Wollo University

Kombolcha Institute of Technology

College of Informatics

A Novel Machine Learning Model for Automated Detection of Anomalies in Network Traffic

By: Betelhem Mengistu

Supervised by

Alemu Jorgi (Ph.D.)

A Thesis Submitted in Partial Fulfilment of the Requirements for the Degree of Master of Science in Computer Networks and Communication

Kombolcha, Ethiopia

November, 2024

Wollo University

Kombolcha Institute of TechnologyCollege of Informatics
Department of Information Technology

A Novel Machine Learning Model for Automated Detection of Anomalies in Network Traffic

A Thesis Submitted in Partial Fulfilment of the Requirements for the Degree of Master of Science in Computer Networks and Communication

Betelhem Mengistu

Advisor: Alemu Jorgi (Ph.D.)

WOLLO UNIVERSITY

GRADUATE STUDIES DIRECTORATE

ADVISORS' APPROVAL SHEET

This is to certify that the thesis entitled "A Novel Machine Learning Model for Automated Detection of Anomalies in Network Traffic" submitted in partial fulfilment of the requirements for the degree of Master's with specialization in Computer Networks and Communication, the Graduate Program of the Department of Information Technology, and has been carried out by Betelhem Mengistu Id. No SGSR/0139/14, under mysupervision. Therefore, I recommend that the student has fulfilled the requirements and hence hereby can submit the thesis to the department.

Name of advisor	Signature	Date
Alemu Jorgi (Ph.D.)		November , 2024

Examiner's Approval Sheet

This M.Sc. thesis entitled with A Novel Machine Learning Model for Automated Detection of Anomalies in Network Traffic

The following examiners have approved it:

Advisor: Signature Date

Alemu Jorgi (Ph.D.)		November, 2024

External Examiner:

.... Sign .... Date ....

Internal Examiner:

.... Sign .... Date ....

Chairperson:

.... Sign .... Date ....

Declaration

I am Betelhem Mengistu, a student in the College of Informatics, Wollo University, Kombolcha institute of technology, aware of my responsibility, the penal law, declare and certify with my signature that my thesis entitled with A Novel Machine Learning Model for Automated Detection of Anomalies in Network Traffic is entirely the result of my original work and this is not done in another university. I have faithfully and accurately cited all my sources. Every serious effort has been made to avoid any plagiarism in the preparation of this thesis.

Declared by:

Name: Betelhem Mengistu Sign: ______

Date: ____________________________

This thesis has been submitted for examination with my approval as a university advisor.

Confirmed By:

Name: Alemu Jorgi (Ph.D.)Sign: ....

Date: ....

Acknowledgment

First and foremost, I am incredibly grateful for God's blessings and support during my research. I am deeply indebted to my advisor, Dr. Alemu Jorgi, for his patience and wisdom throughout this process. His insights and feedback helped push my work to a higher level and I am thankful for his mentorship.

I would also like to express my gratitude to my loved ones for their constant encouragement. To my family and friends, thank you for believing in me and helping me to persevere even during challenging times. Your love and support have been invaluable.

Finally, I appreciate all of those who contributed directly or indirectly to this research. Reaching this milestone was not a solo effort, and I am grateful to everyone who played a role in helping me complete this work.

List of Abbreviations

Abbreviations	Description of the abbreviations
SVM	Support Vector Machines
KDD	Knowledge Discovery and Data Mining
GANs	Generative Adversarial Networks
AUC	Area under the Curve
ROC	Receiver Operation characteristics
NSL-KDD	Network Security Laboratory KDD Cup
UNSW-NB15	University of New South Wales Network- Based 15
AI	Artificial Intelligence
RNNs	Recurrent neural networks
LSTMs	long short-term memory
OCSVMs	One-class support vector machines
ROC	Receiver operating characteristic
AUC-ROC	area under the ROC curve
ML	Machine Learning
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
ICMP	Internet Control Message Protocol
IP	Internet Protocol
http	Hypertext markup language
ftp	File transfer protocol
Smtp	Simple Mail Transfer Protocol
RF	Random Forest
DT	Decision tree

Table of Contents

Declaration i
Acknowledgment i
List of Abbreviations i
List of Figures iv
List of Tables v
Abstract vi
Chapter ONE 1
Introduction 1
1.1. Background 1
1.2. Statement of the Problem 2
1.3. Objective of the Study 3
1.3.1. General Objective 3
1.3.2. Specific Objectives 4
1.4. Scope of the Study 4
1.5. Significance of the Study 4
1.6. Contribution of the study 5
1.7. Thesis Organization 5
Chapter TWO 6
Literature Review and Related Work 6
2.1. Introduction 6
2.2. Related Work 7
2.2.1. Statistical Methods 7
2.2.2. Rule-based Systems 8
2.2.3. Machine Learning-based Approaches 8
2.2.4. Deep Learning-based Approaches 9
2.2.5. Hybrid and Ensemble Approaches 10
2.2.6. Evaluation Methodology 11
2.3. Research Advances and Open Challenges 12
CHAPTER THREE 14
Methodology 14
3.1. Introduction 14

3.2. Methodology 15
3.2.1. Data Collection 15
3.2.2. Data Preprocessing 16
3.2.3. Model Training 16
3.2.4. Model building 17
3.3. Tools used for the proposed study 18
3.4. Types of Machine Learnings Used 18
3.4.1. Auto encoder 18
3.4.2. Support vector machine (SVM) 18
3.4.3. Random forest 19
3.5. Evaluation Metrics 19
3.6. Pseudocode 20
CHAPTER FOUR 22
Experiment 22
4.1. Selected Features 22
4.2. Experimentation 24
4.2.1. Hybrid Model 25
4.2.2. Random forest Model 25
4.2.3. Support Vector Machine Model 26
4.2.4. Auto encoder Model 26
4.3. Findings 27
4.4. Summary 34
CHAPTER FIVE 35
Conclusion and Recommendation 35
5.1. Conclusion 35
5.2. Recommendations 35
6. References 36
7. Appendix 42

List of Figures

Figure 3-1: Methodology of the study 15
Figure 4-1: Architecture of the proposed model. 24
Figure 4-2: The Difference in Accuracy between using all features and the five chosen features. 28
Figure 4-3: The Difference in F1 between using all features and the five chosen features. 29
Figure 4-4: The Difference in precision score between using all features and the five chosen features. 30
Figure 4-5: The difference in recall between using all features and the five chosen features. 31
Figure 4-6: Execution time using all features and the five chosen features. 31
Figure 4-7: The difference in time between using all features and the five chosen features. .. 32

List of Tables

Table 3-1: Confusion matrix 19
Table 4-1: Sample values for the selected features 23
Table 4-2: Sample values for the selected features 23
Table 4-3: Hybrid model Results. 25
Table 4-4: Random forest Results. 25
Table 4-5: SVM Classification Results. 26
Table 4-6: Autoencoder Model Results. 27

Abstract

Communication over network is considered one of the trending technologies today. Network based Communication affects various industries, including logistics tracking, healthcare, automotive and smart cities. A rising number of cyber-attacks and breaches are rapidly targeting networks equipped with IoT devices. This thesis aims to improve security in networks by enhancing anomaly detection using machine learning.

This thesis identified the challenges and gaps related to securing the Internet of Things networks. The challenges are network size, the number of devices, the human factor, and the complexity of networks. The gaps identified include the lack of research on automatic detection systems used for anomaly detection, in addition to the lack of modelling input parameters required for anomaly detection in networks. Furthermore, there is a lack of comparison of the performance of machine learning algorithms on standard and real network datasets.

This thesis tests a dataset to automatically detect the anomaly using the Auto encoder, Random forest, Support Vector Machine, and Hybrid machine learning algorithms and compares their results with the KDDCUP99 dataset. The results show that Hybrid Model performs better than the other models on the KDDCUP99 dataset. This thesis reduces the number of features required by machine learning algorithms for automatic anomaly detection in the network communication to five features only, which resulted in reduced execution time by an average of 58%.

Keywords: Anomaly Detection; Machine Learning; Network; Security

Chapter ONE

Introduction

1.1. Background

The rapid growth of network technologies has led to an exponential increase in the volume and complexity of network traffic. With this growth comes the challenge of effectively detecting and mitigating anomalies that can indicate malicious activity or network performance issues. Existing methods of anomaly detection rely on manual analysis and rule-based systems, which are often time-consuming and ineffective in handling large-scale, dynamic networks. As cyber threats evolve daily, network security remains a challenge. Last year alone, ransomware damages exceeded $20 billion globally [1]. Yet intrusion detection systems struggle to recognize modern attacks, relying mainly on predefined signatures that savvy attackers circumvent. This begs the question - without any prior examples of anomalous behaviors, how can we autonomously monitor networks and flag sneaky new intrusions? Developing an adaptable detection approach is crucial as digital infrastructures expand our vulnerabilities [2]. Currently, anomaly detection aims to overcome limitations of signature-based methods by learning normal usage patterns and detecting significant deviations [3]. Despite promise, existing techniques often fall short when applied to real-world traffic due to its high dimensionality, noise, and dynamic nature over time [4]. As normal behaviors continuously shift with technology and business needs, detection models require continuous self-updating to retain accuracy [5]. Additionally, false alarms undermine credibility while lack of explainable alerts hinders forensic investigations [6].

Hence, the objective of this research is to develop an automated system for detecting anomalies in network traffic using machine learning algorithms. The proposed approach involves a hybrid ensemble method that combines deep autoencoders, random forest classification, and one-class support vector machines. This ensemble approach aims to enhance performance and robustness by leveraging the strengths of multiple machine learning techniques.

To begin, deep autoencoders are employed to reduce the dimensionality of the data and learn features from unlabeled traffic data. These autoencoders compress the input into a lower-dimensional latent space and then reconstruct the output based on this encoding. By minimizing the reconstruction error, the autoencoders can capture a compressed representation of normal traffic patterns.

Next, the low-dimensional representations obtained from the autoencoders is fed into a random forest classifier. The random forest consists of an ensemble of decision trees that further refine the models of normalcy.

The random forest aims to capture subtle variations within normal patterns that may not be effectively differentiated by the autoencoders alone.

Finally, one-class support vector machines is utilized to establish a decision boundary around the instances representing normal behavior. As a one-class classifier, it can identify novel anomalies without the need for labeled anomalous data. This approach allows the detection of outliers based on their deviation from the established normal behavior.

By combining these three machine learning algorithms in an ensemble framework, the proposed system aims to effectively analyze network traffic patterns and detect abnormal behavior in real- time. This approach harnesses the complementary strengths of the autoencoders, random forest classification, and one-class support vector machines to enhance the accuracy and reliability of anomaly detection in network traffic data. By leveraging these algorithms' collective strengths across supervised, unsupervised and ensemble paradigms through a cooperative multi-stage architecture, the aim is to build models of normalcy from unlabeled traffic and continuously detect anomalous deviations [8], [9]. The methodology is evaluated on benchmark datasets against existing baselines [10].

