deep sequence model such as RNN and its varieties in real-world traffic prediction based on deep transfer learning. In this work, a comprehensive analysis of different deep sequence models has been performed for networks with limited training data based on transfer learning. In addition, we evaluate the performance using five different real-world internet traffic datasets. Internet Service Provider (ISP) networks generate a significant amount of traffic data. This data volume, influenced by factors such as the network size and number of users, can be a challenge to manage, especially when developing machine learning models in situations where data might be sparse. Transfer learning, a technique that applies knowledge from a pre-existing model to a new task, can be a highly effective tool in these scenarios, facilitating the creation of reliable prediction models with fewer training samples.

We propose that predictive models, previously trained on large datasets, can be adapted to forecast traffic in smaller networks. By utilizing the insights from these pre-existing models, we can create accurate models for smaller networks that have a limited amount of data available. This method can mitigate the issues surrounding data scarcity, lessening the need for extensive data when building reliable prediction models. While smaller datasets can be used to train machine learning models, these models often struggle to generalize effectively, as they typically require a greater amount of data to learn complex patterns efficiently. Pre-trained models, on the other hand, can be fine-tuned on smaller datasets, improving their performance on the target task. This approach allows for the transfer of knowledge from pre-existing models, which have already recognized important features and patterns, to new, smaller datasets, resulting in more accurate predictions. Although Generative Adversarial Networks (GANs) could be used to augment smaller datasets, this method requires a considerable amount of time and effort to ensure the synthetic dataset aligns with the distribution of real-world internet traffic. Additionally, it may not always be feasible to create a synthetic dataset that accurately represents the target domain's distribution. Using transfer learning can also reduce the resources required to develop traffic prediction models for ISP networks. Given the variety of network structures and traffic patterns across different ISPs, it can be challenging to create individual models for each network. By using pre-trained models, we can develop accurate models with fewer training samples, maintaining good performance while lessening the need for individual models for each network.

In summary, transfer learning presents a robust solution for overcoming data scarcity and reducing the resources required to develop traffic prediction models for ISP networks. By leveraging pre-existing models, it is possible to build more accurate models for smaller networks with limited data, thereby reducing the need for distinct models for each network.

4.3 Proposed Methodology

In this section, we will discuss our proposed methodology, as presented in Fig. 4.1. We utilized four different datasets for our experimentation, and these datasets will be described in subsection 4.3.1. Additionally, we will discuss our data augmentation technique in subsection 4.3.2. Finally, we will describe transfer learning for our traffic prediction task in subsection 4.3.3 and our prediction model in subsection 4.3.4, respectively.

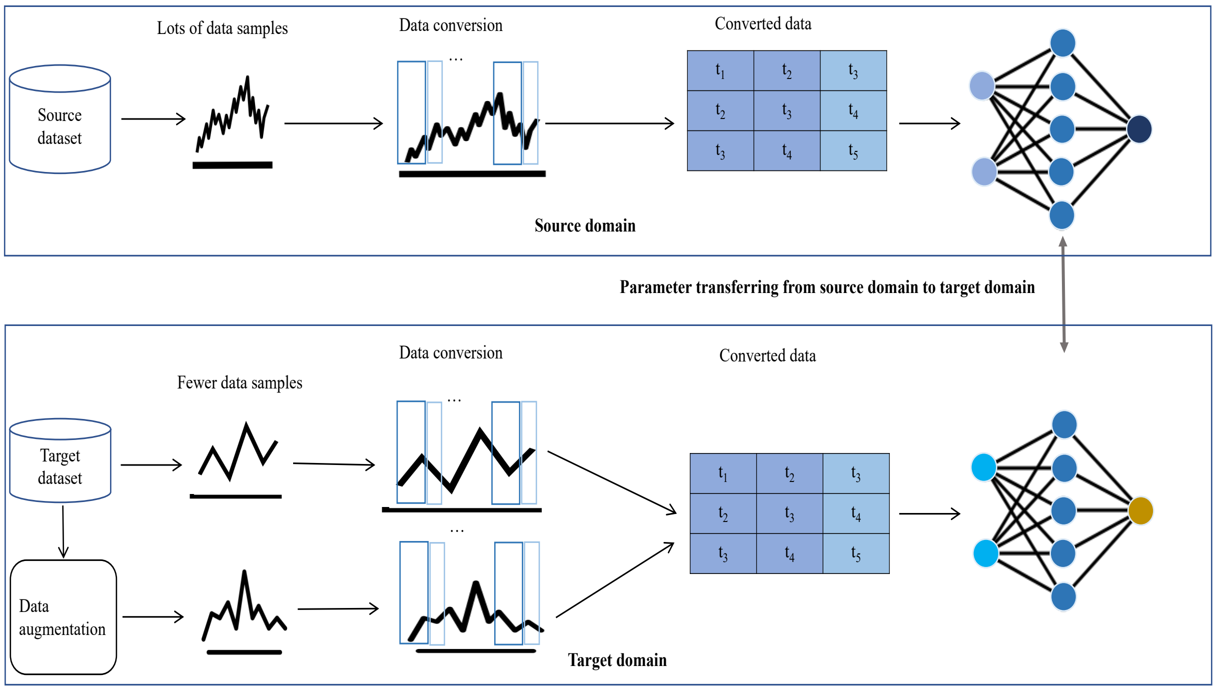


Figure 4.1: High-level overview of our proposed traffic prediction model integrated with transfer learning and data augmentation.

4.3.1 Dataset and Preprocessing Steps

Real internet traffic telemetry on several high-speed interfaces has been used for this experiment. Telemetry data was collected by sampling the value of the SNMP(Simple Network Management Protocol) interface MIB (Management Information Base) counter of a core facing interface on a provider edge router. Samples are taken on 5-minute intervals, with the bps (bit per second) value for the interval being the difference between the samples at each end of the interval times 8. This was a 40 Gbs interface, so at no point in the sampling period were there any discards (ifOutDiscards did not change during the sampling period). A total of four different datasets are used in our experiment. The source domain dataset, Dataset A, consists of 8563 data samples and it is used to build the predictive model for the source task. The other three datasets, Dataset B, C, and D, are comparatively smaller in size, having 363, 369, and 358 data instances, respectively. We used these three datasets for the predictive task in target domain. The smaller datasets for target domain are collected from three specific pair of source and destination node.

Transfer learning was employed in this experiment to address the challenge of limited data availability in the target domain and to leverage the knowledge learned from the larger source domain dataset to improve predictions in the target domain. The source and target domain datasets share similar attributes, such as timestamps and corresponding internet traffic volume at specific time-points. However, there are some key differences between the domains that could impact the transfer learning performance. First, the data distributions differ between the source and target domains. The source domain dataset A has a considerably higher mean traffic volume $(8,364,386,961)$ compared to the target domain datasets B $(552,753,735.6)$, C $(729,400,375.1)$, and D $(553,417,309.4)$ according to Table 4.1. This indicates that the source domain experiences significantly more traffic on average than the target domains. Furthermore, the standard deviation and variance values also exhibit differences between the source and target domains. The source domain dataset A has a higher standard deviation $(4,098,702,833)$ and variance (1.68×10^{19}) compared to the target domain datasets B (standard deviation: 210,546,994.7, variance: 4.43×10^{16}), C (standard deviation: $287,908,086.1$, variance: 8.29×10^{16}), and D (standard deviation: $210,166,274.1$, variance: 4.42^{16}). This implies that the source domain exhibits greater variability in traffic volume compared to the target domains. The data distributions for source and target domains dataset is shown in Figure 4.2.

The skewness values for datasets A, B, C, and D reveal differences in the asymmetry of their data distributions. Dataset A has a positive skewness, indicating a longer right tail with more instances of higher traffic volume values. Datasets B and D exhibit slightly negative skewness, suggesting somewhat longer left tails with more instances of lower traffic volume values, though the asymmetry is not very strong in either case. Dataset C has a skewness value close to 0, indicating a nearly symmetric distribution. These differences in skewness among the datasets could impact the transfer learning performance, as models may need to adapt to varying data asymmetries between the source and target domains. Despite these differences, our transfer learning approach aims to leverage the shared attributes and knowledge gained from the larger source domain dataset to improve predictions in the target domain.

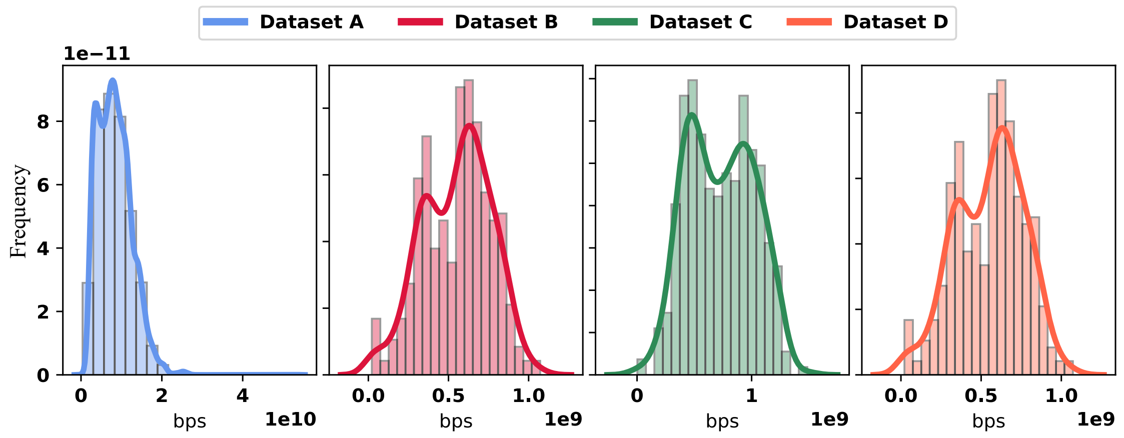


Figure 4.2: Data distribution of source and target domain datasets.

Table 4.1: Summary Statistics Comparison across Different Datasets.

	Mean	STD	VAR	Skewness
Source Domain Dataset A	8364386961	4098702833	1.68E+19	0.793776342
Target Domain Dataset B	552753735.6	210546994.7	4.43E+16	-0.249849673
Target Domain Dataset C	729400375.1	287908086.1	8.29E+16	0.057702382
Target Domain Dataset D	553417309.4	210166274.1	4.42E+16	-0.258100096

Algorithm 1: Discrete Wavelet Transform (DWT) Data Augmentation

```
1:  **Input:** Original dataset $D_{orig}$, wavelet function $w$ (e.g., 'db4'), number of levels $l$,
    augmentation factor range $[a, b]$ (e.g., $[0.5, 1.5]$)
2:  **Output:** Augmented dataset $D_{aug}$
3:  for each data point $x_i$ in $D_{orig}$ do
4:      Perform wavelet decomposition of $x_i$ using $w$ and $l$ to obtain sets of coefficients,
        $C_i = \{c_{i1}, c_{i2}, ..., c_{in}\}$
5:      for each set $c_{ij}$ in $C_i$ corresponding to higher frequency bands do
6:          $c'_{ij} = c_{ij} \times f_{ij}$ where $f_{ij} \sim Uniform(a, b)$
7:      end for
8:      Perform inverse wavelet transform on modified coefficients $C'_i$ to generate new data
        point $x'_i$
9:      $D_{aug} = D_{aug} \cup \{x'_i\}$
10: end for
11: $D_{exp} = D_{orig} \cup D_{aug}$
12: return $D_{exp}$
```

4.3.2 Data Augmentation

Discrete Wavelet Transform (DWT) is a data augmentation technique employed in this study to address the challenge of limited data in the target domains, which can significantly impact the performance of machine learning models [76]. DWT is a powerful tool for time-frequency analysis and has been widely used for signal processing, image compression, and feature extraction [77]. By augmenting the dataset with new, meaningful information using DWT, this study aims to enhance the performance of the LSTM_Seq2Seq and LSTM_Seq2Seq_ATN models by providing a richer and more diverse representation of the target domains. The process of applying the DWT technique to the smaller datasets of the target domains involves several steps as below:

Step 1: Perform wavelet decomposition on the original data using a suitable wavelet function (e.g., 'db4', a Daubechies wavelet) and a specified number of levels. Wavelet decomposition is a multi-scale analysis technique that breaks down the data into multiple frequency bands, each represented by a set of coefficients.

Step 2: Modify the detail coefficients of the higher frequency bands by multiplying them with random factors between 0.5 and 1.5 and we empirically selected this range for coefficient modification. This step introduces controlled variability in the data, while retaining the essential characteristics of the original signal.

Step 3: Perform inverse wavelet transform on the modified coefficients to generate new, augmented data points that preserve the overall structure of the original data.

Step 4: Combine the original data and augmented data to create an expanded dataset for each target domain.

By employing the DWT data augmentation technique summarized in Algorithm 1, the study anticipates improving the performance of the LSTM_Seq2Seq and LSTM_Seq2Seq_ATN models by enabling them to learn more diverse and robust representations of the target domains. This enhancement is expected to lead to better generalization and reduced overfitting when tested on smaller datasets.

4.3.3 Deep Transfer Learning

Transfer learning is an deep learning or machine learning optimization approach in which knowledge is transferred from one domain to another similar domain. We can formally define transfer learning in terms of domain and task. There are two domains such as source and target domain involved in transfer learning while the domain consists of a feature space X and probability distribution $P(X)$ where $X = \{x_1, x_2, ..., x_n\} \in X$. The task T in transfer learning is consists of label space Y and an objective function $f : X \longrightarrow Y$ while X is a feature space for particular domain, $\{X, P(X)\}$. In transfer learning, the source domain D_S and target domain D_T consists of two different task T_S and T_T and the purpose of the transfer learning is to assist the task in target domain to perform better using the knowledge in D_S and T_S.

In this study, we employ transfer learning summarized in Algorithm 2 to adapt a pre-trained Encoder-Decoder LSTM model from a source domain to a target domain, particularly when the

Algorithm 2: Transfer Learning for Time Series Prediction

```
 1: Initialize lists for the number of future time steps $n_{future}$, past time steps $n_{past}$
 2: Prepare the input sequences with $n_{past}$ and $n_{future}$ time steps
 3: Split the input data into training and testing sets
 4: for each $n_{future}$ in future_list do
 5:    for each $n_{past}$ in past_list do
 6:       Prepare the input sequences with $n_{past}$ and $n_{future}$ time steps
 7:       Split the input data into training and testing sets
 8:       Load the pre-trained source model $M_s$
 9:       Remove the last dense layer $D_s$ from the source model $M_s$
10:       Add a new dense layer $D_t$ with the target domain's output features to the source
          model $M_s$
11:       Create a new model $M_t$ with the modified source model's input and the new dense
          layer's output: $M_t = M_s \cup D_t$
12:       Freeze all layers except the new dense layer $D_t$ by setting their 'trainable' attribute to
          'False'
13:       Compile the model with the Adam optimizer (learning rate = 0.001) and the Huber
          loss function
14:       Train the model on the target domain data $D_{train}^{(T)}$ for a predefined number of epochs
15:       Unfreeze all layers by setting their 'trainable' attribute to 'True'
16:       Compile the model with the Adam optimizer (learning rate = 0.0001) and the Huber
          loss function
17:       Fine-tune the model on the target domain data $D_{train}^{(T)}$ for the same number of epochs
18:       Evaluate the adapted model on the testing data $D_{test}^{(T)}$
19:    end for
20: end for
```

target domain dataset is relatively small. The rationale behind this approach is to leverage the knowledge acquired by the model on the source domain, where abundant data is available, to improve its performance on the target domain with limited data. This technique reduces the risk of overfitting and allows the model to generalize better to the target domain. To implement the transfer learning process, we first remove the output layer of the pre-trained source model, which is tailored to the source domain's prediction task, and replace it with a new dense layer designed for the target domain. This modification ensures that the output of the adapted model is compatible with the target domain's prediction task. During the initial training phase, we freeze all layers of the model except the newly added dense layer by setting their 'trainable' attribute to 'False'. This approach allows the model to focus on learning the target domain-specific characteristics without altering the weights of the other layers, which have already captured useful information from the source domain. We use a higher learning rate of 0.001 during this phase to encourage faster convergence. Subsequently, we unfreeze all layers in the model and fine-tune it on the target domain data with a reduced learning rate of 0.0001. This lower learning rate ensures that the fine-tuning process makes small, precise updates to the model's weights, allowing it to adapt to the target domain while preserving the knowledge acquired from the source domain. Throughout the training process, we employ the Adam optimizer and the Huber loss function, which have been proven effective in training deep neural networks, such as LSTM models. By utilizing transfer learning in combination with the Encoder-Decoder LSTM model, we effectively adapt the model to the target domain while leveraging the knowledge gained from the source domain. This approach results in improved performance on the smaller target domain dataset, demonstrating the potential benefits of transfer learning for time series prediction tasks in various domains.

4.3.4 Model Selection and Implementation

We used two differnet model: LSTM_Seq2Seq and LSTM_Seq2Seq_ATN for our prediction task. Both models are based on Long Short-Term Memory (LSTM) networks, which are a type of recurrent neural network (RNN) designed to handle long-range dependencies in sequential data. These models are particularly well-suited for time series prediction tasks, such as traffic forecasting. The addition of the attention mechanism in the LSTM_Seq2Seq_ATN model can potentially improve performance by allowing the model to focus on the most relevant parts of the input sequence during prediction.

LSTM_Seq2Seq (LSTM Encoder-Decoder)

The Encoder-Decoder LSTM summarized in Algorithm 3 for time series prediction is a deep learning technique that leverages the power of recurrent neural networks to effectively capture temporal dependencies in time series data. The model consists of two key components: an encoder LSTM and a decoder LSTM. The encoder LSTM processes the input sequence of past time steps and generates a final hidden state and cell state, which encapsulate the relevant information from the input. This final hidden state is then replicated and used as the initial input for the decoder LSTM. The decoder LSTM predicts future time steps based on the encoder's final hidden and cell states. To obtain the final prediction, a TimeDistributed dense layer is applied to the output sequence of the decoder LSTM. The model is compiled using the Adam optimizer and the Huber loss function and trained on the given training data. This approach effectively captures the temporal dynamics of the input sequence and allows for accurate predictions of future time steps in the time series data.

The Encoder-Decoder LSTM algorithm for time series prediction uses various parameters that contribute to the model's performance. The choice of 100 hidden units for both the encoder and decoder LSTM layers balances model complexity and computational efficiency. A higher number of hidden units would increase the capacity to capture complex patterns in the data but could lead to overfitting and longer training times. Conversely, fewer hidden units might reduce the model's ability to learn and represent the underlying structure of the time series data. The use of the Adam optimizer is motivated by its adaptive learning rate, which adjusts based on the gradient's magnitude, resulting in faster convergence compared to traditional gradient descent methods. This makes it suitable for training deep neural networks such as LSTM models. The Huber loss function is employed as it combines the best properties of Mean Squared Error (MSE) and Mean Absolute Error (MAE) loss functions. It is less sensitive to outliers than MSE and has a smoother gradient than MAE, allowing for a more stable learning process. The TimeDistributed dense layer is applied to the output sequence of the decoder LSTM to obtain the final prediction, enabling the model to process each time step independently while maintaining the same weights across all time steps. This approach helps the model generalize to varying sequence lengths and makes it more robust to the intricacies of time series data. These parameter choices, while not exhaustive, provide a solid foundation for the Encoder-Decoder LSTM model. Further fine-tuning or hyperparameter optimization could be performed based on the specific dataset and problem domain to improve the model's performance.

LSTM_Seq2Seq_ATN (LSTM Encoder-Decoder with Attention)

The model starts by initializing lists for the number of future time steps (n_{future}) and past time steps (n_{past}). Then, the input sequences are prepared using these time steps, and the input data is split into training and testing sets. For each combination of future and past time steps, the algorithm prepares input and output tensors for training and testing data. It initializes an encoder LSTM layer with 100 hidden units and processes the input sequence using this layer, obtaining the final hidden state (h_T), cell state (c_T), and encoder hidden states ($h = (h_1, h_2, \ldots, h_T)$). Next, the final hidden state (h_T) is replicated n_{future} times to initialize the decoder input. A decoder LSTM layer with 100 hidden units is initialized, using the final hidden state (h_T) and cell state (c_T) from the encoder. The decoder input is then processed with the decoder LSTM. Attention scores are computed between the encoder and decoder hidden states, and a context vector is calculated as a weighted sum of the encoder hidden states. The context vector is concatenated with the decoder hidden states, and a TimeDistributed dense layer is applied to the combined context to obtain the final prediction (y'). Finally, the model is compiled using the Adam optimizer and Huber loss function. It is trained on the training data and evaluated on the testing data for each combination of future and past time steps. The algorithmic steps of the model are summarized in Algorithm 4.

Algorithm 3: Encoder-Decoder LSTM for Time Series Prediction

```
1: Initialize lists for the number of future time steps $n_{future}$, past time steps $n_{past}$
2: Prepare the input sequences with $n_{past}$ and $n_{future}$ time steps
3: Split the input data into training and testing sets
4: for each $n_{future}$ in future_list do
5:     for each $n_{past}$ in past_list do
6:         Prepare the input sequences with $n_{past}$ and $n_{future}$ time steps
7:         Split the input data into training and testing sets
8:         Initialize the encoder LSTM layer with 100 hidden units
9:         Process the input sequence $x = (x_1, x_2, \dots, x_T)$ with the encoder LSTM:
10:        for $t = 1$ to $T$ do
11:            $h_t = \text{LSTM}(x_t, h_{t-1})$
12:        end for
13:        Obtain the final hidden state $h_T$ and cell state $c_T$
14:        Replicate the final hidden state of the encoder $n_{future}$ times:
15:        $decoder\_input = \text{RepeatVector}(n_{future})(h_T)$
16:        Initialize the decoder LSTM layer with 100 hidden units, using the final hidden state
           $h_T$ and cell state $c_T$ from the encoder
17:        Process the decoder input with the decoder LSTM:
18:        for $t = 1$ to $n_{future}$ do
19:            $y_t = \text{LSTM}(y_{t-1}, h_{t-1})$
20:        end for
21:        Obtain the output sequence $y = (y_1, y_2, \dots, y_{n_{future}})$
22:        Apply a TimeDistributed dense layer to the output sequence $y$ to obtain the final
           prediction $y'$:
23:        $y' = \text{TimeDistributed}(\text{Dense}(1))(y)$
24:        Compile the model using the Adam optimizer and the Huber loss function
25:        Train the model on the training data, and evaluate on the testing data.
26:    end for
27: end for
```

Algorithm 4: Encoder-Decoder LSTM with Attention for Time Series Prediction

```
1:  Initialize lists for the number of future time steps $n_{future}$, past time steps $n_{past}$
2:  Prepare the input sequences with $n_{past}$ and $n_{future}$ time steps
3:  Split the input data into training and testing sets
4:  **for** each $n_{future}$ in future_list **do**
5:      **for** each $n_{past}$ in past_list **do**
6:          Prepare input and output tensors for the training and testing data
7:          Initialize the encoder LSTM layer with 100 hidden units
8:          Process the input sequence $x = (x_1, x_2, \dots, x_T)$ with the encoder LSTM:
9:          **for** $t = 1$ to $T$ **do**
10:             $h_t = \text{LSTM}(x_t, h_{t-1})$
11:         **end for**
12:         Obtain the final hidden state $h_T$, cell state $c_T$, and encoder hidden states
            $h = (h_1, h_2, \dots, h_T)$
13:         Initialize the decoder input with the final hidden state $h_T$ replicated $n_{future}$ times
14:         Initialize the decoder LSTM layer with 100 hidden units, using the final hidden state
            $h_T$ and cell state $c_T$ from the encoder
15:         Process the decoder input with the decoder LSTM:
16:         **for** $t = 1$ to $n_{future}$ **do**
17:             $y_t = \text{LSTM}(y_{t-1}, h_{t-1})$
18:         **end for**
19:         Compute the attention scores between the encoder and decoder hidden states
20:         $attention = \text{softmax}(dot(decoder\_stack\_h, encoder\_stack\_h, axes = [2, 2]))$
21:         Compute the context vector as a weighted sum of the encoder hidden states
22:         $context = dot(attention, encoder\_stack\_h, axes = [2, 1])$
23:         Concatenate the context vector and the decoder hidden states
24:         $decoder\_combined\_context = concatenate(context, decoder\_stack\_h)$
25:         Apply a TimeDistributed dense layer to the decoder combined context to obtain the
            final prediction $y'$:
26:         $y' = \text{TimeDistributed(Dense(1))}(decoder\_combined\_context)$
27:         Compile the model using the Adam optimizer and the Huber loss function
28:         Train the model on the training data, and evaluate on the testing data
29:     **end for**
30: **end for**
```

Table 4.2: Prediction Model Performance Summary in Source Domain.

	LSTM_Seq2Seq	**LSTM_Seq2Seq_ATN**
	MAPE	**MAPE**
Step 1	3.94	3.95
Step 6	5.08	5.06
Step 9	5.88	5.82
Step 12	6.74	6.77

4.4 Analysis of Experimental Results

In this section, we first evaluate the performance of our prediction models in the source domain for both single and multi-step prediction in subsection 4.4.1. Then, we transfer the knowledge of the source domain model for the prediction task in the target domain and evaluate the model performance before and after data augmentation, respectively, in subsections 4.4.2 and 4.4.3.

4.4.1 Prediction Model Performance in Source Domain

The performance results of the source domain models, LSTM_Seq2Seq and LSTM_Seq2Seq_ATN, on the larger dataset demonstrated their effectiveness in predicting internet traffic. The MAPE values for both single-step and multi-step predictions are summarized in Table 4.2. For single-step predictions, both models achieved consistently low MAPE values. LSTM_Seq2Seq achieved a MAPE of 3.94%, while LSTM_Seq2Seq_ATN achieved a slightly higher MAPE of 3.95%. These results indicate accurate forecasts of the immediate future values in the larger dataset. Moving to multi-step predictions, the MAPE values slightly increased as the prediction horizon extended. At Step 6, both models had similar MAPE values, with LSTM_Seq2Seq at 5.08% and LSTM_Seq2Seq_ATN at 5.06%. The MAPE values continued to increase at Steps 9 and 12, with LSTM_Seq2Seq reaching 5.88% and 6.74%, respectively, and LSTM_Seq2Seq_ATN reaching 5.82% and 6.77%, respectively. These findings highlight the challenges associated with making accurate long-term forecasts due to accumulated errors over time. Interestingly, the comparison between LSTM_Seq2Seq and LSTM_Seq2Seq_ATN models reveals that the inclusion of the attention mechanism in LSTM_Seq2Seq_ATN did not significantly improve the models' predictive accuracy. The performance of attention-based models heavily relies on having sufficient training data to learn the attention weights effectively. If the size of the dataset is limited or if the attention weights are not able to be trained robustly due to data sparsity, the attention mechanism may not be able to effectively capture the important information for prediction. Both models performed similarly for both single-step and multi-step predictions. The achieved MAPE values have several implications for achieving the research objectives and the potential applications of the prediction models. The accurate single-step predictions demonstrate the models' effectiveness in real-time monitoring and decision-making tasks, such as network management, capacity planning, and anomaly detection in internet traffic. However, it is crucial to acknowledge the limitations of multi-step predictions. The increasing MAPE values with an extended forecasting horizon indicate the inherent challenges of making accurate long-term forecasts. Stakeholders should consider these limitations and manage expectations accordingly when using the models for longer-term projections. The comparable performance

Table 4.3: Multi-step Model Performance Summary Before Data Augmentation in Target Domain.

	Target Domain A	**Target Domain B**	**Target Domain C**
	LSTM_Seq2Seq		
	MAPE	MAPE	MAPE
6 step	16.96	22.38	18.05
9 step	17.88	21.77	18.09
12 step	20.99	19.73	19.83
	LSTM_Seq2Seq_ATN		
6 step	17.13	22.81	17.18
9 step	18.69	23.06	19.06
12 step	21.17	24.15	19.64

between LSTM_Seq2Seq and LSTM_Seq2Seq_ATN models suggests that the attention mechanism may not provide substantial benefits for the considered prediction tasks. This finding prompts further exploration of alternative techniques or architectures to improve multi-step prediction accuracy.

4.4.2 Source Domain Model Performance on the Target Domain Before Data Augmentation

The source domain dataset, with 8,352 data points, is significantly larger than the average target domain dataset, which has only 350 data points. This size difference can impact the performance of the transfer learning models when predicting internet traffic for the target domains.

Based on the provided results in Table 4.3, it can be observed that as the prediction steps increase from 6 to 12, the Mean Absolute Percentage Error (MAPE) generally increases for both LSTM_Seq2Seq and LSTM_Seq2Seq_ATN models across all target domains. This suggests that the models become less accurate in predicting internet traffic as the forecasting horizon grows, which is a common trend in time series forecasting due to increasing uncertainty with the forecast horizon. For example, in Target Domain A, the LSTM_Seq2Seq model has a MAPE of 16.96 for 6-step prediction, increasing to 17.88 and 20.99 for 9-step and 12-step predictions, respectively. Similarly, the LSTM_Seq2Seq_ATN model exhibits an increase in MAPE from 17.13 to 21.17 as the prediction steps grow from 6 to 12.

When comparing the two models, it can be seen that the LSTM_Seq2Seq_ATN model generally performs worse than the LSTM_Seq2Seq model, with higher MAPE values across all target domains and prediction steps. This might indicate that the attention mechanism in the LSTM_Seq2Seq_ATN model does not provide additional benefits for this specific task, or it may not have been optimized properly given the significant difference in dataset sizes between the source and target domains. In addition, the performance of the models varies across different target domains. For instance, both models perform best on Target Domain B at the 12-step prediction, while they show the worst performance on Target Domain B at 6-step and 9-step predictions. This suggests that the domains have different characteristics that affect the model's performance, and these characteristics may not be fully captured by the source domain model due to the dataset size discrepancy.

To summarize, the transfer learning models' performance on the smaller target domain datasets is influenced by the size difference between the source and target domains, as well as the varying characteristics of each target domain. A simpler model like LSTM_Seq2Seq might be more suitable for these specific target domains than the more complex LSTM_Seq2Seq_ATN model. Further fine-tuning of the models or the development of new models tailored to the specific characteristics of each target domain could lead to improved performance in internet traffic prediction tasks. This experiment offers valuable insights into the application of transfer learning for internet traffic prediction tasks using different model architectures. The findings have several implications that can inform future research and applications in this domain.

Firstly, the experiment emphasizes the importance of choosing the right model architecture for a given task. The simpler LSTM_Seq2Seq model demonstrated better performance than the more complex LSTM_Seq2Seq_ATN model, which incorporated an attention mechanism. This finding suggests that adding complexity does not always guarantee improved performance and that it is crucial to consider the specific characteristics and requirements of the task at hand. Secondly, the results indicate that the performance of transfer learning models can be influenced by the size of the source and target domain datasets. The significant difference in dataset sizes between the source domain (8,352 data points) and target domains (average of 350 data points) may have affected the models' ability to generalize to the smaller target domains. This finding highlights the need for further research on the impact of dataset size when applying transfer learning in time series prediction tasks. Additionally, the experiment underscores the varying performance of the models across different target domains. This suggests that each target domain has distinct characteristics that influence the models' performance, necessitating a more in-depth understanding of these domain-specific features to optimize model performance. As a result, future work could focus on developing models that can better adapt to the specific characteristics of each target domain, potentially improving prediction accuracy. Furthermore, the increasing MAPE values with longer prediction horizons highlight the challenges associated with predicting internet traffic over longer timeframes. This observation implies the need for better strategies to manage uncertainty in longer-term forecasts, such as incorporating additional features, using ensemble methods, or improving the underlying models.

In conclusion, the experiment provides valuable insights into the performance of transfer learning models for internet traffic prediction tasks across smaller target domains. The findings emphasize the need for careful model selection, consideration of dataset size, and a deeper understanding of domain-specific characteristics. These insights can inform the development of more effective models and strategies for improving internet traffic prediction performance in future work.

4.4.3 Source Domain Model Performance on the Target Domain After Data Augmentation

After applying the DWT data augmentation technique and increasing the target dataset size to approximately 2,000, both the LSTM_Seq2Seq and LSTM_Seq2Seq_ATN models showed improved performance across the target domains as shown in Table 4.4

For the LSTM_Seq2Seq model, the MAPE values decreased compared to the previous

Table 4.4: Multi-step Model Performance Summary After Data Augmentation in Target Domain.

	Target Domain A	**Target Domain B**	**Target Domain C**
	LSTM_Seq2Seq		
	MAPE	MAPE	MAPE
6 step	12.27	15.86	12.18
9 step	12.6	14.28	13.07
12 step	13.1	13.83	13.53
	LSTM_Seq2Seq_ATN		
6 step	13.30	16.99	12.61
9 step	13.25	15.62	13.69
12 step	13.19	15.05	14.46

results. In Target Domain A, the MAPE ranged from 12.27 to 13.10 for 6-step to 12-step predictions. In Target Domain B, the MAPE ranged from 13.83 to 15.86, and in Target Domain C, the MAPE ranged from 12.18 to 13.53. These lower MAPE values indicate that the LSTM_Seq2Seq model achieved better accuracy in predicting internet traffic in the target domains after data augmentation. Similarly, the LSTM_Seq2Seq_ATN model also exhibited improved performance. In Target Domain A, the MAPE ranged from 13.19 to 13.30 for 6-step to 12-step predictions. In Target Domain B, the MAPE ranged from 15.05 to 16.99, and in Target Domain C, the MAPE ranged from 12.61 to 14.46. Although the LSTM_Seq2Seq_ATN model had slightly higher MAPE values compared to the LSTM_Seq2Seq model, it still demonstrated improved performance after data augmentation. The performance improvement can be attributed to the increased dataset size achieved through DWT data augmentation. The expansion of the target domain dataset through DWT data augmentation provided a larger and more diverse set of samples for training the models. The increased dataset size enabled the models to learn more effectively and capture a broader range of patterns present in the target domains. By expanding the target dataset, the models were exposed to a more diverse and representative set of samples. This allowed the models to capture a wider range of underlying patterns and enhance their ability to generalize to unseen data. The larger dataset size also helped mitigate the effects of limited data availability in the smaller target domains, enabling the models to learn more effectively and improve their prediction accuracy. The increased dataset size may have facilitated better adaptation of the models to the specific characteristics of each target domain and aided in fine-tuning their parameters accordingly.

Overall, the DWT data augmentation technique and the resulting increase in dataset size had a positive impact on the performance of both the LSTM_Seq2Seq and LSTM_Seq2Seq_ATN models. These findings underscore the importance of data augmentation in improving model performance in time series prediction tasks, particularly when dealing with limited data availability in smaller target domains. The results highlight the value of data augmentation techniques, such as DWT data augmentation, in expanding the target domain dataset. Augmenting the data helped mitigate the limitations of smaller target domain datasets and improve the models' performance by providing more representative samples for training. The success of the transfer learning approach, even with a smaller target domain dataset, suggests that leveraging knowledge from a larger source domain dataset can still provide benefits. Transfer learning allowed the models to capture domain-agnostic patterns from the source domain and adapt them to the target domains, leading to improved performance. The ratio between the target domain and source domain data is an important factor to consider. While the target domain dataset was still smaller than the source domain dataset, the augmentation techniques helped bridge the gap and improve the model's ability to generalize to the target domains. This finding emphasizes the significance of balancing the size and representativeness of the target domain data to achieve optimal performance. Overall, the results indicate that augmenting the target domain dataset and leveraging transfer learning can effectively improve the performance of internet traffic prediction models. The findings reinforce the importance of data augmentation techniques and highlight the benefits of incorporating prior knowledge from larger source domain datasets, even when the target domain dataset is relatively small.

4.5 Conclusion

In this research, we utilized transfer learning techniques and data augmentation methods to predict internet traffic patterns in smaller network datasets. Our investigation aimed to alleviate the challenges of scarce data in smaller Internet Service Provider (ISP) networks, which makes the development of reliable prediction models a difficult task. The models LSTM_Seq2Seq and LSTM_Seq2Seq_ATN, originally trained on larger datasets, were transferred and fine-tuned on smaller datasets. This approach successfully reduced the need for extensive data collection in smaller ISP networks, saving time and resources.

The findings revealed that both LSTM_Seq2Seq and LSTM_Seq2Seq_ATN models perform comparably well on single-step predictions in the larger dataset, indicating their robustness and reliability. In contrast, the models struggled with multi-step predictions, revealing inherent challenges in long-term forecasts due to accumulated errors over time. Upon transfer to the smaller datasets, it was observed that LSTM_Seq2Seq performed generally better than LSTM_Seq2Seq_ATN. The attention mechanism in LSTM_Seq2Seq_ATN did not seem to provide additional benefits for the prediction tasks at hand, suggesting that increasing model complexity does not always lead to improved performance. It is crucial to consider the specific characteristics and requirements of the task when selecting a model. The significant difference in dataset sizes between the source domain and target domains may have affected the models' ability to generalize to the smaller target domains. The models' performance varied across different target domains, suggesting that each domain has distinct characteristics that influence model performance.

Data augmentation via the Discrete Wavelet Transform (DWT) was utilized to enhance the target dataset size. This significantly improved the performance of both models, suggesting that data augmentation techniques can play a crucial role in improving model performance, especially when dealing with limited data. Despite the promising results, the research has limitations that provide directions for future work. First, the attention mechanism in the LSTM_Seq2Seq_ATN model may need further fine-tuning or alternative approaches to improve its performance. Second, while DWT was effective in this study, other data augmentation techniques could also be explored to further increase the diversity and representativeness of the smaller datasets. Third, we might also need to look into further improving the multi-step prediction performance of these models, as it is crucial for longer-term projections. Lastly, the discrepancies observed in the performance across different target domains suggest a need for further investigation into domain-specific features. Understanding these features better could help design models that are more adaptable and thus more accurate for each specific domain. We also plan to explore other architectures and strategies to handle the uncertainty associated with longer prediction horizons.

In summary, the study presents an approach to tackle the problem of data scarcity in smaller ISP networks. The application of transfer learning and data augmentation techniques show promise in enhancing prediction model performance. This work paves the way for further research in this direction, emphasizing the importance of choosing the right model, augmenting the data effectively, and understanding domain-specific characteristics.

Chapter 5

An Evaluation of Machine Learning Models for Internet Traffic Prediction: Identical vs. Out-of-Distribution Data Samples

Abstract: Internet traffic prediction, crucial for proactive network management and Quality of Service (QoS) in self-organizing networks (SON), often faces challenges due to complex, non-linear, and non-stationary real-world traffic. While existing machine learning and deep learning models demonstrate strong performance, they typically operate on the assumption that data samples are independent and identically distributed (IID), an often untrue presumption in real scenarios. This mismatch leads to performance inconsistencies between IID and out-of-distribution (OOD) samples. Our research analyzes various boosting algorithms and deep sequence models using both IID and OOD samples from diverse actual traffic datasets. We report a notable accuracy drop with OOD samples and propose a hybrid architecture combining deep sequence models and discrete wavelet transformation (DWT). Training models on decomposed hierarchical components instead of the original data, our hybrid models exhibit improved prediction accuracy for both IID and OOD samples, also substantially reducing the performance gap between IID and OOD predictions. The results affirm the superiority of our methodology over traditional models, highlighting its robustness and adaptability in real-world traffic prediction.

