Multi-Paradigm Network Anomaly Identification: Leveraging Supervised, Unsupervised and Hybrid Approaches to Discover Known and Unknown Threats for Enhanced Intrusion Detection

by

Atik Uddola
21241054
MD. Kawsar Habib
21201327
Md. Nafizur Rahman Bhuiya
21201009
Tasmin Ahmed Oni
21201532

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering
Brac University
June 2025

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Atik Uddola 21241054	MD. Kawsar Habib 21201327
Md. Nafizur Rahman Bhuiya 21201009	Tasmin Ahmed Oni 21201532

Approval

The thesis/project titled "Multi-Paradigm Network Anomaly Identification: Leveraging Supervised, Unsupervised and Hybrid Approaches to Discover Known and Unknown Threats for Enhanced Intrusion Detection" submitted by

1. Atik Uddola (21241054)
2. MD. Kawsar Habib (21201327)
3. Md. Nafizur Rahman Bhuiya (21201009)
4. Tasmin Ahmed Oni (21201532)

Of Spring, 2025 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science and Engineering on June 20, 2025.

Examining Committee:

Supervisor:
(Member)

Dr. Muhammad Iqbal Hossain
Associate Professor
Department of Computer Science and Engineering
BRAC University

Co-supervisor::
(Member)

Towshik Anam Taj
Lecturer
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Dr. Sadia Hamid Kazi
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

While network infrastructures grow increasingly complex and expand massively, anomaly detection has become central to ensuring cybersecurity and maintaining operational stability. Traditional and conventional systems struggle to identify new or unknown attack types, making adaptive and intelligent detection essential. This work presents a hybrid approach to network anomaly detection that leverages both supervised and unsupervised machine learning models to address these challenges. The proposed system utilizes a combination of deep learning models, supervised models and unsupervised clustering techniques with extensive preprocessing and class balancing using CTGAN for improved anomaly detection. Experiments were conducted using the UNSW-NB15 dataset, testing various scenarios with different combinations of known and unknown classes. The hybrid algorithm using the CURE-based unsupervised clustering approach achieved a high detection rate across multiple unknown class scenarios, with up to 91.9% detection rate and for known class scenarios up to 99.16% detection rate is obtained which significantly outperformed the conventional models used in real time Intrusion detection.

Keywords: Anomaly; Detection; Network; Intrusion; Hybrid; Clustering; Outlier; Classification; Supervised; Unsupervised; Security

Acknowledgement

Above all, we would like to thank Almighty Allah for giving us the strength, patience and wisdom to complete this thesis work. We are greatly obliged to our supervisor Dr. Muhammad Iqbal Hossain for his continuous support, invaluable suggestions and constant encouragement throughout the period of this research work. His extensive guidance was very useful in our work. We would also like to extend our special thanks to our co-supervisor Towshik Anam Taj who through his endless support, valuable suggestions and vigilant supervision which helped us confront challenges and find ways to polish our work. Lastly, we are greatly indebted to our parents for their unquestioning love, encouragement and prayers which have sustained and motivated us throughout this work.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

$AMEN$ Anomaly detection method based on node neighborhoods

APT Advanced Persistent Threats

AUC Area Under Curve

CNN Convolutional Neural Network

$CTGAN$ Conditional Tabular Generative Adversarial Network

CTI Cyber Threat Information

$CURE$ Clustering Using Representatives

DBN Deep Belief Network

DoS Denial of Service

$EGBAD$ Efficient GAN-based Anomaly Detection

FNN FeedForward Neural Network

GAN Generative Adversarial Network

GBM Gradient Boosting Machines

$kPCA$ Kernel Principal Component Analysis

LOF Local Outlier Factor

$LSTM$ Long Short-Term Memory

$MemAE$ Memory-Augmented Deep Autoencoder

$MTAD\text{-}GAT$ Multivariate Time-series Anomaly Detection via Graph Attention Network

$OCSVM$ One-Class Support Vector Machines

PCA Principal Component Analysis

$R2L$ Remote Local-to-Local

$SCAN$ Structural Clustering Algorithm for Networks

$U2R$ User-to-User Root

Chapter 1

Introduction

In the fast evolving scenario of computer networks, ensuring security and maintaining optimal performance have become one of the main concerns for individuals and organizations. As networks grow in complexity and scale to huge sizes, they become more vulnerable to various forms of attacks and anomalies. This work focuses on the crucial aspect of network anomaly detection. It proposes an effective approach that leverages the combined power of machine learning and deep learning models.