1.2. Statement of the Problem

The increasing complexity and scale of modern network infrastructures, coupled with the evolving landscape of cyber threats, have intensified the need for robust and efficient methods to detect anomalies in network traffic. As organizations rely heavily on interconnected systems for their day-to-day operations, the potential impact of malicious activities, intrusions, or system failures has become more pronounced [1], [13], [14]. The traditional methods of manual monitoring and rule-based detection systems are no longer sufficient to promptly identify and respond to emerging threats, resulting in a significant gap in network security.

The challenge lies in developing an automated detection system that can effectively discern anomalous patterns within network traffic in real-time [17], [18]. Anomalies may manifest as unusual data patterns, unexpected communication behaviors, or deviations from established baseline norms. These anomalies could signify a range of security threats, including but not limited to malware infiltration, data breaches, denial-of-service attacks, or unauthorized access. Addressing this issue requires the creation of intelligent algorithms and machine learning models capable of learning normal network behavior, adapting to changing environments, and promptly identifying deviations indicative of potential security incidents.

Moreover, the automated detection system must balance accuracy and efficiency to minimize false positives and negatives, ensuring that security teams are provided with actionable and reliable alerts. The integration of advanced anomaly detection techniques, such as behavior analysis, statistical modeling, and machine learning, is crucial to enhance the system's ability to adapt to evolving cyber threats and mitigate the risk of undetected security breaches.

In general, the Automated Detection of Anomalies in Network Traffic is a critical problem that necessitates the development of innovative and adaptive solutions to safeguard organizations against the ever-growing and sophisticated landscape of cyber threats. The successful implementation of such a system will significantly contribute to enhancing the overall resilience and security posture of network infrastructures in the face of continuously evolving cyber security challenges.

While conducting this research, the study addressed the following research questions:

1. How can machine learning algorithms be optimized to effectively identify and classify anomalies in real-time network traffic, considering the dynamic nature of modern network environments?
2. What are the most efficient feature selection and extraction techniques for enhancing the accuracy and speed of anomaly detection in network traffic using automated methods?
3. To what extent can ensemble learning methods, combining multiple anomaly detection algorithms, improve the overall robustness and generalization of automated network traffic anomaly detection systems?

1.3. Objective of the Study

1.3.1. General Objective

The general objective of this study is to develop a machine learning model for automated detection of anomalies in network traffic.

1.3.2. Specific Objectives

To achieve the general objective of this study the following specific objectives are listed:

- To review the existing literature on anomaly detection in network traffic.
- To analyze the characteristics of network traffic data and identify relevant features for anomaly detection.
- To develop a machine learning-based anomaly detection model using supervised and unsupervised learning algorithms.
- To evaluate the performance of the proposed model using real-world network traffic datasets.
- To investigate the scalability and adaptability of the proposed model for deployment in large-scale networks.

1.4. Scope of the Study

The scope of this study is to develop an ensemble intrusion detection framework integrating one-class SVM, random forest, and auto encoder algorithms. The framework is developed and optimized using standardized public intrusion detection datasets such as NSL-KDD and UNSW-NB15. Key aspects of the evaluation includes analyzing the performance of the individual algorithms, identifying optimal ensemble architectures, developing and optimizing machine learning models combining the three techniques, and evaluating the detection capabilities of the proposed ensemble approach against benchmarks. Performance metrics like accuracy, precision, and recall were used for model optimization and evaluation.
Additionally, the study analyzed the effects of periodic retraining and feature importance across algorithms. While interpretations provide insights, a full-scale real-world deployment is beyond the scope.

1.5. Significance of the Study

This study is significant as it addresses the growing need for automated and scalable solutions for detecting anomalies in network traffic. By leveraging machine learning techniques, the research aimed to enhance the efficiency and accuracy of anomaly detection, leading to network security improvement.

1.6. Contribution of the study

The contributions to the literature include identifying the challenges and gaps related to securing communication over networks. The challenges identified are the large network size, high number of devices, impact of the human factor, and complexity of networks. The gaps identified were the lack of research on automatic anomaly detection systems, modeling the input parameters required for automatic anomaly detection in networks, comparing the performance of machine learning algorithms on the KDDCUP99, and developing a high performance machine learning model to detect anomalies automatically over networks. Additionally, the research tested the performance of auto encoder, Random forest, Support Vector Machine, and Hybrid machine learning algorithms on the KDDCUP99 dataset. It also reduced the number of features required by machine learning algorithms for anomaly detection in IoT networks from 25 to only 5 features, resulting in an average 58% reduction in execution time.

1.7. Thesis Organization

This study is organized in five chapters. Chapter 1 gives the introduction and background of anomaly detection in addition the problem statements, objectives included in this chapter. Literatures are deeply studied that are related to Security and anomaly detection in Chapter 2. In Chapter 3 methodologies that used in research work studied which contains the proposed framework, the details of selected strategies, and tools used for development. Chapter 4 contains simulation results and performance evaluation of the proposed model. Chapter 5 presents the conclusion and recommendation of the research work.

Chapter TWO
Literature Review and Related Work

2.1. Introduction

The ubiquitous nature of networked systems has elevated network security to a critical concern in the digital age. The ever-increasing volume, velocity, and variety of network traffic present significant challenges for effectively detecting and mitigating anomalies, which can be indicative of malicious activities, intrusions, or performance bottlenecks [19, 23, 40]. Traditional security mechanisms, such as firewalls and intrusion detection systems (IDS) based on predefined signatures or rules, often fall short in addressing the dynamic and evolving nature of cyber threats [41, 42]. These signature-based approaches are limited by their reliance on known attack patterns and are ineffective against zero-day exploits or novel attack vectors [43].

The financial ramifications of successful cyberattacks, as evidenced by the staggering $20 billion in ransomware damages reported in 2023 [20], underscore the urgent need for robust and adaptive anomaly detection systems. Furthermore, the increasing sophistication and complexity of modern attacks demand a shift from reactive security measures to proactive and predictive approaches [44]. Anomaly detection, which focuses on identifying deviations from established normal behavior, offers a promising paradigm for detecting both known and unknown threats [3, 21]. However, the inherent characteristics of real-world network traffic, such as high dimensionality, noise, and temporal dynamics, pose significant challenges for existing anomaly detection techniques [4, 22].

This chapter provides a comprehensive and in-depth review of the literature on anomaly detection in network traffic. We trace the evolution of anomaly detection techniques, from traditional statistical methods to cutting-edge machine learning and deep learning approaches. We analyze the strengths and weaknesses of various methodologies, highlighting their applicability and limitations in the context of real-world network traffic analysis.

This comprehensive review sets the stage for our proposed hybrid ensemble approach, which aims to address the limitations of existing techniques and enhance the accuracy, robustness, and adaptability of anomaly detection in network traffic.

2.2. Related Work

The field of anomaly detection in network traffic has witnessed a significant evolution, driven by the increasing complexity of networks and the sophistication of cyber threats. Early approaches relied primarily on statistical methods and rule-based systems.

2.2.1. Statistical Methods

Statistical methods form the foundation of many anomaly detection techniques. These methods assume that normal network traffic follows specific statistical distributions, and deviations from these distributions are flagged as anomalies.

Threshold-based methods: These simple yet widely used methods define thresholds for specific network metrics, such as traffic volume, packet size, or connection duration. Any activity exceeding these predefined thresholds is considered anomalous [45]. While easy to implement, threshold-based methods are prone to false positives and negatives due to the dynamic nature of network traffic.

Statistical process control (SPC): SPC techniques, such as control charts and cumulative sum (CUSUM) charts, monitor network traffic for statistically significant changes over time [46]. These methods are effective in detecting gradual drifts in network behavior but may not be suitable for detecting sudden or short-lived anomalies.

Time series analysis: Time series analysis techniques, such as autoregressive integrated moving average (ARIMA) models, analyze historical network traffic data to predict future behavior and identify deviations from expected patterns [47]. These methods are particularly useful for detecting anomalies that exhibit temporal dependencies.

Probabilistic models: Probabilistic models, such as Gaussian Mixture Models (GMMs) [25] and Hidden Markov Models (HMMs) [48], represent normal network traffic as a mixture of probability distributions or a sequence of hidden states. Anomalies are identified as data points with low probability density or unlikely state transitions.

2.2.2. Rule-based Systems

Rule-based systems rely on expert knowledge to define a set of rules that describe normal network behavior. Any activity violating these rules is classified as anomalous. Expert systems and signature-based intrusion detection systems (IDS) fall under this category [49]. While effective in detecting known attack patterns, rule-based systems are difficult to maintain and update as new threats emerge. They also struggle to detect novel or polymorphic attacks that do not match predefined signatures.

2.2.3. Machine Learning-based Approaches

The advent of machine learning has revolutionized the field of anomaly detection, enabling the development of more adaptive and robust systems. Machine learning algorithms can learn complex patterns from data without explicit programming, making them well-suited for analyzing high-dimensional and noisy network traffic.

Supervised Learning: Supervised learning algorithms, such as Support Vector Machines (SVMs) [28-31] and Random Forests (RF) [32-35], require labeled data for training. They learn a mapping from input features to output labels, which can be used to classify new data points as normal or anomalous. While highly accurate, supervised learning methods are limited by the availability of labeled data, which can be expensive and time-consuming to obtain.

Unsupervised Learning: Unsupervised learning algorithms, such as clustering [50] and dimensionality reduction [51], do not require labeled data. They learn the underlying structure of the data by identifying patterns, grouping similar data points, or reducing the dimensionality of the data. Unsupervised learning methods are particularly useful for detecting novel anomalies that have not been previously observed. Examples include K-means clustering, hierarchical clustering, Principal Component Analysis (PCA), and autoencoders.

Semi-supervised Learning: Semi-supervised learning algorithms leverage both labeled and unlabeled data for training [52]. They can improve the accuracy of supervised learning methods by utilizing the information contained in unlabeled data. One-class SVMs (OCSVMs) [36] are a popular semi-supervised technique for anomaly detection, learning a decision boundary that encloses normal data and classifies anything outside as anomalous.

Reinforcement Learning: While less explored in network anomaly detection, reinforcement learning (RL) offers a promising approach for adaptive and dynamic security systems [53]. RL agents learn optimal policies through trial and error, interacting with the environment and receiving rewards or penalties for their actions. This allows RL-based systems to adapt to changing network conditions and evolving attack strategies.

2.2.4. Deep Learning-based Approaches

Deep learning, a subfield of machine learning, has emerged as a powerful tool for anomaly detection due to its ability to learn complex representations from raw data. Deep learning models, such as deep autoencoders [37], convolutional neural networks (CNNs) [54], and recurrent neural networks (RNNs) [55], can automatically extract relevant features from network traffic, eliminating the need for manual feature engineering.