5.1 Introduction

The Internet and its applications have become fundamental means of communication for all classes of consumers in modern society, resulting in significant internet traffic. Predicting internet traffic is crucial for any network as it helps to design better strategic plans for the business. Internet traffic prediction can be beneficial in traffic engineering, performance diagnostics, failure recovery, anomaly detection, load balancing, etc. [78]. It also plays a vital role in dynamic bandwidth reservation and allocation, congestion control, admission control [79], and privacy-preserving routing [80] etc., which assure better Quality of Service (QoS), Quality of Experience (QoE). Therefore, a substantial number of research works have been proposed on predicting internet traffic using statistical models, machine learning, and deep learning models.

Due to time-variability, long-term correlation, self-similarity, suddenness, and chaos [81] in internet traffic, it is challenging to develop an accurate prediction model. The majority of recent studies on traffic prediction used data from independent and identical distribution (IID) for model training and testing, i.e., $P_{tr}(X, Y) = P_t e(X, Y)$ where X is the feature set and Y its corresponding target. But the train and test distribution can be different for several reasons, such as the temporal/spatial evolution of data or the sample selection bias in the data collection process. Therefore, there is a high probability of dealing with a dataset different from the training set during the model's deployment phase in the real world. Furthermore, internet traffic is volatile due to heterogeneous sources. Hence, it is challenging to design a robust predictive model that can perform better for both IID and out-of-distribution (OOD) data samples. In addition, there is considerable evidence that AI-based prediction models give unexpected outcomes caused by selection biases, confounding factors, and other biases in the data[82]. As a result, OOD generalization refers to machine learning techniques in which there is a distribution shift between P_{tr} and P_{te}. We must specify how the testing distribution varies from the training distribution in OOD scenarios.

While existing works, based on machine learning and deep learning, have demonstrated promising results in traffic prediction, they have primarily assumed that the train and test data samples come from a similar distribution. This leaves a gap in understanding how these models perform when presented with an inference dataset that has a slightly different or completely unknown distribution than the training dataset. Without this assessment, the effectiveness of these predictive models in unknown scenarios remains unclear. Addressing this challenge, our study evaluates the performance of several boosting and deep sequence models using both IID and OOD data samples. We discovered a significant drop in the prediction accuracy of these classical machine learning models for OOD samples compared to IID data. To address this problem, we propose a novel approach that integrates a hybrid deep learning model with discrete wavelet transformation (DWT). The integration of DWT allows us to extract multiple lower resolution levels from the original time series data, capturing the detailed characteristics of internet traffic [83]. Training the hybrid model on these detailed components offers enhanced prediction accuracy for both IID and OOD data samples, and provides a promising avenue for future exploration in the field of internet traffic prediction.

In summary, this chapter presents our investigation into the effectiveness of boosting algorithms and deep sequence algorithms on OOD data samples and introduces a potential solution in the form of a hybrid machine learning model incorporating DWT. We aim to contribute to the body of knowledge in traffic prediction, helping to develop more effective strategies for network management and enhancing the QoS and QoE. This chapter is organized as follows. Section 5.2 describes the literature review of current traffic prediction using machine learning models. Section 5.3 presents the methodology, including data preprocessing, discrete wavelet transformation, machine learning models, and experiment details. Section 5.4 summarizes the different machine learning and deep learning methods' performance with both identical and out-of-distribution data and draws a comparative picture between standalone and hybrid models. Finally, section 5.5 concludes our paper and sheds light on future research directions.

5.2 Literature Review

Internet traffic prediction has gained considerable attention in recent years due to the need for efficient network management and control. Numerous techniques have been developed for this purpose, which can broadly be categorized into conventional statistical models, machine learning methods, and deep learning-based methods.

Generally, conventional statistical models are incapable of capturing complex underlying patterns in time-series data. As a result, researcher integrated statistical models with others machine learning models to utilize their power in linear prediction. For example, a comparative performance analysis between a conventional statistical model, AutoRegressive Integrated Moving Average (ARIMA), and a deep learning model, Long Short-Term Memory (LSTM), has been conducted in [84] for internet traffic prediction. In addition, they used the signal decomposition technique, discrete wavelet transforms (DWT), to separate the linear and non-linear components from the original data before feeding them into the prediction model. A hybrid model consisting of a statistical and deep learning model is proposed in [85] for better performance than the standalone model. In addition, they applied DWT on the time series data for separating the linear and non-linear components, modeled respectively using Auto Regressive Moving Average (ARMA) and Recurrent Neural Networks (RNN). Since the conventional statistical models, such as ARMA, and ARIMA, are incapable of handling the non-linear components in the time series, the author tried to use the signal decomposition technique here to deal with the complex, non-stationary internet traffic by combining the power of deep learning. According to their experimental results, the combination of ARIMA and RNN performed better than the individual model.

Deep learning models, a subcategory of machine learning models, have become the standard for solving network traffic prediction issues in various contexts. These prediction models grounded in deep learning can be broadly classified based on the method of neuron interconnection in neural networks, such as RNN, convolutional neural networks (CNN), and graph neural networks (GNN).

RNNs, particularly LSTM and Gated Recurrent Units (GRU), are capable of handling long sequence data, including natural languages and time series, by employing recurrent connections and memory mechanisms. These representative RNN variants address the vanishing gradient problem associated with the traditional RNN structure and have been applied successfully to network traffic issues with univariate and multivariate time series forecasting [86]. CNNs, on the other hand, treat traffic data in different grids as image pixel values [87]. Integrating the attention mechanism into CNNs enhances their predictive performance [88]. For instance, a combination of the attention mechanism with ConvLSTM has been successful in capturing long-term spatial-temporal dependencies for cellular traffic prediction, leading to accurate predictions over hourly and daily time scales [89]. Additionally, a spatial-temporal downsampling neural network model based on the Transformer network has been proposed for citywide mobile traffic prediction [90]. More recently, GNNs have emerged as a new frontier in AI research, using input data structured as graphs [91]. GNNs have found successful application in communication networks, including network traffic prediction [92]. To enhance the performance of the GNN-based prediction model in large-scale traffic prediction, transfer learning has been introduced to reuse knowledge and reduce computation in cellular traffic prediction [93].

In the field of internet traffic prediction, an emerging trend is the fusion of signal decomposition techniques with prediction models, specifically those based on deep learning. These models have demonstrated their effectiveness in handling the complex, time-series data typical of network traffic. When coupled with signal decomposition techniques like Discrete Wavelet Transformation (DWT), these models are endowed with the ability to dissect data into simpler, manageable components, subsequently improving prediction capabilities. Several noteworthy studies have embraced this approach, successfully integrating signal decomposition methods into their models to enhance predictive accuracy. For example, an artificial neural network model combined with the multi-fractal DWT is proposed in [94]. The network traffic is discomposed into low-frequency and high-frequency components using Haar Wavelet, which has been considered a target for the ANN model with the input of the original traffic data. In the end, model predictions are combined to reconstruct the actual value. Their model outperformed compared with two existing methodologies. In [95], they performed a comparative analysis among different methods of DWT and spline-extrapolation in predicting the characteristics of the IoT multimedia internet traffic. The spline-extrapolation with the B-splines was the best, giving them the minimum forecast error of 5% compared with Haar-wavelet and quadratic splines having prediction errors respectively 7-10% and 10%.

Moreover, some studies have even combined various technical methods, including wavelet transformation, Hurst exponent analysis, model parameter optimization, and fusion of multiple prediction models, to further improve prediction accuracy. For example, In [96], the author developed a traffic prediction framework by combining the power of several technical methods such as Mallat wavelet transformation, Hurst exponent analysis, model parameter optimization, and fusion of multiple prediction models. Firstly, a three-level decomposition has been carried out on the original traffic data to extract a set of approximate and detailed components. Then, the individual component predictability was analyzed using Hurst exponent analysis, where a higher Hurst exponent (H) indicates more predictableness. According to their study, the approximate component has a higher H than the detailed component. As a result, a more efficient machine learning model, the Least squares Support vector machine (LSSVM), is used to predict detail components while the approximate component is analyzed using ARIMA. The proposed method showed better prediction accuracy compared with the other models.

The use of signal decomposition in real-world internet traffic prediction is prevalent in the current literature. The existing works indicate the outperformance of the hybrid model capable of handling the linear and non-linear components separately using different types of models. However, most research assumes that the train data and test data have come from the same distribution, i.e., $P_{tr}(X, Y) = P_{te}(X, Y)$. But, the train and test distribution can be different due to several reasons, such as the temporal/spatial evolution of data or the sample selection bias in the data collection process. Therefore, out-of-distribution (OOD) generalization discusses machine learning methodology where a distribution shift exists between P_{tr} and P_{te}. In OOD problem settings, we need to define how the test distribution is different from the strain distribution. There are different distribution shifts in the literature, but the most common one is the covariate shift, where the target generation process is the same with the marginal distribution of X shifts from the training set to the test set. According to recent studies, machine learning methods do not guarantee the generalization of out-of-distribution data. And this issue is not considered extensively in the existing literature on internet traffic prediction. Therefore, it is significantly vital to building a robust prediction model so that it can generalize the unknown distribution in the future. In this research, we propose a hybrid machine learning model that can better generalize the covariate shift in the data.

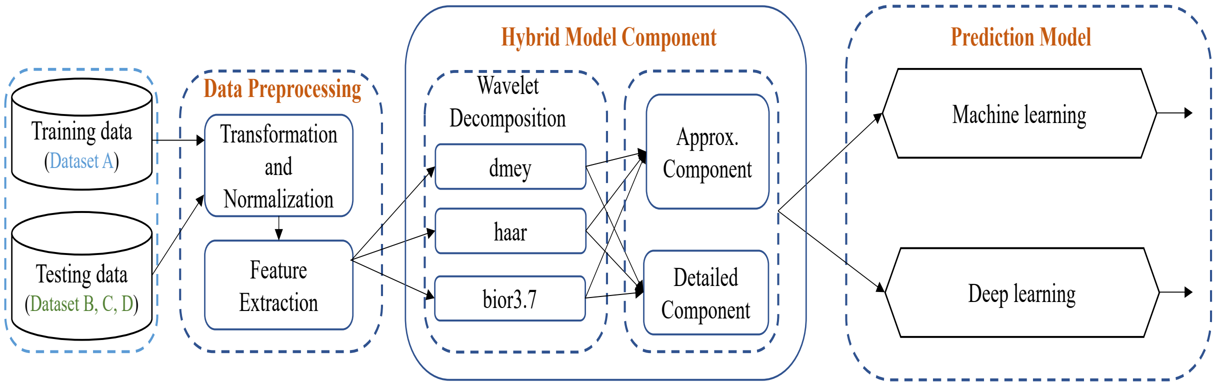


Figure 5.1: High-level framework of our proposed hybrid traffic prediction model for our-of-distribution generalization.

5.3 Proposed Methodology

In this section, we describe each component of our proposed traffic prediction model depicted in Fig. 5.1. Also, we include a detailed description of our experimental datasets used for machine learning model training and evaluation.

Given a sequence of internet traffic data, our goal is to accurately predict future data points. The sequences are non-linear, non-stationary, and may have different distributions (in-distribution and out-of-distribution). Given a time series $X = \{x_1, x_2, \ldots, x_n\}$, our aim is to design a model f such that $f(X_{t-m:t-1})$ can accurately predict x_t for $t = m+1, \ldots, n$ where m is the size of the sliding window. For the OOD generalization, we want our model to perform well not only on the in-distribution data (similar to the training data distribution) but also on out-of-distribution data (different from the training data distribution). In our study, we propose a Discrete Wavelet Transformation (DWT) based hybrid model that takes advantage of the strength of deep learning and the effective feature extraction capability of wavelet transformations. The purpose of this hybrid model is to effectively overcome the OOD generalization problem, which is a common pitfall for conventional machine learning models.

Discrete Wavelet Transformation is a mathematical tool for hierarchically decomposing a signal. For a given signal $X = \{x_1, x_2, \ldots, x_n\}$, it provides two sets of coefficients, Approximation coefficients (Ca) and Detail coefficients (Cd). This process can be mathematically represented as:

$$Ca = DWT_{approx}(X),$$
$$Cd = DWT_{detail}(X).$$

The Approximation coefficients (Ca) capture the overall trend or slow changes in the data, while the Detail coefficients (Cd) represent the specific occurrences or fast changes in the data. This hierarchical decomposition provides a more detailed feature extraction from complex and non-stationary signals like internet traffic. We propose to integrate these wavelet features into a deep learning model for improved prediction. The Approximation coefficients (Ca) and Detail coefficients (Cd) obtained from the DWT are used as input features for the deep learning model. This way, the model not only benefits from the powerful representational learning of deep learning but also from the fine-grained, hierarchical features provided by the wavelet transformation. By including the wavelet transformation as a signal processing step, our model is more capable of understanding and adapting to the non-linear and non-stationary properties of internet traffic data. This feature also empowers the model to handle the OOD generalization problem by learning more abstract and versatile representations. This integration can be represented mathematically as:

$$y = f(Ca, Cd),$$

where f represents the deep learning model, Ca and Cd are the wavelet features, and y is the predicted value.

Our proposed hybrid model exhibits superior out-of-distribution generalization. This is largely due to the wavelet transformation's capability of capturing the essential features from signals with different characteristics and distributions. As such, even when faced with OOD data, our model can effectively identify the underlying patterns and make accurate predictions. We showcase this strength by testing our model on different datasets with diverse distributions. The consistent performance across these datasets proves our model's ability to generalize well to out-of-distribution data, which is a significant improvement over traditional machine learning models. Through our research, we contribute to the field of internet traffic prediction by providing a novel methodology that effectively handles the challenges of non-linear, non-stationary data, and out-of-distribution generalization.

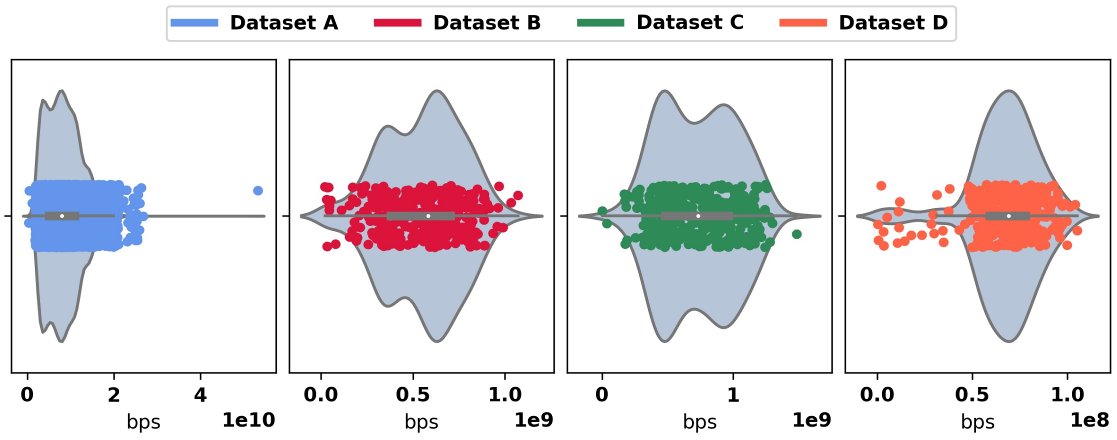


Figure 5.2: Data distribution of different datasets.

5.3.1 Dataset Description

In this research, we rely on four distinct real-world internet traffic datasets labeled as A, B, C, and D. These datasets consist of telemetry data collected through SNMP (Simple Network Management Protocol) interface MIB (Management Information Base), sampled from a core-facing interface on a provider edge router. The sampling process occurred at 5-minute intervals, with the bps (bit per second) value for each interval being determined from the difference between the samples at both ends of the interval, multiplied by 8. Given that the interface's capacity was 40 Gbs, no discards (ifOutDiscards) were observed throughout the sampling period.

For our experimental setup, dataset A was utilized for training our proposed model and evaluating its in-distribution prediction performance, while datasets B, C, and D were employed solely for assessing the model's performance on out-of-distribution (OOD) data. Notably, dataset A is relatively larger, consisting of 8563 data samples, whereas datasets B, C, and D are smaller, comprising 363, 369, and 358 data instances respectively.

Given that each dataset was procured from different network interfaces, they exhibit distinct distributions, as depicted in Fig. 5.2. Specifically, the training dataset, A, demonstrates a shift in distribution in comparison to the testing datasets B, C, and D. The introduction of testing datasets possessing OOD samples allowed us to gauge the proposed hybrid model's proficiency in addressing the out-of-distribution generalization issue typically encountered in conventional machine learning models.

5.3.2 Data Preprocessing

The preprocessing stage of our hybrid traffic prediction framework primarily involves two tasks: data transformation with normalization and feature extraction. Initially, we examined the presence of missing values in our datasets. Within dataset A, we identified 29 missing data instances. These were handled using the forward fill technique, which replaces missing values with the preceding valid data point. Other approaches, including linear interpolation and quadratic interpolation, are often used for dealing with missing data in time series analysis. However, these methods were unsuitable for our data due to its non-linear nature. Linear interpolation, which estimates missing values by connecting points along a straight line, and polynomial interpolation, which requires a predefined order and fills missing data points based on the minimum possible degree fitting the existing points, were therefore not chosen. After weighing different replacement methods, we decided to employ the forward-fill technique for handling missing values in our experiment.

Subsequently, we executed normalization and standardization operations on both our training and testing datasets. This step is vital to prevent potential biases during model fitting resulting from variations in measurement scales. To finalize the preprocessing, we reformatted our time series data from the $(time, values)$ structure to a $(features, target)$ format compatible with supervised learning. This conversion was carried out using a sliding window technique, where we generated lagged features from our time-series dataset. This method uses a set number of preceding data samples, the lagged features, to predict the subsequent value or target in time-series data. The width of the window, indicating the number of features, must be defined for this feature extraction process. We experimented with several window-width parameters, such as 6, 9, 12, and 15, aiming to identify the optimal input size for the prediction model.

5.3.3 Discrete Wavelet Transform

Our proposed hybrid prediction model incorporates a signal transformation element based on wavelet decomposition. Though we considered various machine learning models to capture the non-linear and non-stationary nature of internet traffic, traditional machine learning models have certain drawbacks including overfitting, difficulty in identifying global optima, and suboptimal performance on out-of-distribution data. To counter these challenges, we included a non-classical signal transformation module – the Discrete Wavelet Transformation (DWT)–into our approach. This module aids in extracting hierarchical components of intricate signals, thus revealing finer characteristics. Unlike classical transformation techniques that perform optimally with linear data, DWT has the capacity to comprehend the stochastic properties of non-linear real-world internet traffic.

DWT is an analytical method to uncover concealed patterns within an original signal by transforming this signal into the time-frequency domain. This transformation employs a wavelet – a wave-like oscillation – that extracts multiple lower-resolution levels by adjusting the wavelet's scale and location [83]. We have a selection of wavelets available for signal decomposition, but for our model, we opted for mother wavelets from two primary wavelet categories: orthogonal and bi-orthogonal. Orthogonal wavelet filters-low-pass and high-pass filters-have consistent lengths, unlike bi-orthogonal filters. Furthermore, orthogonal wavelet filters are symmetric, while the low-pass bi-orthogonal filter is symmetric and the high-pass filter is symmetric or anti-symmetric. As signal transformation characteristics are strongly linked with wavelet properties, mother wavelets should be chosen based on the specific problem requirements. In our experiment, we utilized and compared three commonly used mother wavelets: dmey (orthogonal), haar (orthogonal), and bior (biorthogonal). These wavelets have found extensive application in prior works for time-series analysis [97, 98].

In each DWT stage, the signal breaks down into two components: the approximate component (Ca), and the detailed component (Cd), which correspond to the overall trend and intricate events within the data, respectively. The Ca from level i is utilized to calculate Ca in the subsequent level, $i+1$. The signal undergoes convolution through a low-pass filter (l_p) and high-pass filter (h_p) to generate the new Ca_{i+1} and Cd_{i+1}. We trained our prediction model using these decomposed components to provide the model with a more nuanced understanding of the signal's individual components. After predicting the individual components, we reconstructed the original data by merging all level's detailed components and the last level's approximate components. We conducted experiments both with and without this decomposition component to contrast the standalone and hybrid models' performances in predicting testing data non-identical to the training data. The formulas for the components and their reconversion into original data are as follows [99]:

$$Ca_{i+1}[n] = Ca_i * l_p[n] = \sum_{m=-\infty}^{\infty} Ca_i[m] l_p[n-2m] \tag{5.1}$$

$$Cd_{i+1}[n] = Ca_i * h_p[n] = \sum_{m=-\infty}^{\infty} Ca_i[m] h_p[n-2m] \tag{5.2}$$

$$\text{Original data, } y_t = Ca_n + \sum_{i=1}^{n} Cd_i \tag{5.3}$$

Table 5.1: Model Performance: Train and Test on Dataset A Using Gradient Boosting Algorithm.

	Standalone				**Hybrid (dmey)**				**Hybrid (haar)**				**Hybrid (bior3.7)**			
	6	9	12	15	6	9	12	15	6	9	12	15	6	9	12	15
GBR	96.1	96.1	96.2	96.2	97.1	97.1	97.1	97.2	97.1	97.1	97.1	97.1	97.1	97.2	97.2	97.2
XGB	96.1	96.0	95.9	95.9	97.4	97.4	97.4	97.3	97.2	97.3	97.3	97.2	97.2	97.2	97.2	97.3
LGB	96.2	96.2	96.1	96.2	97.4	97.4	97.4	97.4	97.3	97.3	97.3	97.3	97.1	97.2	97.2	97.2
CBR	96.3	96.1	96.2	96.2	97.3	97.2	97.2	97.2	97.2	97.1	97.1	97.1	97.2	97.2	97.2	97.1
SGD	96.0	96.1	96.2	96.1	96.7	96.9	97.0	97.0	96.7	96.6	96.9	96.9	96.7	96.8	96.8	96.9

Table 5.2: Model Performance: Trained on Dataset A, Tested on Dataset B Using Gradient Boosting Algorithm.

	Standalone				**Hybrid (dmey)**				**Hybrid (haar)**				**Hybrid (bior3.7)**			
	6	**9**	**12**	**15**	**6**	**9**	**12**	**15**	**6**	**9**	**12**	**15**	**6**	**9**	**12**	**15**
GBR	82.6	81.6	82.0	81.0	86.7	86.9	86.8	86.7	86.5	86.2	85.6	85.5	82.5	82.5	82.0	81.5
XGB	78.9	78.7	80.1	78.0	87.9	87.6	87.4	87.4	87.3	87.5	87.5	87.1	83.3	81.9	82.1	82.8
LGB	83.4	83.3	82.9	82.8	87.6	87.3	87.1	86.9	87.3	87.4	87.2	87.0	84.8	84.5	84.1	83.9
CBR	80.6	79.8	81.3	80.4	85.7	84.9	84.8	84.0	85.6	85.2	85.3	84.8	83.8	83.2	83.0	81.3
SGD	80.1	79.5	78.6	77.9	84.1	83.8	82.8	82.6	84.0	83.3	82.5	81.7	83.6	83.3	82.3	81.4

Table 5.3: Model Performance: Trained on Dataset A, Tested on Dataset C Using Gradient Boosting Algorithm.

	Standalone				**Hybrid (dmey)**				**Hybrid (haar)**				**Hybrid (bior3.7)**			
	6	**9**	**12**	**15**	**6**	**9**	**12**	**15**	**6**	**9**	**12**	**15**	**6**	**9**	**12**	**15**
GBR	74.1	72.8	73.9	73.0	83.1	83.1	82.6	82.8	80.9	81.1	80.8	80.6	79.1	78.7	78.7	78.9
XGB	72.6	71.7	72.9	71.0	84.0	83.6	83.2	83.1	82.2	82.5	82.3	82.3	80.7	79.7	79.6	80.2
LGB	75.6	75.9	75.4	75.1	83.5	83.3	83.2	83.1	82.2	82.2	82.0	81.9	82.0	81.7	81.4	81.1
CBR	74.0	73.2	73.6	73.8	82.5	81.4	81.8	81.1	80.8	80.6	80.4	80.0	80.9	80.9	81.1	80.0
SGD	73.9	73.6	73.3	72.6	79.7	79.5	78.9	78.9	78.6	78.4	77.9	77.3	79.6	79.2	78.9	78.1

Table 5.4: Model Performance: Trained on Dataset A, Tested on Dataset D Using Gradient Boosting Algorithm.

	Standalone				**Hybrid (dmey)**				**Hybrid (haar)**				**Hybrid (bior3.7)**			
	6	**9**	**12**	**15**	**6**	**9**	**12**	**15**	**6**	**9**	**12**	**15**	**6**	**9**	**12**	**15**
GBR	86.5	86.0	86.3	86.2	88.6	88.7	88.7	88.7	88.7	88.6	88.5	88.8	87.0	87.0	86.9	87.1
XGB	85.3	84.9	85.1	85.0	89.4	89.4	89.1	89.4	89.1	89.1	89.1	89.1	87.8	87.2	87.2	87.4
LGB	86.6	86.5	86.4	86.6	89.6	89.5	89.4	89.6	89.3	89.2	89.1	89.3	89.8	89.5	89.3	89.4
CBR	86.3	86.2	86.0	86.2	89.6	89.5	89.6	89.7	89.3	89.4	89.2	89.3	89.5	89.1	88.7	88.6
SGD	85.0	84.5	84.5	84.7	87.5	87.2	87.0	87.1	87.4	86.9	86.9	87.0	88.2	87.7	87.6	87.7

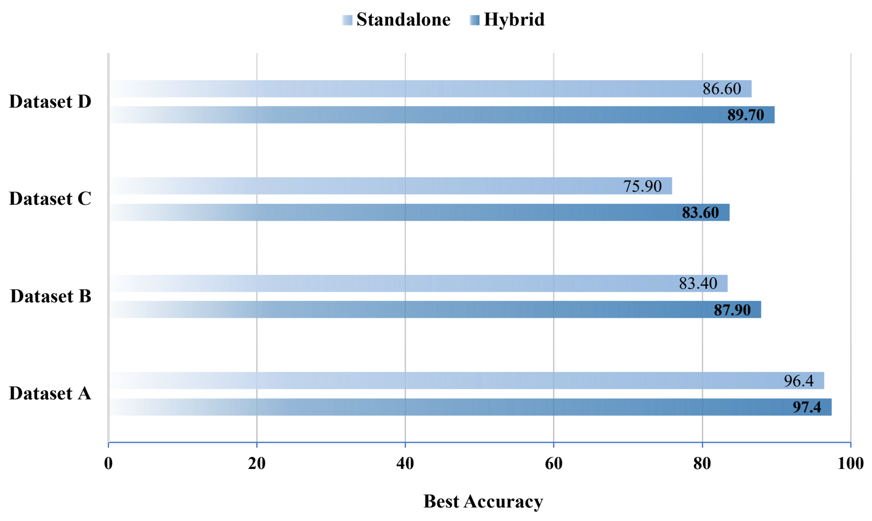


Figure 5.3: Best accuracy comparison between standalone and hybrid model.

5.4 Analysis of Experimental Results

In this section, we analyze our experimental data to find out the result. Firstly, we evaluate the performance of our proposed hybrid model based on dataset *A* where 70% and 30% data are used for model training and in-distribution testing, respectively. After that, the performance was evaluated using three other datasets, *B*, *C*, and *D*, which are entirely unknown to the model, to measure the effectiveness of our proposed model for out-of-distribution generalization. We evaluate the performance of our proposed hybrid model using both boosting and deep sequence algorithm in subsection 5.4.1 and 5.4.2 respectively.

The group column 'Standalone (No Wavelet)' represents the standalone model performance while the other three columns 'Hybrid (dmey)', 'Hybrid (haar)', and 'Hybrid (bior3.7)' represent the hybrid model with the corresponding mother wavelet. In addition, we used four different window widths such as 6, 9, 12, and 15, for our feature extraction process. Therefore, all our experimental results summarize the performance of the prediction models based on different feature sets.

5.4.1 Performance Analysis Using Boosting Algorithms

In this section, we analyze and compare the performance of our proposed prediction model using data samples from a similar distribution of the training dataset. In the case of the standalone model where the signal decomposition module was off, the best prediction accuracy by the XGB model is 96.1% using the six input features according to the Table 5.1. The highest accuracy for LGB, SGD, GBR, and CBR are respectively 96.2%, 96.2%, 96.2%, and 96.3% using inputs 6, 12, 12, and 6.

After evaluating standalone models with a test set having similar distribution as the training set, we used three other datasets from different distributions to validate model performance in case of out-of-distribution. The results are summarized in Table 5.2, Table 5.3, and Table 5.4, respectively, for dataset *B*, dataset *C*, and dataset *D*. As the distribution of these three datasets

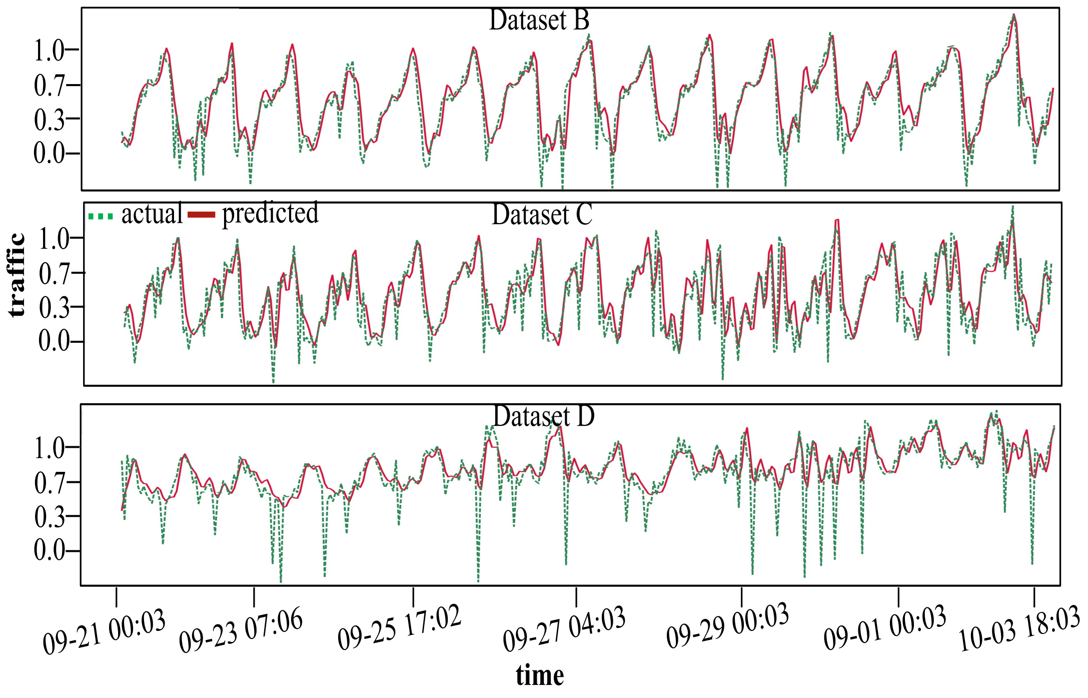


Figure 5.4: Comparison of actual and predicted traffic using our proposed hybrid model for OOD dataset.

differs from the data samples used in training, the standalone model performance decreases significantly in predicting these OOD samples. Fig. 5.3 depicts a comparative view of best performance by the standalone model and proposed hybrid model. Overall, the standalone model performance was at its peak when it was evaluated using the same distribution data, i.e., training and testing on dataset *A*. The performance dropped from 96% to 83% when tested using dataset *B*, where the samples have a distribution shift compared to data samples in *A*. Also, the performance gap between IID and OOD evaluation using data samples from dataset *C* and *D* is significantly large, and it is more than 20% and 10%, respectively. This portion of our experiment indicates a classical out-of-distribution problem of the machine learning model, which assumes both training and testing data are identically distributed.

In our experiment, the classical machine learning model showed a pretty high prediction accuracy of more than 96% for IID samples which indicates their capability of handling non-linear and non-stationary traffic data. But the standalone models were straggling to maintain a similar range of prediction accuracy when tested with OOD data. The accuracy falls from 96% to 75%-86% compared to IID and OOD evaluation. However, there is a high probability of having a distributional shift in the data after deployment of the prediction model in the real world, which we imitated in our work by considering three different data distributions for evaluating our proposed model. Also, we tried to propose a hybrid machine learning model so that it can reduce the IID and OOD performance gap compared to the standalone model. In the next section, we discuss the performance of our proposed hybrid model for identical and out-of-distribution data.

In our hybrid model architecture, the models have an extra module that transforms the

Table 5.5: Model Performance: Train and Test on Dataset A Using Deep Sequence Algorithm.

	Standalone				**Hybrid (dmey)**				**Hybrid (haar)**				**Hybrid (bior3.7)**			
	6	**9**	**12**	**15**	**6**	**9**	**12**	**15**	**6**	**9**	**12**	**15**	**6**	**9**	**12**	**15**
RNN	96.0	95.9	96.1	96.3	97.6	96.9	97.0	97.1	97.2	97.3	96.5	97.0	97.5	97.4	97.4	96.8
LSTM	96.5	96.5	96.6	96.6	97.9	98.0	98.0	97.7	97.7	97.6	97.7	97.5	98.1	98.0	97.9	97.9
LSTM_Seq2Seq	96.6	96.7	96.7	96.7	98.3	98.4	98.4	98.4	97.9	97.8	97.7	98.0	98.2	98.3	98.3	98.4
LSTM_Seq2Seq_ATN	96.6	96.7	96.7	96.7	98.3	98.4	98.4	98.4	97.9	97.9	97.9	98.0	98.2	98.4	98.4	98.4

original signal into several hierarchical detailed components based on wavelet transformation. The hybrid model is trained using decomposed components instead of original time-series data. This operation allowed the prediction model to learn individual components we aggregate later for converting results into original data. However, the performance of the hybrid model evaluated using identically distributed data, i.e., training and testing using both datasets *A* is better than the standalone model. The best hybrid model performance we achieved is 97.4% using the ensemble hybrid model (demy), which is a 1% improvement over the standalone model. It indicates our proposed model better captures the general trend of the traffic compared to the standalone model, as we trained them using detailed components.

Next, we evaluated our hybrid model performance using completely unknown datasets which do not have similar distribution as the training dataset. Our experiment showed a significant performance enhancement after applying wavelet transformation to handling out-of-distributed datasets. For example, the hybrid XGB model with dmey wavelet gave us more than 8% accurate results for dataset *B* compared with the standalone model. Similarly, more than 10% accuracy has been improved in the case of dataset *C* by the hybrid LGB model with haar wavelet. Overall, the hybrid model's performance is better than the standalone model for each dataset. According to the Fig. 5.3, the best prediction accuracy for out-of-distributed data has been jumped by more than 4.5%, 7.5%, and 3% respectively for dataset *B*, *C*, and *D* when compared with the standalone model. Also, the hybrid model reduces the best IID and OOD performance gap from 13% to 10%, 20% to 13%, and 10% to 7%, respectively, for dataset *B*, *C*, and *D* compared with the classical machine learning model. The comparison between actual and predicted traffic by best-performing hybrid models for different datasets has been depicted in Fig. 5.4. Due to the complex and non-linear characteristics of internet traffic, it is challenging to figure out the actual trend and pattern using original time-series data, affecting the standalone model's performance for unknown distribution. In contrast, the detailed hierarchical component of the time-series data gave more information about the general trend of the traffic.

5.4.2 Performance Analysis Using Deep Sequence Models

First, four different deep sequence models have been implemented for single-step traffic prediction. The experimental result is summarized in Table 5.5 where we assumed the test data samples came from a similar distribution of the training dataset. We considered dataset *A* for this experimentation as it is the larger dataset among the other three, and the train-to-test data sample ratio was 70%/30%. However, Table 5.5 contains results of the standalone model with-

Table 5.6: Model Performance: Trained on Dataset A, Tested on Dataset B Using Deep Sequence Model.

	Standalone				**Hybrid (dmey)**				**Hybrid (haar)**				**Hybrid (bior3.7)**			
	6	**9**	**12**	**15**	**6**	**9**	**12**	**15**	**6**	**9**	**12**	**15**	**6**	**9**	**12**	**15**
RNN	82.7	82.2	81.4	79.0	89.8	88.7	88.2	88.1	88.2	88.4	87.2	86.4	90.5	88.3	87.9	87.9
LSTM	83.3	81.6	79.8	81.0	89.8	89.7	89.9	89.5	88.3	88.0	87.5	86.6	89.5	88.6	89.4	87.7
LSTM_Seq2Seq	69.2	69.3	69.5	69.3	91.3	91.7	91.5	91.6	88.6	88.9	88.5	87.3	90.0	90.9	90.4	90.7
LSTM_Seq2Seq_ATN	67.8	67.9	68.2	68.0	91.2	91.6	91.6	91.5	88.9	88.7	87.9	87.9	90.0	90.9	90.5	89.7

Table 5.7: Model Performance: Trained on Dataset A, Tested on Dataset C Using Deep Sequence Model.

	Standalone				**Hybrid (demy)**				**Hybrid (haar)**				**Hybrid (bior3.7)**			
	6	**9**	**12**	**15**	**6**	**9**	**12**	**15**	**6**	**9**	**12**	**15**	**6**	**9**	**12**	**15**
RNN	73.9	73.8	73.9	72.8	85.8	85.0	85.0	84.6	83.6	83.8	83.3	82.2	86.5	84.8	85.2	84.7
LSTM	74.9	74.6	72.7	73.2	85.6	85.7	85.8	85.8	83.2	83.4	83.1	82.2	85.8	83.8	85.1	84.1
LSTM_Seq2Seq	66.3	66.2	66.6	66.6	86.7	87.0	87.0	87.0	84.4	84.6	84.4	83.7	87.2	87.9	87.4	87.2
LSTM_Seq2Seq_ATN	65.2	65.1	65.6	65.6	86.7	86.9	87.0	86.9	84.6	84.6	84.1	84.2	87.2	87.7	87.3	86.7

Table 5.8: Model Performance: Trained on Dataset A, Tested on Dataset D Using Deep Sequence Model.