An anomaly in computer networks refers to unexpected or unusual behavior in system activity or network traffic. These anomalies can indicate various issues ranging from hardware malfunctions to security breaches. It can significantly impact the overall performance of the network system. Network anomalies can manifest in several forms where each presents unique challenges for detection and mitigation. These include Volume anomalies where traffic suddenly increases. It often causes network outages or leads to distributed denial of service attacks. Topological anomalies involve changes in the network's structure such as routing problems or unauthorized modifications. Content anomalies happen when harmful data like malicious files or stolen information is found in network traffic. Statistical anomalies occur when traffic patterns deviate from normal levels and behavioral anomalies involve unusual use of network resources. It often points to insider threats or hacked accounts. Common network attacks include denial of service which floods the network with traffic, unauthorized access through vulnerabilities, malicious activity from harmful IP addresses, phishing to steal personal information, malware that damages systems and injection attacks that add harmful code to programs or databases [16] .

As networks continue to grow and scale in complexity, the necessity of robust anomaly detection systems becomes the most important. The goal of this work is to advance the subject of network security by analysing the possibilities of a hybrid algorithm approach. Machine learning techniques are utilised in this approach. Our goal is to create a more reliable, thorough and flexible anomaly detection model so that network administrators and security specialists can use it as a useful tool in their continuous attempts to keep networks safe and effective. Additionally, we aim to detect and classify multi-class anomalies using unsupervised hybrid algorithm.

1.1 Problem Statement

Anomalies are dynamic and ever-changing, making it difficult to identify and categorise them in real-world systems like industrial monitoring, cybersecurity and healthcare. Current binary classification methods cannot generalise to previously unseen or unlabelled anomalies. They are only capable of detecting preset anomalies. Furthermore, this problem has not been sufficiently addressed by the research or application of multi-class anomaly detection techniques. When new abnormalities appear, existing methods necessitate acquiring fresh datasets, retraining and testing, which results in difficult and resource-intensive procedures. The dependence on continuous dataset updates leads to inefficiencies and impairs the ability to spot anomalies in real time. This work aims to design a context-aware, explainable and incrementally adaptive anomaly detection framework that leverages hybrid unsupervised learning and continual learning techniques to maintain high detection accuracy in dynamic, real-world network environments, even under high feature overlap and limited labeled data.

1.2 Research Objective

Detecting anomalies in dynamic systems can be quite difficult, especially when they are new or unlabeled. Current approaches frequently rely significantly on manually categorizing anomalies, retraining models and obtaining new datasets, all of which are time consuming and wasteful. This study aims to create a hybrid unsupervised framework that integrates clustering techniques to effectively perform multiclass network anomaly detection. The framework aims to accurately classify known attack types, detect unknown anomalies not present in the training data and distinguish normal network behavior.

1. **Multiclass Detection**: Developed a robust framework capable of accurately classifying multiple types of network anomalies. Addressed the limitations of supervised and conventional models that fail to detect new, unseen anomalies.
2. **Hybrid Algorithm Approach**: Designed a hybrid unsupervised algorithm that integrates clustering and outlier detection techniques to effectively identify both known and unknown network anomalies as well as normal class while minimizing false positives.
3. **Unknown Attack and normal Class Detection**: Enable the algorithm to detect novel anomalies absent from training data and accurately identify normal network behavior, while overcoming conventional anomaly detection issues like high false positives and poor normal class identification.

By achieving these objectives, the research aims to establish a more efficient and practical solution for anomaly detection in complex and dynamic systems.

1.3 Thesis Structure

The thesis is structured to address the challenge of detecting both known and unknown network anomalies using a hybrid approach that combines supervised learn-

ing, unsupervised clustering techniques. It presents a comprehensive flow from background research to model evaluation and concludes with key insights and future directions.

Chapter 2 reviews previous studies on network anomaly detection using machine learning and deep learning techniques. It outlines existing challenges and the rationale behind adopting a hybrid approach.