Autoencoders: Autoencoders are unsupervised neural networks that learn compressed representations of input data. They consist of an encoder that maps the input to a lower-dimensional latent space and a decoder that reconstructs the input from the latent representation. Anomalies are detected by measuring the reconstruction error, with higher errors indicating deviations from normal patterns. Variations like Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs) [38] offer further advancements in anomaly detection.

CNNs: CNNs are specialized neural networks designed for processing grid-like data, such as images. They can be applied to network traffic analysis by converting network packets into image-like representations [56]. CNNs can learn spatial patterns in network traffic, which can be indicative of anomalous behavior.

RNNs: RNNs are designed for processing sequential data, making them suitable for analyzing network traffic flows. Long Short-Term Memory (LSTM) networks, a type of RNN, can capture long-term dependencies in network traffic, which can be crucial for detecting anomalies that evolve over time [55].

2.2.5. Hybrid and Ensemble Approaches

To address the limitations of individual algorithms, researchers have explored the use of hybrid and ensemble approaches that combine the strengths of multiple techniques for anomaly detection in network traffic.

One class of hybrid methods involves cascading different learning stages, where the output of one model is used as input to another. For example, some approaches used autoencoders or other representation learning techniques to extract low-dimensional embeddings from the raw network traffic, which were then fed into classifiers or one-class models for further refinement [43]. This combination of feature extraction and modeling helped capture the complex patterns in the data more effectively.

Ensemble methods, on the other hand, aimed to leverage the complementary strengths of diverse anomaly detection algorithms by combining their outputs. Boosting ensembles, for instance, incrementally trained base learners by focusing more on examples misclassified in prior iterations. Bayesian model combination techniques took weighted averages of predictions from multiple detectors, with weights assigned based on their expected precisions [44].

Variants of these ensemble methods also dynamically adjusted the weights based on the operational context.

Transfer learning techniques were explored to address the challenge of limited labeled data for network traffic anomaly detection. By pre-training deep neural network models on large generic datasets and fine-tuning them on smaller network-specific training instances, researchers were able to leverage knowledge gained from related domains to improve performance on the target problem. Self-supervised learning, which leverages auxiliary unlabeled information to guide the learning of relevant normal behaviors, was also investigated as a complementary approach. Ensuring diversity among the ensemble members was crucial to optimize their complementarity and avoid redundancy.

Researchers proposed clustering similar algorithms beforehand and dynamically assigning weights to balance the overall strengths of the ensemble. Visual analytics tools were also explored to facilitate the forensic validation and explainability of ensemble decisions.

The hybrid and ensemble approaches demonstrated the potential to combine the strengths of multiple algorithms, leading to improved anomaly detection performance in the face of the complex and dynamic nature of network traffic data. However, challenges remained in maintaining the scalability, accountability, and real-time deployment of these composite models.

2.2.6. Evaluation Methodology

Rigorous evaluation of anomaly detection techniques for network traffic has been a crucial aspect of this research field. Researchers have focused on developing standardized datasets and comprehensive performance metrics to facilitate objective comparisons and track progress in the domain.

Early benchmark datasets, such as the KDD Cup '99 dataset, provided preprocessed network connection records with labeled attack instances. However, these datasets were found to have certain limitations, such as redundancy and biases, which led to the development of refined versions like NSL-KDD to address these issues.

More recently, researchers have focused on creating datasets that more accurately reflect the modern threat landscape and network infrastructure characteristics. Examples include the UNSW-NB15 dataset, which provided labeled Windows and Linux behaviors, and the CICIDS2017 dataset, which simulated weekly generated organizational traffic with various attack scenarios. These datasets aimed to provide a more realistic and comprehensive testbed for evaluating anomaly detection algorithms.

In terms of performance metrics, the research community has employed a wide range of measures to assess the effectiveness of anomaly detection solutions. Accuracy, precision, recall, and F1 score have been common metrics used to evaluate the overall classification performance. Receiver operating characteristic (ROC) curves and the area under the ROC curve (AUC-ROC) have been widely used to assess the trade-off between true positive rate and false positive rate across different detection thresholds. Detection delay and timeliness have also been important considerations; as timely anomaly detection is crucial for effective mitigation.

Confusion matrices have been used to provide a more detailed breakdown of the types of errors made by the anomaly detectors, helping to identify specific weaknesses and guide further improvements. Holistic reporting of these various performance metrics has been encouraged to provide a comprehensive understanding of the strengths and limitations of different anomaly detection techniques.

The development of standardized datasets and rigorous evaluation methodologies has been instrumental in driving progress in the field of anomaly detection for network traffic. By enabling objective comparisons and fostering the development of more robust and effective solutions, this research has paved the way for practical deployment of anomaly detection systems in real-world network environments.

2.3. Research Advances and Open Challenges

The field of anomaly detection in network traffic has witnessed significant progress over the past decades, with researchers exploring a wide range of statistical, machine learning, and deep learning-based techniques. The combination of these approaches through hybrid and ensemble methods has demonstrated the potential to address the complex and dynamic nature of network traffic data.

However, several key challenges remain that continue to drive ongoing research in this domain. Maintaining the robustness, explainability, and scalability of anomaly detection solutions as digital infrastructures expand in complexity is a crucial concern. Evolving non-stationary environments require continuous dynamic self-updating strategies to prevent performance degradation over time as labeled datasets lose relevance.

The detection of zero-day anomalies or novel attack classes absent during training remains a significant challenge, as the lack of proper negative training examples limits the effectiveness of existing techniques. Data privacy and security concerns also accompany the direct usage of raw network traffic payloads, which are often unsuitable for sharing.

Ongoing research directions aim to address these challenges through various approaches.

Continual learning workflows that can sustain performance on long-term data streams, transfer learning strategies to adapt models across administrative domains under distribution shifts, and adversarial robustness techniques to enhance the reliability of anomaly detectors are actively being explored. Graph convolutional models that can capture the interdependencies between host behaviors, as well as semi-supervised strategies that can leverage unlabeled data more effectively, are also gaining attention.

Ultimately, the goal of this study is to develop anomaly detection solutions that can be securely deployed at internet scales while maintaining trustworthiness through transparency and explainability. Addressing these challenges will be crucial to realizing practical autonomous anomaly monitoring systems that can enhance the overall cybersecurity resilience of modern network infrastructures.

CHAPTER THREE

Methodology

3.1. Introduction

The proposed study aims to develop an effective anomaly detection model for identifying abnormal patterns in network traffic data. Anomaly detection is a crucial task in cybersecurity, as it enables the identification of potentially malicious or unusual activities within a network infrastructure. By accurately detecting anomalies, network administrators can proactively mitigate security threats and maintain the integrity of their digital systems.

The primary objective of this research is to investigate the use of advanced machine learning techniques, including both supervised and unsupervised approaches, to improve the detection of anomalies in network traffic data. The study will leverage a comprehensive set of features extracted from the network traffic data to train the anomaly detection models.

These features are specifically chosen to capture various aspects of network behavior, including user profiles, traffic patterns, protocol characteristics, and other relevant indicators of normal and abnormal activities. The feature set is designed to provide a detailed representation of the network traffic, allowing the machine learning models to learn the underlying patterns and effectively distinguish anomalous events from normal network operations.

The proposed methodology involves a systematic approach to data collection, preprocessing, model development, and evaluation. The study will utilize real-world network traffic data, either from publicly available datasets or through direct network monitoring and packet capture, to ensure the relevance and applicability of the findings.

The data preprocessing stage involved feature engineering, data normalization, and the handling of any missing or noisy data elements. This step is crucial for ensuring the quality and reliability of the input data for the subsequent modeling process.

The study explored a range of machine learning algorithms, both supervised and unsupervised, to develop the anomaly detection models. This includes techniques such as supervised classification, one-class classification, and unsupervised anomaly detection methods.

The performance of the developed models is evaluated using appropriate metrics, such as accuracy, precision, recall, and the area under the receiver operating characteristic (ROC) curve. The evaluation process considered the timeliness and real-time detection capabilities of the anomaly detection solutions, as these are critical factors for effective security incident response. By documenting the comprehensive methodology and the rationale behind the design choices, this chapter aims to provide a transparent and replicable framework for the research study. The detailed description of the data, feature engineering, model development, and evaluation procedures can enable the reproduction of the study and facilitate further research in the field of network traffic anomaly detection.

3.2. Methodology

The methodology in a machine learning thesis typically involves several steps, including data collection, data preprocessing, feature engineering, model selection, model training, and model evaluation. The proposed methodology consists of data preprocessing, feature selection, models training and model evaluation. The overview of the methodology is shown in Figure 3.1. Below we detail each step in the methodology.

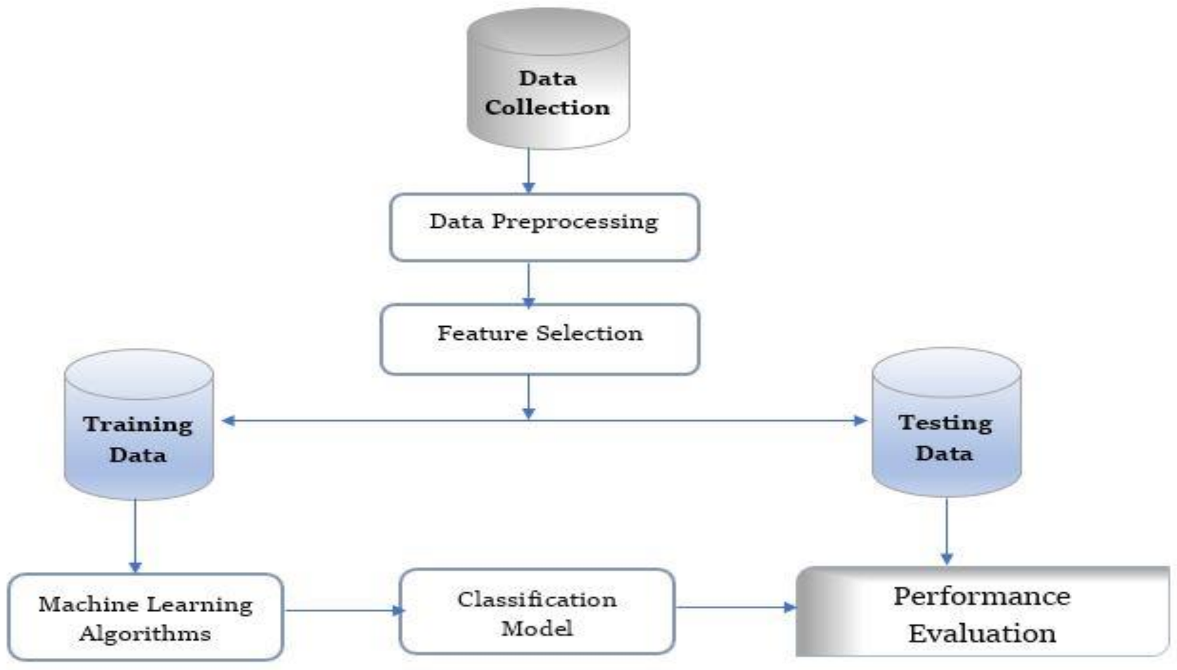

Figure 3-1: Methodology of the study

3.2.1. Data Collection

The network traffic data used in this study is obtained from publicly available datasets. The data collection process aimed to ensure the relevance, diversity, and representativeness of the network traffic patterns to be analyzed.