	Standalone				**Hybrid (demy)**				**Hybrid (haar)**				**Hybrid (bior3.7)**			
	6	**9**	**12**	**15**	**6**	**9**	**12**	**15**	**6**	**9**	**12**	**15**	**6**	**9**	**12**	**15**
RNN	85.1	84.3	85.4	84.4	90.7	90.2	90.4	90.1	89.9	90.0	89.7	89.5	91.4	91.0	91.0	91.0
LSTM	85.8	83.3	83.4	83.2	90.2	90.6	90.6	90.7	89.4	89.3	89.4	89.3	91.0	91.0	90.7	90.4
LSTM_Seq2Seq	81.6	81.6	81.7	82.0	90.9	90.8	91.0	91.2	89.5	89.8	89.7	89.7	91.4	91.7	91.6	91.7
LSTM_Seq2Seq_ATN	81.0	81.0	81.1	81.5	90.8	90.8	90.9	91.2	89.7	89.8	89.7	89.9	91.4	91.6	91.6	91.6

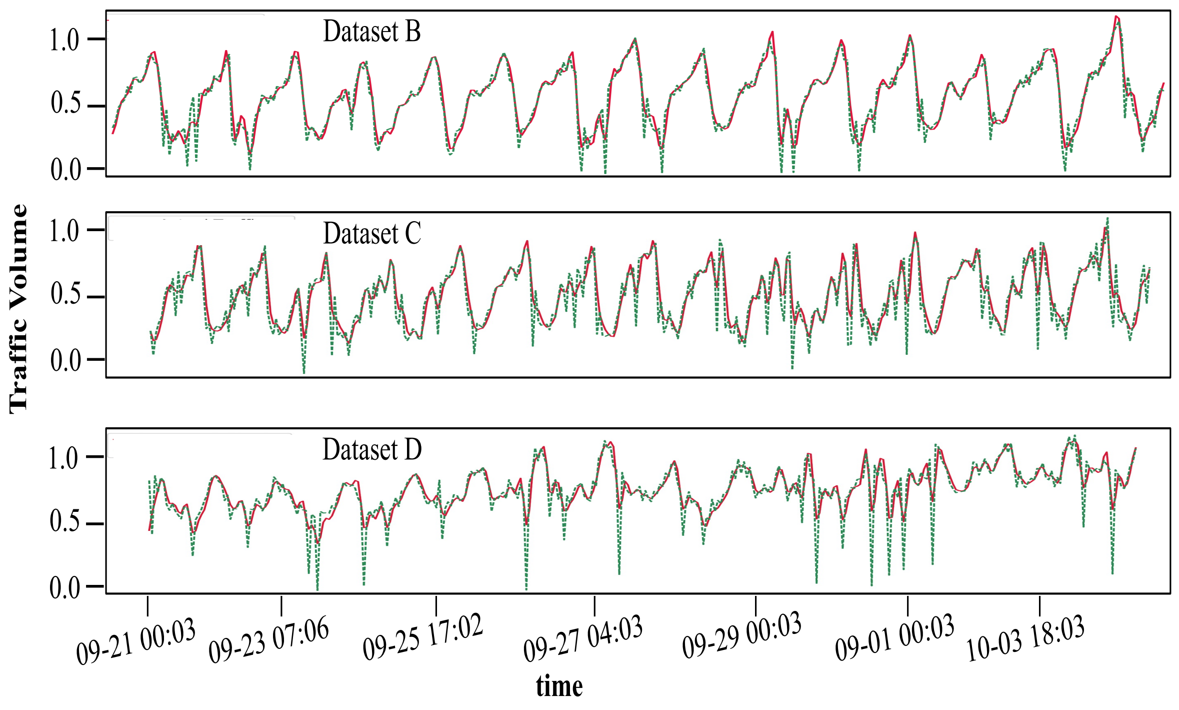


Figure 5.5: Comparison of actual and predicted traffic using our proposed hybrid model for OOD dataset.

out any wavelet decomposition and three hybrid model settings with varying mother wavelets. In addition, we reported results for different feature lengths of 6, 9, 12, and 15.

In the case of the standalone model, the best prediction accuracy of 96.7% we achieved through the encoder-decoder model with nine features, while RNN performance was comparatively poor. When we provided a longer sequence, e.g., 15 features, to train our prediction models, we noticed that the RNN model failed to capture information very well compared to LSTM and encoder-decoder model. The prediction accuracy was 96.3% using 15 input variables for RNN, while LSTM, LSTM_Seq2Seq, and LSTM_Seq2Seq_ATN models, the corresponding accuracy was 96.6%, 96.7%, and 96.7% respectively. It indicates the inherent vanishing gradient problem of RNN in processing longer sequences which have been solved by introducing the gate concept in LSTM architecture. On the other hand, LSTM and LSTM encoder-decoder model performances were very close, and they could retain information from longer sequences, resulting in better accuracy. However, after evaluating the standalone model, we tested similar data samples using our proposed hybrid method. The prediction models were trained using the decomposed detailed and approximate components instead of the original time-series data. Since real-world internet traffic is very complex and might have an irregular pattern, we provided the hierarchical signal components to our deep sequence models for better generalization using unknown samples. As a result, the best prediction accuracy using the hybrid encoder-decoder model is 98.4% which is 1.7% more than the best standalone model performance. When we compared the average accuracy of each input setting of the standalone model with our proposed hybrid model, we noticed more than 1% accuracy improvement for each type of hybrid model with different wavelets. Furthermore, the dmey wavelet function improved the highest average accuracy among three different mother wavelets, while haar gave the lowest enhancement. However, the haar wavelet function might fail to capture the high-frequency changes in the high-frequency coefficients as the haar window is only two elements wide. As a result, it might have performed poorly than the other two wavelets in case of decomposing the time-series data resulting in lower prediction accuracy.

When we evaluate the performance of standalone conventional prediction models using dataset *B*, the average accuracy for different input settings dropped from 96% to 81% for RNN as shown in Table 5.6. Also, the other three prediction models showed a massive gap between in-distribution and out-of-distribution average prediction, which are 15%, 27%, and 28% respectively for LSTM, LSTM_Seq2Seq, and LSTM_Seq2Seq_ATN. It indicates a classical problem of the conventional machine learning model in OOD generalization. However, the similar dataset *B* is used to evaluate our hybrid prediction models, where the average performance gap between in-distribution and out-of-distribution evaluation is 9% for RNN_dmey, which was 15% for standalone RNN. Similarly, the other three hybrid models: LSTM_dmey, LSTM_Seq2Seq_dmey, and LSTM_Seq2Seq_ATN_dmey showed an improved prediction accuracy with respectively 9%, 7%, and 7% gap between in-distribution and out-of-distribution average prediction. When we compared them with the standalone model performance, we noticed a significant performance enhancement by our hybrid model. For example, the encoder-decoder model performance gap reduces from 27% to 7%, indicating the wavelet component's effectiveness in our framework better to understand the complex pattern of real-world internet traffic.

Finally, we validated the performance of our proposed model with varying mother wavelet functions. The original signal has been decomposed into several hierarchical components based on the wavelet characteristics, ultimately impacting the model's accuracy. For example, In the case of haar wavelet, the hybrid RNN model reduces the performance gap from 15% to 10% when compared with the standalone RNN model. Also, the other three prediction models provided better accuracy than the standalone model. For example, the LSTM_haar model reduces the performance gap from 15% to 10%. However, considering all models' performance, we noticed that the dmey wavelet performance was better than the haar. As mentioned in the previous section, the haar window is only two elements wide, which might affect the proper capturing of the high-frequency changes in the high-frequency coefficients. As a result, hybrid models with haar wavelet perform lower than dmey wavelet, although their performance is significantly better than standalone models. Using bior3.7 bi-orthogonal wavelet function for signal decomposition, our prediction models perform better than haar wavelet and close to dmey wavelet function. For example, the average accuracy improvement was 6% for out-of-distributed data using RNN_bior3.7 model, similar to RNN_dmey. The gap between average accuracy for LSTM_bior3.7, LSTM_Seq2Seq_bior3.7, and LSTM_Seq2Seq_ATN_bior3.7 was respectively 10%, 8%, and 8% which is better than haar and very close to dmey wavelet function.

Finally, we used our hybrid model to evaluate two other unknown datasets, *C* and *D*. The details results are summarized in Table 5.7, and Table 5.8 respectively for dataset *C*, and *D*. In the standalone model, the performance of RNN and LSTM models is relatively similar across all parameter settings (6, 9, 12, 15), with the LSTM showing a marginal improvement for dataset *C*. However, when shifted to a hybrid mode, all models demonstrate a significant increase in performance, with the highest results achieved by the LSTM Seq2Seq model in the bior3.7 hybrid variant. The LSTM Seq2Seq model has a lower performance in standalone mode but performs better when combined with hybrid techniques, especially the bior3.7 hybrid type. Interestingly, the addition of an attention mechanism (ATN) in the LSTM Seq2Seq model doesn't significantly change the results, suggesting that the main driver of performance in these settings is the Seq2Seq structure, rather than the attention mechanism.

For dataset D, our findings indicate that the hybrid models, across all network variants, consistently outperform their standalone counterparts. Notably, the RNN and LSTM models exhibit similar performance trends, achieving peak performance in the standalone mode at 85.4 and 85.8, respectively. These models demonstrate a considerable increase in performance in the hybrid mode, with RNN peaking at 91.4 under the bior3.7 condition, and LSTM reaching a high of 91.0, also in the bior3.7 setting. The more complex LSTM_Seq2Seq and LSTM_Seq2Seq_ATN models display a reduced performance in the standalone mode relative to the basic RNN and LSTM models. However, they demonstrate robust performance in the hybrid modes, surpassing the performance of the simpler models. The LSTM_Seq2Seq and LSTM_Seq2Seq_ATN models achieve the highest overall performance, reaching 91.7 and 91.6 respectively under the bior3.7 condition.

The comparison between actual and predicted traffic by best-performing hybrid deep sequence models for different datasets has been depicted in Fig. 5.5. After analyzing all results, we noticed a general performance improvement for out-of-distributed traffic prediction using our hybrid models with varying wavelet functions while the standalone model accuracy dropped significantly when tested with OOD data compared to IID data samples, however, our proposed model reduces this significant gap into marginal quantity by analyzing the decomposed hierarchical component of the original time-series data. The results underscore the performance advantages conferred by hybrid model techniques on various RNN architectures, particularly the more complex LSTM_Seq2Seq and LSTM_Seq2Seq_ATN models. Our results indicate an overall more than 10% accuracy improvement by the hybrid model when it is tested using unknown datasets to the model.

5.5 Conclusion

In this paper, we introduced a novel approach for internet traffic prediction using a hybrid machine learning model incorporating signal decomposition techniques and deep learning models. Our approach aimed to overcome the key challenges in internet traffic prediction, namely the non-linear, non-stationary characteristics of traffic data, and the issue of out-of-distribution generalization.

Our research demonstrated that traditional machine learning models exhibited high prediction accuracy when trained and tested with data from identical distributions, indicating their capability to handle non-linear and non-stationary traffic data. However, they faltered when faced with out-of-distribution data, exhibiting a significant drop in accuracy. In response to this issue, we proposed a hybrid model that applied wavelet transformation to decompose the original time-series data into multiple hierarchical components, allowing our model to capture the complex underlying patterns in internet traffic. Our hybrid model demonstrated a superior performance over standalone models, achieving a prediction accuracy of 97.4% for in-distribution data. The model also significantly outperformed standalone models when tested with out-of-distribution data, demonstrating an improvement in accuracy of more than 8%, 10%, and 3% for datasets *B*, *C*, and *D* respectively. The results thus highlighted the effectiveness of the hybrid model in reducing the performance gap between in-distribution and out-of-distribution data.

Furthermore, we also explored deep sequence models for internet traffic prediction. Our proposed hybrid deep sequence models outperformed their standalone counterparts in terms of prediction accuracy, achieving an accuracy of 98.4% for in-distribution data, and significantly reducing the performance gap when tested with out-of-distribution data. This research contributes to the field of internet traffic prediction by providing a novel methodology to handle non-linear, non-stationary data and out-of-distribution generalization. It offers a promising solution for effective and efficient internet traffic prediction, which could pave the way for better network management and optimization. Nevertheless, future research should focus on improving the robustness of the proposed model and expanding the application of the model to other types of non-stationary and non-linear time series data. In addition, the impact of different wavelet functions on the performance of the hybrid model warrants further exploration. We hope our work will stimulate further research and development in the field of internet traffic prediction and beyond.

Chapter 6

Network Intrusion Detection and Comparative Analysis using Ensemble Machine Learning and Feature Selection

Abstract: Proper security solutions in the cyber world are crucial for enforcing network security by providing real-time network protection against network vulnerabilities and data exploitation. An effective intrusion detection strategy is capable of taking a holistic approach for protecting critical systems against unauthorized access or attack. In this paper, we describe a machine learning (ML) based comprehensive security solution for network intrusion detection using ensemble supervised ML framework and ensemble feature selection methods. In addition, we provide a comparative analysis of several ML models and feature selection methods. The goal of this research is to design a generic detection mechanism and achieve higher accuracy with minimal false positive rates (FPR). NSL-KDD, UNSW-NB15, and CICIDS2017 datasets are used in the experiment, and results show that our detection model can identify 99.3% of intrusions successfully with the lowest 0.5% of false alarms, which depicts better performance metrics compared to existing solutions.

6.1 Introduction

The growing frequency of cyber-attacks is an eminent problem of the modern era. These attacks are detrimental for both individuals and enterprises, and are capable of impacting the confidentiality, integrity, and availability of critical information that is carried through the network. It is essential for enterprises to protect their networks against intruders, hackers, and network threats. Therefore, a network system must incorporate several security tools to protect important data and services from potential threats. The Intrusion Detection System (IDS) is a software or hardware implementation for screening vindictive movement or strategy infringement in the network. It monitors the malicious activity or security violation within the network and notifies the administrator of potential threats. Cyber-attacks are becoming pernicious day by day. In order to keep pace with the constantly changing cyber threats, modern networked business environments require a high level of security for safe and trusted communication of information between organizations. Traditional IDSs (Signature-based and Anomaly-based) are not adequately designed to adapt to the continuously changing patterns of network intrusion. Incorporating artificial intelligence within the IDS has the potential to cope with the new attack patterns and to ensure network security.

In the last few decades, machine learning (ML), a subset of artificial intelligence, has been used extensively to improve intrusion detection by enabling automation and prediction on an advanced level. ML techniques enable dealing with the modifiable, reproducible, and extensible datasets. These techniques help IDSs to become robust by learning and tackling seen and unseen attacks. In addition, a single ML model may not always predict accurately, whereas a combination of ML models (ensemble ML) detect more precisely. Dietterich et al., [100] mentioned that ensemble ML classifiers outperform individual classifiers in solving various classification problems. Besides, Feature Selection (FS) is a process of selecting a subset of appropriate and relevant features from a large feature space. It plays a vital role in achieving higher accuracy as well as minimizing the model training time. The challenges of feature optimization and designing IDSs with the best suitable ML techniques motivated us to investigate the performances of various FS methods. Also, we were motivated to explore the ensemble techniques for feature selection and model classification as it consider several candidates' decisions for optimization rather than individual choices.

In this research, we have extended our existing work [101] and implemented a supervised ensemble ML framework (SupEnML) by combining multiple ML classifiers from various classification families such as Decision Tree, Logistic regression, Naïve Bayes, Neural Network, Support Vector Machine for network intrusion detection. Furthermore, the ensemble feature selection (EnFS) technique has been incorporated inside the classification procedure of this framework. This EnFS technique combines three major FS method types (Filter-based, Wrapper-based and Embedded methods) for feature selection such as Chi-squared, Mutual Information, Pearson Correlation, SFPR, Logistic Regression with L1 penalty, Random Forests, ANOVA, Recursive Feature Elimination, and LASSO. The yielded features are ensembled with majority voting and resulted in an optimized set of features. After getting the feature set, both individual and ensemble classification have been performed using the SupEnML framework. This study shows comparative performance analysis of single models (built from individual classifications) and ensemble models (built from ensemble classifications). Three benchmark intrusion detection datasets: NSL-KDD [102], UNSW-NB15 [103], and CICIDS2017 [104] are used for the experimentation, where the outcome displays significant improvement of performance in terms of the accuracy and false-positive rate for our ensemble-based approach. The proposed approach detects network intrusion more accurately and efficiently than existing ML-based intrusion detection methods mentioned in the literature [105], [106], [107], [108]. The rest of the chapter is organized as follows: Section 6.2 discusses the previous state-of-the-art approach to developing NIDS with AI techniques. The proposed framework for detecting cyberattacks using ensemble techniques is briefly discussed in Section 6.3. Section 6.4 presents and discusses the experimental results of the proposed framework. Finally, section 6.5 concludes the paper with future directions.

6.2 Literature Review

Machine Learning (ML) algorithms focus on the development of computer programs with the system's ability to automatically acquire and improve functionality without human intervention. Tsai et al., [109] reviewed 55 papers on intrusion detection using ML and performed a comparative analysis of algorithms and datasets used. Mukkamala et al., [110] performed a comparative analysis on artificial neural network, support vector machine, and the ensemble of these two models. They concluded that the ensemble classifier outperformed these single classifiers in detecting intrusions. Chebrolu et al., [111] proposed a lightweight intrusion detection technique using two feature selection methods and ensemble learning. Amiri et al., [112] studied the effect of feature selection and proposed a modified mutual information-based feature selection method (MMIFS) using the KDD cup 99 data set. Deep learning based detection system is becoming promising now a days. Vinayakumar, Ravi, et al., [113] proposed a highly scalable and hybrid DNNs framework which is capable of detecting real-time cyberattacks by monitoring network traffic and host level events. These authors have also examined and evaluated the effectiveness of various shallow and deep networks for Network Intrusion Detection Systems (NIDS) in a different research [114]. Both research have been evaluated with the KDDCup' 99 dataset. Moreover, Vinayakumar, Ravi, et al., [115] have proposed a deep learning-based botnet detection system for detecting and classifying domain names, and this system can be deployed at the ISP level for monitoring IoT devices.

Gomes et al., [116] proposed a taxonomy of ensemble methods for data stream classification, identified open-source tools, and discussed the upcoming trends in ensemble learning. Sagi, Omer, and Lior Rokach [117] reviewed the major approaches and techniques for ensemble learning and discussed the future trends, including refinement of popular algorithms for big data compatibility, model transformation into a more straightforward form, and integration with deep neural networks. Gao et al., [105] designed an ensemble adaptive voting algorithm to improve the detection accuracy using the NSL-KDD dataset and compared it with their constructed MultiTree algorithm. Tu Pham et al., [106] proposed an improved IDS using limited FS (25 subsets and 35 subsets of features) and ensemble models based on bagging and boosting techniques which depicted high accuracy for the NSL-KDD dataset. Das et al., [101] designed an ensemble framework for supervised classifiers, which accomplishes an immense increase in accuracy for detecting DDoS attacks based on four ML classifiers. Kittler et al., [118] developed a theoretical framework for classifier combination schemes, and analyzed the sensitivity of these schemes to the estimated errors, especially for sum rule.

To improve the ML algorithm's efficiency, feature selection (FS) plays a vital role during the preprocessing phases by removing unwanted and irrelevant features. Chandrashekhar and Sahin [119] provided a thorough study on the FS technique that demonstrates improved performance in analyzing standard and RF generator datasets with supervised learning. Sheikhpour et al., [120] investigated the semi-supervised FS methods that use both labeled and unlabeled data and illustrated two taxonomies based on a hierarchical structure. Khalid et al., [125] performed a detailed survey on feature selection and extraction to reduce dimensionality for improved data analysis. Luis et al., [126] proposed an evaluation technique to determine the correct usage of feature selection algorithms based on a scoring measure and explained the particularities of relevance, irrelevance, and redundancy. Adams and Beling [127] performed a survey on FS methods for Gaussian mixture models (GMM) and hidden Markov models (HMM), demon-

Table 6.1: A Comparative Analysis of the Study of Literature Review.

Classification Method	Feature selection method	Pre-processing	Pros	Cons	Dataset
Ensemble [110]	Not mentioned	Not mentioned	Comparative analysis and proposed two ensemble methods	Experiment is done on one dataset only.	DARPA, KDD-Cup'99
Supervised [111]	Not mentioned	Not mentioned	Suitable for specific intrusion detection	No preprocessing and not as a general purpose IDS	DARPA, KDD-Cup'99
Supervised [112]	Modified Mutual Information	FS methods for preprocessing	Proposed two FS method and performed comparative analysis	Experiment is done on one dataset only.	KDDCup' 99
Deep Learning [113], [114], [115]	Deep learning	Levenshtein distance function	They examine the effectiveness of shallow and deep networks to NIDS. Highly scalable with real time data processing	Training was not done using advanced hardware through distributed approach	KDDCup 99, NSL-KDD, UNSW-NB15, Kyoto, WSN-DS, and CICIDS 2017
Ensemble [105]	Not mentioned	One hot encoding, PCA, scaling	Ensemble model effectively improves detection accuracy.	No FS method is applied	NSL-KDD
Ensemble [106]	Leave-one-out techniques and NB, Gain Ratio	FS methods for preprocessing	Achieved high accuracy and low false alarm rate (FAR)	Experiment is done on one dataset only.	NSL-KDD
Ensemble [101]	Existing methods	MinMax, transformed, normalized	DDoS attack detection with higher accuracy	only one dataset was used to evaluate the built classifiers.	NSL-KDD
Supervised [119]	Filter, Wrapper, Embedded method	PCA, FFT	Providing overall view for supervised feature selection method with comparison.	Unsupervised learning methods are not covered.	Breast cancer, Diabetes, Ionosphere, Liver disorder,Medical,Fault mode
Supervised [120]	Filter, Wrapper, Embedded method	PCA	Two taxonomies of semi-supervised feature selection methods presented	The have'nt provided the solution for regression problems.	WDBC, WBCD, Diabetes
Supervised [121]	Combination of SVM, DT, and SA	Not mentioned	Obtained decision rules for detecting new attacks	Experiment is done on one dataset only.	KDDCup'99
Supervised [122]	Info gain Gain ratio Chi-squared ReliefF	EMFS	Ensemble-based multi-filter feature selection method provides better accuracy with less complexity.	Needs to incorporate more dataset and algorithm.	NSL-KDD
Ensemble [123]	EnFS	MinMax,sanitized, transformed, normalized	Not mentioned	NSL-KDD	
Ensemble, Unsupervised [124]	Not mentioned	Non numeric to numeric conversion	Detects DDoS attack using ensemble classifiers and a reduced feature dataset.	Not mentioned	NSL-KDD

strating that FS methods developed for GMM can be adapted for HMM and vice versa. Lin et al., [121] performed research on intrusion detection using the KDD'99 dataset and proposed an intelligent algorithm with the FS method. Osanaiye et al., [122] came up with an ensemble-based multi-filtered feature selection technique combining four filter-based FS methods. The technique generated important features from the selected algorithm with higher accuracy for the NSL-KDD dataset. Das et al., [123] proposed an ensemble framework for FS methods to produce an optimal set of features and performed a comparative analysis of the existing approaches in terms of accuracy and false alarms using the NSL-KDD dataset.

In a nutshell, the literature study indicates that the ensemble technique can be broadly applied in feature selection and model classification to obtain better outcomes in intrusion detection. We have found that the researchers were focused on detecting specific cyber-attack from a particular dataset. Here, the goal of our research is to provide a generic intrusion detection framework with higher accuracy using various anomalous datasets, like NSL-KDD, UNSW-NB15 and CICIDS2017. As the critical features are important in model classification, we have increased the number of FS methods in comparison to earlier research. We also ensemble filter-based, wrapper-based, and embedded FS methods in our EnFS framework. Finally, we have incorporated ensemble feature selection (EnFS) with ensemble classification (SupEnML) to get better accuracy with minimal false alarms.

6.3 Proposed Methodology

This section provides an overview of our proposed methodology. The detailed architecture illustrating the process flow is given in Fig. 6.1 and 6.2. The methodology is divided into five main phases, namely a) data collection, b) data preprocessing, c) ensemble feature selection, d) model classification, and e) anomaly detection.

6.3.1 Data Collection

The performance of a machine learning model largely depends on the quality of the data used for model training. The data contain relevant information of the problem domain and need to be cleaned for further analysis. Here, we use three well-known datasets from different research organizations to train, test, and verify our proposed framework, and they are NSL-KDD, CICIDS2017, and UNSW-NB15 from the Canadian Institute of Cybersecurity and the University of New South Wales. Apart from dividing the dataset into train and test data, we have a third portion of the dataset that we call the 'verification data'. This verification data mimics the real time data and is used to verify our proposed model within the IDS.

6.3.2 Data Preprocessing

The crude datasets need to be cleaned, sanitized, transformed, normalized, and feature reduced before feeding the ML classifiers. To clean and sanitize the dataset, we remove the rows containing ',' or 'inf' values. For the CICIDS2017 dataset, the single space before each of the feature names was stripped. We also deleted the column 'id' in the UNSW-NB15 dataset. As the ML models can handle numeric input, we converted the non-numeric data into numeric

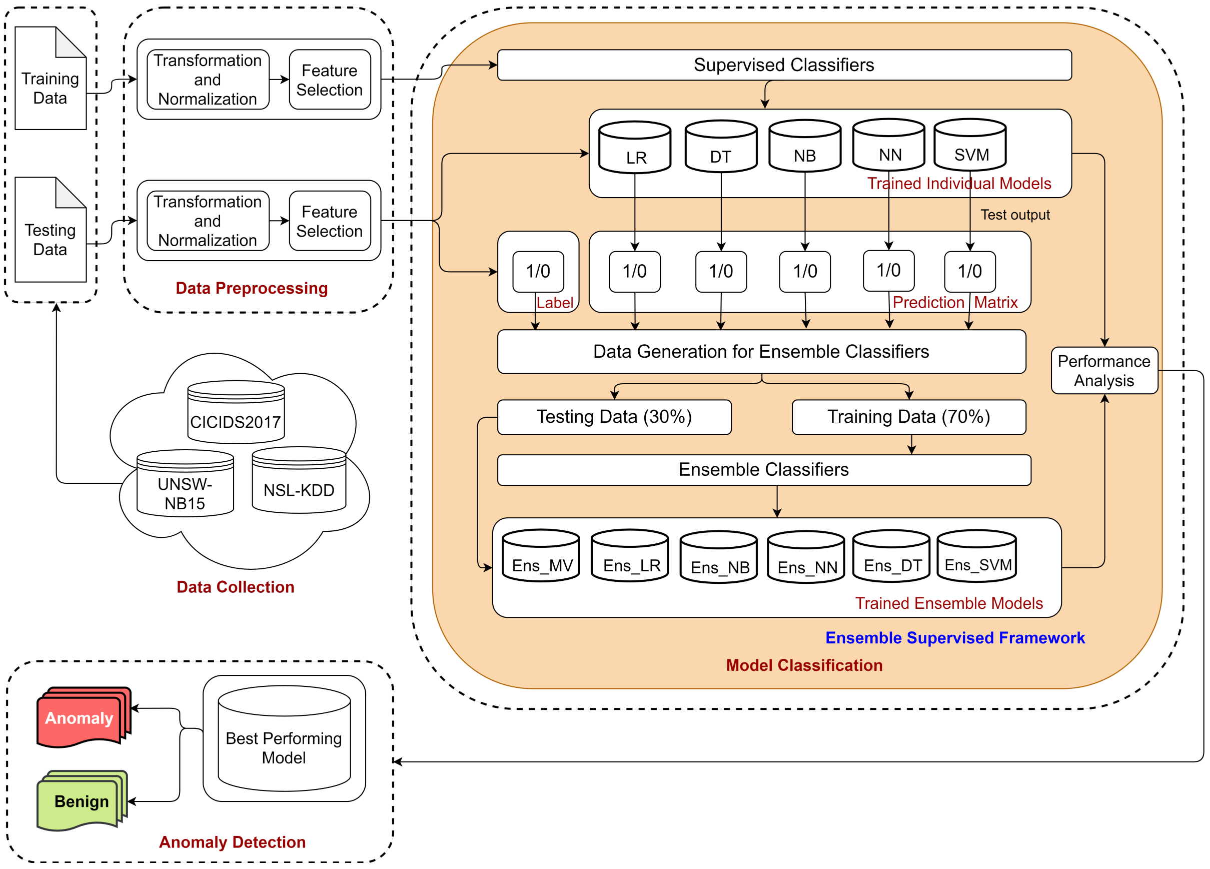


Figure 6.1: High-level framework of our proposed ensemble machine learning model for intrusion detection.

using the data encoding technique for all three datasets. Though level encoding and OneHot encoding are two well known data transformation techniques, we haven't used them as they increase the data dimensionality and do not provide feature names. Instead, we performed this conversion using our algorithm. To normalize the data, we use the MinMax scaling technique that fits the wide range of values within a scale ranging from 0 to 1.

We separated benign and attack data instances from the training datasets provided by the repositories mentioned above. We used supervised methods that perform better with a balanced dataset. To make a training dataset balanced, we first chose the data type (benign or attack) with the minimum count of data instances within that training dataset. Using this minimum count, we have taken almost the same amount of normal (benign) and anomalous (attack) data instances to prepare a balanced training set for NSL-KDD and UNSW-NB15 datasets. The original amount of testing data instances were considered for preparing the testing set for these datasets. However, the CICIDS2017 dataset doesn't have separate training and testing sets. To prepare a balanced dataset, we followed the same procedure as with the two datasets. Then it was split in a 75 to 25 ratio where 75% are training data, and 25% are testing data. The numbers of benign and attack data in the three datasets are shown in TABLE 6.2.

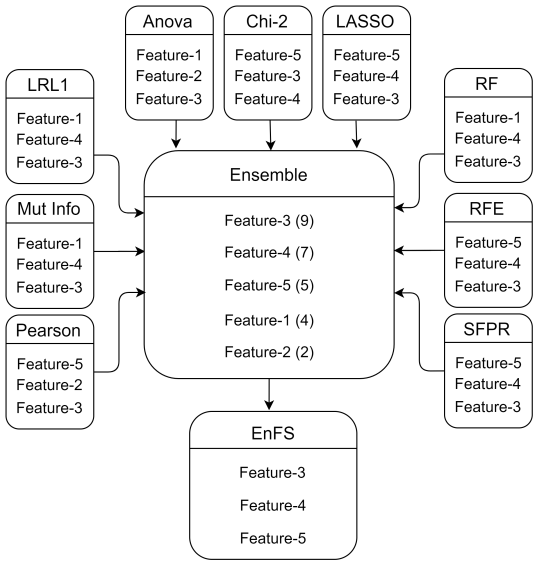


Figure 6.2: Ensemble feature selection approach (EnFS) using majority voting.

6.3.3 Ensemble Feature Selection (EnFS)

Ensemble feature selection is a process of identifying the best feature subset based on the majority voting technique. The process is illustrated in Fig. 6.2, where we use nine individual FS methods. Afterwards, we rank the features according to majority voting where more than half of the FS methods select the features. The major contributions of this research are to use several feature selection algorithms, perform a comparative analysis among them, and ensemble them using majority voting. Based on the performances, we have chosen the top nine FS methods from a list of fifteen well-known methods. They are Anova, Chi-squared, LASSO, Logistic Regression with L1 Penalty, Random Forest, Recursive Feature Elimination, Pearson Correlation, Mutual Information, and SFPR. First, we used these feature selection methods to find the optimal feature sets, and the ensemble supervised framework trained five individual and six ensemble models for performance analysis using these feature sets. We tuned the feature selection methods using several feature counts, like 5, 10, 15, 20, 25, 30, and full set. In most of the cases for all three datasets, feature count 20 generated the optimal feature sets that produced the best results among them. Then we combined all nine feature sets using a majority voting technique and generated a minimal number of features. All features i.e.; nine feature sets from nine FS methods, feature set from EnFS approach, and a full set of features are used to train the models listed in ensemble supervised framework (SupEnML) for analysis purposes.

6.3.4 Model Classification

In our proposed ensemble supervised ML framework, we perform two-stage classification using individual and ensemble classifiers sequentially to discern anomalies from the datasets. We train five supervised individual models, such as Logistic regression (LR), Decision tree (DT), Naïve Bayes (NB), Neural network (NN), and Support vector machine (SVM) using the train-

Table 6.2: Data Instances for NSL-KDD, UNSW-NB15, and CICIDS2017 Datasets Used in Experiment.

Dataset	Training			Testing			Verification		
	Total	Benign	Attack	Total	Benign	Attack	Total	Benign	Attack
NSL-KDD	125974	67343	58630	22544	12833	9711	1000	200	800
UNSW-NB15	112000	56000	56000	70000	35000	35000	1000	200	800
CICIDS2017	75000	38000	38000	25000	13000	12000	1000	200	800

Table 6.3: Hyper-Parameter Values Used for Different Individual and Ensemble Classifiers.

	Classifier	**Short Names**	**Hyper-parameter Values**
Supervised Models	Logistic Regression	LR	random_state=0, solver='lbfgs', multi_class='multinomial'
	Decision Tree	DT	default parameters
	Naïve Bayes	NB	alpha=1.0, binarize=0.0, fit_prior=True, class_prior=None
	Neural Network	NN	solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(5, 2), random_state=1
	Support Vector Machine	SVM	C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=True
Ensemble Models	Majority Voting	Ens_MV	none
	Decision Tree	Ens_DT	default parameters
	Naïve Bayes	Ens-NB	alpha=1.0, binarize=0.0, fit_prior=True, class_prior=None
	Logistic Regresion	Ens_LR	random_state=0, solver='lbfgs', multi_class='multinomial'
	Neural Network	Ens_NN	solver='lbfgs', alpha=1e-5, novelty=True hidden_layer_sizes=(5, 2), random_state=1
	Support Vector Machine	Ens_SVM	C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=True

ing data. These models are tested using the testing data, and the corresponding test outputs (e.g; 1 for anomalous, 0 for benign) generate a prediction matrix. Appended by the label from testing data, this prediction matrix forms the data for ensemble classifiers. This dataset is divided into training and testing data with a ratio of 70 to 30. Then we train and test six ensemble classifiers, namely Ensemble majority voting (Ens_MV), Ensemble logistic regression (Ens_LR), Ensemble naive bayes (Ens_NB), Ensemble neural network (Ens_NN), Ensemble decision tree (Ens_DT), and Ensemble support vector machine (Ens_SVM). We tuned the hyper-parameters of these classifiers using a grid search algorithm to obtain the best valued hyper-parameters that are listed in our paper earlier [124]. We have performed 10-fold cross-validation during the model training time over a randomly divided dataset to avoid the model overfitting. After a comparative performance analysis of these eleven models using the evaluation metrics, the best performing model is obtained for anomaly detection. The mechanism of combining individual classifiers using the ensemble technique is illustrated in Fig. 6.1.

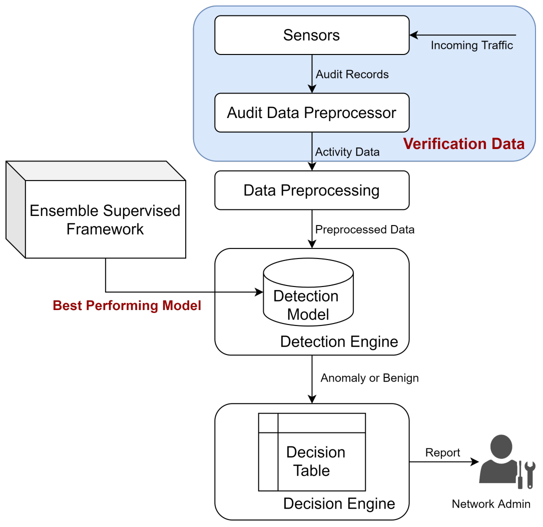


Figure 6.3: Ensemble machine learning framework as the detection technique of an IDS.

6.3.5 Anomaly Detection

In the real world, an IDS as shown in Fig. 6.3 can be placed at the gateway in a secured network. The IDS consists of sensors and an audit data preprocessor, to convert the incoming traffic into activity data. We use 'Verification data' to mimic the real time data stream. The best performing model obtained from the ensemble supervised framework is used as the detection model of the IDS. The detection model within the detection engine analyzes the verification data and identifies the data as anomaly or benign. Afterwards, based on the rules from the decision table, the decision engine takes necessary actions and reports the network admin about the potential threat.

6.3.6 Proposed Algorithm

We implement an ensemble feature selection and ensemble classification algorithm to improve the overall performance of our proposed machine learning model. As defined in Algorithm 1, it takes input as features and targets for training and testing purposes of our classification models. We initialize a list of feature selection methods, machine learning models for individual classification, machine learning models for ensemble classification, and the number of best features as shown between lines 1 to 8. Also, the data structure for storing the individual model prediction result has been initialized here. The ensemble feature selection and classification approach execute for all considered best feature sets controlled by the outer loop in line number 9. The k-best features extracted by individual feature selection algorithms have been stored between lines 11 to 14 for ensembling them later. We use the training features and target to extract the best feature set from the dataset. The ensemble feature set has been calculated based on the majority voting technique, i.e.,a feature is essential when it is selected by at least half of the feature selection algorithm. The code block starting from line number 15 to 20 executes to identify the ensemble feature set, and it is the end of the ensemble feature selection part of our algorithm. The ensemble classification part of the algorithm has started from line number 21, where the code block from line number 21 to 27 trains our machine learning models and stores the classification result in the data structure initialized at the beginning of the algorithm. The accumulated result has been used as our new dataset where features are the individual classification results, and the target is the ground truth from the dataset. The new dataset has been trained and classified again by all machine learning models considered for our experiment, and it shows in between line numbers 30 to 34. Finally, we analyze the result at line number 35 to evaluate the performance of the individual classifier to identify the best-performing model.

6.4 Analysis of Experimental Results

6.4.1 Software and Hardware Preliminaries

We have used Python and the machine learning library scikit-learn [128] to conduct the experiments in computers with the configuration of Intel®CORE™i5-6600K CPU@3.50GHz, 8GB memory & 64-bit Ubuntu operating system, and Intel (R) i5-5250U CPU@1.6GHz, 4GB memory & MacOS operating system.

6.4.2 Datasets

In this research, three intrusion detection benchmark datasets are used to perform all of the experiments. Details of these datasets are as follows:

NSL-KDD: NSL-KDD [102] dataset is the improved version of the KDD'99 dataset proposed by the Canadian Institute of Cybersecurity and is widely used for detecting network intrusion. It comprises 41 features and a class label to describe 39 common cyberattacks.

UNSW-NB15: The Cyber Range Lab of the Australian Centre for Cyber Security created the UNSW-NB15 dataset [103] using the IXIA PerfectStorm tool. They captured 100 GB of raw traffic that consists of nine types of cyberattacks. The dataset has a total of 49 features with a class label.

CICIDS2017: The Canadian Institute for Cybersecurity generated the CICIDS2017 [104] dataset that represents recent day cyber-attacks by analyzing network traffic using CICFlowMeter. This benchmark dataset exclusively covers 11 necessary criteria that make it reliable to cybersecurity data analysts.

6.4.3 Evaluation Metrics

In this research, well established evaluation metrics are required to find the best performing model which can be incorporated within an IDS. Sensitivity, specificity, accuracy, precision, recall, and f-measure are the well-known performance evaluation metrics. These are derived from four basic terms, such as true positive, false positive, true negative, and false negative rates. The evaluation metrics are defined as follows:

1. *Accuracy* $[(TP + TN)/Total]$: the percentage of true DDoS detection over total data instances.

2. *Precision* $[TP/(FP + TP)]$: is the measurement of how often the model correctly identifies a DDoS attack.

3. *False Positive Rate (FPR)* $[FP/(FP + TN)]$: indicates how often the model raise a false alarm by identifying a benign as a DDoS attack.

4. *Recall* $[TP/(FN+TP)]$: is the measurement of how many of the DDoS attacks the model does identify correctly. Recall is also known as true-positive rate, sensitivity, or DDoS detection rate,

5. *F1-Score* $[2TP/(2TN + FP + FN)]$: is the harmonic average of precision and recall.

Additionally, ROC curve is used to portrait the relationship between the True Positive and False Positive rates.

6.4.4 Discussion of Results

In this section, we describe the results obtained from our experiments. As mentioned earlier, we have performed similar experiments with three datasets, NSL-KDD, UNSW-NB15 and CICIDS2017, in two phases: feature selection and data modeling.