Chapter 3 describes the dataset, preprocessing pipeline, feature engineering and the proposed algorithm architectures. It includes both supervised and unsupervised anomaly detection methods, emphasizing data balancing and hybrid clustering strategies.

Chapter 4 presents the performance evaluation of various models across different scenarios, comparing detection rates for known and unknown anomalies. The effectiveness of the hybrid approach is highlighted through confusion matrices and metric comparisons.

Chapter 5 mentioned the key findings and contributions of the work. It also discusses potential areas for improvement and future directions such as enhancing algorithm generalization, reducing false positives and integrating the system into real-time detection environments.

Chapter 2

Related Works

2.1 Related Works

Teng [35] (2024) introduced an optimized algorithm for network traffic anomaly detection using CNN. KDD Cup 99 [1] and NSL-KDD [8] datasets are used here for analysis. The CNN integrates multi-scale feature extraction, residual connections and distributed computing for improving detection accuracy. SVM and DT are also used here to compare with CNN's performance. The CNN achieved outstanding results with an accuracy of 98.5%, precision of 97.8%, recall of 99.0% and an F1 score of 98.4%, outperforming SVM (accuracy 95.2%, F1 score 94.7%) and Decision Trees (accuracy 93.1%, F1 score 92.8%). DDoS, U2R, port scanning are the focused attacks in this paper. The limitations include lower performance with imbalanced datasets, difficulty working well with different datasets, and sensitivity to noisy data. So, it needs further optimization and better reliability.

Shahu et al., [33](2024) presents the study on multi-class network anomaly detection using machine learning techniques. The UNSW-NB15 dataset is used here for analysis which includes 45 features and has 9 types of attacks like DDoS, Fuzzer, Backdoor etc. Stacking, Extra Trees, MLP, XGBoost, KNN, DT, Logistic Regression, Naive Bayes, SVM and Random Forest are the used models here. Precision, Recall, F1-score, MAE, RMSE and accuracy are used as the performance metrics. Among all the models, Random Forest achieved the highest accuracy rate of 98.06%. On the other hand, XGBoost had 97.99%, Extra Trees had 97.80%, and Stacking had 97.30%. KNN and MLP had accuracy of 91.9% and 90.35%, respectively, while Logistic Regression and Naive Bayes achieved 89.5% and 88.2%, respectively. Here high false positive rates, high-dimensional data is a big challenge which can lead to over fitting with imbalanced data. In the paper the authors tried to visualize the effectiveness of different types of machine learning algorithms in detecting network anomalies.

Bharadiya [27] (2023) focuses on using machine learning to detect anomalies in cyberspace. With the growing number of internet users, attackers have more opportunities to carry out attacks like spamming, phishing and bypassing security measures. Supervised machine learning models can effectively detect these anomalies and help prevent malware attacks. The paper explains which models work best for different issues. Decision trees, random forest and explainable AI models are good for spam

detection while logistic regression, random forest and neural networks work well for phishing detection. It also highlights the importance of using balanced and updated datasets with carefully selected features to train models effectively. For example, variance can be checked using PCA or features can be prioritized based on their contribution to reducing decision tree impurity using the Gini Index . Bharadiya [27](2023) emphasized the need for updated datasets to reduce errors and improve accuracy. The paper concludes that logistic regression is often preferred for its low false positive rate and high precision although it doesn't provide specific statistical results. Using optimized datasets can also reduce computational complexity and improve model speed. Overall, machine learning offers powerful tools to strengthen cybersecurity, helping organizations detect and respond to threats more efficiently while protecting critical systems and data.

Zhang et al., [30] (2023) introduced a CTMAANet model that helps to find unusual patterns in images. The model combines Convolutional Neural Networks (CNN) for capturing small details with transformers for understanding larger patterns. It has a special memory system that helps differentiate between normal and abnormal features, making it simpler to identify unexpected patterns in images. CTMAANet includes a dual view discriminator that examines both the image and its hidden features to improve detection accuracy. It uses a multi-scale memory module and a special loss function to reduce errors in classification. Examples of attacks include network anomalies, fraudulent activities and defective products. The model has been tested on datasets such as CIFAR10, MNIST and MVTec AD. It showed a 21% improvement in AUC compared to PCA/kPCA, a 12% improvement over other reconstruction methods and a 5% improvement over GAN-based models. In the CIFAR100 dataset, CTMAANet performed well across different categories. However, it does have some limitations like high computational demands, sensitivity to settings and risk of overfitting.