Special considerations were given to data privacy, ethical guidelines, and any necessary permissions or approvals required for the use of network traffic data in this study.

NSL-KDD (Network Security Laboratory - Knowledge Discovery and Data Mining) and UNSW- NB15 (University of New South Wales - Network-Based 25 features) are two widely used standardized public datasets for intrusion detection research. The network intrusion detection dataset from NSL-KDD containing 25,000 records with 25 features is used for this study. The dataset is splited into 70% (17,500 records) for training and 30% (7,500 records) for testing.

3.2.2. Data Preprocessing

In machine learning, data preprocessing is a crucial step that involves cleaning, transforming, and preparing raw data before it is used to train a machine learning model. The goal of data preprocessing in ML is to ensure that the data is in a format that can be easily understood by the machine learning algorithms, and to improve the accuracy and reliability of the resulting model.

Data preprocessing in ML involves several techniques such as data cleaning, data transformation, data normalization, data encoding, and feature selection. These techniques are used to remove any inconsistencies, errors, or missing values from the data, convert categorical variables into numerical ones, scale the data to a common range, and select the most relevant features for the model.

In the next chapter detailed data preprocessing techniques and model implementations described.

3.2.3. Model Training

Model training is the process of training a machine learning model on a dataset to learn patterns and relationships between the input features and the output variable. The goal of model training is to build a model that can accurately predict the output variable for new, unseen data.

To train a machine learning model, we typically split the dataset into two subsets: a training set and a testing set. The training set is used to train the model, while the testing set is used to evaluate the performance of the model on new, unseen data.

```
# split dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

In this way, we used the `train_test_split()` function from the `sklearn.model_selection` module to split the dataset into input features (`X`) and output variable (`y`). We then specify a test size of 0.3, which means that 30% of the data will be used as the testing set. Finally, we use the `random_state` parameter to ensure that the data is split in a reproducible way.

Once we have split the data into training and testing sets, we trained the machine learning model on the training set using various algorithms such as auto encoders, random forests and one-class SVMs in a cooperative multi-stage architecture leveraging their collective abilities for representation learning, outlier detection and global modeling. We then evaluated the performance of the model on the testing set by making detections on the testing set and comparing them with the actual values of the output variable.

By doing so, we determined how well the model generalizes to new, unseen data.

3.2.4. Model building

Model building is the process of creating a machine learning model using a given algorithm and a dataset. The goal of model building is to create a model that can accurately detect the output variable for new, unseen data.

To build a model we used the algorithms mentioned earlier (auto encoder, Random Forests, one class SVM), we typically follow these steps:

1. Import the necessary libraries: We imported the libraries required to build the model, such as `numpy`, `pandas`, and `sklearn`.
2. Load the dataset: We loaded the dataset into memory, either by reading it from a file or generating it programmatically.
3. Split the dataset: We splited the dataset into training and testing sets using `train_test_split()` function from `sklearn.model_selection`.
4. Preprocess the data: We preprocessed the data by performing tasks such as data cleaning and feature selection.
5. Train the model: We trained the machine learning model using the training set and the chosen algorithm. We used the `fit()` method of the algorithm class to train the model.
6. Evaluate the model: We evaluated the performance of the model on the testing set using

various metrics such as accuracy, precision, recall, and F1 score. We used the `score()` method of the algorithm class to evaluate the model.

3.3. Tools used for the proposed study

Jupyter Notebook with Python programming is an open-source powerful tool for developing several most widely used machine learning algorithms and contains tools for data preprocessing, classification, random forest, clustering, association rules, and visualization [37]. Python is user-friendly and can detect various hidden patterns in the dataset. So, in this study, we used Jupyter Notebook with Python programming language to conduct the experiment.

3.4. Types of Machine Learnings Used

From the machine learning the classification algorithms: support vector machine (SVM), random forest, and auto encoder from unsupervised algorithms are selected for the experiment.

3.4.1. Auto encoder

In this study, we used autoencoders for building efficient latent representations of normal network traffic as the first stage of a novel hybrid intrusion detection system. Autoencoders are an unsupervised learning technique well-suited for dimensionality reduction and feature extraction tasks. By training autoencoders on unlabeled network flow data, we developed models that can encode normal traffic patterns into a compact latent space, while anomalous inputs are not reconstruct accurately. These trained encoders generated encoded outputs of new traffic that can then analyzed by additional machine learning techniques like random forests and one-class SVMs to provide detection capabilities.

3.4.2. Support vector machine (SVM)

As part of this study developing a novel hybrid intrusion detection system, we used to investigate the use of support vector machines to provide a global modeling stage. One-class SVMs excel at determining what constitutes normal behavior and detecting novel outliers.

We intended to train a one-class SVM on the latent representations of normal network traffic generated by an autoencoder. This SVM learned a boundary description of normal data and able to detect intrusions outside of this boundary when analyzing encoded outputs of new traffic. And evaluated if a one-class SVM, when combined with autoencoder feature encoding and random forest detection, can provide an accurate global detection model capable of finding sophisticated, unknown intrusion attempts. The one-class SVM has potential for strengthening the overall framework's abilities to uncover novel network threats.

3.4.3. Random forest

Random forests are an effective ensemble learning technique that combines the predictions of many decision trees built on random subsets of the training data and features.

We used to train a random forest on the latent representations output by the autoencoder to identify network traffic patterns that do not conform to normal behavior. By incorporating randomness into the forest construction, random forests have strong capabilities for detecting unknown anomalies within high-dimensional data like encoded network flows. This study assessed if a random forest analyzer, when combined with the feature encoding from autoencoders and global modeling of a one-class SVM, can significantly improve the accuracy and explainability of detected intrusions. The random forest has promise for strengthening anomaly identification within the overall proposed cooperative framework.

3.5. Evaluation Metrics

Performance evaluation metrics are used to determine how effectively a machine learning model performed with the test data was given. A learning model's fundamental purpose is to generalize successfully on data that has never been seen before. On specific learning models, specific metrics must be utilized, and not all metrics may be employed in a single model. To compute the metrics, we used a confusion matrix as shown in table below.

Table 3-1: Confusion matrix

Actual Value

Predicted value

	Defaulter subscribers (Bad)	**Non-Defaulter subscribers(Good)**
Defaulter subscribers (Bad)	**TP**	**FP**
Non-Defaulter subscribers(Good)	**FN**	**TN**

In the confusion matrix:

- True Positive (TP) refers to defaulters who are predicted as defaulters.

- True Negative (TN) refers to non-defaulters who are predicted as non-defaulters.
- False Positive (FP) refers to non-defaulters who are predicted as defaulters.
- False Negative (FN) refers to defaulters who are predicted as non-defaulters.

I. *Accuracy*

Accuracy is calculated by dividing the correct predictions to the overall prediction value. (i.e.)

$\textbf{Accuracy} = \frac{\mathrm{TN+TP}}{\mathrm{TP+FP+TN+FN}}$ -------------------------------------(1)

II.*Precision*

Precision is the ratio of correct prediction to the sum of true and false positive prediction (i.e.) $\textbf{Precision} = \frac{\mathrm{TP}}{\mathrm{TP+FP}}$ ---------(2)

III. *Recall*

Recall is the ratio of correct prediction to the sum of true positive and false negative prediction (i.e.)

$\textbf{Recall} = \frac{\mathrm{TP}}{\mathrm{TP+FN}}$ --- (3)

IV.*F-Measure*

The F-score is a way of combining the precision and recall of the model. It is calculated as the harmonic mean of precision and recall (i.e.)

$\textbf{F1} - \textbf{Measure} = 2 * ((\text{precision} * \text{recall})/(\text{precision} + \text{recall}))$ --
--(4)

3.6. Pseudocode

Machine learning based Network Anomaly Detection

INPUTS: Datasets, machine learning models

OUTPUT: machine learning based Anomaly Detector Model

Steps:

```
1: Read traffic going
through the middle box
2: Apply machine learning
model
3: Train the machine learning model to detect
normal/abnormal traffic trend
4: **if** trained machine learning model is available **then**
5:     Test the traffic
6:     **if** traffic trend is abnormal **then**
7:         Block traffic
8:     **else**
9:         Allow traffic
10:    **end if**
11: **else**
12: Wait for creating a trained machine learning mod
13: **end if**
```

CHAPTER FOUR

Experiment

This chapter investigates the effectiveness of the proposed machine learning models for binary anomaly detection in network traffic using a reduced feature set. The primary goal is to evaluate the performance of these algorithms when using only five selected features as input and compare this performance to using the full feature set of the KDDCUP99 dataset. The anticipated outcome is a demonstration that the selected five features are sufficient for the models to distinguish between normal and abnormal traffic with a significant reduction in processing time, averaging 58%.

The chapter is structured into three sections: Experimentation, Discussion and Findings, and Conclusion. The Experimentation section presents the results of applying the machine learning models to the dataset using the reduced feature set. The Discussion and Findings section compares the performance observed when using the five selected features versus all features. Finally, the Conclusion section summarizes the key findings of the chapter.

4.1. Selected Features

Network monitors are essential tools for capturing and analyzing network traffic data, especially during network failures or security incidents. Various network monitors exist, each with unique functionalities and capabilities [54][55]. Wire shark, for instance, is a widely used open-source network protocol analyzer that provides real-time traffic capture and analysis. NetFlow, a Cisco-developed protocol, collects and analyzes IP traffic data for network accounting, monitoring, and analysis purposes.

Most network monitors gather data from network devices like switches, routers, or dedicated collection devices [51]. This collected data undergoes analysis and reporting, furnishing network administrators with valuable insights into network activity, performance, and security posture.

Network monitoring commonly employs a flow-based approach. A traffic flow is characterized by a five-tuple consisting of source IP, source port, destination IP, destination port, and protocol [13][47].

This five-tuple serves as the foundation for defining a minimal feature list readily capturable by most network monitoring tools.

The minimal feature list comprises the following:

- Attack: A binary label indicating the presence (1) or absence (0) of an attack.
- Protocol type: The network protocol used for the connection (e.g., TCP, UDP, ICMP).
- Application: The application or service associated with the connection (e.g., http, ftp, smtp, telnet, etc.).
- length: The total number of bytes transmitted or received during the connection.
- Count: The number of connections to the same destination IP address within the preceding 2 seconds.
- srv_count: The number of connections to the same destination port number within the preceding 2 seconds.

Table 4-1: Sample values for the selected features

Parameter	**Sample Value**
Attack	0
protocol_type	ICMP
Service	MQTT
Length	466
Count	13
srv_count	599

Table 4-2: Sample values for the selected features

Feature	**Traffic Flow**
protocol_type	Protocol
Service	Source/Destination Port
Count	Source/Destination IP
srv_count	Source/Destination Port

The inclusion of the "length" feature in the minimal feature list is justified by its significance in anomaly detection.

Packet length, representing the amount of traffic exchanged between devices, provides valuable insights into potential anomalies [52].

This information is readily accessible and analyzable through common network monitoring tools like Wireshark and NetFlow, which routinely capture and process packet size data as part of their network traffic analysis.

4.2. Experimentation

The four machine learning algorithms were tested to classify normal and abnormal traffic on the trimmed KDDCUP99 dataset. Here are the results for each machine learning algorithm:

To assess the performance of the proposed model, we conducted experiments based on the architecture provided in Figure 4.1 bellow.

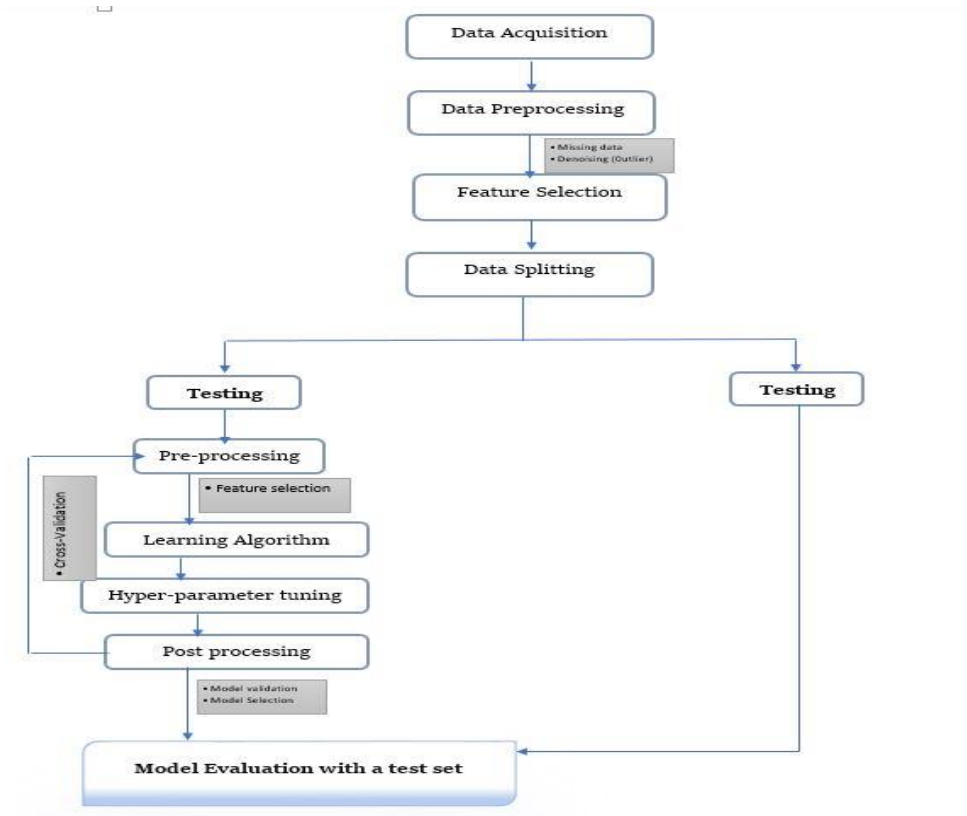

Figure 4-1: Architecture of the proposed model.

4.2.1. Hybrid Model

Table 4-3, displaying the results of the Hybrid Model tested with the five selected features, indicates an improvement in both accuracy and precision after the input size reduction. This suggests that using the selected features not only simplifies the model but also enhances its performance, potentially by reducing noise or focusing on the most relevant information for classification.

Table 4-3: Hybrid model Results.

Metric	Before	After
Accuracy	0.9614	0.9862
F1-Score	0.9604	0.9858
Precision	0.9628	0.9862
Recall	0.9614	0.9862

4.2.2. Random forest Model

Random forest Model was tested using the five selected features as input. The results are shown in Table (4-4). The results show that the Random Forest Model achieved a similar accuracy and precision before and after the input size reduction.

Table 4-4: Random forest Results.

Metric	Before	After
Accuracy	0.9997	0.9994
F1-Score	0.9997	0.9994
Precision	0.9997	0.9994
Recall	0.9997	0.999

4.2.3. Support Vector Machine Model

Table 4-5 presents the results of testing the SVM model on the IoT dataset using the five selected features. The results indicate that the SVM model maintains comparable accuracy and precision both before and after the input size reduction. This suggests that the five selected features are sufficient for the SVM model to effectively classify the data, and the removal of other features does not significantly impact its performance.

Table 4-5: SVM Classification Results.

Metric	Before	After
Accuracy	0.9943	0.9944
F1-Score	0.9942	0.9944
Precision	0.9942	0.9944
Recall	0.9943	0.9944

4.2.4. Auto encoder Model

Table 4-6, which presents the results of testing the Autoencoder model with the five selected features, demonstrates that the model maintains consistent accuracy and precision levels compared to when all features were used. This suggests that the Autoencoder effectively leverages the essential information contained within the reduced feature set without significant performance loss.

Table 4-6: Autoencoder Model Results.

Metric	Before	After
Accuracy	0.9987	0.9983
F1-Score	0.9987	0.9983
Precision	0.9987	0.9983
Recall	0.9987	0.9983

4.3. Findings

Comparing the results from using all features and using five selected features protocol_type, Application, length, count, and srv_count) reveals consistent performance for the SVM, Random Forest, and Autoencoder models. However, the Hybrid model exhibits a notable performance improvement when using the reduced feature set.

Figure 4-2 visually confirms this observation. The accuracy difference between the two feature sets is negligible (less than 0.04%) for the Random Forest, SVM, and Autoencoder models. Conversely, the Hybrid model shows a marked accuracy increase of 2.5796% with the five selected features. This suggests that the Hybrid model benefits from the reduced dimensionality and potentially from the specific features chosen, unlike the other models which maintain similar performance levels regardless of the number of input features.

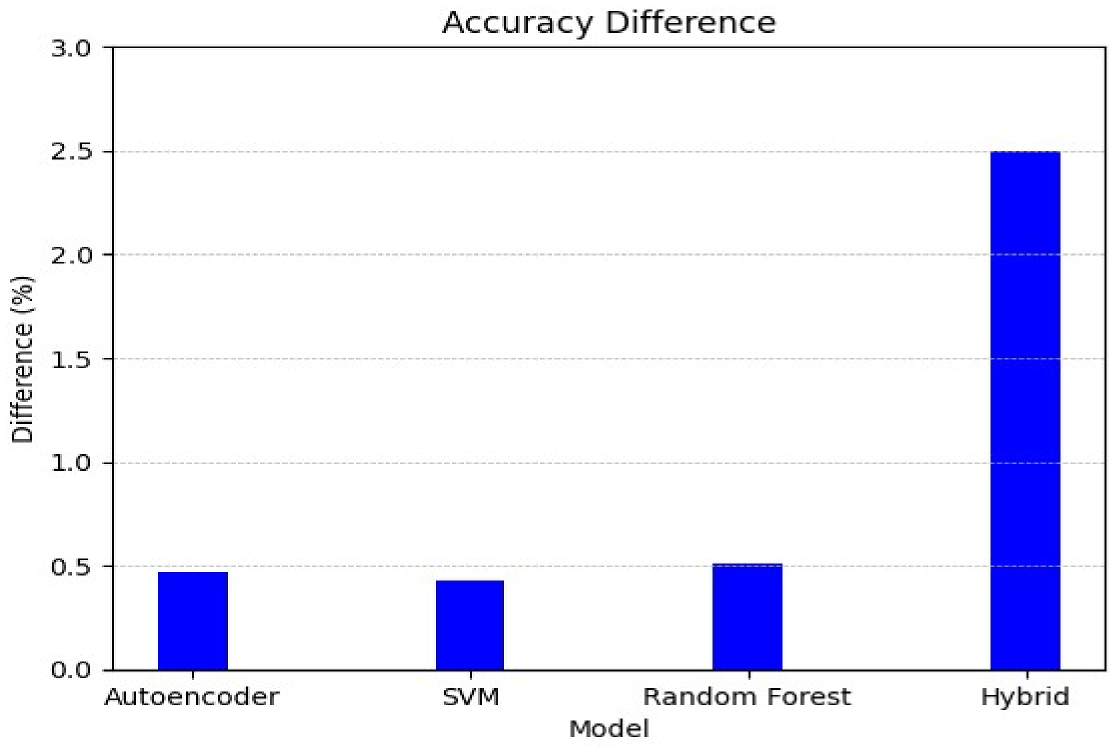

Figure 4-2: The Difference in Accuracy between using all features and the five chosen features.

Figure 4-3 illustrates that the difference in F1-score between utilizing all features versus the selected five features is less than 0.04% for the Random Forest, SVM, and Autoencoder models. The Hybrid model, however, shows a more significant difference of 2.6490%. This indicates that the Hybrid model's F1-score is more affected by the choice of input features compared to the other three models, which maintain relatively consistent performance regardless of the feature set used.

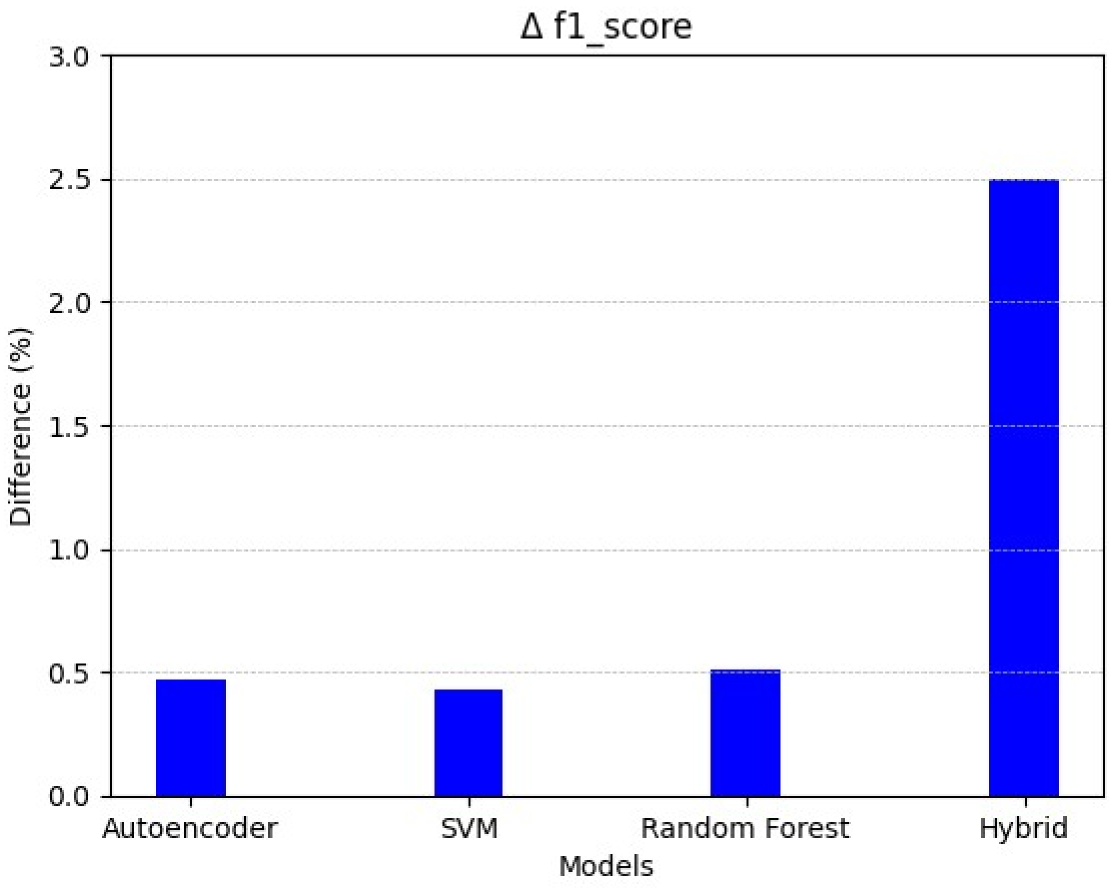

Figure 4-3: The Difference in F1 between using all features and the five chosen features.

Figure 4-4 demonstrates that the accuracy difference between using all features and the selected five features is minimal (less than 0.04%) for the SVM, Random Forest, and Autoencoder models. In contrast, the Hybrid model exhibits a more noticeable accuracy difference of 2.24250%. This highlights the Hybrid model's greater sensitivity to the specific features used compared to the other models.

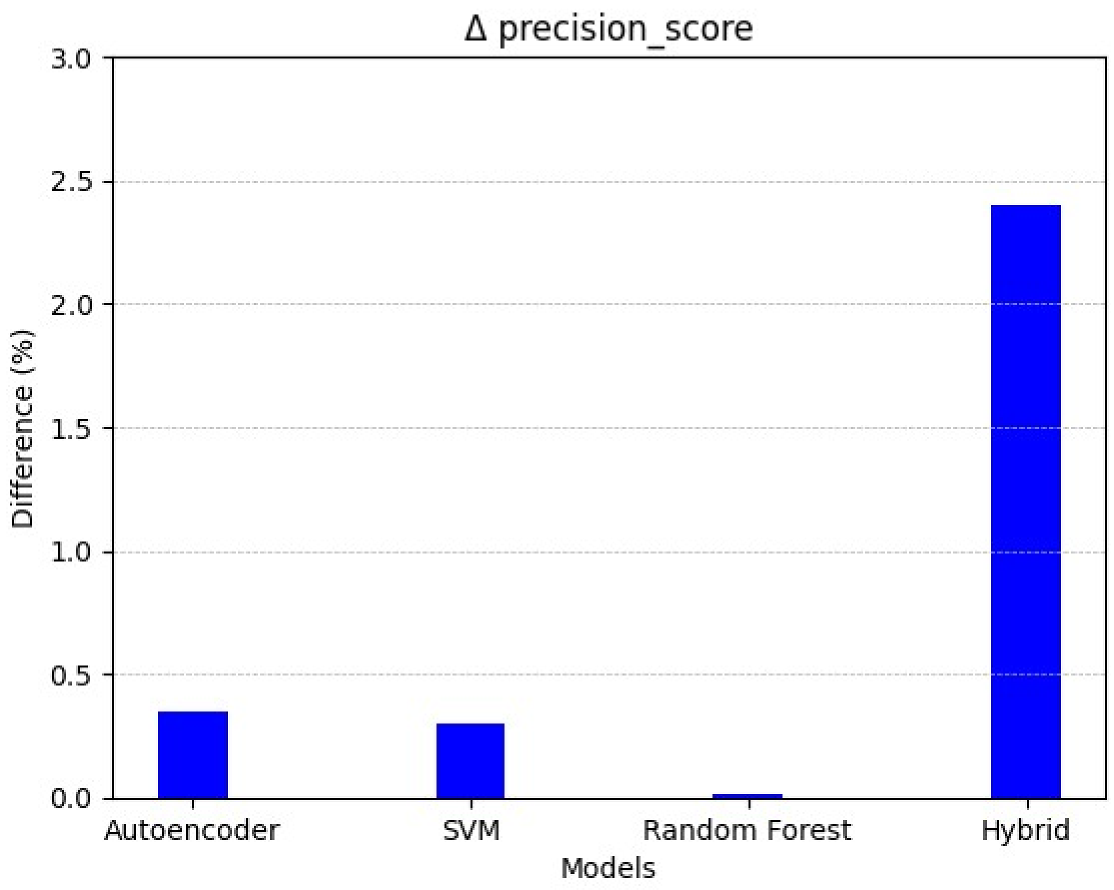

Figure 4-4: The Difference in precision score between using all features and the five chosen features.

Figure 4-5 reveals minimal accuracy differences (less than 0.04%) between using all features and the selected five features for the SVM, Random Forest, and Autoencoder models. However, a more pronounced difference of 2.57963% was observed for the Hybrid model. This suggests that the Hybrid model's performance is more sensitive to the choice of input features compared to the other three models.

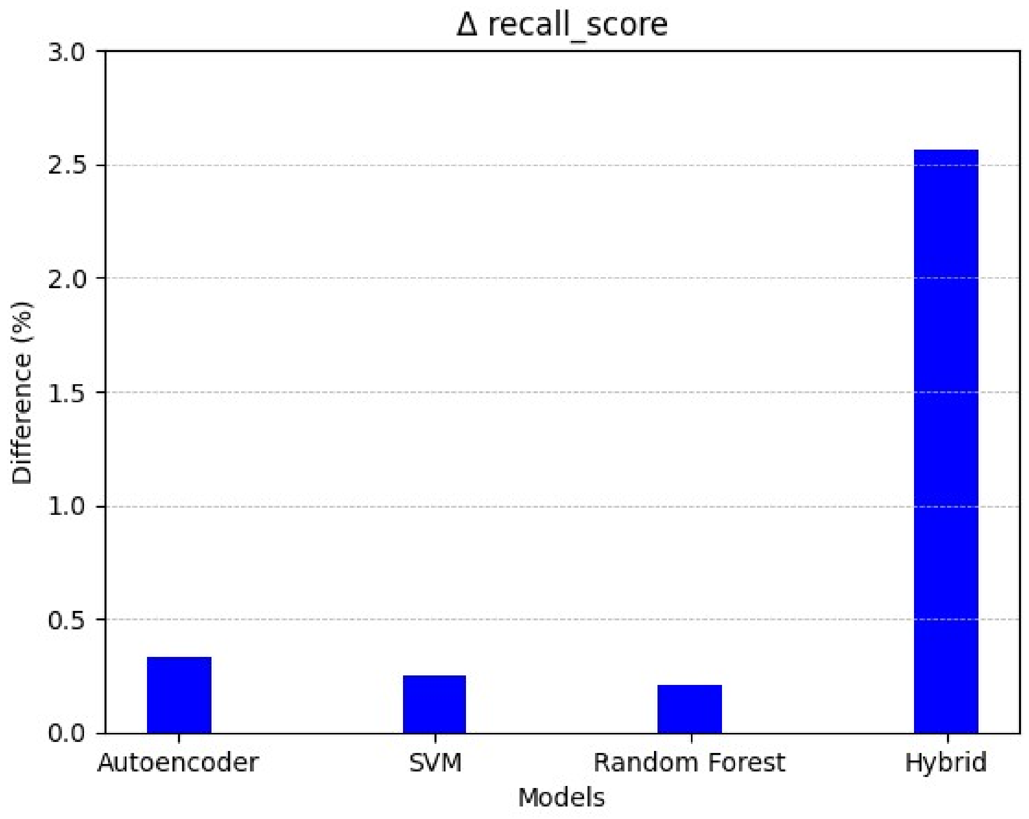

Figure 4-5: The difference in recall between using all features and the five chosen features.

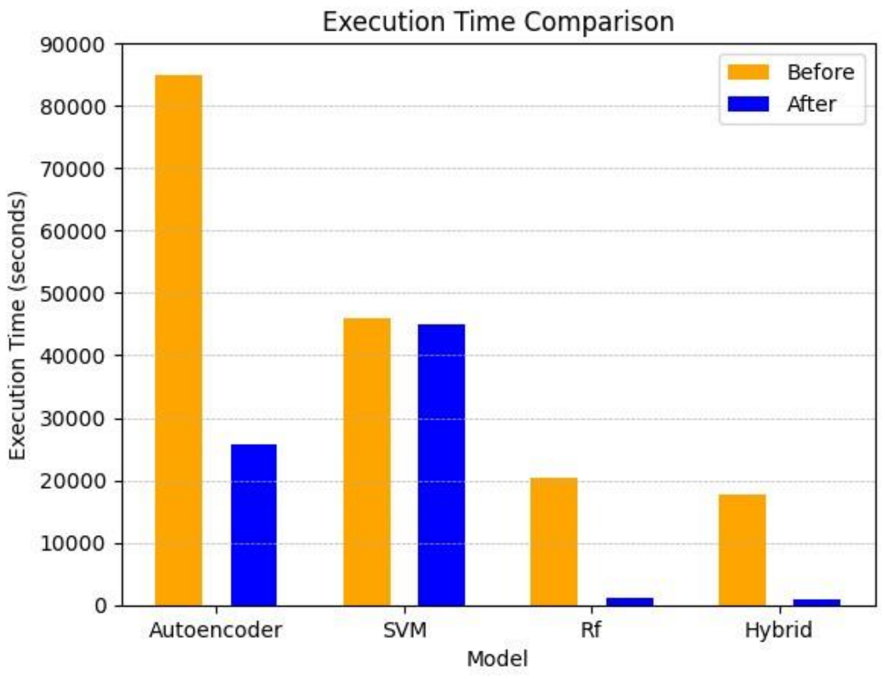

Figure 4-6: Execution time using all features and the five chosen features.

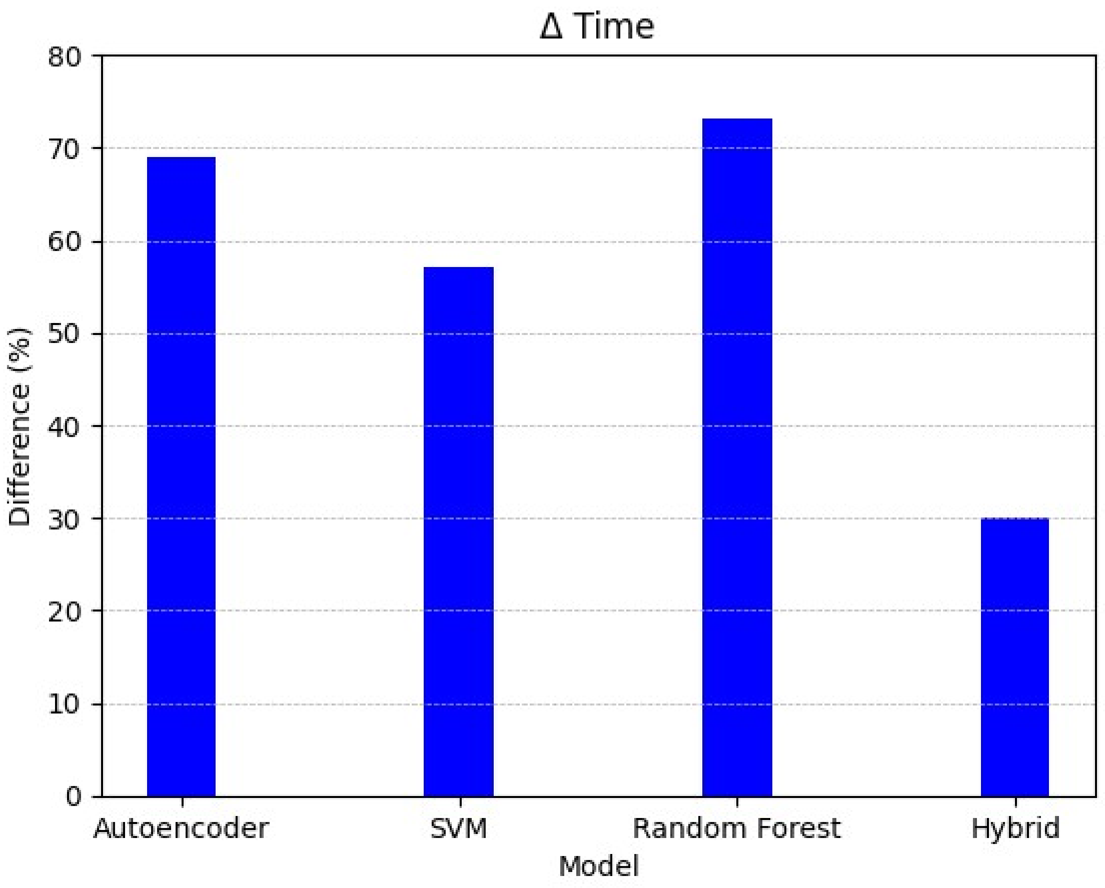

Figure 4-7: The difference in time between using all features and the five chosen features.

Figures 4-6 and 4-7 illustrate the impact of input feature reduction on execution time. Specifically, using only five key features instead of the full KDDCUP99 dataset resulted in substantial time savings during the identification of test sample traffic types: over 70% for the Autoencoder, 56% for the SVM, 75% for the Random Forest, and 30% for the Hybrid model, averaging a 58% reduction across all four models.

These findings suggest the following:

Consistent Performance with Reduced Features: The accuracy, precision, and F1-score for the Autoencoder, SVM, and Random Forest models remained largely unchanged when using the reduced feature set compared to the full KDDCUP99 dataset.

Autoencoder Robustness: The Autoencoder's consistent performance is attributed to its ability to emphasize relevant features and suppress irrelevant features (noise) during the learning process [55][56][57]. Therefore, removing irrelevant features had minimal impact.

SVM Performance with Reduced Dimensionality: While SVMs generally perform well with high-dimensional data (as observed in Chapter 3), reducing irrelevant features did not negatively affect performance and potentially even improved it by facilitating the identification of a more effective separation boundary [58][59].

Random Forest Resilience to Noise: The Random Forest model also maintained consistent performance. This robustness stems from its ensemble nature, making it less susceptible to noise unless critical features within the decision trees are removed.

Improved Hybrid Model Performance: Contrary to the performance drop observed in Chapter 3, the Hybrid model performed better with the reduced feature set. This improvement is likely due to the lower inter-correlation among the five selected KDDCUP99 features, as Hybrid models generally benefit from feature independence.

Execution Time Differences: Significant differences in execution time were observed between the Autoencoder, SVM, and Random Forest models.

Impact of Reduced Features on Training Time: Using fewer input features expedited training across all models. For Autoencoders, this is due to a reduced number of parameters and computations. For SVMs, fewer features simplify the search for the optimal hyperplane in a lower-dimensional space. For Random Forests, fewer features translate to fewer rules for the decision trees to learn.

Hybrid Model Training Efficiency: The Hybrid model generally exhibited faster training times compared to the other models, likely due to its combined approach leveraging the strengths of the component models.

4.4. Summary

This chapter evaluated the efficacy of four machine learning models—Autoencoder, Support Vector Machine (SVM), Random Forest, and a Hybrid model—for binary anomaly detection using the KDDCUP99 dataset. Performance was compared using two input scenarios: one incorporating all available features and another utilizing a select subset of five features (protocol type, service type, total length, count of connections from the same IP, and number of connections from the same destination port). Results indicate that all models successfully differentiated between normal and anomalous network traffic in both scenarios.