TABLE I, II and III listed in the above mentioned URL show the features versus feature selection methods for NSL-KDD, UNSW-NB15 and CICIDS2017, respectively, where 1 represents a feature (in a row) is selected by a corresponding FS method (in a column) and 0 represents not selected. The last column counts the total selection by the FS methods for a single feature. The features are listed in a descending order of total count value. As we have used nine FS methods, majority wins when the count is greater than 4.5 (i.e; 5). The count values for the features below 5 are discarded from the feature list and are not shown on these tables. This majority voting technique is used to select a total of 20 out of 43, 19 out of 42, and 22 out of 80 optimized features from NSL-KDD, UNSW-NB15, and CICIDS2017 datasets,

Table 6.4: Best Performing Results Using Nine Feature Sets, Full Feature Sets and Ensemble Feature Sets for NSL-KDD Dataset.

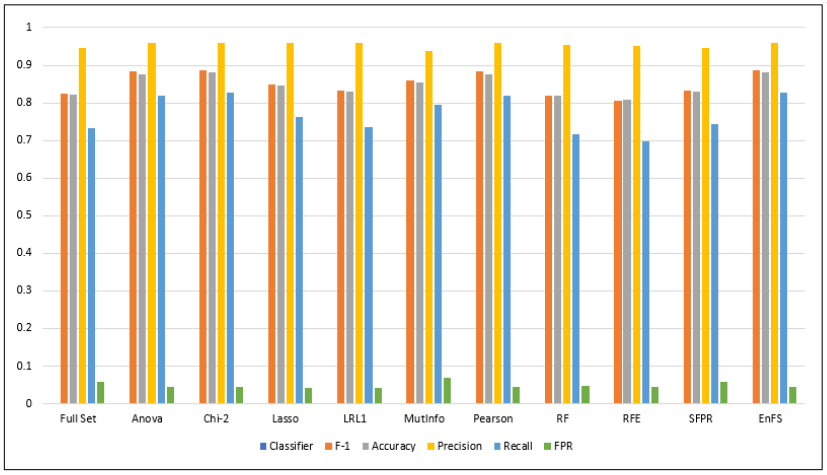
Feature Sets	Classifier	F-1	Accuracy	Precision	Recall	FPR	ROC_auc	Elp_time
Full Set	Ens_DT	0.825	0.823	0.945	0.732	0.057	0.889	0.209
Anova	Ens_LR	0.884	0.877	0.96	0.819	0.045	0.895	0.296
Chi-2	Ens_LR	0.887	0.881	0.959	0.826	0.046	0.898	0.278
Lasso	Ens_DT	0.85	0.846	0.959	0.763	0.043	0.871	0.238
LRL1	Ens_DT	0.832	0.831	0.959	0.735	0.041	0.857	0.263
MutInfo	Ens_NB	0.86	0.853	0.938	0.795	0.07	0.841	0.211
Pearson	Ens_LR	0.884	0.877	0.96	0.818	0.045	0.895	0.3
RF	Ens_NN	0.818	0.819	0.953	0.717	0.047	0.843	1.455
RFE	Ens_DT	0.805	0.807	0.952	0.697	0.046	0.835	0.232
SFPR	Ens_DT	0.832	0.829	0.945	0.743	0.057	0.858	0.509
EnFS	Ens_NB	0.887	0.881	0.959	0.826	0.046	0.892	0.234

respectively. Our proposed EnFS technique significantly reduces the feature set for classification model by 53.5%, 54.8%, and 72.5 % respectively for NSL-KDD, UNSW-NB15, and CICIDS2017 dataset.

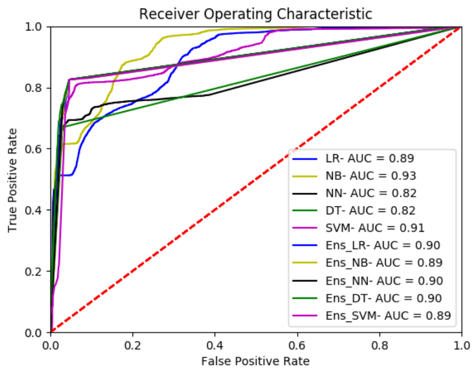
From each of these three datasets, eleven feature sets are extracted: nine of them are from nine individual FS methods, a feature set from ensemble feature selection framework, and the full feature set (i.e., no feature selection was applied). Using these eleven feature sets, we trained and tested our classifiers which are listed in ensemble supervised framework. For each feature set, the ensemble supervised framework is used to train five individual models and six ensemble models. Therefore, a total of 121 models are generated for eleven feature sets. After analyzing the performances of the eleven trained models for each feature set in terms of F-1 score, FPR and training time (Elp_time as elapsed time in tables), the best performing models are listed in TABLE 6.4, 6.5 and 6.6 for NSL-KDD, UNSW-NB15 and CICIDS2017 dataset, respectively. Our experimental results shows an out-performance of proposed ensemble classification model in comparison with the individual classifier. In case of NSL-KDD and CICIDS2017 dataset, our proposed SupEnFS framework shows 100% better classification accuracy for all individual and ensemble feature set while 7 out of 11 (63.3%) best performing classifier were our proposed ensemble machine learning model for UNSW-NB15 dataset. Overall, our proposed ensemble classification model outperform than the individual classification model for all datasets in the experiments.

From TABLE 6.4, a better performance measure in terms of F-1 (0.825), Accuracy (0.881), Precision (0.959), Recall (0.826), False Positive Rate (0.046), etc. is achieved using our EnFS method compared to any individual FS method except Chi-square. However, our EnFS method requires comparatively smaller training time (0.234 Sec.) than that for Chi-square (0.278 Sec.). The graphical representation of this table is illustrated in Fig. 6.4 (a). From this bar chart, it is clear that the bunch of bars at the rightmost side which is for EnFS is found better compared to any other bunches. Fig. 6.4 (b) represents the ROC curves for all 11 models using EnFS feature set for NSL-KDD dataset.

TABLE 6.5 shows a better performance measure in terms of Accuracy (0.881), Precision



(a) Performance analysis.



(b) ROC curves.

Figure 6.4: a) Comparative performance analysis using nine feature sets, full feature sets and ensemble feature sets for NSL-KDD dataset, b) ROC curves for ensemble feature selection using NSL-KDD datasets.

(0.808), False Positive Rate (0.22), etc. except the similar F-1 score (0.867) with the Chi-squared and Recall (0.935) deviation of 0.064 from the best score (0.999 for SFPR and RFE) using our EnFS method compared to any individual FS method. The table is graphically represented in Fig. 6.5 (a) in order to visualize the performances of all FS methods. Fig. 6.5 (b) represents the ROC curves for all 11 models using EnFS feature set for UNSW-NB15 dataset, where Ens_NN model shows the best auc value.

In TABLE 6.6, we see the better performance measure in terms of F-1 (0.995), Accuracy (0.995), Precision (0.995), Recall (0.996), False Positive Rate (0.006), etc. using our EnFS method compared to any individual FS method except the LASSO method, which has lesser FPR (0.005) than that for the EnFS method (0.006). Fig. 6.6 (a) shows the visual representation of the table. In Fig. 6.6 (b), we see the ROC curves for all 11 models using EnFS feature set for CICIDS2017 dataset.

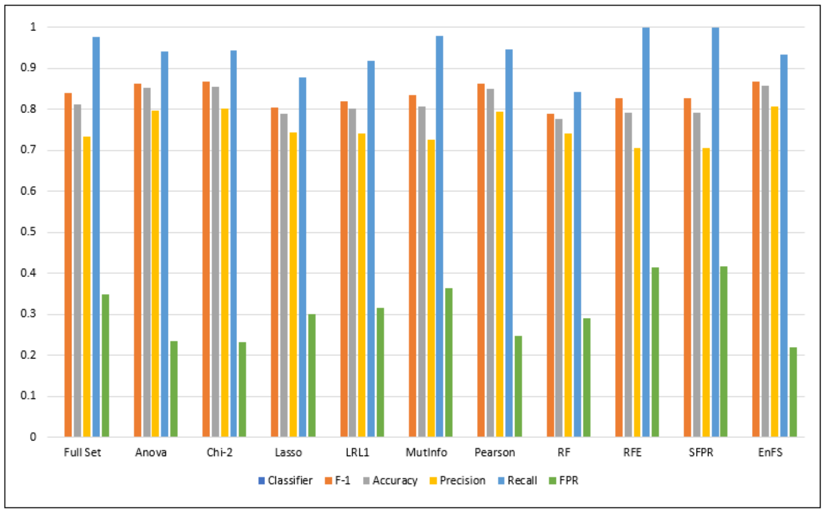
From TABLE 6.4, 6.5, and 6.6, it is clear that Chi-squared and LASSO methods generate equal or better performances for some metrics like, FPR in CICIDS and Recall in UNSW-NB15

Table 6.5: Best Performing Results Using Nine Feature Sets, Full Feature Sets and Ensemble Feature Sets for UNSW-NB15 Dataset.

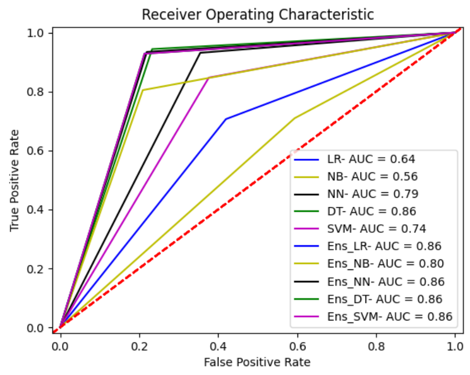
Feature Set	Classifier	F-1	Accuracy	Precision	Recall	FPR	ROC_auc	Elp_time
Full Set	Ens_DT	0.839	0.813	0.735	0.977	0.348	0.815	0.241
Anova	Ens_DT	0.864	0.853	0.798	0.942	0.235	0.853	0.242
Chi-2	DT	0.867	0.856	0.802	0.944	0.233	0.856	1.196
Lasso	Ens_SVM	0.805	0.789	0.743	0.879	0.3	0.79	301.68
LRL1	Ens_DT	0.821	0.801	0.742	0.919	0.316	0.802	0.245
MutInfo	Ens_DT	0.835	0.808	0.727	0.98	0.363	0.809	0.242
Pearson	DT	0.863	0.85	0.794	0.946	0.246	0.85	1.193
RF	Ens_SVM	0.789	0.776	0.741	0.844	0.291	0.776	315.367
RFE	NN	0.828	0.792	0.706	0.999	0.415	0.792	14.363
SFPR	SVM	0.828	0.792	0.706	0.999	0.416	0.792	6681.006
EnFS	Ens_NN	0.867	0.857	0.808	0.935	0.22	0.858	6.019

Table 6.6: Best Performing Results Using Nine Feature Sets, Full Feature Sets and Ensemble Feature Sets for CICIDS2017 Dataset.

Feature Set	Classifier	F-1	Accuracy	Precision	Recall	FPR	ROC_auc	Elp_time
Full Set	Ens_DT	0.945	0.942	0.941	0.948	0.063	0.978	0.105
Anova	Ens_DT	0.892	0.884	0.861	0.925	0.16	0.955	0.102
Chi-2	Ens_DT	0.938	0.933	0.9	0.978	0.116	0.961	0.103
Lasso	Ens_DT	0.995	0.995	0.995	0.995	0.005	0.998	0.1
LRL1	Ens_DT	0.94	0.936	0.907	0.975	0.106	0.967	0.103
MutInfo	Ens_DT	0.945	0.944	0.965	0.925	0.036	0.97	0.101
Pearson	Ens_DT	0.892	0.884	0.861	0.925	0.16	0.955	0.101
RF	Ens_DT	0.92	0.92	0.944	0.898	0.057	0.969	0.102
RFE	Ens_DT	0.946	0.946	0.985	0.911	0.015	0.982	0.101
SFPR	Ens_SVM	0.94	0.935	0.904	0.978	0.111	0.953	4.037
EnFS	Ens_NN	0.995	0.995	0.995	0.996	0.006	0.998	0.286



(a) Performance analysis.

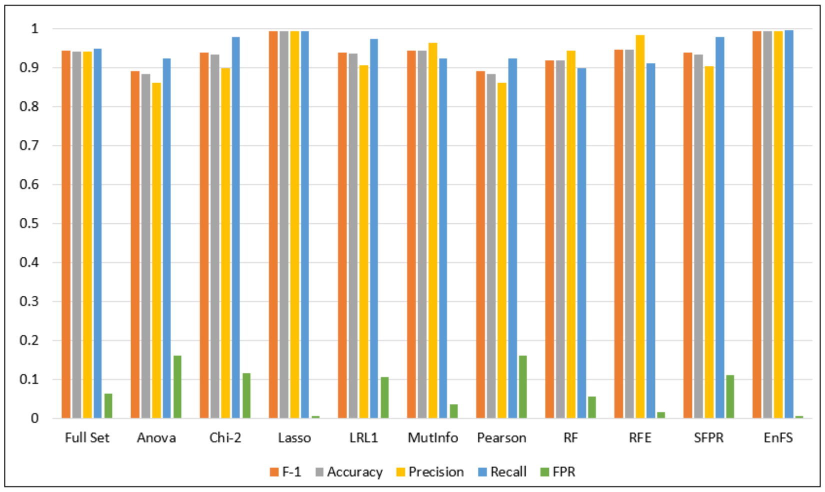


(b) ROC curves.

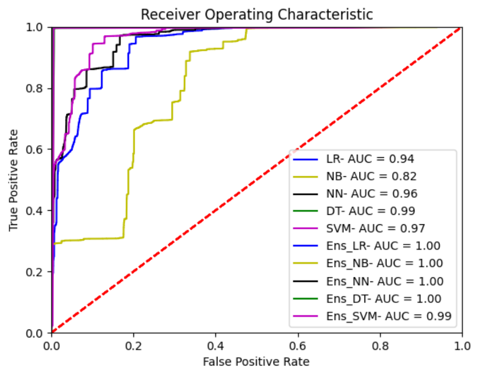
Figure 6.5: a) Comparative performance analysis using nine feature sets, full feature sets and ensemble feature sets for UNSW-NB15 dataset, b) ROC curves for ensemble feature selection using UNSW-NB15 datasets.

dataset but not for all performance metrics when compared with our EnFS method. On the other hand, our EnFS method always yields better performances for most of the performance metrics and especially in case of classification accuracy and F1-score which is very crucial for any machine learning based IDS.

Comparing TABLE 6.4, 6.5 and 6.6, and Fig. 6.4, 6.5, and 6.6, we can summarize that the feature set obtained from the EnFS framework provides the best performances in terms of F-1 score and Accuracy for all experimental setups. Also the FPR (False Positive Rate) was better for most of the dataset (2 out of 3) where for CICIDS2017 dataset the FPR was a slightly lower (0.001) than the best FPR (0.006). From these tables, it is also clear that our proposed ensemble classification models (i.e., the Classifier in the tables that starts with Ens_) outperform single models for more than 85% (29 classification model out of total 33 model) of our experimental models. So, it is very clear from extensive experimental result that our proposed ensemble feature selection (EnFS) based SupEnML, an anomaly classification framework, gen-



(a) Performance analysis.



(b) ROC curves.

Figure 6.6: a) Comparative performance analysis using nine feature sets, full feature sets and ensemble feature sets for CICIDS2017 dataset, b) ROC curves for ensemble feature selection using CICIDS2017 datasets.

erally perform better than the individual feature selection and classification approach. As per our knowledge, we have come across a very few existing related works on either ensemble classification or feature selection research, and results are not produced for similar datasets that we have used. Therefore, we were unable to provide a detailed comparison between our methods and results with the existing related works. However, we have added a comparison of the results from any of the three datasets irrespective of existing works' ensemble techniques which is summarized in TABLE 6.7.

Furthermore, we verified the best performing models for each datasets obtained from our proposed methods using the verification data consisting of 1000 instances. Among these instances, 800 of them are attacks and 200 are benign. Table 6.8 shows the performances of three best models using the verification data. From the table, we observe that the performance measures are almost the same with the results of EnFS from TABLE 6.4, 6.5, and 6.6.

Table 6.7: A Performance Comparison of Our Method Compared to Existing Methods.

Method	Ensemble feature Selection	Ensemble machine learning	Verification Dataset	Best performance (Accuracy)
Ensemble, [105]	Not implemented	Adaptive voting	NSL-KDD	85.2%
Ensemble, [106]	Not implemented	Bagging (Base classifier - J48)	NSL-KDD	84.25%
Ensemble, [107]	Not implemented	DAREnsemble	NSL-KDD	78.8%
SVM, [108]	Not Implemented	Not Implemented	NSL-KDD	82.68%
Deep Learning, [113], [114], [115]	Not Implemented	Not Implemented	NSL-KDD, KDD-Cup 99, UNSW-NB15, WSN-DS, CICIDS 2017	NSL-KDD (DNN): 78.9%, UNSWNB-15 (DNN): 78.4%, CICIDS 2017 (DNN): 96.3%, NSL-KDD (ML): 93.4%, UNSWNB-15 (ML): 90.3%, CICIDS 2017 (ML): 94.1%
Our work	EnFS	SupEnML	NSL-KDD, UNSWNB-15 CICIDS	NSL-KDD: 88.1%, UNSWNB-15: 85.7%, CICIDS 2017: 99.5%

Table 6.8: Verified the Best Performing Models' Performances Using Verification Set for Three Datasets.

Dataset	Best Model	Instances		Predicted Results				F-1 Score	Accuracy	Precision	Recall	FPR
		Attack	Benign	Attack		Benign						
				TP	FP	TN	FN					
NSL-KDD	Ens_NB	800	200	640	9	191	160	0.883	0.831	0.986	0.8	0.04
UNSW-NB15	Ens_NN	800	200	618	21	179	182	0.859	0.797	0.967	0.773	0.105
CICIDS2017	Ens_NN	800	200	790	1	199	10	0.993	0.989	0.999	0.988	0.005

6.5 Conclusion

Typically, the network anomalies can be detected using an intrusion detection system (IDS). In this research, we have combined ensemble feature selection and ensemble machine learning approaches as the detection mechanism within an IDS to detect the network anomalies. For the ensemble feature selection framework, we first have experimented with the feature sets obtained from nine feature selection methods and then have combined these feature sets to get the minimal number of features using majority voting. Our experiments demonstrate that the feature set obtained from the EnFS approach has better outcomes compared to any individual FS method. In addition, we have used this feature set in our ensemble supervised ML framework to find the best performing model which can be incorporated into any IDS. A total of eleven feature sets (nine from nine different FS methods, a feature set from EnFS, and a full set of features) have been used for the experimentation. We have also performed a comparative analysis among these feature sets where the EnFS set outperforms individual feature sets in most cases. Moreover, for each feature set, we have used our ensemble supervised ML framework to train eleven models (five individual and six ensemble models). Almost 80% of ensemble models outperform single models. For this extensive experimentation, we have used NSL-KDD, UNSW-NB15, and CICSIDS2017 datasets.

Using a full set of data instances for all three datasets can exact a toll in terms of longer experimentation runtime. Therefore, sometimes we have used a reduced amount of data for training purposes to boost the simulation time, and thus compromising the probability of a near-perfect performance. In future, we plan to incorporate the full amount of data as well as new datasets (including other domains) to verify the efficiency of our method. Furthermore, we plan to consider unsupervised learning with our ensemble classification which will solidify our method. In addition, adversarial machine learning (AML) can be added as an extension of our research in the future to remove the malicious adversaries by enabling safe adoption of the ML techniques in adversarial settings, and thus to retain the whole system security [129].

Chapter 7

ENIDS: A Hybrid Model Using Stacking Ensemble and Deep Learning for Network Intrusion Detection and Classification of Cyber-Attacks

Abstract: Rapid and widespread adoption of emerging Information Technology (IT) infrastructures and services in commercial and private endeavors opens new horizons for novel cyberattacks. Network Intrusion Detection Systems (NIDS) gained attention as an effective means of combating various cyber threats. Recent research demonstrates the potency of machine learning (ML) and deep learning (DL) approaches in the development of NIDS. In this study, we propose a DL-based framework called the Ensemble approach for Network Intrusion Detection System (ENIDS) to detect various types of cyberattacks, which includes dynamic data pre-processing, optimal feature selection, the handling of imbalanced data samples, and a DL-based ensemble model. Our DL-based ensemble model is comprised of two layers: the base learner and the meta-learner. The base learner is composed of three robust DL models: convolutional neural networks (CNN), long short-term memory (LSTM), and gated recurrent units (GRU), and the meta-learner is a deep neural network (DNN) model. The proposed framework experimented with two publicly available and popular network traffic datasets, namely UNSW-15 and CICIDS-2017. In the UNSW-15 and CICIDS-2017 datasets, our proposed framework detects cyberattacks with an accuracy of 90.6% and 99.6% and an f1-score of 90.5% and 99.6%, respectively. According to experimental findings, the proposed ensemble framework outperforms existing state-of-the-art approaches and demonstrates better performance than benchmark DL methods in terms of accuracy, f1-score, and execution time for training and testing.

7.1 Introduction

IT infrastructures have been expanding rapidly in capacity, system loads, and complexity in recent years. People's lives are becoming more reliant on the applications of smart appliances, smartphones, and cloud services. Particularly, when the COVID-19 outbreak began in 2020, the internet became a lifeline for people in practically every industry. According to the Pew Research Center, 90% of Americans regarded various online tools as a vital part of their daily lives during the coronavirus pandemic, with most of them using those tools for business, video meetings, academics, and communication purposes [130]. As per DataReportal, a total of 4.94 billion individuals, or 62.4% of the world's population, use the Internet [131]. Besides that, the deployment of 5G and its advanced features is expected to boost the number of 5G users to 1.02 billion in the coming year [132]. This rapid expansion of the users, network infrastructures, and protocols of emerging technologies is posing security issues and opening horizons for novel threats [133]. 5G networks, for example, provide greater bandwidth and reduced latency but expand the attack surface, increasing system vulnerabilities [134].

Any offensive action conducted towards diverse systems to access, steal, or modify valuable data, install malicious software, or interrupt routine services without the administrator's consent is considered a cyberattack. Any cyberattack can be categorized as targeted or untargeted. Targeted attacks attempt to infiltrate specific businesses, while untargeted attacks breach as many systems or devices as possible. Targeted cyberattacks include spear-phishing and deploying botnets, while untargeted attacks include ransomware and phishing [135]. Malware, phishing, denial-of-service (DoS), distributed denial-of-service (DDoS), reconnaissance, backdoors, etc. are some common types of cyberattacks.

Cyberattacks on both the core network and host levels have increased in recent years, disrupting a broad range of business and government domains. According to Check Point Research, the weekly incidence of cyberattacks soared to 50% during the COVID-19 pandemic, with education and research being the most afflicted sectors [136]. In 2021, a large-scale cyberattack hampered nine US government agencies, where attackers deployed untraceable backdoors in Microsoft Exchange and compromised access to the server to gain all information residing inside the server [137]. According to CSIS (Center for Strategic and International Studies), a Russian hackers group launched an ”information attack” in February 2022 to obtain access to two of Ukraine's financial institutions, causing service outages [138]. Throughout the first half of 2021, 5,591 network layer DDoS attacks were reported, according to the ”2021 DDoS Threat Landscape Report”, and the attack technique is changing in dimension, volume, regularity, and complexity [139].

In order to detect and prevent cyberattacks on the networks, firewalls are used as the primary defense mechanism, and a network intrusion detection system, or NIDS, is employed as an advanced security mechanism. NIDS is mainly a security application that can analyze network traffic, detect security threats, and take measures when an anomaly is detected. Based on Donghwoon et al. [140] intrusion detection system (IDS) should consist of three main steps: First, IDS needs to collect and track network flow data. After that, IDS needs to reformat and reshape the network data by cleaning the raw data, processing them, and making them feasible in the next step. Finally, using the processed data as input, a model will determine if network traffic is normal or possesses any abnormalities. Developing an effective and efficient model is the most crucial part of building a NIDS. IDS are categorized as signature-based and anomaly-based. Traditional signature-based intrusion detection systems (IDS) compare network packets to known signatures. In contrast, models in anomaly-based IDS produced by integrating ML and DL are capable of learning attack signatures to create the attack pattern. Because of the dynamic nature of diverse cyberattack patterns, anomaly-based IDS is more viable than traditional IDS. However, DL techniques outperform shallow ML-based approaches in the case of high-dimensional and substantially large datasets [141].

Many previous studies utilized UNSW-15 [103] and CICIDS-2017 [104] to detect network abnormalities; however, the majority followed binary classification or shallow ML and DL approaches. Because each type of threat follows a different attack procedure, multi-classification is getting attention because it aids in selecting prevention techniques [142]. Also, ensemble approaches were utilized to get better accuracy than baseline ML and DL approaches [133]. To address the previous challenges, we proposed a DL-based stacking ensemble approach to identify and categorize various DDoS attacks [143]. This is an extension of our previous work, where we present an efficient and dynamic NIDS that can detect a wide range of cyberattacks. This time we introduced the Synthetic Minority Oversampling Technique (SMOTE) in data pre-processing to tackle imbalanced datasets, and we included a few more datasets to evaluate the efficacy of our proposed hybrid model. The rest of the chapter is organized as follows: Section 7.2 discusses the previous state-of-the-art approach to developing NIDS with AI techniques. The proposed framework for detecting cyberattacks using ensemble techniques is briefly discussed in Section 7.3. Section 7.4 presents and discusses the experimental results of the proposed framework. Finally, section 7.5 concludes the paper with future directions.

7.2 Literature Review

This chapter will review the related literature of previous works in the network intrusion detection system (NIDS) field as well as pinpoint the shortcomings of existing studies, which is the motivation for this research. This section discusses three distinctive phases of developing a NIDS: statistical-based approach, machine learning-based approach, and deep learning-based approach.

7.2.1 Machine Learning-Based Approach

In 1999, Chris Sinclair used an ML-based expert system to automatically classify a network connection by utilizing network patterns [144]. They employed a genetic algorithm and decision tree to develop the IDS to detect "low and slow" attacks, which may contain intrusion behavior. Zhang et al. presented a hierarchy-based network intrusion detection system (HIDE) using a hybrid model comprised of perceptrons and backpropagation to distinguish normal and abnormal traffic flows [145]. The HIDE model is composed of three tiers, and each tier has multiple agents to detect network intrusion. Tier-1 pre-processes the network traffic collected by the probe and sends periodic reports about the traffic to Tier-2. Tier-2 observes the LAN to check network status, and finally, Tier-3 receives data from both Tier-1 and Tier-2. After pre-processing, all data are sent to the statistical processor to be converted into stimulus vectors and feed them to the perceptron and backpropagation processor to classify network traffic. Ripon et al. employed an ML-based approach in an old dataset named NSL-KDD to detect cyberattacks generated from the Internet of Things (IoT) devices, cloud computing, and social networking sites [146]. Their research focused on comparing the effectiveness of different ML models in detecting four different types of cyberattacks, and they concluded that their utilized classification algorithms, Random Forest (RF) and Support Vector Machine (SVM), can detect cyberattacks effectively.

Eskin et al. first presented a network intrusion detection technique using unsupervised ML methods in 2002. It includes clustering algorithms, an SVM, and the K-Nearest Neighbor algorithm [147]. Their research introduces a geometrical paradigm for unsupervised anomaly detection that maps typical metadata into a feature space. Chih-Fong Tsai and his research team reviewed 55 ML-related intrusion detection systems developed between 2000 and 2007 [109]. They divided the system into single, hybrid, and multi-classifiers and compared them based on the datasets, classification method, and experimental setup. Their findings conclude that K-Nearest Neighbor and SVM are more popular single classifiers, integrated-hybrid classifiers are the most commonly used hybrid classifiers, and ensemble techniques did not receive much attention. Zhang et al. [148] introduced a hybrid NIDS based on Random Forest (RF) that can detect misuse, patterns of intrusions, and outliers. The authors used an unsupervised learning technique to train the anomaly detection components to detect anomalies and outliers in network traffic flow. Finally, their work combines misuse detection with anomaly detection, allowing anomaly detection to detect novel cyberattacks while misuse detection filters out known intrusions. Phurivit et al. [149] developed a real-time IDS named RT-IDS, which could distinguish between normal traffic and anomalies. The authors identify 12 essential features from network traffic that are important for detecting network anomalies, and several ML algorithms such as Decision Tree (DT), and Neural Network with back-propagation are used, where DT with the Rippler rule outperforms all other algorithms.

Injadat et al. [150] introduced a multi-level optimum ML approach for network anomaly detection. Their strategy showed a 99% detection accuracy and reduced the number of false positives by 1% to 2% using the CICIDS-2017 and UNSW-15 datasets. Raisa et al. [151] used Gini Impurity-based Weighted Random Forest to select features from UNSW-15 and Network TON_IoT datasets and compare the performance of different ML models with different feature sets. Roberto et al. [152] proposed an ML-based intrusion detection system to identify the cyberattacks from the network traffic dataset called UGR'16 generated from heterogeneous devices. Saikat et al. [153] performed ensemble feature selection and used ensemble ML techniques to detect different types of cyberattacks in the CICIDS-2017, UNSW-NB15, and NSL-KDD datasets. They proposed ensemble feature selection (EnFS) which achieved better accuracy and f1-score in all three datasets, however, they performed binary classification which did not resemble the detection rate of different types of cyberattacks. In their other study, Das et al. [123] developed an ensemble framework for feature selection (FS) methods that aimed to generate an optimal set of features. They also conducted a comparative analysis of various existing approaches and focused on accuracy and false positives, using the NSL-KDD dataset.

7.2.2 Deep Learning-Based Approach

A research team from the University of Toronto led by Geoffrey Hinton proposed ImageNet and addressed the fact that deep learning (DL) can outperform any ML algorithms in image classification tasks [154]. In the LSVRC-2010 contest, they trained 1.2 million images using Convolutional neural networks (CNN) with thousands of features. They achieved a lower error rate than any other advanced ML model in the ILSVRC-2012 contest, where they achieved only a 15.3% error rate in the test data, which outperformed every other model. In 2015, Yann et al. proposed that traditional ML cannot process raw forms of natural data like pixel values of images, matching news items, user interests in a particular product, and many more [155].

However, DL makes up multiple levels of representation layers, which take raw input in one representation layer and transform it into a higher layer that can learn complex functions more abstractly.

In 2016, Niyaz et al. proposed a DL-based strategy named STL (self-taught learning) to detect network anomalies using the NSL-KDD dataset [156]. Their self-taught learning approach can learn about features from different network sources. These pre-processed features are passed through the auto-encoder and regression with the SoftMax function to classify normal and attack traffic. Their experiments outperform some other research conducted using DL techniques. The authors used the KDD Cup 1999 (KDD-99) dataset for their IDS by using a DL-based method named deep neural network (DNN) and an ML-based method called SVM, where the accuracy of DNN is 15% higher than SVM. Vinayakumar et al. [113] proposed a hybrid deep neural network (DNN) called the scale-hybrid-IDS-AlertNet framework that is both scalable and capable of identifying real-time cyberattacks by analyzing network traffic and host-level events. In a related study [114], the authors evaluated the efficacy of a variety of shallow and deep networks for use in NIDS, utilizing the KDD-99 dataset. Finally, Vinayakumar [115] presented a DL-based botnet detection system designed to detect and classify domain names, which can be implemented at the Internet Service Provider (ISP) level to monitor IoT devices. Before 2016, most of the researchers used KDD-98, KDD-99, and NSL-KDD datasets for their decade-old IDS, which do not match the current network scenario. This dataset crisis was solved when, in 2015, Moustafa et al. published the UNSW-15 Network Flow datasets, which have both normal and anomalous network traffic [103]. The UNSW-15 network dataset is more useful than previous datasets to evaluate NIDS perfection because it represents contemporary network traffic contexts.

Hanif et al. utilized Artificial Neural Networks (ANN) to detect network intrusion on Internet of Things (IoT) devices, and in their experiment, they used the UNSW-15 dataset [157]. In their experiment, instead of predicting different types of network attacks, they only detected the normal attribute of the traffic. In order to examine network traffic characteristics of Internet of Things (IoT) devices in 2018, Moustafa et al. [158] utilized the UNSW-15 dataset, where the authors used AdaBoost ensemble algorithms to discern between normal and abnormal traffic. To construct the NIDS dedicated to identifying attacks in IoT networks, this framework focuses on Domain Name Systems (DNS) and Hypertext Transfer Protocols (HTTP), together with MQ Telemetry Transport (MQTT) and associated flows. They merged three techniques in their framework: ANN, Naive Bayes, and DT, and then passed them through AdaBoost ensemble methods. The DL-based NIDS got more attention from the research community when the Canadian Institute of Cybersecurity (CIC) published two large network traffic datasets, namely CICIDS-2017 [104] and CSE-CIC-IDS-2018.

In 2021 Aleesa et al. [159] utilized UNSW-15 datasets and proposed an AI-based approach to detect network anomalies from the traffic flow. The authors suggested a framework with two levels: level 1 could determine whether a traffic flow was normal or abnormal, and level 2 could classify the many forms of attacks when abnormal traffic was discovered. In their work, instead of incorporating any DL models, shallow ML models are used. In order to evaluate DL and ML-based methods for detecting network anomalies, Liu et al. [160] presented a taxonomy for NIDS. The proposed taxonomy in this work can answer several questions about feature selection, data type selection to predict certain types of attacks, as well as ML and DL model selection based on the type of network data available. Ana et al. [161] made a comparison between incremental and non-incremental broad learning systems (BLS) to identify DoS attacks on the communication networks using the CSE-CIC-IDS-2018 and CICIDS-2017 datasets, where they concluded that in both datasets, the non-incremental approach gives better accuracy; in contrast, the incremental process takes less training time. Faker et al. [162] employed big data as well as DL methods and used UNSW-15 and CICIDS-2017 datasets to evaluate the performance of the models. They incorporated DNN, Random Forest (RF), and Gradient Boosting Tree (GBT), where DNN shows better performance in both datasets for multiclass classification and GBT shows better performance in CICIDS-2017 during binary classification compared to DNN. Azriel et al. [163] employed bidirectional DL methods and showed a performance comparison between traditional ML and DL models.

Unlike the above-mentioned research efforts, we used two different recent network traffic datasets, UNSW-15 released by the University of New South Wales [103], and CICIDS-2017 published by the Canadian Institute of Cybersecurity (CIC) [104], to train our DL-based stacking ensemble model and analyze how it performs in different attack scenarios. Many researchers have proposed ML and DL-based techniques to detect and predict network anomalies. Moreover, some researchers proposed DL-based methods, but most of them were focused on binary classification and used shallow ML and DL methods. Moreover, some researchers used an out-of-date dataset where traffic patterns no longer resemble today's diverse network traffic [164]. In our proposed NIDS, we used a data resampling technique to balance the imbalanced data samples and employed a stacking ensemble technique composed of different DL models to detect the different kinds of cyberattacks.

7.3 Proposed Methodology

This work aims to propose an ensemble NIDS framework to identify several types of cyberattacks that will impact network services. The proposed framework comprises four phases: dataset collection, data pre-processing and normalization, optimal feature selection, data resampling, and base and ensemble model learning. Fig. 7.1 demonstrates the overall overview of the proposed AI-based NIDS framework to detect cyberattacks from network traffic. In the overall experiment, we first performed multi-step data pre-processing. In section 7.3.1, we discuss the utilized datasets in developing the NIDS and performed the first step of data pre-processing to remove the data redundancy as well as improve data quality; the dataset and data pre-processing are discussed in this section. Moreover, we also perform the data normalization method using a min-max scaler to handle outliers.The third phase will perform the optimal feature selection to select the appropriate features for specific types of attacks. Random feature elimination (RFE) is used with 5-fold cross-validation where it defines two options to remove the irrelevant features: select the number of features and utilization of a base algorithm to select the features based on their importance. The feature selection approach is discussed in section 7.3.2. In the fourth phase, we perform data resampling techniques, to solve the data imbalance classification problems. The data resampling method dynamically oversamples and undersamples the data samples to balance the underrepresented minority class in the datasets. The synthetic minority oversampling technique (SMOTE) [165] is used to oversample the minority class, and then the edited nearest neighbor (ENN)[166] is used to oversample the majority class. The details of the data resampling are discussed in section 7.3.3. The final phase in

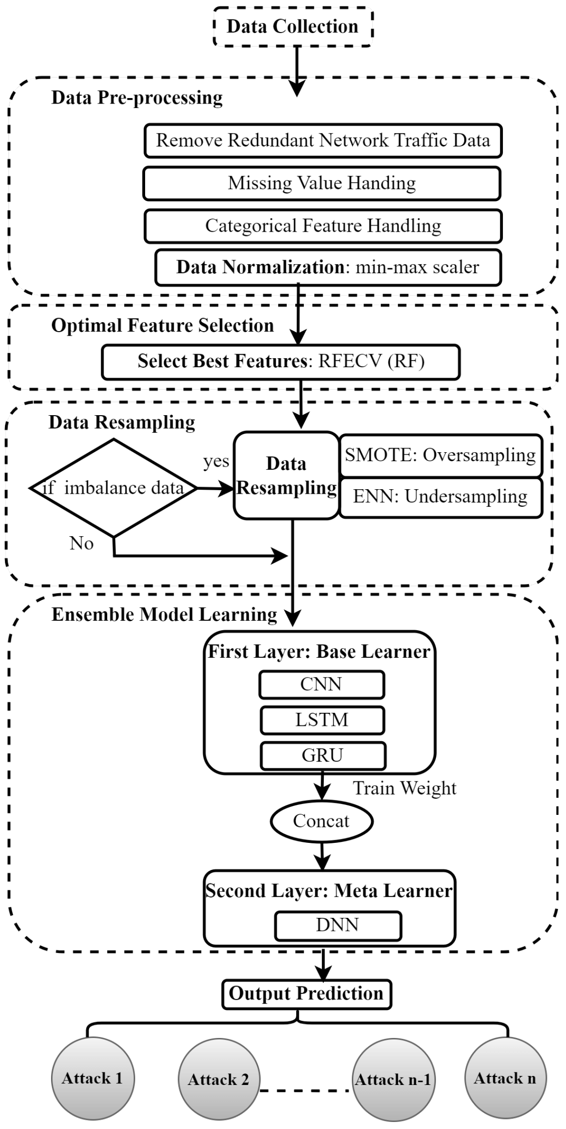


Figure 7.1: The framework of the proposed NIDS.

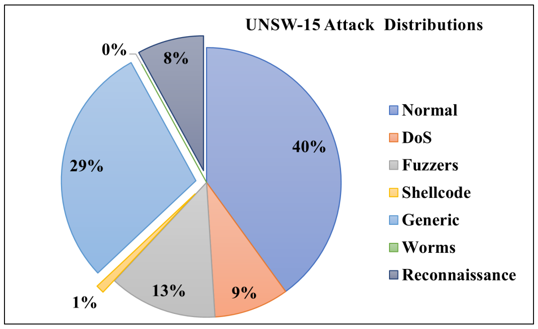


Figure 7.2: UNSW-15 dataset attack distributions.

section 7.3.4 consists of base model learning and ensemble model learning. We use three base learners in our experiments including CNN, LSTM, and GRU, which resided in the ensemble model's first layers. The three base learners are trained with the pre-processed data, and their training weights are saved; their performance with the test data is recorded. The three base learners are discussed in section III-E.1. The weight of the three base learners is concatenated and passed to the second layer of the ensemble model, which is a DNN, and then the ensemble model has trained again. The details of the ensemble model have discussed in sections III–E.2. Finally, section III–E.3 shows the runtime complexity of the proposed ENIDS model.