Wazid et al.,[25] (2022) explain how machine learning can improve cybersecurity especially as more IoT devices connect to the internet and become vulnerable to threats like malware and data theft. Machine learning can help detect a wide range of threats including eavesdropping, replay attacks, man-in-the-middle attacks, impersonation and DoS attacks. Their study shows that combining machine learning with blockchain technology can improve threat detection leading to high accuracy rates. The "Blockchain-enabled ML-driven detection" model achieved an F1score of 98.00%, the "EveDroid" model reached 99.00% and the "Graph-based Convolutional Neural Network (CNN)" model showed 94.00%. The paper introduces two main approaches: machine learning in cybersecurity and cybersecurity in machine learning. Wazid et al., [25](2022) proposed merging both ML and cybersecurity for greater benefits. Finding the right balance between security and system performance requires more effort because stronger security often leads to slower response times and operations. Machine learning can help automate this balance but human supervision is still essential.

Islam et al., [21] (2022) introduced Smartvalidator, an AI-powered tool that helps automate the validation of cyber threat alerts allowing security teams to respond to threats faster. Normally, security experts have to manually update rules which

can be slow and less effective when new threats emerge. Smartvalidator tackles the problem by collecting cyber threat information using machine learning to build prediction models and prioritizing alerts based on relevance. It relies on eight algorithms including Decision Tree, Random Forest, SVM and Neural Networks to create custom models that reduce errors and improve accuracy by using trusted threat data. The study highlights threats like malicious IP addresses, phishing attempts and anomaly detection grouping them into volume based, content based and behavior based anomalies. Smartvalidator showed strong performance with 75% of models scoring above 0.8 in F1 Scores while using 99% fewer models than traditional approaches which saves both time and resources. However, it has some drawbacks such as limited data quality, threat coverage, high resource needs and a complex setup process.

Al-Jeshi et al., [22] (2022) mentioned a blockchain based system to enhance the fintech sector and strengthen core banking systems. They highlighted common risks like ATM malware and phishing attacks targeting customer credentials, sophisticated SWIFT network breaches and insider threats exploiting weak physical security. Unlike traditional systems that can take days for international transactions and rely on multiple intermediaries, this blockchain setup demonstrated the ability to process 30 transactions among 30 peers in under 10 seconds showing major improvements in speed and scalability. The study highlights how blockchain can improve security, efficiency and cost effectiveness in the financial sector. By enabling direct peer to peer transactions and removing intermediaries, it significantly cuts transaction costs and processing times for cross-border transfers. Automating processes with smart contracts further reduces operational costs. By 2022, banks could save $15–20 million by cutting out middlemen and reducing manual intervention and compliance costs. Overall, the proposed blockchain system has the potential to transform traditional banking by solving inefficiencies and building a more secure and efficient fintech ecosystem.

Alalmaie et al., [19] (2022) presented a network security model for anomaly detection called the zero trust security model. This model operates by continuously checking and not trusting any data. The proposed model uses a balanced partitioning technique to enhance protection against attackers and extend multiview anonymization to various data fields . The model used an autoencoder to extract features. It detects anomalies with 92.23% accuracy. This is greater than the two-stage ensemble technique (91.27%) and other models such as Decision Tree + XGBoost (90.85%). It follows Zero Trust principles by anonymizing data before analysis and continuously checking network traffic while treating all data as untrusted. To sum up, the zero trust based architecture improves security and detects network breaches more accurately.

Anand et al., [20](2022) proposed an autoencoder neural network for the DIII-D tokamak's magnetic sensors, which are essential for maintaining plasma balance. For tokamak fusion reactors to monitor and control the stability, shape and position of the plasma, accurate sensor readings are crucial for maintaining plasma confinement and preventing disruptions. The autoencoder was trained on more than 4,000 plasma discharges to achieve this. When reconstruction errors exceed a

certain threshold, defective sensors are detected. This method works well for real-time applications, identifying problems like drift and noise faults within roughly 0.005 milliseconds. However, the threshold setting has a significant impact on its effectiveness, with false positives from low thresholds and false negatives from high thresholds. The investigation emphasizes the importance of establishing thresholds carefully and suggests strategies like ensemble approaches, continuous learning, and context-aware analysis to improve detection accuracy. In conclusion, the autoencoder-based system significantly improves sensor reliability and safety in the DIII-D tokamak.