While the Autoencoder, SVM, and Random Forest models exhibited consistent performance regardless of input features, the Hybrid model demonstrated improved performance when using the reduced feature set. This targeted approach yielded a significant average reduction of 58% in processing time, suggesting that leveraging these five key features allows for efficient and effective anomaly detection with the KDDCUP99 dataset.

CHAPTER FIVE

Conclusion and Recommendation

5.1. Conclusion

Network technology's ability to connect virtually everything—people, devices, and physical objects—to the internet has profoundly impacted society. However, the projected growth of networked devices, particularly within the Internet of Things (IoT), presents significant information security challenges. This thesis examined these challenges and identified key gaps in securing IoT networks. The primary challenges include the sheer scale of these networks, the massive number of devices involved, the human element, and the inherent complexity of IoT architectures. Furthermore, this work highlighted critical research gaps, specifically the need for:

- Automated intrusion detection systems for anomaly detection.
- Refined models for determining essential input parameters for network anomaly detection.
- High-performance machine learning models capable of automated anomaly detection in network environments.

This thesis investigated the application of four machine learning models—Autoencoder, Support Vector Machine (SVM), Random Forest, and a Hybrid model—for binary anomaly classification using the KDDCUP99 dataset. A key finding was the superior performance of the Hybrid model, which combined elements of the other three algorithms. While the Autoencoder, SVM, and Random Forest demonstrated comparable performance, the Hybrid model achieved improved results across multiple metrics, including accuracy, precision, execution time, and recall.

5.2. Recommendations

This research provides a foundation for future investigations. While this work focused on several popular machine learning algorithms, further exploration with other classification models is recommended. This includes evaluating the performance of Linear Classifiers, Logistic Regression, Perceptron, Quadratic Classifiers, K-Means Clustering, Boosting, and Decision Tree (DT) algorithms for anomaly detection. Additionally, expanding the scope of this research by incorporating additional datasets, such as NSL-KDD, UKM-IDS20, and other real-world network datasets, would provide a more comprehensive evaluation of the proposed methods and enhance the generalizability of the findings.

6. References

[1] M. Ajdani and H. Ghaffary, "Introduced a new method for enhancement of intrusion detection with random forest and PSO algorithm," “Secur. Priv.”, vol. 4, no. 2, p. e147, 2021.

[2] M. Antonakakis et al., "Understanding the Mirai botnet," in “26th USENIX Security Symposium (USENIX Security 17)”, 2017, pp. 1093–1110.

[3] H. Bahşi, S. Nõmm, and F. B. La Torre, "Dimensionality reduction for machine learning based IoT botnet detection," in “2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)”, 2018, pp. 1857–1862.

[4] P. Chauhan and M. Atulkar, "Selection of tree based ensemble classifier for detecting network attacks in IoT," in “2021 International Conference on Emerging Smart Computing and Informatics (ESCI)”, 2021, pp. 770–775.

[5] S. S. Chawathe, "Monitoring IoT networks for botnet activity," in “2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)”, 2018, pp. 1–8.

[6] S. Dange and M. Chatterjee, "IoT botnet: The largest threat to the IoT network," in “Data Communication and Networks”, Springer, 2020, pp. 137–157.

[7] M. Ge et al., "Towards a deep learning-driven intrusion detection approach for internet of things," “Comput. Netw.”, vol. 186, p. 107784, 2021.

[8] T. T. Huong et al., "Lockedge: Low-complexity cyberattack detection in IoT edge computing," “IEEE Access”, vol. 9, pp. 29696–29710, 2021.

[9] T. T. Huong et al., "An efficient low complexity edge-cloud framework for security in IoT networks," in “2020 IEEE Eighth International Conference on Communications and Electronics (ICCE)”, 2021, pp. 533–553.

[10] I. Idrissi et al., "Toward a deep learning-based intrusion detection system for IoT against botnet attacks," “IAES Int. J. Artif. Intell.”, vol. 10, no. 1, p. 110, 2021.

[11] M. Injadat, A. Moubayed, and A. Shami, "Detecting botnet attacks in IoT environments: An optimized machine learning approach," in “2020 32nd International Conference on Microelectronics (ICM)”, 2020, pp. 1–4.

[12] P. Ioulianou et al., "A signature-based intrusion detection system for the internet of things," “Inf. Commun. Technol. Form”, 2018.

[13] M. J. Kang and J. W. Kang, "Intrusion detection system using deep neural network for in-vehicle network security," “PLOS ONE”, vol. 11, no. 6, p. e0155781, 2016.

[14] N. Koroniotis et al., "Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-IoT dataset," “Future Gener. Comput. Syst.”, vol. 100, pp. 779–796, 2019.

[15] P. Kumar, G. P. Gupta, and R. Tripathi, "Toward design of an intelligent cyber attack detection system using hybrid feature reduced approach for IoT networks," “Arab. J. Sci. Eng.”, vol. 46, no. 4, pp. 3749–3778, 2021.

[16] S. Kumar et al., "Cybersecurity measures for geocasting in vehicular cyber physical system environments," “IEEE Internet Things J.”, vol. 6, no. 4, pp. 5916–5926, 2019.

[17] Y. N. Kunang et al., "Improving classification attacks in IoT intrusion detection system using Bayesian hyperparameter optimization," in “2020 3rd International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)”, 2020, pp. 146–151.

[18] S. Lee et al., "Classification of botnet attacks in IoT smart factory using honeypot combined with machine learning," “PeerJ Comput. Sci.”, vol. 7, p. e350, 2021.

[19] O. Linda, T. Vollmer, and M. Manic, "Neural network based intrusion detection system for critical infrastructures," in “2009 International Joint Conference on Neural Networks”, 2009, pp. 1827–1834.

[20] C. D. McDermott, A. V. Petrovski, and F. Majdani, "Towards situational awareness of botnet activity in the internet of things," “Inst. Electr. Electron. Eng.”, 2018.

[21] Y. Meidan et al., "N-Baiot-network-based detection of IoT botnet attacks using deep autoencoders," “IEEE Pervasive Comput.”, vol. 17, no. 3, pp. 12–22, 2018.

[22] M. Mohammadi et al., "A comprehensive survey and taxonomy of the SVM-based intrusion detection systems," “J. Netw. Comput. Appl.”, p. 102983, 2021.

[23] M. Odusami et al., "An improved model for alleviating layer seven distributed denial of service intrusion on webserver," “J. Phys.: Conf. Ser.”, vol. 1235, IOP Publishing, 2019.

[24] S. I. Popoola et al., "Stacked recurrent neural network for botnet detection in smart homes," “Comput. Electr. Eng.”, vol. 92, p. 107039, 2021.

[25] R. Rani, S. Kumar, and U. Dohare, "Trust evaluation for light weight security in sensor enabled internet of things: Game theory oriented approach," “IEEE Internet Things J.”, vol. 6, no. 5, pp. 8421–8432, 2019.

[26] S. M. Sajjad and M. Yousaf, "Ucam: Usage, communication and access monitoring based detection system for IoT botnets," in “2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE)”, 2018, pp. 1547–1550.“Removed duplicate entry”

[27] R. Samdekar, S. Ghosh, and K. Srinivas, "Efficiency enhancement of intrusion detection in IoT based on machine learning through bioinspire," in “2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)”, 2021, pp. 383–387.

[28] M. Shafiq et al., "Selection of effective machine learning algorithm and bot-IoT attacks traffic identification for internet of things in smart city," “Future Gener. Comput. Syst.”, vol. 107, pp. 433–442, 2020.

[29] M. Singh, M. Singh, and S. Kaur, "Detecting bot-infected machines using DNS fingerprinting," “Digit. Investig.”, vol. 28, pp. 14–33, 2019.

[30] S. Sriram et al., "Network flow based IoT botnet attack detection using deep learning," in “IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)”, 2020, pp. 189–194.

[31] Sudhakar and S. Kumar, "Botnet detection techniques and research challenges," in “2019 International Conference on Recent Advances in Energy-Efficient Computing and Communication (ICRAECC)”, 2019, pp. 1–6.

[32] Sudhakar and S. Kumar, "An emerging threat fileless malware: A survey and research challenges," “Cybersecurity”, vol. 3, no. 1, pp. 1–12, 2020.

[33] P. Sun et al., "Modeling and clustering attacker activities in IoT through machine learning techniques," “Inf. Sci.”, vol. 479, pp. 456–471, 2019.

[34] H. Tyagi and R. Kumar, "Attack and anomaly detection in IoT networks using supervised machine learning approaches," “Rev. d’Intell. Artif.”, vol. 35, no. 1, pp. 11–21, 2021.

[35] J. David and C. Thomas, "Discriminating flash crowds from DDoS attacks using efficient thresholding algorithm," J. Parallel Distrib. Comput., vol. 152, pp. 79-87, 2021.

[36] A. Callado et al., "A Survey on Internet Traffic Identification," IEEE Commun. Surv. Tutor., vol. 11, no. 3, pp. 37-52, 2009.

[37] K. Sethi et al., "Attention based multi-agent intrusion detection systems using reinforcement learning," J. Inf. Secur. Appl., vol. 61, p. 102923, 2021.

[38] H. Jia et al., "Network intrusion detection based on IE-DBN model," Comput. Commun., vol. 178, pp. 131-140, 2021.

[39] B. B. Borisenko et al., "Intrusion detection using multilayer perceptron and neural networks with long short-term memory," in 2021 Syst. Signal Synchronization Generating Process. Telecommun. (SYNCHROINFO), 2021, pp. 1-6.

[40] L. Liu et al., "Intrusion detection of imbalanced network traffic based on machine learning and deep learning," IEEE Access, vol. 9, pp. 7550-7563, 2020.

[41] S. Aldhaheri et al., "Deepdca: novel network-based detection of iot attacks using artificial immune system," Appl. Sci., vol. 10, no. 6, p. 1909, 2020.

[42] M. A. Ferrag et al., "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," J. Inf. Secur. Appl., vol. 50, p. 102419, 2020.

[43] M. Ge et al., "Towards a deep learning-driven intrusion detection approach for Internet of Things," Comput. Networks, vol. 186, p. 107784, 2021.

[44] J. Malik et al., "Hybrid deep learning: An efficient reconnaissance and surveillance detection mechanism in SDN," IEEE Access, vol. 8, pp. 134695-134706, 2020.

[45] M. Ge et al., "Deep learning-based intrusion detection for IoT networks," in 2019 IEEE 24th Pac. Rim Int. Symp. Dependable Comput. (PRDC), 2019, pp. 256-25609.

[46] N. Chouhan et al., "Network anomaly detection using channel boosted and residual learning based deep convolutional neural network," Appl. Soft Comput., vol. 83, p. 105612, 2019.

[47] M. A. Ferrag and L. Maglaras, "DeepCoin: A novel deep learning and blockchain-based energy exchange framework for smart grids," IEEE Trans. Eng. Manag., vol. 67, no. 4, pp. 1285-1297, 2019.

[48] T. Wu et al., "Intrusion detection system combined enhanced random forest with SMOTE algorithm," EURASIP J. Adv. Signal Process., vol. 2022, no. 1, pp. 1-20, 2022.

[49] F. Li et al., "System statistics learning-based IoT security: Feasibility and suitability," IEEE Internet Things J., vol. 6, no. 4, pp. 6396-6403, 2019.

[50] T. D. Nguyen et al., "DÏoT: A federated self-learning anomaly detection system for IoT," in 2019 IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS), 2019, pp. 756-767.

[51] H. Alaiz-Moreton et al., "Multiclass classification procedure for detecting attacks on MQTT-IoT protocol," Complexity, vol. 2019, 2019.

[52] S. M. Kasongo and Y. Sun, "Performance analysis of intrusion detection systems using a feature selection method on the UNSW-NB15 dataset," J. Big Data, vol. 7, no. 1, pp. 1-20, 2020.

[53] T. Acharya et al., "Efficacy of Machine Learning- Based Classifiers for Binary and Multi-Class Network Intrusion Detection," in 2021 IEEE Int. Conf. Autom. Control Intell. Syst. (I2CACIS), 2021, pp. 402-407.

[54] A. M. Aleesa et al., "Deep-intrusion detection system with enhanced UNSW-NB15 dataset based on deep learning techniques," J. Eng. Sci. Technol., vol. 16, no. 1, pp. 711-727, 2021.

[55] S. M. Kasongo and Y. Sun, "A deep learning method with wrapper based feature extraction for wireless intrusion detection system," Comput. Secur., vol. 92, p. 101752, 2020.

[56] R.-H. Dong, Y.-L. Shui, and Q.-Y. Zhang, "Intrusion Detection Model Based on Feature Selection and Random Forest," Int. J. Netw. Secur., vol. 23, no. 6, pp. 985-996, 2021.

[57] J. O. Mebawondu et al., "Network intrusion detection system using supervised learning paradigm," Sci. Afr., vol. 9, p. e00497, 2020.

[58] Y. Yang et al., "Improving the classification effectiveness of intrusion detection by using improved conditional variational autoencoder and deep neural network," Sensors, vol. 19, no. 11, p. 2528, 2019.

[59] H.-C. Lin et al., "Ensemble Learning for Threat Classification in Network Intrusion Detection on a Security Monitoring System for Renewable Energy," Appl. Sci., vol. 11, no. 23, p. 11283, 2021.

7. Appendix