7.3.1 Dataset Description and Preprocessing

The proposed ensemble NIDS was built using two up-to-date datasets, UNSW-15[103], and CICIDS-2017[104] which contain the different types of most recent cyberattacks. The attack distributions and the number of features are different in both datasets. Data pre-processing is crucial for ML/DL methods as the performance of the model largely depends on pre-processing. In the network traffic datasets, network availability and traffic failure are common issues that can be solved by pre-processing the data from different extents. Data pre-processing removes noise from data, handles missing values, and reduces redundancy from datasets.

Dataset Description

The UNSW-15 dataset captures raw packets with the IXIA PerfectStorm tool. The Australian Centre for Cyber Security (ACCS) published this network flow data available for public use, including both normal and malicious traffic. For archiving 100 GB of raw traffic, the tcpdump device is employed. ACCS used three networks with 45 different IP addresses to create the dataset, which took 31 hours to collect. The UNSW-15 dataset is divided into four CSV files and comprises 2.5 million records with 49 features. The dataset is further divided into 82,332 testing records and 1,75,341 training records. Backdoor, worms, generic, exploits, fuzzers, reconnaissance, DoS, and shellcode are among the nine categories of network attacks classified as anomalous traffic. In our experiments, we worked with worms, generic, fuzzers, reconnaissance, DoS, and shellcode. The attack distributions in the UNSW-15 dataset are similar to Fig. 7.2.

The Canadian Institute of Cybersecurity (CIC) published the CICIDS-2017 network traffic

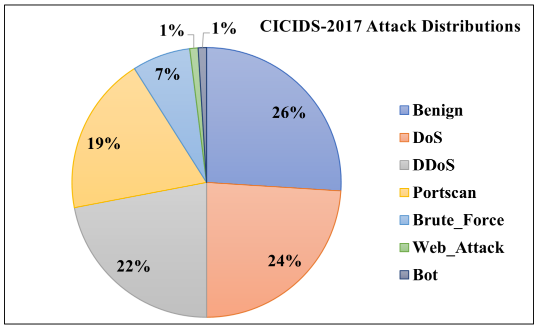


Figure 7.3: CICIDS-2017 dataset attack distributions.

data, which contains eight separate CSV files containing five days of regular and aberrant activity of the network flows from Monday to Friday. They build an attack network with a router and a switch for the testbed, and a victim network with a firewall, router, and switches. They produced normal and anomalous traffic using CICFlowMeter software, and 80 features were collected from traffic-generated pcap files. The heartbleed attack, infiltration attack, brute_force attack, DDoS attack, DoS attack, web attack, and botnet are among the cyberattacks covered in CICIDS-2017. It comprises 2,09,417 records, and the attack distributions of the CICIDS-2017 dataset are like in Fig. 7.3.

Data Pre-processing

We excluded various features from both datasets that have little impact on normal or abnormal traffic in the initial round of our data pre-processing phase. We remove the following features from the UNSW-15 datasets:

1) The 'id' column is removed as it has no impact.

2) Time-based features ('stime', 'ltime') are deleted from the UNSW-15 dataset since they are redundant.

3) Switch-related information like 'sport', 'srcip', 'dstip', and 'dsport' is also removed from the UNSW-15.

4) Categorical features such as 'proto', and 'service' have a wide range of values, thus we employ the label-encoder approach, which generates new, unique numerical values for each category.

5) We also eradicate features like 'ct ftp cmd', 'is ftp login', and 'ct flw http method' which have many missing values.

The CICIDS-2017 dataset is pre-processed by considering the following steps:

1) Less relevant features such as 'timestamps', and 'IP addresses' were eliminated from the CICIDS-2017.

2) As the network flow of this dataset is created using CICFlowMeter, it collects some network-related redundant features such as 'Bwd PSH Flags', 'Bwd Avg Bytes/Bulk', 'Bwd Avg Packets/Bulk', 'Bwd Avg Packets/Bulk', 'Bwd URG Flags', 'Fwd Avg Bytes/Bulk', 'Fwd Avg Packets/Bulk', 'Fwd Avg Bulk Rate', which have many missing values.

3) The 'Fwd Header Length' feature is also excluded as it exists twice in the dataset.

4) Due to the similarity in network traffic behavior of web attack-brute_force, web attack-xss, and web attack SQL injection, they are labeled as web-attack. Additionally, various forms of DoS attacks, including DoS slowHTTP, dos hulk, DoS slow loris, and DoS goldeneye, are labeled as DoS.

After removing redundant features, we performed data normalization, also known as scaling the features. It is a data pre-processing strategy where all the data transforms into the same scale. In deep learning, normalization is a process to transform data within a range between 0 and 1. Moreover, when we do not know about the data distribution then normalization is a good approach. If we do not normalize our data that are measured in different scales, they will not contribute equally during model training. Scaling the data equalizes all features, which also assists the algorithm to converge faster, along with optimizing with the gradient descent algorithm. Data distribution is different in both UNSW-15 and CICIDS-2017 datasets, and to fit the data appropriately into the model to get better accuracy we need to normalize our data. As we frequently encounter extremely large or small values among data samples in network traffic data, the min-max scaler is better suited for network anomaly identification due to its high sensitivity to outliers. In our proposed framework, we employed the min-max data normalization technique which scales the data from minimum range 0 to maximum range 1. The mathematical formulation for the min-max scaler is given below:

$$min - max = \frac{(x - min(x))}{(max(x) - min(x))} \tag{7.1}$$

The real-time lowest and maximum values from all the processed data samples are represented here by min and max.

7.3.2 Optimal Feature Selection

Feature selection is the process of selecting the most relevant features from a huge number of feature sets and eliminating irrelevant features to improve model performance and reduce the computational cost of the predictive models. Moreover, removing noisy features improves learning efficiency, and reduces the training time of the models. Feature importance determines the contribution of each feature in the prediction model. Selecting important features is critical for developing a better NIDS, as it contains network traffic-related irrelevant information which does not contribute to occurring a cyberattack in the network traffic. In our experiments, we utilized Random Feature Elimination (RFE) with a 5-fold cross-validation which is easy to configure, and its configuration consists of two options: selecting the number of features and a base algorithm to select the number of features.

RFE followed both wrapper-style and filter-based feature selection algorithms, where it first searched a subset of features from the entire datasets and removed features until the desired

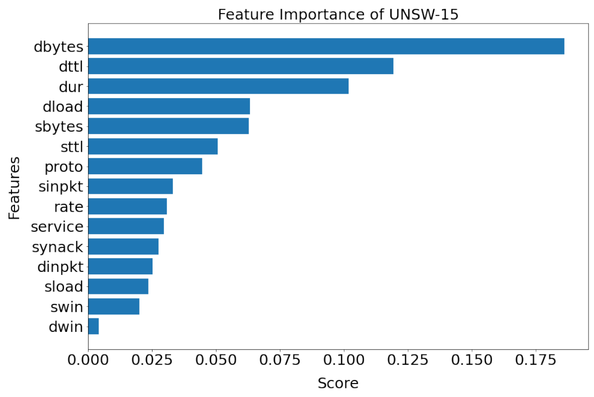


Figure 7.4: Feature importance in UNSW-15 dataset.

number of features remained. RFE performs k-fold cross-validation, removing characteristics that are less significant for the model with each cross-validation. The process continues until RFE has read all the features from the dataset and only keeps those that improve the overall cross-validation performance. Random Feature Elimination with Cross-Validation (RFECV) is a time-consuming method for selecting features yet yields the best results. In our experiments, we did not define how many features we wanted to select, rather we let the algorithm decide to select the optimum number of features. We used Random Forest (RF), as a base algorithm which is a supervised ML method that employs both the bagging method and DT, to identify the optimal features for our NIDS model. To calculate the soft voting for classification, RF takes the original columns, fits them into decision trees, and then mixes them. The RFECV with Random Forest gave us 25 relevant features from the UNSW-15 dataset and 49 applicable features from the CICIDS-2017 dataset. We also calculated the feature importance using the RF by increasing the purity of the child nodes. One feature gains relevance when the purity of its related child nodes is improved. Each tree estimates the importance of each feature, which is then averaged to obtain the total feature importance. Fig. 7.4 and Fig. 7.5 portray the feature importance of the 15 best features from the UNSW-15 and CICIDS-2017 datasets.

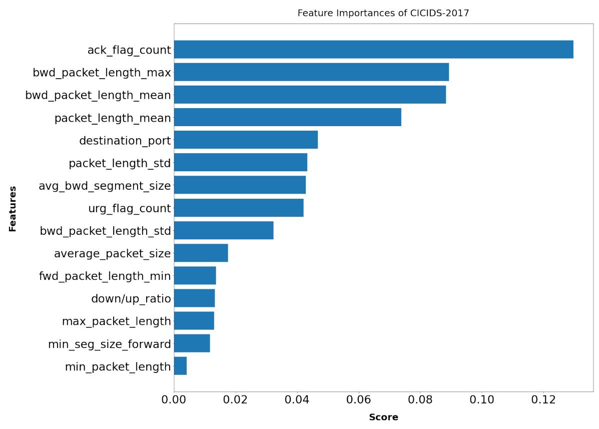


Figure 7.5: Feature importance in CICIDS-2017 dataset.

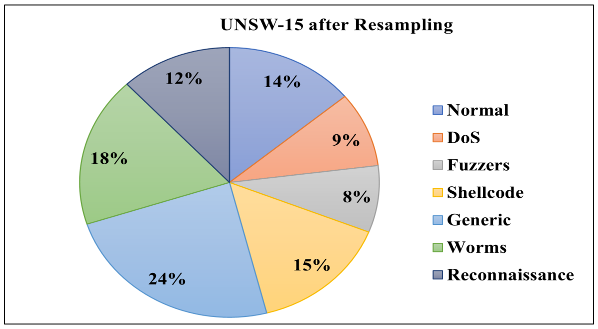


Figure 7.6: Attack distribution of UNSW-15 after resampling.

7.3.3 Data Resampling

In an ML-based approach, imbalanced data is a classification problem. All the classes are not distributed equally in imbalanced data, meaning that the dataset is biased towards one or more classes, with only a few samples for others. As a result, while training a model using imbalanced data, the model was biased toward one or two classes. The majority class is balanced using the under-sampling technique, while the minority class is balanced using the over-sampling technique. To overcome the imbalanced class problems Synthetic Minority Oversampling Technique (SMOTE) is used [165]. The less frequent samples were oversampled using SMOTE, whereas the more frequent samples were under-sampled using Edited Nearest Neighbors (ENN). Imbalanced learn, imported as imblearn from the scikit-learn library, is a widely known machine learning library that deals with imbalanced classes. SMOTE takes the following three steps to oversample the minority class:

1) It calculates the distance between each sample using Euclidean distance and then modifies these samples using the k-nearest neighbor.

2) Then they take n samples and calculate the imbalance ratio from them, as well as the number of samples that need to be made from the samples.

$$n = round(imbalancedratio) - 1 \tag{7.2}$$

$$imbalancedratio = \frac{S_{max}}{S_{min}} \tag{7.3}$$

Where n is the number of samples.

3) Finally, the set of produced samples y is taken from the k-nearest neighbor, and new synthetic samples are constructed from those neighbors.

In order to under-sample, the majority class was chosen. The majority of their k-nearest neighbors sample is removed by ENN. If one sample is owned by a most frequent class also if the classification of the original class is disputed by its three nearest neighbors, it is deleted from the samples; otherwise, it belongs to the minority class. Worms and shellcode traffic have only 130 and 1133 samples in the UNSW-15 dataset, which is relatively low when compared to normal and generic traffic, which includes 56,000 and 40,000 samples, respectively. Fig. 7.6 depicts the attack distribution of the UNSW-15 dataset after data resampling using SMOTEEN

On the other hand, only 1% of samples in the CICIDS-2017 dataset belong to web-attack and bot attack traffic, while benign, DoS, and DDoS attack traffic each include 26%, 24%, and 22% of samples, respectively. Fig. 7.7 depicts the attack distribution of the CICIDS-2017 dataset before and after data resampling using SMOTEENN.

Algorithm 1 illustrates the primary learning procedure of the proposed ENIDS model. In the first stage from lines 1-4, we checked the training set IDS_{train} and if it is imbalanced performed class resampling, and the resampling strategy is only applied to the training data when it is imbalanced. After that, from lines 5–9, we train the three base learners, BL_1, BL_2, and BL_3, with training data samples in IDS_{train}. Base learner training is the process of training base learners to make predictions on new data using training data. The algorithm concatenates the trained weights gained from each base learner into a list called $Concat - BLW$ after training the base learners. This list is then used to train the meta-learner (ML) in IDStrain using the meta-learner training function ($meta - learner_{training}$) on all training data samples. The Concat-BLW list, as well as the training data samples and the weights of the base learners (BL_1, BL_2, and BL_3), are used to train the meta-learner. Line 10 concatenates all the training weight gained from three base learners. The weights of the base learners, on the other hand, are marked as non-trainable, so they are not updated during the training process. Finally, the algorithm makes predictions on the test data samples using the trained base learners and the meta-learner ML. Individual predictions are made using the base learners, and the results are saved in BL_{O1}, BL_{O2}, and BL_{O3}. After that, the meta learner is used to generate an ensemble prediction, which is saved in ML_P. The algorithm's final output is ML_P, which represents the predicted results for the various types of cyberattacks in the test data set IDS_{test}.

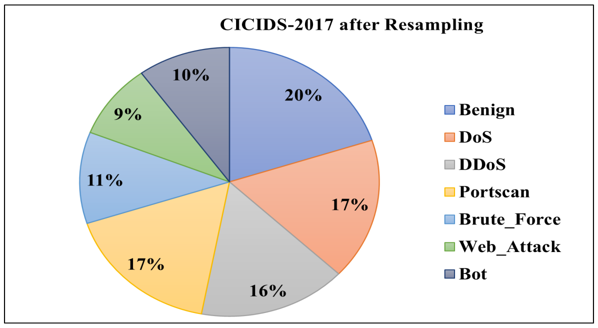


Figure 7.7: Attack distribution of CICIDS-2017 after resampling.

7.3.4 Stacking Model Description

Our overall stacking ensemble architecture is comprised of two layers; in the first layer, we employed three different DL-based architectures for our IDS system, which consists of convolutional neural networks (CNN), gated recurrent units (GRU), and long short-term memory (LSTM). We used a deep neural network (DNN) in the second layer, which takes prediction results from the previous three DL models. Each DL architecture has input, hidden, fully connected, and output layers for multi-classification.

Base Model

Convolution neural network (CNN) also known as ConvNet designed to process grid-like topological data such as images. The first CNN [167] was developed by Yann LeCun, where he proposed LeNet-5 which was able to recognize handwritten characters like postal codes. CNN consists of multiple layers where earlier layers are responsible for extracting features and later layers combine the features and make predictions. CNN model consists of convolution and pooling layers that are used for feature extraction and fully connected layers are responsible for classifying results. Dropout layers are used to prevent the model from becoming overfitted and the sigmoid or SoftMax activation function is to derive class label predictions.

CNN model generally takes an image as input. We reshaped the train and test dataset to (5, 5, 1) dimensions for the UNSW-15 dataset where the model takes 5×5 matrix and ImageDataGenerator from Keras [168] library generating an image type data. For the CICIDS-2017 train and the test is reshaped to (7, 7, 1) dimensions. In our CNN model of the first layer, we used six two-dimensional filters with a kernel size of (2×2) for the UNSW-15 dataset and a kernel size of (3×3) for the CICIDS-2017 dataset. Every layer of CNN deals with a ReLU activation function. After every two convolution layers, we set one two-dimensional max-pooling layer with different pool sizes. The output is then delivered to fully connected layers, where it is used to train representations of higher-order features that may be used to classify the output into distinct class labels.

Hochreiter and Schmidhuber initially proposed Long Short-Term Memory (LSTM)[53] as an advanced variant of RNN in 1997, to address the problem of exploding and disappearing gradients. The only objective of LSTM is to avoid long-term reliance issues, and it is capable of automatically remembering information for lengthy periods of time. The forget gate of the LSTM cell decides whether the cells need to keep information gained from the previous cell or forget it. The input gate, also known as the store gate, is in control of storing and quantifying new information. A sigmoid function is utilized in the output gate to determine which elements of the cell state will be the output. The LSTM model of the first layer of our proposed stacking ensemble model consists of two LSTM layers, one dense layer, followed by one dropout layer. Like the CNN model, every layer in the LSTM model interacts with a corrected ReLU activation function. The LSTM model's loss function is categorical cross-entropy, and an Adam optimizer with a learning rate of 0.001 was employed for optimization. For LSTM, the input shape is (batch, timestamps, and features), and here batch size means the number of samples we send to the model at a time. For UNSW-15, the timesteps are 5, features are 25 and the input shape is (5, 25); for CICIDS-2017, it is (5, 49).

The Gated Recurrent Unit (GRU) is another short-term memory solution, and its essential principle is virtually identical to that of the LSTM. GRU used a hidden state to transport information instead of a cell state, and its design consists of two hidden states: the reset gate and the update gate. Tangen's hyperbolic activation function (Tanh), as opposed to CNN and LSTM models, is employed with the GRU model. GRU also takes an input of a 3D tensor, with shape (batch size, timesteps, features), here batch size means the number of samples we send to the model at a time. In UNSW-15, we take 5 timesteps, and 25 features and the input shape becomes (5, 25); for CICIDS-2017 we take 5 timesteps and 49 features and the input shape becomes (5, 49). The details of the three base model learning using training data and their prediction approach using test data in the proposed framework are shown in Algorithm 1.

The primary reasons for selecting CNN, LSTM, and GRU as base learners are given below.

1) All three DL models can identify important features during training, which increases learning efficiency in terms of time and resources.

2) These three DL models are robust and able to handle large amounts of data samples with multiple classes, which is suitable for multiclass classifications.

3) All three DL models can learn from a long sequence of data samples, have little dependence on pre-processing, and require a low computational cost.

4) All three DL models support Graphics Processing Units (GPU), which employ parallel computing and accelerate the training time.

Ensemble Model

In our work, we are using stacking-based ensemble methods because it uses a second label, also known as a ”meta-learner,” through which it can define which classifiers are appropriate and which are not. While bagging and boosting employ homogeneous weak learners, the ensemble uses heterogeneous weak learners to train them in parallel and aggregate them. The idea of stacking was initially presented by David H. Wolpert in 1992 when he divided the dataset into J equal pieces and utilized Jth fold cross-validation during training while the remaining samples were used for testing purposes [169]. He later trained numerous models using the training test pairs as input for the meta-model. Parameter estimates, model selection, and hyperparameter tuning are all part of the stacking framework. Algorithm 1 shows the learning of the meta-learner in the proposed model, where it first concatenates the training weight of the base learners and passes them as input to the meta-learner. Fig. 7.8 shows the architecture of the proposed stacking ensemble technique.

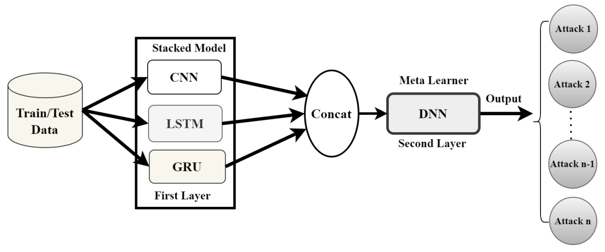


Figure 7.8: The stacking ensemble architecture.

In our proposed hybrid deep learning model, called the stacking ensemble model, the meta-learner in the second layer is composed of a deep neural network (DNN). All the results obtained from the first three DL models in the first layer are concatenated and delivered to the DNN model. The non-linear nature of deep learning neural networks can pose difficulties in learning specific features from large datasets, leading to a reduced predictive capacity of a single deep learning model. Stacking ensembles is a technique that addresses this issue by employing a meta-learner, which takes the output of first-layer feature sub-models, combines them, and trains the meta-learner to make improved predictions. This method enhances the prediction ability and robustness of the model by directly passing the output weights of sub-models to the meta-learner. Previous experiments have encountered challenges in detecting multiple types of cyberattacks using large, imbalanced datasets. In this study, we aimed to enhance predictive performance and reduce variance by utilizing the stacking ensemble technique. Specifically, we applied this technique to two large, imbalanced datasets, where the meta-learner of the stacking ensemble model receives and combines predictions from existing models generated by the first-layer base learners. By doing so, we aimed to improve the model's ability to accurately predict and classify multiple types of cyberattacks.

The proposed stacking ensemble configuration is shown in Fig. 7.9. The Ensemble model considers the following steps during the model training:

1) The weights of the three first-layer base learners—CNN, LSTM, and GRU—are loaded as a list; each loaded model is used as a separate input head to the ensemble model, and because the base learners are marked as non-trainable, their weights won't change throughout training.

2) The output weight of the three distinct DL models of the base learners merged using a single concatenation merge that creates a single vector of 21 elements from 7 different class labels predicted by each of the 3 models.

3) The meta-learner will then analyze the input employing two hidden layers and make predictions using an output layer. During the training of the ensemble model, only the weights of the new hidden and output layer will be changed, as the three base learners are marked as non-trainable. Once the model has been fitted, the stacking model is used to predict the unseen data.

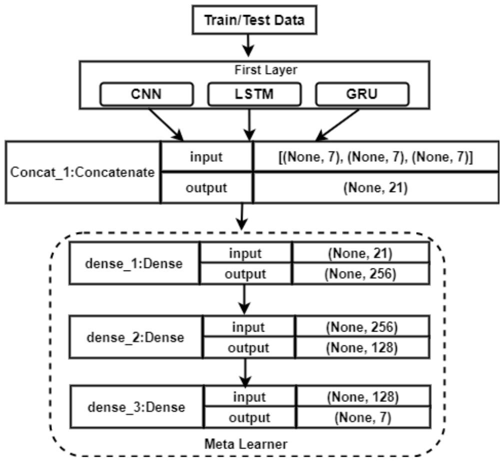


Figure 7.9: Configuration of the proposed stacking model.

7.3.5 Complexity Analysis of Proposed Model

Runtime complexity is an important metric for assessing algorithm performance and efficiency. Developing an intrusion detection system with a low runtime complexity will aid in performance and reduce the computational resources required. The proposed ENIDS can be trained on a high-speed server; however, the proposed model's deployment may differ depending on the system where it is deployed.

The class resampling operation in line 3 will take $O(N)$, where N is the number of samples in the training set. Lines 5–9 will execute N times, and each base learner training operation in lines 6–8 will take $O(T)$, where T is the time required to train the base learner on one sample. As a result, the time complexity of lines 5-9 is $O(N * T)$. Because it is a simple concatenation operation, the Concat-BLW operation in line 10 will take $O(1)$ time. Lines 11–13 will execute N times, and line 12 will take $O(M)$, where M is the time required to train the meta-learner on one sample. As a result, the time complexity of lines 11-13 is $O(N * M)$. Lines 14–19's for loop will run M times, and each IDS-prediction test operation in lines 15–18 will take $O(P)$, where P is the time it takes to predict one sample. Thus, the time complexity of lines 14-19 is $O(M * P)$.

Therefore, the total time complexity of the given algorithm will be $O(NT + NM + M * P)$. The actual time taken by the algorithm will be determined by the values of N, T, M, and P.

7.4 Analysis of Experimental Results

The proposed NIDS framework is trained with two publicly available datasets namely UNSW-15 and CICIDS-2017. Each of the datasets is pre-processed, and an optimal number of features is selected from both datasets. Each dataset contains the seven most recent types of cyberattacks. To train the proposed AI-based NIDS, we divided our datasets into three parts: the training set, testing set, and validation set. We used the stratified training-test split of the Scikit-Learn library. The stratified train-test split divided the data samples into similar proportions. A total of 70% of data samples are used for training the models, while 10% of the data samples are used for validation and the remaining 20% of the data samples are used for testing purposes.

7.4.1 Software and Hardware Requirements

In this experiment, a Windows 10 PC with an AMD Ryzen 9 5900HX processor is used. It has 16 GB of RAM, 512 GB of solid-state drive (SSD), and 4GB of NVIDIA GeForce RTX 3050 graphics processing unit (GPU). The NIDS model is implemented using different libraries of Python 3.7 including Keras [168], TensorFlow 2.8.0 [170], and Scikit-learn [128]. The Pandas package for data analysis, NumPy for numerical analysis, and Matplotlib and Seaborn to generate graphs for the experiment results.

7.4.2 Evaluation Metric

In order to evaluate the performance of the proposed NIDS we used a confusion matrix, which is a graphical depiction of the performance of a classification problem and gives the output in matrix format. Using a confusion matrix, a classification model's outcomes can be characterized as true positive (TP), true negative (TN), false positive (FP), and false negative (FN). A classification model's outcomes can be characterized as follows using a confusion matrix:

1) True Positive: These values are predicted and labelled as positive. For example, if traffic is forecasted as a DoS assault in our IDS system and it is indeed a DoS attack, we can conclude that the IDS made an accurate prediction.

2) True Negative: These are the values labeled as negative and also predicted as negative and correct. For instance, in the NIDS model, if traffic is predicted not as a DoS attack and it was not a DoS attack then the model correctly predicted the traffic.

3) False Positive: This occurs when a model predicts a positive value for a class, but the actual value is negative. For example, if our IDS model predicted a DoS attack but it turned out to be typical traffic, we'd have a false-positive result.

4) False Negative: This is used to describe results that were anticipated to be negative but ended up being positive. For example, if a NIDS model projects a packet as regular traffic, however, the traffic was actually a DoS attack. Higher false-negative numbers indicate a defective model.

Fig. 7.10 shows a perceptible representation of a binary classification confusion matrix. The confusion matrix for multi-class classification will be the same size as the number of classes the model must predict.

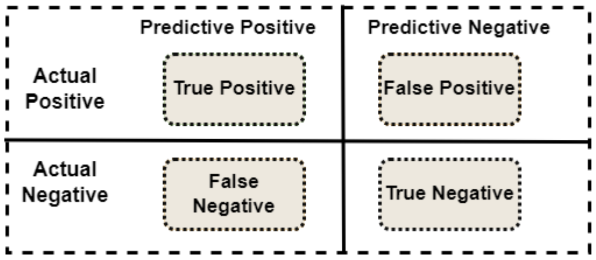


Figure 7.10: Confusion matrix.

In our experiments, we used two imbalanced datasets, and it is difficult to obtain accurate results from the imbalanced data. The classification accuracy does not always correspond to the model's real performance, especially when the misclassification rate for minor classes is high. It also ignores the problem of class imbalance in a dataset, which occurs when the number of positive and negative levels is vastly different. As a result, various performance evaluation indicators must be considered to achieve actual model performance. In order to get a better overview of the performance of our proposed NIDS, we considered four important performance metrics for multiclass classification: accuracy, recall, precision, and f1-score. The term accuracy is a ratio of the total number of actual predictions made by the model to the total number of predictions. Precision is determined by dividing the true positives predicted to belong to a specific class by the total number of positive outcomes predicted by the classifier, and recall is a metric that measures how well our model detects true positives. Finally, the f1-score is used to determine how well our model detects actual positives. It's calculated by dividing the number of real positive results by the total number of positive values. The training and testing times of the models are also used to measure the efficacy of the proposed NIDS.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{7.4}$$

$$Precision = \frac{TP}{TP + FP} \tag{7.5}$$

$$Recall = \frac{TP}{TP + FN} \tag{7.6}$$

$$F1Score = 2 \times \frac{Precision * Recall}{Precision + Recall} \tag{7.7}$$

7.4.3 Analysis of Experimental Results

Three different DL models, including CNN, LSTM, and GRU, are used in the first layer of the proposed stacking ensemble-based NIDS, and their training weights are used as the input for the meta learner in the second layer, which is a DNN. The ensemble model, as well as the base learners of the first layer, are evaluated with the test set to make a comparison with the proposed ensemble NIDS.

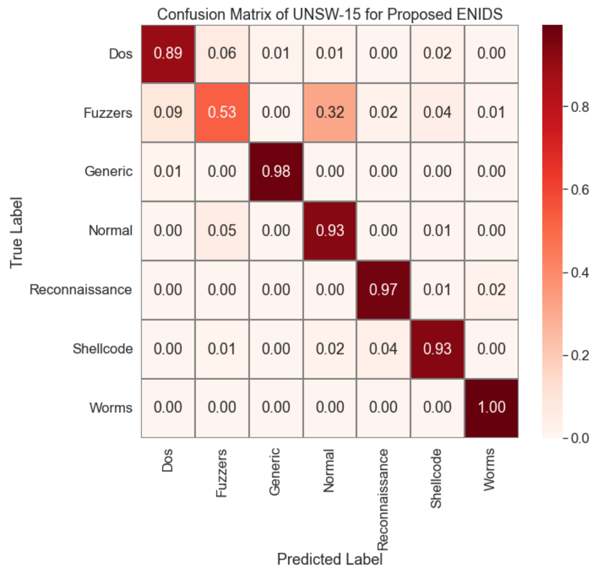
Table 7.1 presents a performance comparison of the proposed Ensemble-based Network Intrusion Detection System (ENIDS) with single deep learning (DL) models and some previous state-of-the-art works on the UNSW-15 dataset proposed in [153], [157], and [159]. ENIDS achieved the best overall performance of the models in the table, with an accuracy of 90.64% and an F1 score of 90.50%. This is superior to the performance of the individual DL models, including Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU), which attained accuracy scores ranging from 88.4% to 89.1% and F1 scores ranging from 88.2% to 89.1%. While requiring more training time, the proposed ensemble approach outperforms the CNN and GRU models (7.3 and 7.5 seconds, respectively) in terms of attack detection rates with unseen data, taking only 6.9 seconds. ENIDS outperformed the other models due to its ability to integrate multiple DL models into an ensemble framework, allowing it to capture different aspects of network traffic and make more informed decisions by combining the predictions of the base models. Additionally, the ENIDS model used a meta-learner in the second layer to learn how to combine the weights of the base models to make a final prediction, which improved the model's accuracy.

Furthermore, ENIDS addressed the issue of imbalance classification by using data augmentation techniques or class weighting, which helped to balance the class distribution and led to improved classification performance. In contrast, the proposed EnFS model in [153] achieved a high precision score of 96.7%, which was higher than the proposed ENIDS model's precision score of 90.94%. However, EnFS is limited to detecting only one type of attack and does not address the issue of imbalance classification. Overall, the superior performance of the proposed ENIDS model is due to its ability to integrate multiple DL models into an ensemble framework, to use a meta-learner to combine the predictions of the base models and to address the issue of imbalance classification. These techniques allowed ENIDS to achieve high accuracy and F1 score while still maintaining a relatively high precision score, outperforming the individual DL models and the state-of-the-art EnFS model. Fig. 7.11 presents a confusion matrix that provides insights into the detection rates of different types of attacks in the UNSW-15 dataset using the proposed ensemble approach. The results show that the ensemble approach achieves

Table 7.1: Model Performance Comparison for the UNSW-15 Dataset.

Model	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)	Train Time (s)	Test Time (s)
CNN	89.1	89.7	89.2	89.1	322	7.3
LSTM	89.0	89.5	89.0	89.0	381	7.8
GRU	88.4	89.4	88.4	88.2	364	7.5
EnFS_EnNN[153]	79.7	96.7	77.3	85.9	-	-
ANN[157]	84	-	-	-	-	-
RNN-LSTM[159]	85.38	-	-	-	-	-
Proposed ENIDS	90.6	90.9	90.6	90.5	460	6.9

high detection rates for several types of attacks, including worms, reconnaissance, and generic attacks, with detection rates exceeding 95%. The proposed model also achieves a detection rate of more than 90% for shellcode and normal traffic. However, approximately 6% of these are misclassified as reconnaissance and fuzzers attacks. The model can detect 89% of DoS attacks but only achieves a 53% detection rate for fuzzer attacks, with 32% of them misclassified as normal traffic. This is because fuzzers attacks are designed to closely resemble normal traffic, making them difficult to distinguish from legitimate traffic. Moreover, hackers may use fuzzers to simulate normal traffic, further complicating the detection process. Overall, the proposed ensemble approach performs well in detecting most attacks in the UNSW-15 dataset but struggles with identifying fuzzers attacks. Further research is needed to improve the model's ability to detect these attacks, which could include incorporating additional features or developing more advanced detection techniques.



True \ Predicted	Dos	Fuzzers	Generic	Normal	Reconnaissance	Shellcode	Worms
Dos	0.89	0.06	0.01	0.01	0.00	0.02	0.00
Fuzzers	0.09	0.53	0.00	0.32	0.02	0.04	0.01
Generic	0.01	0.00	0.98	0.00	0.00	0.00	0.00
Normal	0.00	0.05	0.00	0.93	0.00	0.01	0.00
Reconnaissance	0.00	0.00	0.00	0.00	0.97	0.01	0.02
Shellcode	0.00	0.01	0.00	0.02	0.04	0.93	0.00
Worms	0.00	0.00	0.00	0.00	0.00	0.00	1.00

Figure 7.11: Confusion matrix of the proposed ENIDS model on the UNSW-15 dataset.

Table 7.2 presents the performance comparison of different models on the CICIDS-2017 dataset in terms of accuracy, precision, recall, and F1 score. The proposed ENIDS model achieves outstanding results, outperforming all other models in the table. Specifically, the proposed ENIDS model achieves an accuracy of 99.6%, precision of 99.5%, recall of 99.7%, and F1 score of 99.6%. In comparison, the EnFS model [153] achieves an accuracy of 98.9%, which is comparable to the state-of-the-art models in previous works. However, the EnFS model [153] does not address the issue of attack variations or imbalanced datasets, which can be critical for practical deployment. Moreover, the proposed ENIDS model outperforms other traditional models, such as RBF-BLS [161] and Bidirectional RNN-GRU [163], achieving an accuracy of 96.63% and 98.99%, respectively. The proposed ENIDS model demonstrates a high recall rate of 99.7%, indicating a low false-negative rate and high detection rate. Despite the longer training time of the proposed ensemble approach (1153 seconds) compared to the training time of the base learners, the attack detection rate with unseen data is achieved in only 8.6 seconds. This is faster than the LSTM and GRU models, which require 12.7 seconds and 12.3 seconds, respectively, for detection. The confusion matrix of CICIDS-2017 illustrates that the proposed ensemble NIDS can successfully detect all types of attacks, including Bot, Brute-force, DoS, DDoS, Port scan, and Web-attack, while accurately detecting 98% of benign network traffic. Moreover, Fig. 7.12 depicts the confusion matrix for the CICIDS-2107 dataset.

Table 7.2: Model Performance Comparison for the CICIDS-2017 Dataset.

Model	Acc.(%)	Prec.(%)	Recall(%)	F1(%)	Train Time (s)	Test Time (s)
CNN	97.5	97.7	97.6	97.3	499	8.6
LSTM	97.2	97.4	97.3	97.1	677	12.7
GRU	96.0	96.4	96.1	96.0	637	12.3
EnFS_EnNN[153]	98.9	99.9	98.8	99.3	-	-
RBF-BLS[161]	96.63	-	-	96.87	15.60	-
Bi RNN-GRU[163]	98.99	-	-	-	-	-
Proposed ENIDS	99.6	99.5	99.7	99.6	1153	8.6

The proposed ensemble approach requires additional training time compared to the base models, as it involves training both the base models in the first layer and the meta-learner in the second layer. This is because, during training, the ensemble model takes both the training data and the training weights of the base learners from the first layer. However, this extra training time is offset by the improved performance of the ensemble model, as it combines the predictions of multiple models to improve generalization performance. During testing, the proposed ensemble approach requires less time than the base learners and other existing works, as it makes predictions using a combination of multiple models. This reduces the likelihood of overfitting to the training data, as the ensemble model is more robust to variations in the data. Additionally, the ensemble model can handle different types of network traffic more effectively than the individual base models, as it is able to capture different aspects of the data and combine them to make more informed decisions. Overall, although the proposed ensemble approach requires more training time than the base models, it ultimately leads to improved performance and requires less testing time, making it a promising approach for network intrusion detection.

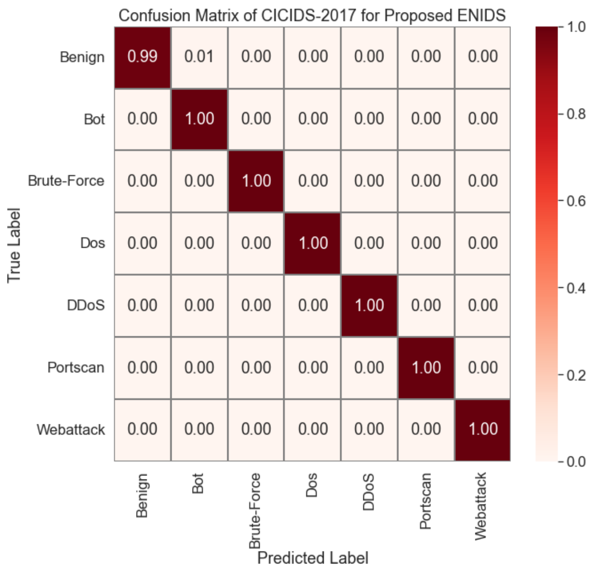


Figure 7.12: Confusion matrix of the proposed ENIDS model on the CICIDS-2017 dataset.

In our experiments, we found that LSTM outperformed GRU in terms of detection accuracy on both datasets. This is because LSTM is better equipped to handle larger datasets, and the datasets used in this study were substantially large. LSTM's complex architecture with three gates, including an input gate, an output gate, a forget gate, and an additional memory cell, allows it to capture more long-term dependencies in the input data. On the other hand, GRU has a simpler architecture with only two gates, an update gate and a reset gate, which makes it less effective in capturing long-term dependencies. However, despite LSTM's superior detection accuracy, GRU has a faster training time and is more computationally efficient than LSTM. This is because the GRU's simpler architecture requires fewer calculations and parameters, resulting in faster training times. Therefore, if training time and computational efficiency are more significant factors than detection accuracy, GRU may be a better choice than LSTM.

In summary, ENIDS integrates multiple deep-learning models into an ensemble framework to capture different aspects of network traffic and make informed decisions. A meta-learner combines the predictions of the base models to improve accuracy. ENIDS addresses imbalanced classification with data augmentation techniques or class weighting, outperforming individual models and state-of-the-art models on UNSW-15 and CICIDS-2017 datasets. However, it struggles with identifying fuzzers attacks and requires further research. ENIDS achieves an attack detection rate with unseen data in only 6.9 seconds and 8.6 seconds respectively for UNSW-15 and CICIDS-2017 datasets respectively, despite longer training time.

7.5 Conclusion

Driven by the advancement in ultra-high speed network technologies such as 5G, the number of connected devices to the Internet is growing rapidly. Hackers/intruders continue to use new techniques to launch large-scale cyberattacks, making network traffic more vulnerable. This research suggested an AI-based ensemble approach for identifying various types of cyberattacks.

In contrast to earlier research, our proposed novel ensemble NIDS model not only determines if the network traffic is benign or normal but also the type of assault in the flow. Compared to the existing models, our suggested stacking ensemble model provides a more accurate forecast. Overall, stacking ensemble DNN, a hybrid deep learning approach, and a well-defined data pre-processing technique are presented in this study. Two well-known datasets, UNSW-15 and CICIDS-2017, were selected for this study because they closely mimic real network traffic flow, which improves the efficiency, robustness, and practicality of our approach. The best possible traffic flow features are selected using random feature elimination with cross-validation (RFECV), and the SMOTEENN technique is employed to resample the imbalanced classes. With the help of the suggested technique, we trained various robust and large deep learning models in the first layer of the proposed model, concatenated their training weights, and then passed them to the second layer of the stacking ensemble technique, which consists of a DNN model. By retraining the model, we can increase the classification accuracy for identifying various cyberattacks. Through the experiments, the proposed NIDS model achieved higher accuracy of 90.4% and 99.6% in the UNSW-15 and CICIDS-2017 datasets, respectively. Moreover, the experiments show that the proposed model achieves a higher F1-score of 90.0% and 99.6% in both datasets, which outperformed other state-of-the-art approaches.

The main limitation of the proposed NIDS is that it can't detect new attacks before they are detected since it is constrained by the labels used in the training phase. Additionally, the proposed NIDS cannot distinguish it from legitimate traffic or other intrusions if new traffic comes in. To overcome this, we will employ the transfer learning technique to assess how well they function on unknown network traffic. Moreover, we will deploy the proposed NIDS into live network traffic so that the model can analyze the traffic flow of networks and distinguish between normal and malicious traffic.

Chapter 8

A Dynamic Weighted Voting Approach to Improve Android Malware Detection in Mobile Cyber-Physical Systems

Abstract: The Mobile Cyber-Physical System (MCPS) integrates the mobility of various smart devices to exchange information between physical and cyber systems. Among those intelligent devices, Android-powered smartphone usage increased significantly due to its low cost and simplicity. But this global prominence of Android operating system also makes it more appealing for cyber-attacks to obtain users' physical private information. Since attackers mostly prefer malicious applications to spread different viruses and take control of the user's device, it is crucial to classify and categorize the malignant application for secure MCPS. Modern machine learning algorithms have shown promising performance in identifying dangerous applications compared to traditional signature-based methods. But most existing works identify only the malicious application where category identification is essential for proper precaution. Also, the static analysis is insufficient for polymorphic malware, which includes regenerating code and changing its properties frequently to evade the detection process. In this study, we compare several state-of-the-art deep learning methods for malapps classification and categorization. Moreover, we propose an ensemble Dynamic Weighted Voting model to identify and label a wide variety of malicious applications using the CCCS-CIC-AndMal-2020 dataset, which contains an extensive collection of Android malware samples. Our proposed ensemble model outperforms the baseline ensemble method Majority Voting by 1% and the classical LSTM model by 2%.

8.1 Introduction

Mobile Cyber-Physical Systems (MCPS) are an essential subgroup of Cyber-Physical Systems (CPS), which integrate the mobility power of the smart devices with the typical cyber components for a better physical system. Nowadays, various mobile devices, e.g., smartphones, smartwatches, tablet computers, netbooks, etc., are available everywhere and equipped with multiple sensors. And because of their unique characteristic, they are involved in many application domains such as vehicular networking systems, healthcare systems, mobile education, etc. [171]. Among them, smartphones play a vital role in exchanging information between physical and cyber systems using various in-built sensors such as MIC, camera, and GPS. Also, the recent advancement in processing and storage chips for intelligent devices enables them to operate different types of applications for everyday usage [172]. Therefore, many operating systems (OS) such as iOS, windows, and blackberry has been developed for smartphones. Among them, Android is one of the most commonly used OS due to its open-source nature, low cost, and simplicity [173].

Due to the global prominence of android OS and the massive number of users, it became a more appealing target for attackers to obtain users' physical private information. The adversary follows different ways to attack devices. For example, they can operate different malware, such as Trojan horses, in the end devices and take control of specific sensors for stealing sensitive information [173]. Attackers offer various malicious applications to install in the smart devices to perform these adversary operations. According to Norton's security blog, there was a 54% increase in mobile malware variants between 2016 and 2017 [174]. These malicious applications can take control of the device on which it is being installed and the other devices connected to the same network. As a result, the smartphone is a vital entry point for cyber-attacks in MCPS. A successful cybercrime could have disastrous, severe, or even lethal consequences on CPS and MCPS [175]. Therefore, defending against attacks through a malicious mobile application is crucial.

Using only tried-and-true techniques are insufficient since sophisticated malware constantly changes and becomes difficult to recognize. The accuracy of traditional signature-based methods is compromised, mainly when malicious programs use polymorphism or code obfuscation [176]. Additionally, merely identifying as a malignant entity is no longer adequate. The category needs to be determined in order to begin employing the appropriate mitigation methods. Since machine learning approaches do not rely on specific rules and are, therefore, more automated and resilient, they have been actively explored for malware detection over the past ten years [177].

Although several android malware detection techniques have been proposed, the existing solutions contain several similar flaws. First, it is no longer sufficient to find malicious programs. It is essential to determine the kind of android malware to understand the threat to which we are correctly exposed. Second, most of them utilize static analysis, which uses statically extracted features to identify malicious applications. However, these static approaches are insufficient to detect complicated malware programs that employ evasion techniques like polymorphism which includes regenerating code to evade the signature-based detection process. As a result, several static-based malware detection algorithms' detection rates declined when assessing recent malware incidents. In this study, we obtain dynamic analysis to detect a wide variety of malicious applications and their corresponding malware categories. We propose an ensemble technique called Dynamic Weighted Voting that outperforms Majority Voting and other baseline deep learning models.

This chapter is organized as follows. Section 8.2 describes the literature review of current malware detection techniques. Section 8.3 presents our proposed methodology. Section 8.4 summarizes the different deep learning methods' performance and draws a comparative picture between baseline ensemble method and our proposed weighted voting model. Finally, section 8.5 concludes our paper and sheds light on future research directions.

8.2 Literature Review

Utilizing the description of the Android application, they were categorized into several types and these data have been used for identifying malicious applications [178]. Their proposed TFDroiduses method used SVM (Support Vector Machine) to classify the malware based on the application description data. The model was trained using benign application data while a small cross-validation dataset used to measure the performance. The proposed classifier was 93.65% accurate in recognizing malicious applications. Suleiman et al.[179] used static characteristics such as permissions, intents, API calls, date of appearance which were derived from date-labeled benign and malware datasets to examine the effectiveness of several machine learning classifiers: Naive Bayes (NB), SVM, Random Forest (RF) etc. In order to define an application, Zhang et al.[180] calculated the consistency of association rules between abstracted API calls, and then utilized multiple machine learning methods: K-Nearest Neighbour (KNN), RF, SVM for detection. The model performed better than MaMaDroid[181], a model created utilizing the same traits and machine learning techniques.

By applying dynamic analysis to extract API calls from apps, Tan, Li, Wang, and Xu[182] enhanced the use of the dynamic feature approach. Their model was optimized for model accuracy and computation burden using the model portioning and early exit strategies. Even though they conducted excellent research, they did it using sample devices rather of actual devices. Another detection model called MaxNet has been proposed in [183] using API calls and system calls that were taken from Android apps. In order to increase the temporal complexity of their model, they combined the recurrent neural network approach with the LSTM. They utilized a dataset of 36000 samples, and their model had an accuracy rate of 96.2%. In [184], they proposed a dynamic model by combining two LSTM models in which system call sequences data were used. Two individual LSTM model in their proposed framework were trained using two different dataset containing either benign and malicious samples. On the basis of the results of trained models, similarity scores were computed to categorize a new malware/benign sample. Dhanya and Kumar[185] proposed a hybrid analysis approach that uses 77 hybrid best features set where application permissions acting as static features and network, file system, cryptographic activities, and information leakage acting as dynamic features. Machine learning methods including NB, J48, and RF were used to characterize the malapps. In order to define the behaviours of the apps, Wang et al.[186] retrieved 11 different kinds of static characteristics from each app, mostly from API calls, permissions, intents, and hardware information. To identify safe applications and identify malware, it used an ensemble of many classifiers, including SVM, KNN, NB, CART, and RF. Another hybrid detection model was proposed by S. Morales-Ortega et al.[187] where they combine feature extraction, feature selection, and ensemble approaches for classifying android malware. The chi-square, relief, and information gain feature selection strategies are utilized to extract key features and an ensemble technique has been used for detection task.

The process of evaluating malware samples without actually running or executing them is known as static malware analysis. In contrast, dynamic malware analysis examines the code while it is being executed inside a controlled setting. The malware is operated in a secured, separated, simulated environment, and its actions are monitored. Though many studies conducted static analysis and dynamic analysis both, larger portion conducted either static analysis or dynamic analysis. However, dynamic analysis which is more resilient to code obfuscation

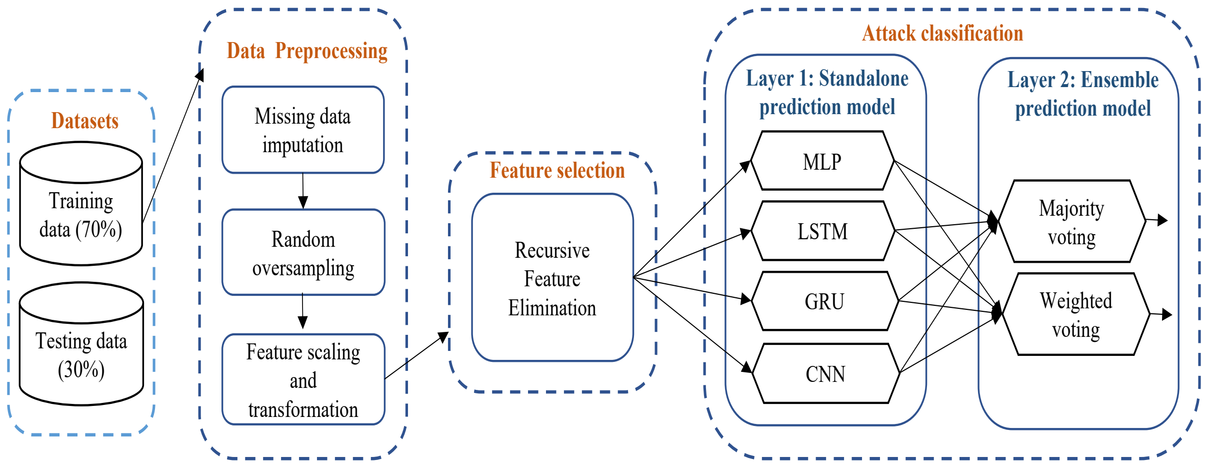


Figure 8.1: High-level framework of proposed methodology of Android malware detection.

and polymorphism evasion techniques, and categorization of the genre of malware were suggested by the recent studies. Modern research also recommended the categorization of detected malware to better prepare against the vulnerability posed by the malicious applications. In this study, we proposed a hybrid deep learning approach combining feature selection and different ensemble strategies such as Majority Voting and our proposed Weighted Majority Voting for efficient classification of different malware categories.

8.3 Proposed Methodology

8.3.1 Dataset and Data Preprocessing

A publicly accessible dataset named as CCCS-CIC-AndMal-2020 was created by the Canadian Institute for Cybersecurity and the Canadian Centre for Cyber Security in 2020. There are 400K android applications in this enormous collection, of which 200K are typically harmless apps and the other 200K are malware apps. The 141 features in this dataset divided into Memory (23 features), API (105 features), battery (2 features), network (4 features), logcat (6 features), and process for dynamic analysis (1 feature). This dataset contains 53439 samples that belong to 14 distinct categories. The authors of the dataset suggest excluding some categories owing to the lack of comprehensive data in these labels [188]. We considered 10 prominent malware types for classification in this study, such as Adware, Backdoor, Scareware, Ransomeware, No-Category, Zero-Day, Trojan, Trojan-SMS, Trojan-Spy, Trojan-Banker, Potentially Unwanted Apps (PUA), and FileInfector.

We used SimpleImputer from the Scikit-Learn package with the ”mean” technique to ensure that the dataframe was missing no values. First, we used the NumPy np.inf value to replace any infinite values, whether they were positive or negative. Then, we substituted a NumPy np.nan value for each empty value. After that, we once again substituted all NumPy np.nan and np.inf values with the mean value of a particular feature using SimpleImputer from the Scikit-Learn module.

According to the attack distribution, CCCS-CIC-AndMal-2020 is a highly imbalanced

Algorithm 5: Dynamic Weighted Voting

```
1  Input: Trained models, Test-set;
2  Initialization: models ← [CNN, LSTM, GRU, MLP];
3  labels ← [Adware, Backdoor, ..., Trojan_Spy];
4  success_rate[number of models][number of labels] ← ∅;
5  final_prediction[no. of samples in test_set] ← ∅;
6  foreach model $m$ in models do
7      foreach label $l$ in labels do
8          success_rate[$m$][$l$] ← True Positive[$m$][$l$] - Sum(False Positive[$m$][$l$]);
9      end
10 end
11 foreach sample $s$ in test_set do
12     Initialization: fittest_model ← ∅;
13     prediction[number of models] ← ∅;
14     rating_point[number of models] ← ∅;
15     max_rating ← 0;
16     foreach model $m$ in models do
17         prediction[$m$] ← predicted label by $m$;
18         rating_point[$m$] ← success_rate[$m$][prediction[$m$]] - MAX(success_rate[models
             \$m$][prediction[$m$]]);
19     end
20     foreach model $m$ in models do
21         if rating_point[$m$] > max_rating then
22             fittest_model ← $m$;
23             max_rating ← rating_point[$m$];
24         end
25     end
26     final_prediction[$s$] ← prediction[fittest_model];
27 end
28 Output: final_prediction;
```

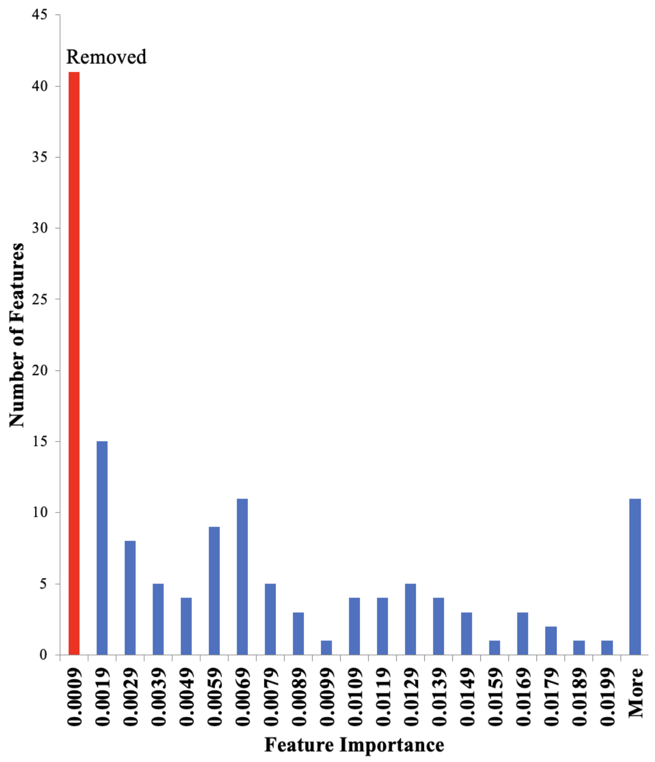


Figure 8.2: Frequency of features over feature importance.

dataset where the ratio of Adware to Trojan-Banker is 45:1. Generally imbalanced datasets are not a good choice for ML/DL algorithms. There are many techniques suggested to deal with imbalanced dataset. One of them is random oversampling of minority labels. We utilized Synthetic Minority Oversampling Technique (SMOTE). It synthesizes new data point from augmentation of existing minority classes. Then to normalise the data in each feature in accordance with the lowest and maximum values provided in the feature, we employed the MinMax Scaler function in the Scikit-Learn package. Finally, we split the whole dataset into 70:30 ratio and provided 70% to the models while training and kept 30% for validation.

8.3.2 Feature Selection

CCCS-CIC-AndMal-2020 dataset contains a large number of features totalling to 141. As choosing optimum number of features by discarding less important features increases DL model accuracy and lessens the complexity, we employed Random Forest regressor to find out the feature importances. The minimum and maximum values of the feature importances are 0 and 0.05998, respectively. The standard deviation of this distribution is 0.00926064. We considered from the minimum value, which is 0, to 1/10 of the standard deviation, which is 0.0009, as range for less important features. We found 41 number of features belong to this range. To remove 41 less important features from dataset, we employed Recursive Feature Elimination with Random Forest Classifier. Fig. 8.2 depicts the number of features over feature importance.

8.3.3 Malicious Application Detection Model

Our malicious application detection model is comprised of two layers as depicted in Fig. 8.1. In first layer, we utilized four DL methods for classification of malware those are found effective in many studies [189, 190, 191, 192]. These are Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU) and Multilayer Perceptron (MLP). We provided training dataset separately to each DL model and collected predictions from each model for evaluation. In the second layer, we propose an ensemble voting technique named as Dynamic Weighted Voting. However, to compare and contrast the performance of our proposed model, we also arrange another ensemble technique named as Majority Voting.

Majority Voting

In Majority Voting each of the four DL models implemented in first layer, has equal right to vote. The final prediction for each data in validation set depends on how the four DL model predict that data. However, there are even number of models, there are chances for 2-2 votes for any case. In that situation, we set rule that, the prediction from model with highest accuracy would be taken into account.

Proposed Dynamic Weighted Voting

We propose an ensemble weighted voting where the weights of the voter models dynamically change depending on their predicted labels while testing, and accuracies observed when predicting those labels while training. For every voter models while training, it computes a Success Rate for each label, that is true positive subtracted by the sum of all false positives for that label. Then while testing for each instance in the test set, we have four predictions from four individual DL models. Our proposed model selects the prediction from the fittest model. The model computes a Rating Point for each voter model to assess, how fit that model is for predicting that particular instance. This Rating Point is computed as its Success Rate in its current predicted label subtracted by maximum value for Success Rates of other voter models in predicting the same label. For an example, if CNN predicts Adware for any instance in test set, its Rating Point for voting would be calculated as its Success Rate in predicting Adware subtracted by the largest Success Rate among other three models in predicting Adware. If LSTM for the same instance predict Trojan, its Rating Point would be Success Rate in Predicting Trojan subtracted by the largest Success Rate among other three models in predicting Trojan. Thus Rating Points for the other two models would be computed. Then the prediction from the model having highest Rating Points would be added to final prediction. This algorithm is given in Algorithm 1.

8.3.4 Evaluation Metrics

We used different evaluation metrics, e.g., Accuracy, Precision, Recall, and F-1 Score estimate the performance of our classification models. There are four measurement parameters in the confusion or error matrix: True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN), which are used to define the evaluation metrics stated above.

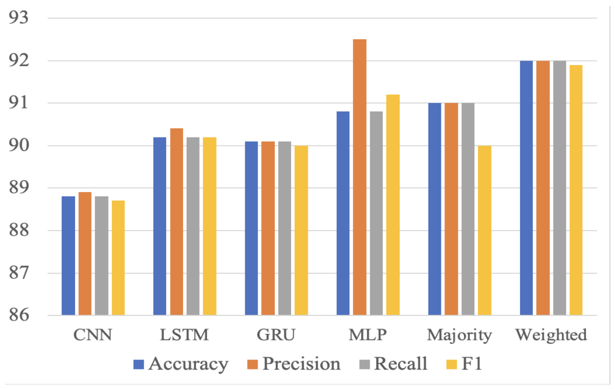


Figure 8.3: Performance comparison among detection models.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{8.1}$$

$$Precision = \frac{TP}{TP + FP} \tag{8.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{8.3}$$

$$F1 = 2 * \frac{Precision + Recall}{Precision * Recall} \tag{8.4}$$

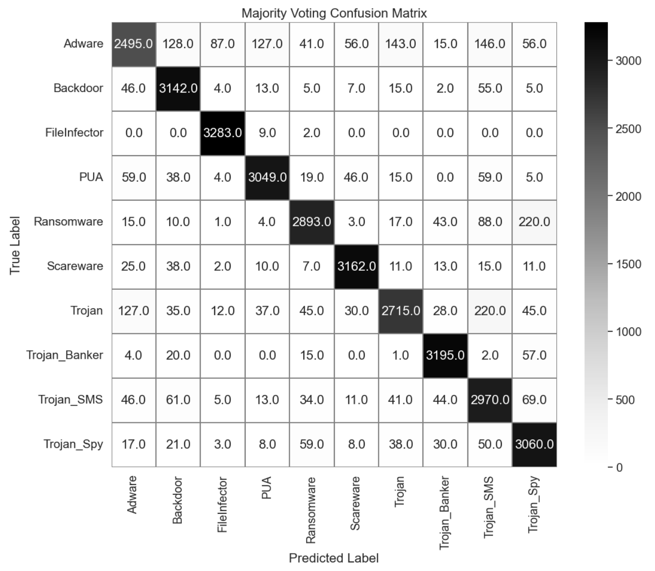
Here, the accuracy can be defined as the percentage of true attack detection over total data samples. Precision measures how often the model can correctly identify the DoS attack from the dataset. Recall is the measurement of how many of the DDoS samples from dataset the model does distinguish correctly. Finally, F-1 score is the harmonic average of precision and recall.

8.3.5 Software and Hardware Requirement

We used Python and deep learning library TensorFlow-Keras[168] to conduct the experiments. Our computer has the configuration of M1-Max Chip, 64GB memory, MacOS V.12.6.

8.4 Analysis of Experimental Results

MLP in the first layer showed highest accuracy among all four individual models. It obtained 90.8% accuracy with 92.5% precision. It mostly failed with Trojan kinds and mistakenly categorised as Adware. LSTM and GRU showed almost similar results, reaching around 90% in all performance metrics, while CNN showed slightly reduced accuracy than other individual models. LSTM and GRU also suffered with Trojan kinds, and Adware. In addition, those two models also mixed up with Ransomeware and Trojan_Spy. However, for CNN it's a different story, it mostly struggled among Trojan kinds. That means it mistakenly categorized one Trojan kind as other kind of Trojan. This is due to the fact that these kinds of Trojan belong to same family of malwares and they have similarity in using resources. MLP performed better



	Adware	Backdoor	FileInfector	PUA	Ransomware	Scareware	Trojan	Trojan_Banker	Trojan_SMS	Trojan_Spy
Adware	2495.0	128.0	87.0	127.0	41.0	56.0	143.0	15.0	146.0	56.0
Backdoor	46.0	3142.0	4.0	13.0	5.0	7.0	15.0	2.0	55.0	5.0
FileInfector	0.0	0.0	3283.0	9.0	2.0	0.0	0.0	0.0	0.0	0.0
PUA	59.0	38.0	4.0	3049.0	19.0	46.0	15.0	0.0	59.0	5.0
Ransomware	15.0	10.0	1.0	4.0	2893.0	3.0	17.0	43.0	88.0	220.0
Scareware	25.0	38.0	2.0	10.0	7.0	3162.0	11.0	13.0	15.0	11.0
Trojan	127.0	35.0	12.0	37.0	45.0	30.0	2715.0	28.0	220.0	45.0
Trojan_Banker	4.0	20.0	0.0	0.0	15.0	0.0	1.0	3195.0	2.0	57.0
Trojan_SMS	46.0	61.0	5.0	13.0	34.0	11.0	41.0	44.0	2970.0	69.0
Trojan_Spy	17.0	21.0	3.0	8.0	59.0	8.0	38.0	30.0	50.0	3060.0

Figure 8.4: Confusion matrix for majority voting model.

than RNNs (Recurrent Neural Network) like LSTM and GRU because RNNs are usually best for time series data, however, we utilized tabular dataset. With CNNs, we assume that pixels close to one another are connected. In a tabular dataset, it may not be the case. Additionally, CNNs have translational equivariance, which a tablular dataset does not require.

Majority Voting improved the individual model performance by slight margin. We observe that different individual models struggled with different labels. When it comes to the opinion of majority, there is higher probability for an increased accuracy. We observe in the confusion matrix that, Majority Voting model improved detection where individual models were struggling. However, it underperformed where individual models showed peak accuracy. Fig. 8.4 depicts the confusion matrix of the Majority Voting. Our proposed Weighted Voting model showed the highest performance among all the models previously discussed, climbing up to 92% in terms of evaluation metrics. It raised the accuracy by 1% than even Majority Voting model, while other individual model lagging 1.2-3.2%. The performance metrics of all the models are depicted in the Fig. 8.3. Though Weighted Voting was beaten by Majority Voting only in Trojan_spy label by only 5 instances, Weighted Voting outperformed in all other categories. This model showed strikingly better performance with Adware, while most individual models and Majority Voting struggled with this kind. Though MLP showed largest true positive for Adware, it also carries heavy false positive for this kind. On the other hand, Weighted Voting came up with less true positive than MLP, (which is still much higher than other models) and reduced number of false positive. Fig. 8.5 shows the confusion matrix of the Weighted Voting. The reason for better performance by our proposed Weighted Voting lies in the idea of setting dynamic weights to the models while voting, according to the category-wise prediction performance by the individual models. And it is worth mentioning that, it does not only considers the true positive rate, but also false positive rate in predicting the labels, while evaluating category-wise performance. As a result, when multiple individual model come up with multiple predictions for any instance, this Weighted Voting can pick the best model for prediction according to that scenario.

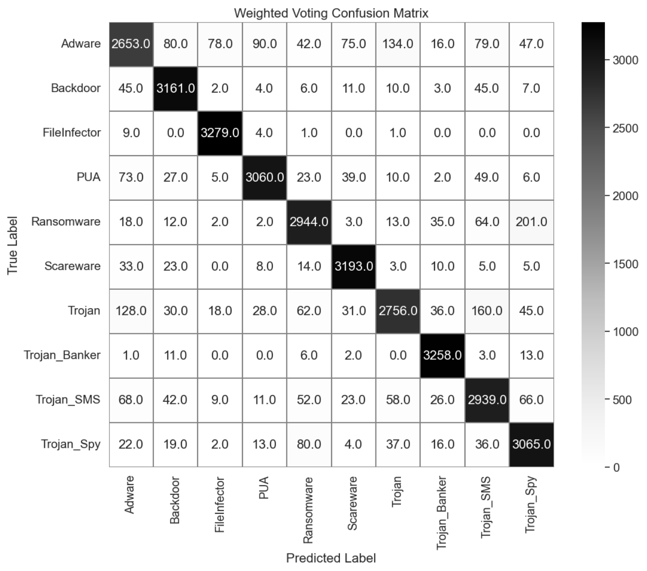


Figure 8.5: Confusion Matrix for dynamic weighted voting model.

Table 8.1: Result Comparison with Existing Works.

Ref.	Model	Dataset	Target	Type	Acc.
[193]	Dynamic	CICMalDroid2020	5	PLDNN	97.84
[191]	Static	CICMalDroid2020	2	CNN	95.9
[190]	Static	Drebin	2	LSTM	92.94
[189]	Dynamic	CCCS-CIC-AndMal-2020	14	DNN	78.82
Proposed	Dynamic	CCCS-CIC-AndMal-2021	10	Ensemble	92

We compare the performance of our proposed model with other similar studies that carried out DL methods. Table 8.1 shows the comparison of performance of other studies. Some of the studies showed higher accuracy than our proposed model, however, with reduced number of categories. Generally, binary classification shows better accuracy than multi-classification. For two different datasets, using LSTM and CNN, [191] and [190] carried out binary classification and showed accuracy of 95.9% and 92.94%, respectively. S. Mahdavifar et al. [193] with 5 labels reached up to 97.84% accuracy deploying Semi-supervised DL method. This model utilized half the number of labels than our proposed model. P. Musikawan et al. [189] conducted experiment having more number of categories than our study, however, it showed 78.82% accuracy.

8.5 Conclusion

The Android-powered smartphones are the most commonly used devices engaged in Mobile Cyber-Physical systems due to their open-source nature, low cost, and ease. As malicious applications are prevalent for carrying out adversary operations on Android devices, it is crucial to identify and classify malapps to ensure security in MCPS consisting of Android-operated devices. The classical signature-based detection approach is insufficient for effective identification of malignant applications that obtain modern techniques like code obfuscation. However, machine learning models showed promising performance in malware detection and classification, while malware classification is crucial for deciding mitigation methods. In this study, we compare several conventional deep learning model performances for malware multi-classification using dynamic analysis. Also, the individual model predictions have been ensembled using the majority voting technique for better accuracy, and it outperforms standalone models by 1%. Finally, we proposed dynamic weighted voting technique to improve further and increase accuracy by 2% compared to the best individual model performance. As part of our future work, we plan to use additional datasets to evaluate the efficacy of the dynamic weighted voting approach.

Chapter 9

Examining Generative Adversarial Network for Smart Home DDoS Traffic Generation

Abstract: Adversarial attacks have become a common place in network security. Neural network-based traffic classifiers have been regarded as effective tools against malicious attacks. However, their performance highly depends on the quality of the training dataset that is often hard to obtain. IoT-centric smart home network is vulnerable to adversarial attacks with a high cost to the individual. In this research, we perform a thorough study on the performance of the original GAN model towards generating flow-based IoT traffic in smart home DDoS attacks. Based on a unique IoT traffic dataset of smart home, we implemented four versions of the original GAN model by using four batch sizes per epoch during training. We captured synthetic IoT traffic at different epochs of the models, which results in a total of 200 IoT traffic datasets. Then, we evaluate the quality of the 200 synthetic datasets using an approach called train-on-synthetic, test-on-real (TSTR). Our study suggests that the original GAN can produce a quality IoT traffic of smart home DDoS attacks at most of the epochs but lacks in providing consistent performance across all the epochs of the GAN model. However, by using TSTR metrics, it is possible to identify the datasets of good quality to be used for real applications.

9.1 Introduction

The Internet of Things (IoT) has brought about a new era to the concept of home automation, now being referred to as a smart home. IoT-based smart home system is a network of home devices enabled with programmability, sensing capability, and connectivity to the Internet, providing access and control both locally and remotely. The smart home system is exposed to many kinds of cyber-attacks (e.g., DoS/DDoS attack, Man in the Middle attack, Sybil attack, etc.) that impede the operability of this system [194]. A machine learning based intrusion detection system (IDS) is a de facto solution now-a-day. However, adversarial attacks that subtly alter original data in an undetectable way could easily be misclassified by responsible classification systems and capable of evading the classifiers that are based on state-of-the-art neural networks [195]. Thus, it becomes a fundamental shortcoming of traditional neural network models against adversarial attacks.

In such context, Generative Adversarial Networks (GAN) [196], capable of generating realistic synthetic datasets, opens up a unique opportunity to explore its applicability in network traffic generation. GAN creates a learning framework that can generate realistic adversary samples after gradually approximating the data distribution of the input dataset. It has shown promising results in generating realistic datasets in the field of image generation [197]. More recently, the prospect of GAN to generate internet traffic data has also been investigated through several researches [198], [199], [200], [201]. However, IoT traffic pertaining to the smart home environment exhibits distinct properties from internet traffic that are dealt with by the above-mentioned body of work [202]. This is due to the diversified household devices communicating in the IoT environment through distinct communication protocols (e.g., CoAP, Zigbee, and MQTT) [203]. Further, different kinds of applications and services in an IoT network exhibit unique traffic patterns. To the best of our knowledge, there is a severe lacking scientific literature on generating IoT traffic specific to smart home environments. In this research, we attempt to fill up the gap by investigating the regular GAN [196] for generating IoT traffic of DDoS attack traces in a smart home network. It is worthwhile to mention that a DDoS attack is the most prominent kind of attack in the context of a smart home IoT environment [204].

We used a labeled smart home flow-based IoT traffic dataset of DDoS attack traces for training the GAN model at different settings (i.e., attributed by the batch size and the number of epochs), which we had previously captured from an emulated smart home network (originally presented in [203]). This dataset constituted the real dataset of IoT traffic in our experiment. Then, we captured 200 synthetic datasets as they were generated by the GAN model in different settings. Next, we evaluated the quality of the synthetic datasets by using the Train-on-Synthetic, Test-on-Real (TSTR) approach [205]. In this approach, using each synthetic dataset, we first trained four machine learning classifiers (e.g., logistic regression (LR), naive Bayes (NB), neural network (NN), and support vector machine (SVM)). Then, we tested the classification accuracy of the trained classifiers against the actual dataset of smart home IoT traffic (used to train the GAN model). This evaluation metric is decisive since it validates the ability of the synthetic data to be used for real applications.

The implication of such research is manifolds, including: (i) the synthetic labeled dataset can be used as an alternative source for privacy-preserving training datasets; (ii) it can augment existing real training datasets for enriching their quality by adding up more anonymity, volumes, varieties, and dimensions. The remainder of this work is organized as follows: Section 9.2 discusses related literature. Section 9.3 describes the IoT traffic dataset, construction of the GAN model, and TSTR method. Experimental results pertaining to the quality of the synthetic IoT traffic datasets is analyzed in Section 9.4. Finally, we conclude the paper with directions to future work in Section 9.5.

9.2 Literature Review

In recent times, Generative Adversarial Network (GAN) is being investigated to generate synthetic network traffic by several research. The authors in [198] propose a GAN model, based on CIDDS-001 dataset, for generating flow-based traffic dataset. But the dataset does not include sequencing information among individual traffic in a flow. In [199], the authors present a framework of GAN to generate adversarial malicious traffic examples capable of evading intrusion detection systems. This framework is tested on the benchmark dataset NSL-KDD. In [200], the authors propose a Wasserstein Generative Adversarial Networks (WGAN)-based model that is integrated with gradient penalty technology for DoS attack generation. The model used the KDD Cup 99 dataset for generating 41 network attributes data. In [201], another GAN version is adapted to generate realistic traffic at the IP packet level (such as ICMP Pings, DNS queries, and HTTP web requests). A network traffic encoding scheme is presented, which converts and maps network traffic data from the IP domain into image-based matrix representations that are typically used in CNN frameworks. The authors in [206] propose a semi-supervised approach that enables training of Deep Convolutional GAN (DCGAN) model with a limited dataset. A feature engineering method is proposed in [207] for selecting features that are the most appropriate to traffic data. This combination of filter and wrapper-based approach is able to reduce the 41 features of the KDD-99 and NSL-KDD dataset into only ten features. The effectiveness of the variants of GAN models is yet to be fully explored by research community. According to the analysis provided in [208], unlike Conditional GAN (CGAN) and Adversarial Autoencoders (AAE), the regular GAN [196] often fails to generate new instances of certain class and shows lack of feature extraction capabilities. On the other hand, CGAN cannot detect anomalies that AAEs are capable of.

The abovementioned research have focused on internet traffic constituting distinct properties than IoT traffic [203] [209] [210]. Since it represents the traffic generated by new kind of participating household devices made by different manufacturers with various communication protocols (CoAP, MQTT, and Zigbee). Moreover, the IoT traffic pattern exhibits new kinds usage and attack scenarios reflecting unique applications, services, and protocol-specific security threats with regards to IoT devices [202]. Therefore, examining IoT traffic generation of smart home by using GAN framework needs to be evaluated on the merit of the tools and methods built upon smart home IoT traffic dataset.

There is severe scarcity of publicly available IoT traffic dataset, especially in smart home environment. A common understanding on the characteristics of IoT traffic is still warranted in the scientific community. In [211], the authors characterize IoT traffics collected from 200 sources and highlight concerning security and privacy areas of smart home IoT. It includes third-party advertising and tracking through supplied devices, less policy-based access control, and weaker application layer encryption. In [209], the authors proposed an open-source traffic generator tool IoT-Flock specific to IoT environment. This tool allows a user to customize his own use case of attack limited to four MQTT-based and CoAP-based attacks, and to collect the generated traffic from the IoT network. Yet another IoT packet level traffic generator tool is proposed in [210]. This tool helps to analyze traffic characteristics of given dataset and enables to model different IoT environment.

The IoT traffic generator tools [209] [210] are helpful for easily emulating an IoT environment and to generate IoT traffic. We adopted similar approach to emulate a smart home network and collected IoT traffic [203]. On the contrary, in this work, we explore the prospect of generating "synthetic" dataset using GAN model that uses "emulated" or "real" dataset as its input. Synthetic dataset is more privacy-preserving and can be used as useful alternatives to real dataset for training IDS classifiers.

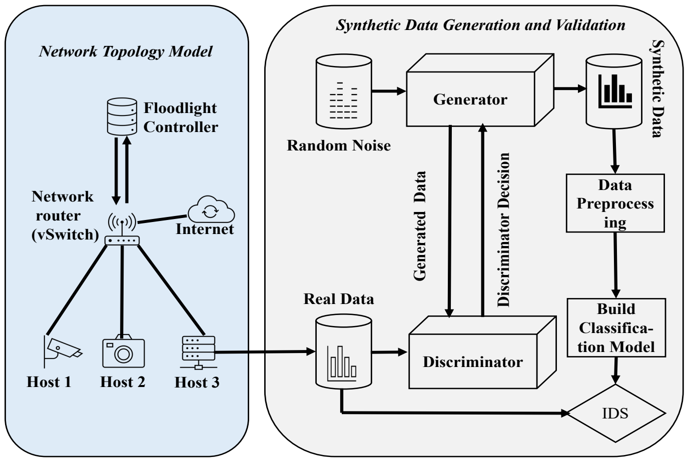


Figure 9.1: High-level overview of proposed framework for adversarial attack generation using GAN.

9.3 Proposed Methodology

In this section, we first briefly discuss the characteristics of IoT dataset and its associated smart home IoT environment in the Section 9.3.1. Then, we describe our proposed GAN-based synthetic data generation model in Section 9.3.2.

9.3.1 Smart Home Emulated Network and IoT Dataset

In this subsection, we discuss our data collection simulation environment as depicted in Figure 9.1. Firstly, we explain our network topology considered for attack data generation in IoT environment in subsection 9.3.1. Then, we briefly discuss our data collection approach and describe attack features in subsection 9.3.1.

Network Topology Model

The dataset used in the GAN models was originally captured within an emulated smart home IoT network environment. The topology model of the IoT network was created using Mininet 2.2.2 with the vSwitch 2.5.4 that is supported by OpenFlow 1.3 and Floodlight controller 1.2. This was a small network consisting of three hosts, i.e.: (i) a HTTP webserver; (ii) a Nest thermostat, and (iii) a Wyze home camera. All the hosts were connected to a network controller through a vSwitch. The vSwitch acts as a WIFI router.

Data Collection and Description

The collected IoT traffic dataset of Smart-home contains 6590 flows of traffic (i.e., each flow contains 100 IoT traffic packets). The dataset includes both benign traffic and malicious traffic of DDoS attack. It is worthwhile to mention that DDoS attacks appears to be extremely difficult to detect by the typical IDS due to their similarity to regular benign traffic at a packet level. A combination of packet-level and flow-level features were extracted to be used by IDS

Table 9.1: Collected Features for Attack Classification

Feature	Description
avg_pkt_size	The average size of a packet per batch
number_uniq_ips	Number of unique incoming IP addresses
number_of_FA	Number of False Fast Re-transmit Avoidance (FA) packets
ack_rst_ratio	This is the ratio of the number of acknowledgment (ACK) packets to the number of reset (RST) packets in a batch.
syn_rst_ratio	This is the ratio of the number of SYN packets to the number of RST packets in a batch.
Avg_number_des_port_50	The average number of destination ports constituting 50% of the batch
Avg_number_des_ip_50	The average number of destinations constituting 50% of the batch

classifiers. The features capture critical information signifying typical DDoS attack, including packet size, proportion of SYN vs. ACK requests, volume of RST (hinting a web server fails due to hijack a TCP connection), abnormal density of a particular destination port or source port in a flow, etc. Total seven features were derived from traffic flow dataset, which constitutes the IoT traffic dataset (the dataset is made available for public use and accessible in [212]). The features are given in Table 9.3.1. The actual formulations of the above feature metrics are provided in [203]. The packets in a batch are considered malicious if more than half of the packets represent attack traffic. Otherwise, they are considered benign. It is important to mention that all features (metadata about a traffic flow) have numerical and continuous values, unlike some packet-level data containing categorical values (e.g., IP addresses, port numbers, and transport protocol).