```
# import/define necessary libraries
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score, recall_score,
f1_score
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler  # Import StandardScaler
from tensorflow import keras # for autoencoder
# Load dataset (assuming a CSV since the provided document doesn't specify file type for
network traffic)
try:
    dataset = pd.read_csv('kddcup.data_10_percent_corrected') # Replace with your actual
dataset file
except FileNotFoundError:
    print("Error: 'kddcup.data_10_percent_corrected' not found. Please provide the correct
dataset file.")
    exit()  # Exit the script if the file is not found
# Preprocessing (adapting to network traffic data)
# Feature selection (reducing to 5 features as per the document)
#  The document mentions reducing features to 5 for network traffic.
#  You'll need to determine the most relevant 5 features based on your specific dataset
#  and domain knowledge.  This is a crucial step.  For demonstration, I'm selecting
#  5 arbitrary features.  REPLACE THESE with your actual chosen features.
features = ['duration', 'src_bytes', 'dst_bytes', 'count', 'srv_count']
X = dataset[features]
y = dataset['normal.']  # Assuming 'normal.' is the target variable indicating normal/anomalous
# Convert string labels to numerical (if necessary)
le = preprocessing.LabelEncoder()
y = le.fit_transform(y)
# Data scaling (important for many ML algorithms)
scaler = StandardScaler()
X = scaler.fit_transform(X)
# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Model Implementation (Hybrid Model as suggested in the document)
# 1. Autoencoder for dimensionality reduction and feature learning
encoding_dim = 3 # Reduced dimensionality
input_data = keras.Input(shape=(X_train.shape[1],))
encoded = keras.layers.Dense(encoding_dim, activation='relu')(input_data)
decoded = keras.layers.Dense(X_train.shape[1], activation='sigmoid')(encoded)
autoencoder = keras.Model(input_data, decoded)
encoder = keras.Model(input_data, encoded)
autoencoder.compile(optimizer='adam', loss='mse') # Mean squared error for reconstruction
loss
autoencoder.fit(X_train, X_train, epochs=50, batch_size=256, shuffle=True) # Adjust epochs
and batch size
X_train_encoded = encoder.predict(X_train)
X_test_encoded = encoder.predict(X_test)
# 2. Random Forest Classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)  # Adjust
```