9.3.2 Synthetic Attack Data Generation and Validation Approach

In this section, the second module of our proposed methodology has been discussed. First, we explain our GAN based data generation through feature space model in subsection 9.3.2. And finally, we summarize our synthetic data validation method in 9.3.2.

GAN Based Data Generation in Feature Space

Generative Adversarial Networks (GANs) [196] are a method to generate synthetic data by learning from a given set of input data distribution. GANs consist of two neural networks: a generator network and a discriminator network as depicted in Figure 1. The generator network is trained to generate synthetic data from random noise. The discriminator network is trained to distinguish generated synthetic data from real world data. The generator network is trained by the output signal gradient of the discriminator network. Both networks are trained iteratively until the generator network can deceive the discriminator network. In our methodology, we followed feature space model for designing the discriminator model of our GAN where the discriminator operates on features extracted from the input data. This is in contrast to the problem space model where the discriminator operates directly on the input data, without first extracting features.

Our proposed generator is a neural network that takes a noise vector of shape (100,) as input and outputs a synthetic data sample of shape (8,). The architecture of the generator network consists of the four dense layers with 256, 512, 1024, and 8 unit followed by LeakyReLU activation with an alpha of 0.2. We chose LeakyReLU activation for both generator and discriminator as it helps mitigates the vanishing gradient problem, common in deep neural networks, by allowing a small negative slope (controlled by the alpha parameter) when the input is negative. This ensures that gradients flow even for negative input values, promoting better learning and helping the GAN to generate more realistic data. Batch Normalization layers are added after certain layers in the generator network. These layers normalize the output of the previous layers, maintaining the mean activation close to 0 and the activation standard deviation close to 1. This helps in improving the training process by reducing internal covariate shift, leading to faster convergence and better generalization. As a result, the generator can produce more realistic synthetic data. Our discriminator is a neural network that takes a data sample of shape (8,) as input and outputs a scalar value representing the probability that the input data is real. The architecture of the discriminator network consists of the four dense layers consisting of 1024, 512, 256, and 1 neuron. GANs excel at capturing the underlying feature distribution of the input data through the adversarial training process. The generator network is trained to create synthetic data resembling the original DDoS attack features, while the discriminator network is trained to differentiate between real and generated data. The iterative training process of GANs involves a minimax game between the generator and discriminator. The adversarial nature of this training process encourages the generator to create more realistic and diverse samples, benefiting the feature space model's ability to generalize across different models and architectures. Our generator network uses a noise vector of length 100 as input. The rationale behind using a noise vector is to provide a source of randomness for the generator, which helps it to explore the data distribution and to learn to create diverse synthetic samples. The choice of 100 dimensions is arbitrary but it is large enough to represent complex patterns in the data while remaining manageable for the neural network. We use *Adam* optimizer for training both the generator and the discriminator. Adam combines the benefits of two other popular optimization algorithms, AdaGrad and RMSProp, and is known for its efficiency and robustness. The learning rate and the beta_1 parameter are manually tuned to prevent oscillations and ensure stable training. We performed GAN training for different batch size of 16, 32, 64, and 128. Investigating the impact of different batch sizes on the GAN model's performance is worth exploring due to several influencing factors that contribute to the model's effectiveness. For example, varying batch sizes can have a significant impact on the GAN model's convergence rate and training stability. Smaller batch sizes may lead to faster convergence but introduce more noise in the gradient updates, while larger batch sizes provide more stable gradient updates at the cost of longer training times. Also, the choice of batch size may affect the likelihood of mode collapse, a common issue in GAN training where the generator learns to produce only a limited set of samples instead of covering the entire data distribution. The training process is repeated for a specified number of epochs, with the generator and discriminator being updated alternately in each epoch. Finally, we saved our fifty individual model at regular epoch interval for generating different set of synthetic dataset for analysis.

Table 9.2: Proportion of Quality Synthetic Dataset Generated by Various Batch Sizes (BS) of GAN.

IDS	**BS*-16**	**BS-32**	**BS-64**	**BS-128**	**Total**
LR	35 (70%)	25 (50%)	38 (76%)	36 (72%)	134 (67%)
NB	31 (62%)	24 (48%)	32 (64%)	25 (50%)	112 (56%)
NN	35 (70%)	21 (42%)	33 (66%)	25 (50%)	114 (57%)
SVM	25 (50%)	18 (36%)	26 (52%)	16 (32%)	85 (43%)
Total (200)	126 (63%)	88 (44%)	129 (65%)	102 (51%)	

Evaluation of Synthetic IoT Traffic Dataset

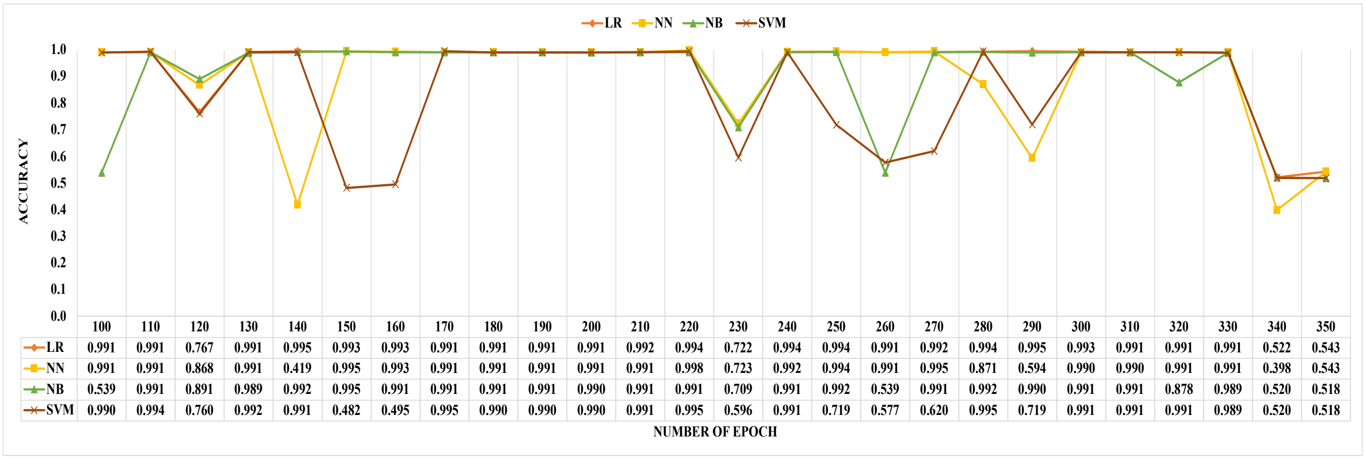
We followed the "Train on Synthesis and Test on Real" (TSTR) approach [205] originally developed to assess the performance of time-series GAN in the health domain. In this approach, the effectiveness of GAN is indirectly evaluated in terms of the quality of generated traffic for informing sufficient pattern to the anomaly-based IDS it is being used to train. That said, the IDS classifiers are trained on the synthetic data produced by GAN model under evaluation, and then the performance of the classifiers is tested against the real dataset used by the GAN. Thus, the test accuracy would imply the quality of the GAN model. This evaluation metric confirms the ability of the synthetic data to be used for real applications. The TSTR metric is also later used in evaluating traffic GAN for the CIDDS dataset [213].

9.4 Analysis of Experimental Results

In this section, we first describe the various sets of synthetic IoT traffic datasets that we generated using the GAN model described in the Subsection 9.4.1. We also implemented the TSTR approach to evaluate the quality of the produced IoT datasets. Our evaluation results are presented, interpreted and analyzed in the subsection 9.4.2.

9.4.1 Generated Synthetic IoT Traffic Dataset

Using the methodology described in the Section III (A), we generated 200 different IoT traffic datasets of DDoS attack to smart home environment. We examined four different batch size for training the GAN to analyze the impact of batch size on the model's convergence rate and training stability. We collected a set of synthetic traffic data for each individual training settings. Firstly, we trained our GAN models using four different batch sizes (i.e., batch size 16, batch size 32, batch size 64, and batch size 128). During each training phase, we saved our GAN model snapshot at every 10 epochs while our training continued for total of 500 epochs, and we got 50 different GAN models for each training session with different batch size. As a result, a total of 200 (4 different batch * 50 model in each batch) different GAN model were employed to generate 200 different synthetic attack datasets for our analysis. Each of the datasets includes 3000 traffic flow (can be accessed from our online repository in [212]).



	100	110	120	130	140	150	160	170	180	190	200	210	220
LR	0.991	0.991	0.767	0.991	0.995	0.993	0.993	0.991	0.991	0.991	0.991	0.992	0.994
NN	0.991	0.991	0.868	0.991	0.419	0.995	0.993	0.991	0.991	0.991	0.991	0.991	0.998
NB	0.539	0.991	0.891	0.989	0.992	0.995	0.991	0.991	0.991	0.991	0.990	0.991	0.991
SVM	0.990	0.994	0.760	0.992	0.991	0.482	0.495	0.995	0.990	0.990	0.990	0.991	0.995

	230	240	250	260	270	280	290	300	310	320	330	340	350
LR	0.722	0.994	0.994	0.991	0.992	0.994	0.995	0.993	0.991	0.991	0.991	0.522	0.543
NN	0.723	0.992	0.994	0.991	0.995	0.871	0.594	0.990	0.990	0.991	0.991	0.398	0.543
NB	0.709	0.991	0.992	0.539	0.991	0.992	0.990	0.991	0.991	0.878	0.989	0.520	0.518
SVM	0.596	0.991	0.719	0.577	0.620	0.995	0.719	0.991	0.991	0.991	0.989	0.520	0.518

Figure 9.2: Accuracy of different classifiers trained by synthetic datasets.

9.4.2 Evaluation of Synthetic Traffic for Quality

The 200 synthetic datasets were evaluated indirectly through the accuracy of the classifiers they trained to. The quality of a classification model depends on the quality of the training dataset. For each setting of GAN models (characterized by batch-size used per epoch during training), 50 different IoT datasets are collected during different epochs. We implemented four types of classifiers for each dataset, resulting in a total of 200 classification models to test. Then, we tested each of their classification accuracy against real IoT traffic dataset.

We determined a classifier producing "satisfactory performance" (reflecting "good quality" of the underlying datasets and hence, the effectiveness of the GAN models) if it demonstrates classification accuracy above 90% when tested against the real dataset. Table I summarizes the number and proportion of quality traffic datasets that reflects satisfactory performance. Each of the columns (2^{nd} to 5^{th}) of the Table I presents results from classifiers trained with a particular batch size (i.e., 16, 32, 64, and 128). Each row (2^{nd} to 6^{th}) shows results pertaining to a particular type of classifier. For instance, with reference to the 1st column of the 1st row of Table I, 35 LR classifiers out of the total 50 (70%) that are trained with batch size 16, shows satisfactory performance. Therefore, 70% of the datasets generated by GAN models involving batch size 16 is of good quality. From our experimental analysis on the effect of batch-size, we observe that the GAN model trained with different batch sizes (i.e., 16, 32, 64, 128) achieves diverse performance (please see the variations of performance in all the rows from the 2^{nd} to 6^{th}). In all, the GAN model that is trained with batch size 64 exhibits the best performance (76%) in terms of generating quality datasets as evident by the LR classifier. However, batch size 16 exhibits the overall best performance (63%) in producing good quality dataset.

Overall, most of the 200 synthetic datasets exhibit satisfactory quality as validated by at the performance of at least four types of trained classifiers. These are LR (134 vs. 200), NB (112 vs. 200), and NN (114 vs. 200) (shown in the last column of the Table I). In contrast, SVM achieves the least performance (85 vs. 200 (43%)) with the same dataset. Please note that the results also include the datasets collected during the initial period (e.g., less than 130 epochs) when the models were still learning, which indicates a higher percentage of quality datasets in the later epochs. In Figure 2, we illustrate how the performance of GAN evolves with the number of epochs during training. Fig 2 shows the classification accuracy of the IDS classifiers that are trained by datasets collected between 100 and 350 epochs with BS-64. The colored line curves depict the classification accuracy of the classifiers while testing against real dataset. All of the classifiers, with a few exception, trained by datasets collected during 100-250 epoch show above 99% classification accuracy. However, the infrequent degradation of performance of the classifiers also reflects the lack of stability in GAN training.

The datasets unveil the best results in quality when tested by the performance of the implementation of the LR classifiers (shown in orange color) that only degrade performance at three occasions (i.e., 120 epoch, 230 epoch, 340 epoch, and 350 epoch). The performance patterns are quite similar for Naïve Bayes classifiers (shown in green color) and Neural Network classifiers (shown in yellow color). The finding is that the datasets generated by the GAN models between 130-330 epochs are of expected quality to be used by real applications. During the other epochs (< 100 and > 330), the GAN models show more frequent instability to producing dataset of similar quality. The reason for the occurring of such instability during GAN training could be due to the mode collapse problem inherently persist with the GAN [214]. On the other hand, the SVM classifier (brown colored line) demonstrates lack of consistency in accuracy. Further, it has no trend of performance improvement with the increase of epoch.

In summary, the regular GAN does not show stable performance throughout the training. This finding supports the existence of the mode collapse problem in the regular GAN reported in [208]. Despite the instability of learning, the regular GAN is able to generate quality datasets during most of the epochs. This is evident by the performance of at least three types of classifiers (i.e., LR, NB, and NN) that we examined in our study. The evaluation by TSTR metrics shows that out of the total 200 dataset generated by the different settings of GAN model, most of them (e.g., LR (134 vs. 200), NB (112 vs. 200), and NN (114 vs. 200) demonstrated satisfactory quality (shown in the Table I). One implication of this finding is that TSTR method can be effectively complemented with the regular GAN for differentiating the quality datasets to be used for real applications.

9.5 Conclusion

The new era of IoT brings new security threats, especially to the smart home network. To provide protection against the cyber-attacks through IDS for IoT centric smart home network, there is a need to have labeled diverse and up-to-date training traffic dataset. Although GAN was initially developed in the field of image generation but recently it has been investigated into the field of traffic generation specific to internet traffic. However, IoT traffic differs from internet traffic due to the difference in protocols used by the communicating devices, especially in smart home environments. In this paper, we explore the performance of the original GAN model towards generating IoT traffic dataset of smart home environment. We captured 200 IoT traffic datasets from the implemented GAN model. The quality of the datasets collected at different epochs is evaluated based on the TSTR method. The results of our study suggest that the original GAN model can produce quality datasets during most of the epochs, while the best performance is observed during 130 - 330 epochs (shown in Fig 2). Further, the GAN model trained by different batch sizes exhibits variations in performance, achieving the best results (76% quality datasets) when trained with the batch size 64 (please see the 2nd row and the 4th column of the Table I). With respect to Table I, as evident by the three different types of classifiers, most of the 200 synthetic datasets (i.e., LR (134 vs. 200), NB (112 vs. 200), and NN (114 vs. 200)) exhibits good quality (i.e., above 99% classification accuracy achieved by their trained classifiers) (shown in the Table I). Though its performance is not stable throughout the GAN training. This is potentially due to the mode collapse problem, which supports the previous analysis presented in. Nevertheless, the quality of the datasets can be realistically measured through the TSTR metric, which would help to distinguish quality datasets for use in real applications. This finding contributes to producing IoT traffic datasets of smart home to be used by real applications and conducting network security research. Such synthetic datasets would also reduce privacy-related concerns typically arise in research work conducted on real dataset in smart home and other applications of IoT. In the future, we plan to create and use more diversified dataset involving more complex IoT network and different types of cyber-attacks by the use of IoT traffic generator tool. Further, we look forward to implementing or upgrade other versions of GAN that show more stability in learning.

Chapter 10

Conclusion and Future Work

In this chapter, we present the conclusions and future work of this thesis. In Section 10.1, we revisit the research problem and the emerging theory from Chapter 2-9. Then, we discuss future work in Section 10.2.

10.1 Conclusion

The digital age has accelerated the rise of global internet usage and, consequently, the volume of network traffic. Predicted to reach 5.3 billion users by 2023, the internet and its surrounding technologies have transformed the way we communicate, work, and entertain. This rapid expansion has brought along some significant challenges, particularly the strain on network capacity due to the escalating growth of internet traffic. Key factors such as increasing user numbers, Machine-to-Machine connections, IPv6 adoption, and the rise in cyber-attacks, including DDoS attacks, significantly contribute to this growth. With this complexity in mind, it becomes evident that accurate traffic forecasting is crucial for service providers to optimize network performance, reduce costs, ensure Quality of Service (QoS) and Quality of Experience (QoE), and deliver reliable and consistent internet speeds. Leveraging machine learning techniques and advanced analytics, major industry players like Verizon, Huawei, and Nokia have been able to predict future traffic patterns and thereby improve their service provision.

In this study, we emphasized four main challenges that hinder the accurate prediction of real-world internet traffic. Firstly, the data anomalies or outliers significantly affect the performance of the predictive models by causing incorrect inferences and skewing the overall analysis. Therefore, proper detection and removal of these outliers are critical to improving the model's accuracy. Secondly, the noise present in wireless network traffic can obscure valuable data patterns, hence the need for effective noise reduction techniques. Thirdly, the issue of having a limited amount of historical data makes it challenging to create prediction models for different geographical or network sectors. Here, transfer learning offers a promising solution. Lastly, the problem of overfitting caused by the non-identical distribution of internet traffic data and the lack of Out-of-Distribution (OOD) generalization in most current approaches requires more focus.

In light of these challenges, we proposed several approaches. We conducted an analysis of deep sequence models and gradient-boosting algorithms for traffic prediction, integrated outlier detection, and employed noise reduction techniques. We also presented a novel approach that combines transfer learning and data augmentation for efficient forecasting in limited data scenarios. Lastly, we addressed the OOD generalization issue by proposing a solution that combines a hybrid deep learning model with Discrete Wavelet Transformation.

Additionally, our research delved into the realm of cyber threats, focusing on the detection of DDoS attacks, which contribute significantly to the surge in global network traffic. The escalating frequency and intensity of such cyber attacks pose considerable challenges to accurate internet traffic volume forecasting. The unpredictability of traffic spikes caused by these attacks, often concealed, introduces substantial uncertainty into the forecasting process. Moreover, the complexities of these challenges are intensified by the rapidly evolving techniques employed by cybercriminals, resulting in new forms of traffic that can outpace the adaptability of existing forecasting models. Increased noise due to heightened attack activities, the lack of historical data due to the rapidly changing cyber threat landscape, and the inconsistent impact of such attacks obscure the discernment of patterns essential for accurate forecasts.

To tackle these challenges, we advocated for a phased approach, initially employing supervised learning for attack detection, followed by a transition to forecasting-based models. This strategy, grounded in several strategic advantages, ensures a comprehensive understanding of the domain and the data, which will prove invaluable during the planned future transition to forecasting-based attack detection models. Our core contributions in the cyber-attack detection domain are manifold. We have proposed innovative methodologies for network anomaly detection, introduced a novel Ensemble Network Intrusion Detection System (ENIDS), addressed the challenges in Mobile Cyber-Physical Systems (MCPS) security, specifically concerning Android devices, and aimed to counter the vulnerability of traditional neural network models used for intrusion detection systems (IDS) in IoT-based smart homes to adversarial attacks.

In conclusion, as the digital world continues to expand, understanding and addressing these challenges are of paramount importance for ensuring network capacity, integrity, and a seamless internet experience for the growing global user base. This study provides invaluable insights and suggests potential solutions for improving internet traffic prediction and cyber-attack detection. Future work should aim to build upon these findings, exploring more efficient techniques for traffic prediction, cyber-attack detection, and addressing any new challenges that may emerge due to the continually evolving nature of internet traffic.

10.2 Future Work

In this section, we explain how we are going to extend some of the research that we have discussed above, either by improving performance or by broadening the scope of the application. We also present some novel ideas on which we plan to work.

10.2.1 Adaptation of External and Internal Factors

We are aiming to improve our traffic prediction model with the first enhancement being the incorporation of both external and internal factors affecting internet traffic. In reality, IP network traffic is highly sensitive to influences such as the integration of new internet services, traffic migration, the use of various internet applications, and more. These factors introduce non-linearity and complexity, making long-term forecasting of internet traffic quite challenging. Consequently, our goal is to develop a method that overlays the impacts of these influences onto our traffic predictions.

10.2.2 Internet Traffic Synthetic Dataset Generation using GAN

Despite the progress made in the internet traffic prediction task, there are still several challenges to be addressed. For instance, most existing approaches assume that the traffic data is independent and identically distributed (i.i.d.), which is not always true in real-world scenarios. Furthermore, the impact of outliers on traffic forecasting accuracy is often ignored. Another challenge is the limited availability of labeled data, which can hinder the development of accurate prediction models. Additionally, most existing models are designed to handle a specific type of traffic and may not be generalizable to different types of traffic. To overcome these challenges, there is a need for more advanced techniques that can model the complex, dynamic nature of internet traffic accurately. In recent years, Generative Adversarial Networks (GANs) have shown great potential in generating synthetic data that can be used for training machine learning models. Using GANs, it is possible to generate synthetic traffic data that closely resembles real-world traffic, thereby providing a way to augment the limited labeled data. Therefore, we would like to develop novel approaches that leverage GANs to generate internet traffic datasets that can be used to train accurate traffic prediction models.

10.2.3 Real-time Anomaly Detection Based on Traffic Prediction

Traffic forecasting-based anomaly detection is a technique that uses traffic prediction models to identify unusual patterns in network traffic. By analyzing predicted traffic patterns against actual traffic patterns, the system can detect if there are any discrepancies or deviations from normal traffic behavior. This approach is particularly useful in identifying network attacks, faults, or congestion that may impact the network's performance. For example, if the predicted traffic pattern shows a sudden spike in network usage, but the actual traffic pattern shows a decline, the system can alert network administrators of a possible anomaly. By detecting these anomalies in real time, network administrators can take action to address any issues that may impact network performance, improving the overall reliability and efficiency of the network. Therefore, we would like to explore the real-time anomaly detection capabilities of our proposed traffic prediction model in the wireless network.

10.2.4 Anomaly Detection Based on Traffic Forecasting

In this thesis, we have extensively engaged in supervised cyber attack detection, utilizing ensemble feature selection and ensemble machine learning. Concurrently, we also explored the domain of internet traffic forecasting. As we look towards the future, our interest pivots towards an innovative fusion of these areas: cyber attack detection based on traffic forecasting. We believe that by applying the principles of traffic forecasting to the realm of cyber security, we could anticipate and identify cyber attacks more effectively. This ambitious intersection of forecasting and security would facilitate the proactive detection of threats, thereby enhancing the robustness of our network security measures.

10.2.5 Enhancing Attack Detection through Reinforcement Learning

We also want to plan a comprehensive strategy for advancing the field of attack detection by utilizing the capabilities of reinforcement learning (RL). With the increasing complexity and diversity of cyber threats, conventional intrusion detection systems face challenges in accurately identifying evolving attack patterns. Therefore, we aim to integrate RL's adaptive learning and decision-making capabilities to develop a novel approach that enhances the accuracy and adaptability of attack detection systems.

Bibliography

[1] Cisco Annual Internet Report - Cisco Annual Internet Report (2018–2023) White Paper — cisco.com. https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html. [Accessed 15-Mar-2023].

[2] Zhiming Hu, Yan Qiao, and Jun Luo. Coarse-grained traffic matrix estimation for data center networks. Computer Communications, 56:25–34, 2015.

[3] Guangjie Han, Hao Wang, Xu Miao, Li Liu, Jinfang Jiang, and Yan Peng. A dynamic multipath scheme for protecting source-location privacy using multiple sinks in wsns intended for iiot. IEEE Transactions on Industrial Informatics, 16(8):5527–5538, 2019.

[4] https://www.marketresearchfuture.com/ Market Research Future. Wireless Network Infrastructure Ecosystem Market Report-Forecast to 2030 — MRFR — marketresearchfuture.com. https://www.marketresearchfuture.com/reports/wireless-network-infrastructure-ecosystem-market-936. [Accessed 15-Mar-2023].

[5] Fact sheet: Verizon network analytics — verizon business. https://www.verizon.com/business/resources/articles/factsheets-verizon-network-analytics/. (Accessed on 03/15/2023).

[6] Ai-in-network-en.pdf. chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://carrier.huawei.com/~/media/cnbgv2/download/products/wireless-network/ai-in-network-en.pdf. (Accessed on 03/15/2023).

[7] Nokia: Nokia traffica brochure. https://onestore.nokia.com/asset/200785?_ga=2.260889806.2114475127.1678883910-303896294.1678883909. (Accessed on 03/15/2023).

[8] Xu Zhou, Yong Zhang, Zhao Li, Xing Wang, Juan Zhao, and Zhao Zhang. Large-scale cellular traffic prediction based on graph convolutional networks with transfer learning. Neural Computing and Applications, pages 1–11, 2022.

[9] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. Time series analysis: forecasting and control. John Wiley & Sons, 2015.

[10] Steven Wheelwright, Spyros Makridakis, and Rob J Hyndman. Forecasting: methods and applications. John Wiley & Sons, 1998.

[11] Vicente Alarcon-Aquino and Javier A Barria. Multiresolution fir neural-network-based learning algorithm applied to network traffic prediction. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 36(2):208–220, 2006.

[12] Cong Wang, Xiaoxia Zhang, Han Yan, and Linlin Zheng. An internet traffic forecasting model adopting radical based on function neural network optimized by genetic algorithm. In First International Workshop on Knowledge Discovery and Data Mining (WKDD 2008), pages 367–370. IEEE, 2008.

[13] B Ricky Rambharat, Anthony E Brockwell, and Duane J Seppi. A threshold autoregressive model for wholesale electricity prices. Journal of the Royal Statistical Society: Series C (Applied Statistics), 54(2):287–299, 2005.

[14] Nobuhiko Terui and Herman K Van Dijk. Combined forecasts from linear and nonlinear time series models. International Journal of Forecasting, 18(3):421–438, 2002.

[15] Mojtaba Nabipour, Pooyan Nayyeri, Hamed Jabani, Shahab S., and Amir Mosavi. Predicting stock market trends using machine learning and deep learning algorithms via continuous and binary data; a comparative analysis. IEEE Access, 8:150199–150212, 2020.

[16] Mohil Maheshkumar Patel, Sudeep Tanwar, Rajesh Gupta, and Neeraj Kumar. A deep learning-based cryptocurrency price prediction scheme for financial institutions. Journal of information security and applications, 55:102583, 2020.

[17] Xiaoli Ren, Xiaoyong Li, Kaijun Ren, Junqiang Song, Zichen Xu, Kefeng Deng, and Xiang Wang. Deep learning-based weather prediction: a survey. Big Data Research, 23:100178, 2021.

[18] Ming Li, Yuewen Wang, Zhaowen Wang, and Huiying Zheng. A deep learning method based on an attention mechanism for wireless network traffic prediction. Ad Hoc Networks, 107:102258, 2020.

[19] Francisco Rau, Ismael Soto, Pablo Adasme, David Zabala-Blanco, and Cesar A Azurdia-Meza. Network traffic prediction using online-sequential extreme learning machine. In 2021 Third South American Colloquium on Visible Light Communications (SACVLC), pages 01–06. IEEE, 2021.

[20] Shah Zeb, Muhammad Ahmad Rathore, Aamir Mahmood, Syed Ali Hassan, JongWon Kim, and Mikael Gidlund. Edge intelligence in softwarized 6g: Deep learning-enabled network traffic predictions. In 2021 IEEE Globecom Workshops (GC Wkshps), pages 1–6. IEEE, 2021.

[21] Qinghai Li and Rui-Chang Lin. A new approach for chaotic time series prediction using recurrent neural network. Mathematical Problems in Engineering, 2016, 2016.

[22] Zheyan Shen, Jiashuo Liu, Yue He, Xingxuan Zhang, Renzhe Xu, Han Yu, and Peng Cui. Towards out-of-distribution generalization: A survey. arXiv preprint arXiv:2108.13624, 2021.

[23] Bernhard Schölkopf, Francesco Locatello, Stefan Bauer, Nan Rosemary Ke, Nal Kalchbrenner, Anirudh Goyal, and Yoshua Bengio. Toward causal representation learning. Proceedings of the IEEE, 109(5):612–634, 2021.

[24] Márcio LF Miguel, Manoel C Penna, Julio C Nievola, and Marcelo E Pellenz. New models for long-term internet traffic forecasting using artificial neural networks and flow based information. In 2012 IEEE Network Operations and Management Symposium, pages 1082–1088. IEEE, 2012.

[25] Qingtian Zeng, Qiang Sun, Geng Chen, Hua Duan, Chao Li, and Ge Song. Traffic prediction of wireless cellular networks based on deep transfer learning and cross-domain data. IEEE access, 8:172387–172397, 2020.

[26] Audrey Wilmet, Tiphaine Viard, Matthieu Latapy, and Robin Lamarche-Perrin. Outlier detection in ip traffic modelled as a link stream using the stability of degree distributions over time. Computer Networks, 161:197–209, 2019.

[27] Ning Li, Lang Hu, Zhong-Liang Deng, Tong Su, and Jiang-Wang Liu. Research on gru neural network satellite traffic prediction based on transfer learning. Wireless Personal Communications, 118(1):815–827, 2021.

[28] Ali R Abdellah, Omar Abdul Kareem Mahmood, Alexander Paramonov, and Andrey Koucheryavy. Iot traffic prediction using multi-step ahead prediction with neural network. In 2019 11th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), pages 1–4. IEEE, 2019.

[29] Sebastian Fischer, Katerina Katsarou, and Oliver Holschke. Deepflow: Towards network-wide ingress traffic prediction using machine learning at large scale. In 2020 International Symposium on Networks, Computers and Communications (ISNCC), pages 1–8. IEEE, 2020.

[30] Daniel Szostak, Adam Włodarczyk, and Krzysztof Walkowiak. Machine learning classification and regression approaches for optical network traffic prediction. Electronics, 10(13):1578, 2021.

[31] Yue Xu, Wenjun Xu, Feng Yin, Jiaru Lin, and Shuguang Cui. High-accuracy wireless traffic prediction: A gp-based machine learning approach. In GLOBECOM 2017-2017 IEEE Global Communications Conference, pages 1–6. IEEE, 2017.

[32] Sajal Saha, Anwar Haque, and Greg Sidebottom. An empirical study on internet traffic prediction using statistical rolling model. In 2022 International Wireless Communications and Mobile Computing (IWCMC), pages 1058–1063, 2022.

[33] Tiago Prado Oliveira, Jamil Salem Barbar, and Alexsandro Santos Soares. Multilayer perceptron and stacked autoencoder for internet traffic prediction. In Network and Parallel Computing: 11th IFIP WG 10.3 International Conference, NPC 2014, Ilan, Taiwan, September 18-20, 2014. Proceedings 11, pages 61–71. Springer, 2014.

[34] Yunjuan Zang, Feixiang Ni, Zhiyong Feng, Shuguang Cui, and Zhi Ding. Wavelet transform processing for cellular traffic prediction in machine learning networks. In 2015 IEEE China Summit and International Conference on Signal and Information Processing (ChinaSIP), pages 458–462. IEEE, 2015.

[35] Paulo Cortez, Miguel Rio, Miguel Rocha, and Pedro Sousa. Multi-scale internet traffic forecasting using neural networks and time series methods. Expert Systems, 29(2):143–155, 2012.

[36] Rayner Alfred et al. A genetic-based backpropagation neural network for forecasting in time-series data. In 2015 International Conference on Science in Information Technology (ICSITech), pages 158–163. IEEE, 2015.

[37] Chih-Wei Huang, Chiu-Ti Chiang, and Qiuhui Li. A study of deep learning networks on mobile traffic forecasting. In 2017 IEEE 28th annual international symposium on personal, indoor, and mobile radio communications (PIMRC), pages 1–6. IEEE, 2017.

[38] Weitao Wang, Yuebin Bai, Chao Yu, Yuhao Gu, Peng Feng, Xiaojing Wang, and Rui Wang. A network traffic flow prediction with deep learning approach for large-scale metropolitan area network. In NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium, pages 1–9. IEEE, 2018.

[39] Rishabh Madan and Partha Sarathi Mangipudi. Predicting computer network traffic: a time series forecasting approach using dwt, arima and rnn. In 2018 Eleventh International Conference on Contemporary Computing (IC3), pages 1–5. IEEE, 2018.

[40] Ying Han, Yuanwei Jing, and Kun Li. Multi-step prediction for the network traffic based on echo state network optimized by quantum-behaved fruit fly optimization algorithm. In 2017 29th Chinese Control And Decision Conference (CCDC), pages 2270–2274. IEEE, 2017.

[41] Abdolkhalegh Bayati, Kim Khoa Nguyen, and Mohamed Cheriet. Multiple-step-ahead traffic prediction in high-speed networks. IEEE Communications Letters, 22(12):2447–2450, 2018.

[42] Frank J Molnar, Brian Hutton, and Dean Fergusson. Does analysis using "last observation carried forward" introduce bias in dementia research? Cmaj, 179(8):751–753, 2008.

[43] Michael R Chernick. The essentials of biostatistics for physicians, nurses, and clinicians. John Wiley & Sons, 2011.

[44] Alan Kvanli, Robert Pavur, and Kellie Keeling. Concise managerial statistics. Cengage Learning, 2005.

[45] KT Chui, BB Gupta, and P Vasant. A genetic algorithm optimized rnn-lstm model for remaining useful life prediction of turbofan engine. electronics 2021, 10, 285, 2021.

[46] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12:2825–2830, 2011.

[47] Tianqi Chen and Carlos Guestrin. Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. 2016.

[48] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. Advances in neural information processing systems, 30, 2017.

[49] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. Advances in neural information processing systems, 31, 2018.

[50] David Saad. Online algorithms and stochastic approximations. Online Learning, 5(3):6, 1998.

[51] Mirosław Kordos, Álvar Arnaiz-González, and César García-Osorio. Evolutionary prototype selection for multi-output regression. Neurocomputing, 358:309–320, 2019.

[52] Larry R Medsker and LC Jain. Recurrent neural networks. Design and Applications, 5:64–67, 2001.

[53] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.

[54] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259, 2014.

[55] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. Advances in neural information processing systems, 27, 2014.

[56] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in neural information processing systems, 30, 2017.

[57] Shaohe Li, Junping Song, Luyang Xu, Yahui Hu, Wanming Luo, and Xu Zhou. Network traffic prediction based on the feature of newly-generated network flows. In 2022 IFIP Networking Conference (IFIP Networking), pages 1–8. IEEE, 2022.

[58] Javad Dogani, Farshad Khunjush, and Mehdi Seydali. Host load prediction in cloud computing with discrete wavelet transformation (dwt) and bidirectional gated recurrent unit (bigru) network. Computer Communications, 2022.

[59] Yu Chen, Wei Wang, Xuedong Hua, and De Zhao. Survey of decomposition-reconstruction-based hybrid approaches for short-term traffic state forecasting. Sensors, 22(14):5263, 2022.

[60] Yang Yu, Qiang Shang, and Tian Xie. A hybrid model for short-term traffic flow prediction based on variational mode decomposition, wavelet threshold denoising, and long short-term memory neural network. Complexity, 2021, 2021.

[61] Kun Zhang, Yinping Chai, and Xing-an Fu. A network traffic prediction model based on recurrent wavelet neural network. In Proceedings of 2012 2nd International Conference on Computer Science and Network Technology, pages 1630–1633. IEEE, 2012.

[62] Yi Lu, Huan Li, Bin Lu, Yun Zhao, Dongdong Wang, Xiaoli Gong, and Xin Wei. Network traffic model with multi-fractal discrete wavelet transform in power telecommunication access networks. In International Conference on Simulation Tools and Techniques, pages 53–62. Springer, 2019.

[63] Irina Strelkovskaya, Irina Solovskaya, Anastasiya Makoganiuk, and Taisa Rodionova. Multimedia traffic prediction based on wavelet-and spline-extrapolation. In 2020 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom), pages 1–5. IEEE, 2020.

[64] Zhongda Tian. Approach for short-term traffic flow prediction based on empirical mode decomposition and combination model fusion. IEEE Transactions on Intelligent Transportation Systems, 22(9):5566–5576, 2020.

[65] Xinqiang Chen, Jinquan Lu, Jiansen Zhao, Zhijian Qu, Yongsheng Yang, and Jiangfeng Xian. Traffic flow prediction at varied time scales via ensemble empirical mode decomposition and artificial neural network. Sustainability, 12(9):3678, 2020.

[66] Dingde Jiang, Yuqing Wang, Zhihan Lv, Sheng Qi, and Surjit Singh. Big data analysis based network behavior insight of cellular networks for industry 4.0 applications. IEEE Transactions on Industrial Informatics, 16(2):1310–1320, 2019.

[67] Bhusana Premanode, Jumlong Vongprasert, and Christofer Toumazou. Prediction of exchange rates using averaging intrinsic mode function and multiclass support vector regression. Artif. Intell. Res., 2(2):47–61, 2013.

[68] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep learning. MIT press, 2016.

[69] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In International conference on machine learning, pages 1597–1607. PMLR, 2020.

[70] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. IEEE Transactions on knowledge and data engineering, 22(10):1345–1359, 2010.

[71] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. In Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part III 27, pages 270–279. Springer, 2018.

[72] Qiong Wu, Kaiwen He, Xu Chen, Shuai Yu, and Junshan Zhang. Deep transfer learning across cities for mobile traffic prediction. IEEE/ACM Transactions on Networking, 2021.

[73] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. IEEE Transactions on knowledge and data engineering, 22(10):1345–1359, 2009.

[74] Mei Wang and Weihong Deng. Deep visual domain adaptation: A survey. Neurocomputing, 312:135–153, 2018.

[75] Aicha Dridi, Hossam Afifi, Hassine Moungla, and Chérifa Boucetta. Transfer learning for classification and prediction of time series for next generation networks. In ICC 2021-IEEE International Conference on Communications, pages 1–6. IEEE, 2021.

[76] Brian Kenji Iwana and Seiichi Uchida. Time series data augmentation for neural networks by time warping with a discriminative teacher. In 2020 25th International Conference on Pattern Recognition (ICPR), pages 3558–3565. IEEE, 2021.

[77] Stéphane Mallat. A Wavelet Tour of Signal Processing: The Sparse Way. Academic Press, 2009.

[78] Aggelos Lazaris and Viktor K Prasanna. Deep learning models for aggregated network traffic prediction. In 2019 15th International Conference on Network and Service Management (CNSM), pages 1–5. IEEE, 2019.

[79] Hashem Kalbkhani, Mahrokh G Shayesteh, and Nasser Haghighat. Adaptive lstar model for long-range variable bit rate video traffic prediction. IEEE Transactions on Multimedia, 19(5):999–1014, 2016.

[80] Guangjie Han, Hao Wang, Xu Miao, Li Liu, Jinfang Jiang, and Yan Peng. A dynamic multipath scheme for protecting source-location privacy using multiple sinks in wsns intended for iiot. IEEE Transactions on Industrial Informatics, 16(8):5527–5538, 2019.

[81] Qinghai Li and Rui-Chang Lin. A new approach for chaotic time series prediction using recurrent neural network. Mathematical Problems in Engineering, 2016, 2016.

[82] Zheyan Shen, Jiashuo Liu, Yue He, Xingxuan Zhang, Renzhe Xu, Han Yu, and Peng Cui. Towards out-of-distribution generalization: A survey. arXiv preprint arXiv:2108.13624, 2021.

[83] Zhiyong Liu and Lucas Menzel. Identifying long-term variations in vegetation and climatic variables and their scale-dependent relationships: A case study in southwest germany. Global and Planetary Change, 147:54–66, 2016.

[84] Guilherme N. N. Barbosa, Martin Andreoni Lopez, Dianne S. V. Medeiros, and Diogo M. F. Mattos. An entropy-based hybrid mechanism for large-scale wireless network traffic prediction. In 2021 International Symposium on Networks, Computers and Communications (ISNCC), pages 1–6, 2021.

[85] Rishabh Madan and Partha Sarathi Mangipudi. Predicting computer network traffic: a time series forecasting approach using dwt, arima and rnn. In 2018 Eleventh International Conference on Contemporary Computing (IC3), pages 1–5. IEEE, 2018.

[86] Chen Qiu, Yanyan Zhang, Zhiyong Feng, Ping Zhang, and Shuguang Cui. Spatio-temporal wireless traffic prediction with recurrent neural network. IEEE Wireless Communications Letters, 7(4):554–557, 2018.

[87] Shanjun Zhan, Lisu Yu, Zhen Wang, Yichen Du, Yan Yu, Qinghua Cao, Shuping Dang, and Zahid Khan. Cell traffic prediction based on convolutional neural network for software-defined ultra-dense visible light communication networks. Security and Communication Networks, 2021:1–10, 2021.

[88] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in neural information processing systems, 30, 2017.

[89] Zihuan Wang and Vincent WS Wong. Cellular traffic prediction using deep convolutional neural network with attention mechanism. In ICC 2022-IEEE International Conference on Communications, pages 2339–2344. IEEE, 2022.

[90] Yahui Hu, Yujiang Zhou, Junping Song, Luyang Xu, and Xu Zhou. Citywide mobile traffic forecasting using spatial-temporal downsampling transformer neural networks. IEEE Transactions on Network and Service Management, 2022.

[91] Weiwei Jiang. Graph-based deep learning for communication networks: A survey. Computer Communications, 185:40–54, 2022.

[92] Zi Wang, Jia Hu, Geyong Min, Zhiwei Zhao, Zheng Chang, and Zhe Wang. Spatial-temporal cellular traffic prediction for 5 g and beyond: A graph neural networks-based approach. IEEE Transactions on Industrial Informatics, 2022.

[93] Xu Zhou, Yong Zhang, Zhao Li, Xing Wang, Juan Zhao, and Zhao Zhang. Large-scale cellular traffic prediction based on graph convolutional networks with transfer learning. Neural Computing and Applications, pages 1–11, 2022.

[94] Yi Lu, Huan Li, Bin Lu, Yun Zhao, Dongdong Wang, Xiaoli Gong, and Xin Wei. Network traffic model with multi-fractal discrete wavelet transform in power telecommunication access networks. In International Conference on Simulation Tools and Techniques, pages 53–62. Springer, 2019.

[95] Irina Strelkovskaya, Irina Solovskaya, Anastasiya Makoganiuk, and Taisa Rodionova. Multimedia traffic prediction based on wavelet-and spline-extrapolation. In 2020 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom), pages 1–5. IEEE, 2020.

[96] Zhongda Tian. Network traffic prediction method based on wavelet transform and multiple models fusion. International Journal of Communication Systems, 33(11):e4415, 2020.

[97] Youngmin Seo, Sungwon Kim, Ozgur Kisi, and Vijay P Singh. Daily water level forecasting using wavelet decomposition and artificial intelligence techniques. Journal of Hydrology, 520:224–243, 2015.

[98] Ashish Kumar, Rama Komaragiri, and Manjeet Kumar. Time–frequency localization using three-tap biorthogonal wavelet filter bank for electrocardiogram compressions. Biomedical engineering letters, 9(3):407–411, 2019.

[99] Seyed Omid Mousavizadeh Kashi and Meisam Akbarzadeh. A framework for short-term traffic flow forecasting using the combination of wavelet transformation and artificial neural networks. Journal of Intelligent Transportation Systems, 23(1):60–71, 2019.

[100] Thomas G Dietterich. Ensemble methods in machine learning. In International workshop on multiple classifier systems, pages 1–15. Springer, 2000.

[101] Saikat Das, Ahmed M Mahfouz, Deepak Venugopal, and Sajjan Shiva. Ddos intrusion detection through machine learning ensemble. In 2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C), pages 471–477. IEEE, 2019.

[102] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. A detailed analysis of the kdd cup 99 data set. In 2009 IEEE symposium on computational intelligence for security and defense applications, pages 1–6. IEEE, 2009.

[103] Nour Moustafa and Jill Slay. Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In 2015 military communications and information systems conference (MilCIS), pages 1–6. IEEE, 2015.

[104] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In ICISSP, pages 108–116, 2018.

[105] Xianwei Gao, Chun Shan, Changzhen Hu, Zequn Niu, and Zhen Liu. An adaptive ensemble machine learning model for intrusion detection. IEEE Access, 7:82512–82521, 2019.

[106] Ngoc Tu Pham, Ernest Foo, Suriadi Suriadi, Helen Jeffrey, and Hassan Fareed M Lahza. Improving performance of intrusion detection system using ensemble methods and feature selection. In Proceedings of the Australasian Computer Science Week Multiconference, pages 1–6, 2018.

[107] Dwarkoba Gaikwad and Ravindra Thool. Darensemble: Decision tree and rule learner based ensemble for network intrusion detection system. In Proceedings of First International Conference on Information and Communication Technology for Intelligent Systems: Volume 1, pages 185–193. Springer, 2016.

[108] Muhammad Shakil Pervez and Dewan Md Farid. Feature selection and intrusion classification in nsl-kdd cup 99 dataset employing svms. In The 8th International Conference

on Software, Knowledge, Information Management and Applications (SKIMA 2014), pages 1–6. IEEE, 2014.

[109] Chih-Fong Tsai, Yu-Feng Hsu, Chia-Ying Lin, and Wei-Yang Lin. Intrusion detection by machine learning: A review. expert systems with applications, 36(10):11994–12000, 2009.

[110] Srinivas Mukkamala, Andrew H Sung, and Ajith Abraham. Intrusion detection using ensemble of soft computing paradigms. In Intelligent systems design and applications, pages 239–248. Springer, 2003.

[111] Srilatha Chebrolu, Ajith Abraham, and Johnson P Thomas. Feature deduction and ensemble design of intrusion detection systems. Computers & security, 24(4):295–307, 2005.

[112] Fatemeh Amiri, MohammadMahdi Rezaei Yousefi, Caro Lucas, Azadeh Shakery, and Nasser Yazdani. Mutual information-based feature selection for intrusion detection systems. Journal of Network and Computer Applications, 34(4):1184–1199, 2011.

[113] Ravi Vinayakumar, Mamoun Alazab, KP Soman, Prabaharan Poornachandran, Ameer Al-Nemrat, and Sitalakshmi Venkatraman. Deep learning approach for intelligent intrusion detection system. IEEE Access, 7:41525–41550, 2019.

[114] R Vinayakumar, KP Soman, and Prabaharan Poornachandran. Evaluating effectiveness of shallow and deep networks to intrusion detection system. In 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), pages 1282–1289. IEEE, 2017.

[115] R Vinayakumar, Mamoun Alazab, Sriram Srinivasan, Quoc-Viet Pham, Soman Kotti Padannayil, and K Simran. A visualized botnet detection system based deep learning for the internet of things networks of smart cities. IEEE Transactions on Industry Applications, 56(4):4436–4456, 2020.

[116] Heitor Murilo Gomes, Jean Paul Barddal, Fabrício Enembreck, and Albert Bifet. A survey on ensemble learning for data stream classification. ACM Computing Surveys (CSUR), 50(2):1–36, 2017.

[117] Omer Sagi and Lior Rokach. Ensemble learning: A survey. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 8(4):e1249, 2018.

[118] Josef Kittler, Mohamad Hatef, Robert PW Duin, and Jiri Matas. On combining classifiers. IEEE transactions on pattern analysis and machine intelligence, 20(3):226–239, 1998.

[119] Girish Chandrashekar and Ferat Sahin. A survey on feature selection methods. Computers & Electrical Engineering, 40(1):16–28, 2014.

[120] Razieh Sheikhpour, Mehdi Agha Sarram, Sajjad Gharaghani, and Mohammad Ali Zare Chahooki. A survey on semi-supervised feature selection methods. Pattern Recognition, 64:141–158, 2017.

[121] Shih-Wei Lin, Kuo-Ching Ying, Chou-Yuan Lee, and Zne-Jung Lee. An intelligent algorithm with feature selection and decision rules applied to anomaly intrusion detection. Applied Soft Computing, 12(10):3285–3290, 2012.

[122] Opeyemi Osanaiye, Haibin Cai, Kim-Kwang Raymond Choo, Ali Dehghantanha, Zheng Xu, and Mqhele Dlodlo. Ensemble-based multi-filter feature selection method for ddos detection in cloud computing. EURASIP Journal on Wireless Communications and Networking, 2016(1):130, 2016.

[123] Saikat Das, Deepak Venugopal, Sajjan Shiva, and Frederick T Sheldon. Empirical evaluation of the ensemble framework for feature selection in ddos attack. In 2020 7th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2020 6th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom), pages 56–61. IEEE, 2020.

[124] Saikat Das, Deepak Venugopal, and Sajjan Shiva. A holistic approach for detecting ddos attacks by using ensemble unsupervised machine learning. In Future of Information and Communication Conference, pages 721–738. Springer, 2020.

[125] Samina Khalid, Tehmina Khalil, and Shamila Nasreen. A survey of feature selection and feature extraction techniques in machine learning. In 2014 Science and Information Conference, pages 372–378. IEEE, 2014.

[126] Luis Carlos Molina, Lluís Belanche, and Àngela Nebot. Feature selection algorithms: A survey and experimental evaluation. In 2002 IEEE International Conference on Data Mining, 2002. Proceedings., pages 306–313. IEEE, 2002.

[127] Stephen Adams and Peter A Beling. A survey of feature selection methods for gaussian mixture models and hidden markov models. Artificial Intelligence Review, 52(3):1739–1779, 2019.

[128] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. the Journal of machine Learning research, 12:2825–2830, 2011.

[129] Wikipedia. Adversarial machine learning. accessed on August 15, 2021.

[130] Sara Atske. The internet and the pandemic, Sep 2021.

[131] kepios DataReportal. Digital around the world &mdash; datareportal – global digital insights, 2022.

[132] Nahmany Thales. 5g security solutions: protect devices, identity and data, 2022.

[133] Ansam Khraisat, Iqbal Gondal, Peter Vamplew, and Joarder Kamruzzaman. Survey of intrusion detection systems: techniques, datasets and challenges. Cybersecurity, 2(1):20, 2019.

[134] Ning Hu, Zhihong Tian, Hui Lu, Xiaojiang Du, and Mohsen Guizani. A multiple-kernel clustering based intrusion detection scheme for 5g and iot networks. International Journal of Machine Learning and Cybernetics, pages 1–16, 2021.

[135] Lockheed Martin. Cyber kill chain®, Jun 2022.

[136] gmcdouga checkpoint. Check point research: Cyber attacks increased 50% year over year - check point software, Jan 2022.

[137] Security Imperva. What is a cyber attack — types, examples & prevention — imperva, 2021.

[138] iDeas CSIS. Strategic technologies program — center for strategic and international studies, Nov 2022.

[139] DDoS Imperva. Ddos threat landscape report q1 2022 — resource library, 2022.

[140] Donghwoon Kwon, Hyunjoo Kim, Jinoh Kim, Sang C Suh, Ikkyun Kim, and Kuinam J Kim. A survey of deep learning-based network anomaly detection. Cluster Computing, 22(1):949–961, 2019.

[141] Jamil Ahmad, Haleem Farman, and Zahoor Jan. Deep learning methods and applications. In Deep learning: convergence to big data analytics, pages 31–42. Springer, 2019.

[142] Rong Hu, Zhongying Wu, Yong Xu, and Taotao Lai. Multi-attack and multi-classification intrusion detection for vehicle-mounted networks based on mosaic-coded convolutional neural network. Scientific Reports, 12(1):1–16, 2022.

[143] Moinul Islam Sayed, Ibrahim Mohammed Sayem, Sajal Saha, and Anwar Haque. A multi-classifier for ddos attacks using stacking ensemble deep neural network. In 2022 International Wireless Communications and Mobile Computing (IWCMC), pages 1125–1130. IEEE, 2022.

[144] Chris Sinclair, Lyn Pierce, and Sara Matzner. An application of machine learning to network intrusion detection. In Proceedings 15th annual computer security applications conference (ACSAC'99), pages 371–377. IEEE, 1999.

[145] Zheng Zhang, Jun Li, CN Manikopoulos, Jay Jorgenson, and Jose Ucles. Hide: a hierarchical network intrusion detection system using statistical preprocessing and neural network classification. In Proc. IEEE Workshop on Information Assurance and Security, volume 85, page 90, 2001.

[146] Ripon Patgiri, Udit Varshney, Tanya Akutota, and Rakesh Kunde. An investigation on intrusion detection system using machine learning. In 2018 IEEE Symposium Series on Computational Intelligence (SSCI), pages 1684–1691, 2018.

[147] Eleazar Eskin, Andrew Arnold, Michael Prerau, Leonid Portnoy, and Sal Stolfo. A geometric framework for unsupervised anomaly detection. In Applications of data mining in computer security, pages 77–101. Springer, 2002.

[148] Jiong Zhang, Mohammad Zulkernine, and Anwar Haque. Random-forests-based network intrusion detection systems. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 38(5):649–659, 2008.

[149] Phurivit Sangkatsanee, Naruemon Wattanapongsakorn, and Chalermpol Charnsripinyo. Practical real-time intrusion detection using machine learning approaches. Computer Communications, 34(18):2227–2235, 2011.

[150] MohammadNoor Injadat, Abdallah Moubayed, Ali Bou Nassif, and Abdallah Shami. Multi-stage optimized machine learning framework for network intrusion detection. IEEE Transactions on Network and Service Management, 18(2):1803–1816, 2020.

[151] Raisa Abedin Disha and Sajjad Waheed. Performance analysis of machine learning models for intrusion detection system using gini impurity-based weighted random forest (giwrf) feature selection technique. Cybersecurity, 5(1):1, 2022.

[152] Roberto Magán-Carrión, Daniel Urda, Ignacio Díaz-Cano, and Bernabé Dorronsoro. Towards a reliable comparison and evaluation of network intrusion detection systems based on machine learning approaches. Applied Sciences, 10(5):1775, 2020.

[153] Saikat Das, Sajal Saha, Annita Tahsin Priyoti, Etee Kawna Roy, Frederick T Sheldon, Anwar Haque, and Sajjan Shiva. Network intrusion detection and comparative analysis using ensemble machine learning and feature selection. IEEE Transactions on Network and Service Management, 2021.

[154] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. Communications of the ACM, 60(6):84–90, 2017.

[155] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. nature, 521(7553):436–444, 2015.

[156] Ahmad Javaid, Quamar Niyaz, Weiqing Sun, and Mansoor Alam. A deep learning approach for network intrusion detection system. In Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS), pages 21–26, 2016.

[157] Sohaib Hanif, Tuba Ilyas, and Muhammad Zeeshan. Intrusion detection in iot using artificial neural networks on unsw-15 dataset. In 2019 IEEE 16th international conference on smart cities: improving quality of life using ICT & IoT and AI (HONET-ICT), pages 152–156. IEEE, 2019.

[158] Nour Moustafa, Benjamin Turnbull, and Kim-Kwang Raymond Choo. An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things. IEEE Internet of Things Journal, 6(3):4815–4830, 2018.

[159] AM Aleesa, MOHAMMED Younis, AHMED A Mohammed, and NM Sahar. Deep-intrusion detection system with enhanced unsw-nb15 dataset based on deep learning techniques. Journal of Engineering Science and Technology, 16(1):711–727, 2021.

[160] Hongyu Liu and Bo Lang. Machine learning and deep learning methods for intrusion detection systems: A survey. applied sciences, 9(20):4396, 2019.

[161] Ana Laura Gonzalez Rios, Zhida Li, Kamila Bekshentayeva, and Ljiljana Trajković. Detection of denial of service attacks in communication networks. In 2020 IEEE international symposium on circuits and systems (ISCAS), pages 1–5. IEEE, 2020.

[162] Osama Faker and Erdogan Dogdu. Intrusion detection using big data and deep learning techniques. In Proceedings of the 2019 ACM Southeast Conference, pages 86–93, 2019.

[163] Azriel Henry and Sunil Gautam. Intelligent intrusion detection system using deep learning technique. In Computing, Communication and Learning: First International Conference, CoCoLe 2022, Warangal, India, October 27–29, 2022, Proceedings, pages 220–230. Springer, 2023.

[164] Bo Dong and Xue Wang. Comparison deep learning method to traditional methods using for network intrusion detection. In 2016 8th IEEE international conference on communication software and networks (ICCSN), pages 581–585. IEEE, 2016.

[165] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. Journal of artificial intelligence research, 16:321–357, 2002.

[166] Zhaozhao Xu, Derong Shen, Tiezheng Nie, and Yue Kou. A hybrid sampling algorithm combining m-smote and enn based on random forest for medical imbalanced data. Journal of Biomedical Informatics, 107:103465, 2020.

[167] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 1998.

[168] François Chollet et al. Keras. `https://keras.io`, 2015.

[169] David H Wolpert. Stacked generalization. Neural networks, 5(2):241–259, 1992.

[170] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. {TensorFlow}: a system for {Large-Scale} machine learning. In 12th USENIX symposium on operating systems design and implementation (OSDI 16), pages 265–283, 2016.

[171] Yanxiang Guo, Xiping Hu, Bin Hu, Jun Cheng, Mengchu Zhou, and Ricky YK Kwok. Mobile cyber physical systems: Current challenges and future networking applications. IEEE Access, 6:12360–12368, 2017.

[172] Lingguang Lei, Yuewu Wang, Jian Zhou, Lei Wang, and Zhongwen Zhang. A threat to mobile cyber-physical systems: Sensor-based privacy theft attacks on android smartphones. In 2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, pages 126–133, 2013.

[173] Arash Habibi Lashkari, Andi Fitriah A Kadir, Laya Taheri, and Ali A Ghorbani. Toward developing a systematic approach to generate benchmark android malware datasets and classification. In 2018 International Carnahan Conference on Security Technology (ICCST), pages 1–7. IEEE, 2018.

[174] Akshay Ashok Wakhare. Malware Detection in Android platform using DNN. PhD thesis, Dublin, National College of Ireland, 2021.

[175] Xing Yang, Lei Shu, Jianing Chen, Mohamed Amine Ferrag, Jun Wu, Edmond Nurellari, and Kai Huang. A survey on smart agriculture: Development modes, technologies, and security and privacy challenges. IEEE/CAA Journal of Automatica Sinica, 8(2):273–302, 2021.

[176] Raden Budiarto Hadiprakoso, Herman Kabetta, and I Komang Setia Buana. Hybrid-based malware analysis for effective and efficiency android malware detection. In 2020 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS), pages 8–12. IEEE, 2020.

[177] Liangyi Gong, Zhenhua Li, Feng Qian, Zifan Zhang, Qi Alfred Chen, Zhiyun Qian, Hao Lin, and Yunhao Liu. Experiences of landing machine learning onto market-scale mobile malware detection. In Proceedings of the Fifteenth European Conference on Computer Systems, pages 1–14, 2020.

[178] Songhao Lou, Shaoyin Cheng, Jingjing Huang, and Fan Jiang. Tfdroid: Android malware detection by topics and sensitive data flows using machine learning techniques. In 2019 IEEE 2Nd international conference on information and computer technologies (ICICT), pages 30–36. IEEE, 2019.

[179] Suleiman Y Yerima and Sarmadullah Khan. Longitudinal performance analysis of machine learning based android malware detectors. In 2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security), pages 1–8. IEEE, 2019.

[180] Hanqing Zhang, Senlin Luo, Yifei Zhang, and Limin Pan. An efficient android malware detection system based on method-level behavioral semantic analysis. IEEE Access, 7:69246–69256, 2019.

[181] E Mariconti, O Lucky, and P Andriotis. Detecting android malware by building markov chains of behavioural models. In NDDS'17. 2017.

[182] Xinrui Tan, Hongjia Li, Liming Wang, and Zhen Xu. End-edge coordinated inference for real-time byod malware detection using deep learning. In 2020 IEEE Wireless Communications and Networking Conference (WCNC), pages 1–6. IEEE, 2020.

[183] Petr Gronát, Javier Alejandro Aldana-Iuit, and Martin Bálek. Maxnet: Neural network architecture for continuous detection of malicious activity. In 2019 IEEE Security and Privacy Workshops (SPW), pages 28–35. IEEE, 2019.

[184] Xi Xiao, Shaofeng Zhang, Francesco Mercaldo, Guangwu Hu, and Arun Kumar Sangaiah. Android malware detection based on system call sequences and lstm. Multimedia Tools and Applications, 78(4):3979–3999, 2019.

[185] KDT Gireesh Kumar. Efficient android malware scanner using hybrid analysis. International Journal of Recent Technology and Engineering (TM), 7:76–80, 2019.

[186] Wei Wang, Yuanyuan Li, Xing Wang, Jiqiang Liu, and Xiangliang Zhang. Detecting android malicious apps and categorizing benign apps with ensemble of classifiers. Future generation computer systems, 78:987–994, 2018.

[187] Salvador Morales-Ortega, Ponciano Jorge Escamilla-Ambrosio, Abraham Rodriguez-Mota, and Lilian D Coronado-De-Alba. Native malware detection in smartphones with android os using static analysis, feature selection and ensemble classifiers. In 2016 11th International Conference on Malicious and Unwanted Software (MALWARE), pages 1–8. IEEE, 2016.

[188] Abir Rahali, Arash Habibi Lashkari, Gurdip Kaur, Laya Taheri, Francois Gagnon, and Frédéric Massicotte. Didroid: Android malware classification and characterization using deep image learning. In 2020 The 10th international conference on communication and network security, pages 70–82, 2020.

[189] Pakarat Musikawan, Yanika Kongsorot, Ilsun You, and Chakchai So-In. An enhanced deep learning neural network for the detection and identification of android malware. IEEE Internet of Things Journal, 2022.

[190] Akshay Ashok Wakhare. Malware Detection in Android platform using DNN. PhD thesis, Dublin, National College of Ireland, 2021.

[191] Mohammad Al-Fawa'reh, Amal Saif, Mousa Tayseer Jafar, and Ammar Elhassan. Malware detection by eating a whole apk. In 2020 15th International Conference for Internet Technology and Secured Transactions (ICITST), pages 1–7. IEEE, 2020.

[192] Janaka Senanayake, Harsha Kalutarage, and Mhd Omar Al-Kadri. Android mobile malware detection using machine learning: A systematic review. Electronics, 10(13):1606, 2021.

[193] Samaneh Mahdavifar, Andi Fitriah Abdul Kadir, Rasool Fatemi, Dima Alhadidi, and Ali A Ghorbani. Dynamic android malware category classification using semi-supervised deep learning. In 2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), pages 515–522. IEEE, 2020.

[194] Home. `https://www.juniperresearch.com`. Accessed: 2023-4-30.

[195] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572, 2014.

[196] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[197] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In International conference on machine learning, pages 214–223. PMLR, 2017.

[198] Markus Ring, Daniel Schlör, Dieter Landes, and Andreas Hotho. Flow-based network traffic generation using generative adversarial networks. Computers & Security, 82:156–172, 2019.

[199] Zilong Lin, Yong Shi, and Zhi Xue. Idsgan: Generative adversarial networks for attack generation against intrusion detection. In Advances in Knowledge Discovery and Data Mining: 26th Pacific-Asia Conference, PAKDD 2022, Chengdu, China, May 16–19, 2022, Proceedings, Part III, pages 79–91. Springer, 2022.

[200] Qiao Yan, Mingde Wang, Wenyao Huang, Xupeng Luo, and F Richard Yu. Automatically synthesizing dos attack traces using generative adversarial networks. International journal of machine learning and cybernetics, 10(12):3387–3396, 2019.

[201] Adriel Cheng. Pac-gan: Packet generation of network traffic using generative adversarial networks. In 2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), pages 0728–0734. IEEE, 2019.

[202] Lei Bai, Lina Yao, Salil S Kanhere, Xianzhi Wang, and Zheng Yang. Automatic device classification from network traffic streams of internet of things. In 2018 IEEE 43rd conference on local computer networks (LCN), pages 1–9. IEEE, 2018.

[203] Yaser Al Mtawa, Harsimranjit Singh, Anwar Haque, and Ahmed Refaey. Smart home networks: Security perspective and ml-based ddos detection. In 2020 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), pages 1–8. IEEE, 2020.

[204] Joel Margolis, Tae Tom Oh, Suyash Jadhav, Young Ho Kim, and Jeong Neyo Kim. An in-depth analysis of the mirai botnet. In 2017 International Conference on Software Security and Assurance (ICSSA), pages 6–12. IEEE, 2017.

[205] Cristóbal Esteban, Stephanie L Hyland, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional gans. arXiv preprint arXiv:1706.02633, 2017.

[206] Auwal Sani Iliyasu and Huifang Deng. Semi-supervised encrypted traffic classification with deep convolutional generative adversarial networks. IEEE Access, 8:118–126, 2019.

[207] B Selvakumar and Karuppiah Muneeswaran. Firefly algorithm based feature selection for network intrusion detection. Computers & Security, 81:148–155, 2019.

[208] Viacheslav Belenko, Valery Chernenko, Maxim Kalinin, and Vasiliy Krundyshev. Evaluation of gan applicability for intrusion detection in self-organizing networks of cyber physical systems. In 2018 International Russian Automation Conference (RusAutoCon), pages 1–7. IEEE, 2018.

[209] Syed Ghazanfar, Faisal Hussain, Atiq Ur Rehman, Ubaid U Fayyaz, Farrukh Shahzad, and Ghalib A Shah. Iot-flock: An open-source framework for iot traffic generation. In 2020 International Conference on Emerging Trends in Smart Technologies (ICETST), pages 1–6. IEEE, 2020.

[210] Hung Nguyen-An, Thomas Silverston, Taku Yamazaki, and Takumi Miyoshi. Generating iot traffic: A case study on anomaly detection. In 2020 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN, pages 1–6. IEEE, 2020.

[211] M Hammad Mazhar and Zubair Shafiq. Characterizing smart home iot traffic in the wild. In 2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI), pages 203–215. IEEE, 2020.

[212] `https://drive.google.com/drive/folders/`. Accessed: 2023-4-30.

[213] Pasquale Zingo and Andrew Novocin. Can gan-generated network traffic be used to train traffic anomaly classifiers? In 2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), pages 0540–0545. IEEE, 2020.

[214] Shahin Mahdizadehaghdam, Ashkan Panahi, and Hamid Krim. Sparse generative adversarial network. In Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops, pages 0–0, 2019.

Appendix A

Copyright Forms of the Papers

There are eleven published papers included in this thesis. Copyright release forms are included in this appendix.

CCC RightsLink

Home | Help | Live Chat | Sign in | Create Account

IEEE Requesting permission to reuse content from an IEEE publication

Deep Learning Based Malapps Detection in Android Powered Mobile Cyber-Physical System

Conference Proceedings:
2023 International Conference on Computing, Networking and Communications (ICNC)

Author: Moinul Islam Sayed

Publisher: IEEE

Date: 20 February 2023

Copyright © 2023, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

RightsLink

Home | Help | Live Chat | Sign in | Create Account

IEEE
Requesting permission to reuse content from an IEEE publication

Towards an Optimal Feature Selection Method for AI-Based DDoS Detection System

Conference Proceedings:
2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)

Author: Sajal Saha

Publisher: IEEE

Date: 08 January 2022

Copyright © 2022, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

CCC RightsLink

Home | Help | Live Chat | Sign in | Create Account

IEEE Requesting permission to reuse content from an IEEE publication

An Empirical Study on Internet Traffic Prediction Using Statistical Rolling Model

Conference Proceedings:
2022 International Wireless Communications and Mobile Computing (IWCMC)

Author: Sajal Saha

Publisher: IEEE

Date: 30 May 2022

Copyright © 2022, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

CCC RightsLink

Home | Help | Live Chat | Sign in | Create Account

IEEE Requesting permission to reuse content from an IEEE publication

A Multi-Classifier for DDoS Attacks Using Stacking Ensemble Deep Neural Network

Conference Proceedings:
2022 International Wireless Communications and Mobile Computing (IWCMC)

Author: Moinul Islam Sayed

Publisher: IEEE

Date: 30 May 2022

Copyright © 2022, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

RightsLink

Home | Help | Live Chat | Sign in | Create Account

Requesting permission to reuse content from an IEEE publication

Deep Sequence Modeling for Anomalous ISP Traffic Prediction

Conference Proceedings: ICC 2022 - IEEE International Conference on Communications

Author: Sajal Saha

Publisher: IEEE

Date: 16 May 2022

Copyright © 2022, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

CCC RightsLink

Home | Help | Live Chat | Sign in | Create Account

IEEE Requesting permission to reuse content from an IEEE publication

Towards an Ensemble Regressor Model for ISP Traffic Prediction with Anomaly Detection and Mitigation

Conference Proceedings:
2022 International Symposium on Networks, Computers and Communications (ISNCC)

Author: Sajal Saha

Publisher: IEEE

Date: 19 July 2022

Copyright © 2022, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

CCC RightsLink

Home | Help | Live Chat | Sign in | Create Account

IEEE Requesting permission to reuse content from an IEEE publication

Analyzing the Impact of Outlier Data Points on Multi-Step Internet Traffic Prediction using Deep Sequence Models

Author: Sajal Saha

Publication: IEEE Transactions on Network and Service Management

Publisher: IEEE

Date: Dec 31, 1969

Copyright © 1969, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

RightsLink

Home Help Live Chat Sign in Create Account

Transfer Learning Based Efficient Traffic Prediction with Limited Training Data

IEEE Requesting permission to reuse content from an IEEE publication

Conference Proceedings:
2023 IEEE 20th Consumer Communications & Networking Conference (CCNC)

Author: Sajal Saha

Publisher: IEEE

Date: 08 January 2023

Copyright © 2023, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

RightsLink

Home | Help | Live Chat | Sign in | Create Account

Requesting permission to reuse content from an IEEE publication

Wavelet-Based Hybrid Machine Learning Model for Out-of-distribution Internet Traffic Prediction

Conference Proceedings:
NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium

Author: Sajal Saha

Publisher: IEEE

Date: 08 May 2023

Copyright © 2023, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

Curriculum Vitae

Name:	Sajal Saha
Education	University of Western Ontario London, ON, Canada Ph.D. in Computer Science, August 2023
	Brock University St. Catharines, ON, Canada M.Sc. in Computer Science, August 2020
	Patuakhali Science and Technology University Dumki, Patuakhali, Bangladesh B.Sc. in Computer Science and Engineering, May 2012
Honours and Awards:	Distinguished Graduate Student Award, Brock University. Western Summer Student Internship (WSSI), Western University. Mitacs Accelerate Research Fellowship, Western University. Western Graduate Research Scholarship (WGRS), Western University. Provost's Brock University International Scholarship, Brock University. DGS Spring 2019 Research Fellowship, Brock University. Research and Graduate Fellowship, Brock University.
Related Work Experience:	Research and Teaching Assistant The University of Western Ontario 2020 - 2023
	Research and Teaching Assistant Brock University 2018 - 2020
	Faculty Member Patuakhali Sci. & Tech. University 2015 - 2018

Publications:

1. Saha, Sajal, Anwar Haque, and Greg Sidebottom. ”Analyzing the Impact of Outlier Data Points on Multi-Step Internet Traffic Prediction using Deep Sequence Models.” IEEE Transactions on Network and Service Management (2023). Impact Factor 4.68

2. Addison, George, Anahita Izadpanahi, Sajal Saha, and Michael Winter. ”L-fuzzy concept analysis using fuzzy categories.” Fuzzy Sets and Systems 460 (2023): 72-102. Impact Factor 4.46

3. Islam, Rejwana, Moinul Islam Sayed, Sajal Saha, Mohammad Jamal Hossain, and Md Abdul Masud. ”Android malware classification using optimum feature selection and ensemble machine learning.” Internet of Things and Cyber-Physical Systems 3 (2023): 100-111.

4. Saha, Sajal, Annita Tahsin Priyoti, Aakriti Sharma, and Anwar Haque. ”Towards an Optimized Ensemble Feature Selection for DDoS Detection Using Both Supervised and Unsupervised Method.” Sensors 22, no. 23 (2022): 9144. Impact Factor 3.847

5. Sayed, Moinul Islam, Sajal Saha, Ibrahim Mohammed Sayem, and Sarna Majumder. ”A Comparative Study on Load Balancing Techniques in Software Defined Networks.” International Journal of Advanced Networking and Applications 13, no. 4 (2022): 5016-5023.

6. Das, Saikat, Sajal Saha, Annita Tahsin Priyoti, Etee Kawna Roy, Frederick T. Sheldon, Anwar Haque, and Sajjan Shiva. ”Network intrusion detection and comparative analysis using ensemble machine learning and feature selection.” IEEE Transactions on Network and Service Management (2021). Impact Factor 4.68.

7. Bashir, Golam Md Muradul, Sajal Saha, Md Raihan Talukder, Joy Karmaker, and Md Saiful Islam. ”QA Website Study For Programming Language And Platform For Developers-A Stack Overflow Study.”

8. Islam, Taohidul, Sajal Saha, Ali Ahmed Evan, Nabonita Halder, and Shakti Chandra Dey. ”Monthly weather forecasting through ANN model: A case study in Barisal, Bangladesh.” International Journal of Advanced Research in Computer and Communication Engineering 5, no. 6 (2016): 1-6.

9. Sajal Saha, Moinul Islam Sayed, and Anwar Haque, “Empirical Mode Decomposition and Stationary Wavelet Transformation in Internet Traffic Prediction”, IEEE 42nd International Conference on Computer Communications (IEEE INFOCOM), 2023. Acceptance Rate 33%

10. Sajal Saha, and Anwar Haque, “Wavelet-Based Hybrid Machine Learning Model for Out-of-distribution Internet Traffic Prediction”, IEEE/IFIP 36th Network Operations and Management Symposium (IEEE NOMS), 202, Acceptance Rate 25%

11. M. I. Sayed, Sajal Saha and A. Haque, ”Deep Learning Based Malapps Detection in Android Powered Mobile Cyber-Physical System,” 2023 International Conference on Computing, Networking and Communications (ICNC), Honolulu, HI, USA, 2023, pp. 443-449, doi: 10.1109/ICNC57223.2023.10074208. Acceptance Rate 25%

12. Saha, Sajal, Anwar Haque, and Greg Sidebottom. ”Transfer learning based efficient traffic prediction with limited training data.” In 2023 IEEE 20th Consumer Communications Networking Conference (CCNC), pp. 477-480. IEEE, 2023. Acceptance Rate 34%

13. Saha, Sajal, Moinul Islam Sayed, and Rejwana Islam. ”Detecting DNS over HTTPS Traffic Using Ensemble Feature-based Machine Learning.” In Applied Intelligence for Industry 4.0, pp. 118-131. Chapman and Hall/CRC, 2023.

14. Saha, Sajal, Anwar Haque, ”Out-of-distribution Internet Traffic Prediction Generalization Using Deep Sequence Model” IEEE 56th International Conference on Communications (ICC). Acceptance Rate 39%

15. Sayed, Moinul Islam, Ibrahim Mohammed Sayem, Sajal Saha, and Anwar Haque. ”A Multi-Classifier for DDoS Attacks Using Stacking Ensemble Deep Neural Network.” In 2022 International Wireless Communications and Mobile Computing (IWCMC), pp. 1125-1130. IEEE, 2022. Acceptance Rate 37%

16. Saha, Sajal, Anwar Haque, and Greg Sidebottom. ”Towards an Ensemble Regressor Model for ISP Traffic Prediction with Anomaly Detection and Mitigation.” In 2022 International Symposium on Networks, Computers and Communications (ISNCC), pp. 1-6. IEEE, 2022. Acceptance Rate 37%

17. Saha, Sajal, Anwar Haque, and Greg Sidebottom. ”An empirical study on internet traffic prediction using statistical rolling model.” In 2022 International Wireless Communications and Mobile Computing (IWCMC), pp. 1058-1063. IEEE, 2022. Acceptance Rate 37%

18. Sayed, Moinul Islam, Ibrahim Mohammed Sayem, Sajal Saha, and Anwar Haque. ”A Multi-Classifier for DDoS Attacks Using Stacking Ensemble Deep Neural Network.” In 2022 International Wireless Communications and Mobile Computing (IWCMC), pp. 1125-1130. IEEE, 2022. Acceptance Rate 37%

19. Saha, Sajal, Anwar Haque, and Greg Sidebottom. ”Deep sequence modeling for anomalous isp traffic prediction.” In ICC 2022-IEEE International Conference on Communications, pp. 5439-5444. IEEE, 2022. Acceptance Rate 39%

20. Saha, Sajal, Annita Tahsin Priyoti, Aakriti Sharma, and Anwar Haque. ”Towards an Optimal Feature Selection Method for AI-Based DDoS Detection System.” In 2022 IEEE 19th Annual Consumer Communications Networking Conference (CCNC), pp. 425-428. IEEE, 2022. Acceptance Rate 34%

21. Saha, Sajal, Golam Md Muradul Bashir, Md Raihan Talukder, Joy Karmaker, and Md Saiful Islam. ”Which Programming Language and Platform Developers Prefer for the

Development? A Study Using Stack Overflow." In 2018 International Conference on Innovations in Science, Engineering and Technology (ICISET), pp. 305-310. IEEE, 2018.

22. Saha, Sajal, Rakibul Hasan Rajib, and Sumaiya Kabir. "IoT based automated fish farm aquaculture monitoring system." In 2018 International Conference on Innovations in Science, Engineering and Technology (ICISET), pp. 201-206. IEEE, 2018.

23. Saha, Sajal, Sudipto Baral, and Anwar Haque. "DEK-Forecaster: A Novel Deep Learning Model Integrated with EMD-KNN for Traffic Prediction." arXiv preprint arXiv:2306.03412 (2023). [Submitted to Elsevier Journal of Network and Computer Application].

24. Saha, Sajal, Anwar Haque, and Greg Sidebottom. "Multi-Step Internet Traffic Forecasting Models with Variable Forecast Horizons for Proactive Network Management" [Submitted to Elsevier Journal of Computer Networks].

25. Sajal Saha, and Anwar Haque," Overcoming Data Scarcity Challenges: Predicting Internet Traffic in Small ISP Networks with Transfer Learning and Data Augmentation", [Submitted to Elsevier Journal of Computer Networks].

26. Ibrahim Mohammed Sayem, Moinul Islam Syed, Sajal Saha, and Anwar Haque, "ENIDS: A Multi-Classifier for an AI-Based Ensemble Approach for Network Intrusion Detection", [Submitted to IEEE Access]

27. Rased Nekvi, Sajal Saha, Yasir Matwa, and Anwar Haque, "Examining Generative Adversarial Network for Smart Home DDoS Traffic Generation" [Submitted to 10th International Symposium on Networks, Computers and Communications].