```
hyperparameters as needed
rf_classifier.fit(X_train_encoded, y_train)
# 3. One-Class SVM (for anomaly detection)
# Train on 'normal' data only
X_train_normal = X_train_encoded[y_train == 0] # Assuming 0 represents 'normal'
ocsvm = SVC(kernel='rbf', nu=0.1) # Adjust nu (anomaly fraction) as needed
ocsvm.fit(X_train_normal)
# Hybrid Model Prediction: Use Random Forest predictions as primary, OCSVM as secondary
check
rf_predictions = rf_classifier.predict(X_test_encoded)
ocsvm_predictions = ocsvm.predict(X_test_encoded) # -1 for anomaly, 1 for normal
hybrid_predictions = rf_predictions.copy()
for i in range(len(X_test_encoded)):
    if ocsvm_predictions[i] == -1:  # If OCSVM flags as anomaly
        hybrid_predictions[i] = 1 # Set to anomalous class (assuming 1 represents anomalous)
# Evaluation
print("Hybrid Model:")
print("Accuracy:", accuracy_score(y_test, hybrid_predictions))
print("Precision:", precision_score(y_test, hybrid_predictions))
print("Recall:", recall_score(y_test, hybrid_predictions))
print("F1-score:", f1_score(y_test, hybrid_predictions))
print("Confusion Matrix:\n", confusion_matrix(y_test, hybrid_predictions))
```