Wang et al., [24] (2022) proposed ATSAD a semi-supervised learning for detecting network intrusions that can work with fewer labeled data. To test the model the paper uses CSE-CIC-IDS2018 [28] dataset that includes various types of attacks like Brute-force, Heartbleed, DoS and DDoS. The model builds on an existing model called Deep SAD by adding an attention network to help the model focus on the most important feature to make it more effective. Other models like One-Class SVM, IF, KDE, SS-DGM and Deep SAD itself are also used here to make a comparison with ATSAD. The AUC score of ATSAD is 99.88% with just 1% of the data labeled. In more noise in the data where Deep SAD loses 0.13% in performance ATSAD loses only 0.05%. The model still has some limitations that it highly depends on having some labeled samples which can cause problems in real world situations with more complex and messier data. So use of more dataset to test ATSAD is needed for the future.

Qin et al., [18] (2021) introduced a new model called AttentionAE, an autoencoder designed for detecting anomalies in networks that include both node attributes and network structure. The model was tested on datasets like BlogCatalog, Flickr and ACM. It uses a node attention mechanism to refine embeddings and an anomaly score generator to calculate scores based on reconstruction error, residual direction and embedding vectors. The node attention mechanism focuses on important nodes in a network, identifying relevant neighbors for each node. By assigning higher weights to these key nodes, the model gains a better understanding of the network's structure and behavior. Reconstruction error evaluates how closely the model's predictions align with the original data. The model outperformed other methods like LOF, SCAN and Dominant with achieving AUC scores of 0.8576 for BlogCatalog, 0.8730 for Flickr and 0.8286 for ACM. A higher AUC indicates better performance by reflecting the model's ability to distinguish between normal and abnormal nodes. Future improvements will focus on dynamic detection and enhancing accuracy.

Min et al., [16] (2021) proposed enhancing the identification of unusual activity in computer networks to defend against cyberattacks. Traditional methods often have trouble handling advanced persistent threats (APTs) which are complex cyberattacks caused by highly skilled attackers. Min et al., [16] (2021) introduced the Memory-Augmented Deep Autoencoder (MemAE) model combining a memory module with autoencoders. The model is trained using regular data to recognize typical patterns. When it detects an anomaly, it shows a higher reconstruction error, indicating a possible attack. The MemAE model was tested on three datasets which are NSL-KDD [8], UNSW-NB15 and CICIDS 2017 [29]. The MemAE model

outperformed AE and OCSVM with an AUROC of 0.9. Its reconstruction errors effectively identified anomalies with average error values of 0.92 (normal) vs. 10.28 (attack) in NSL-KDD and 0.72 (normal) vs. 4.79 (attack) in UNSW-NB15. MemAE improved classification accuracy, reduced false positives and was better at detecting unusual and unknown threats. However, it has limitations like over generalization, imbalanced data and dataset constraints.

Shaukat et al., [12] (2020) mentioned the challenges of cyber threats that are growing day by day in this digital world. This work focuses on applying machine learning techniques to enhance cybersecurity. As cyberspace usage continues to grow, traditional safeguards struggle to keep up with evolving threats like zero day attacks and advanced persistent threats (APTs). This study explores the potential of three machine learning techniques which are deep belief networks (DBNs), decision trees and support vector machines (SVMs). This model helps to detect three major cyber threats like spam, malware and anomalies. For instance, DBNs achieved 98.39% accuracy for spam detection on the Enron dataset whereas decision trees showed 99.96% accuracy in anomaly detection on the KDD [1] dataset. The study helps identify the most suitable models for specific cases to reduce future misclassification risks although it doesn't entirely eliminate them. Overall, the study highlights how machine learning techniques can build more adaptive and effective defenses against the increasing number of digital attacks.

Zhao et al., [13] (2020) focused on solving problems in detecting anomalies in multivariate time series data, which is important for industrial use. Traditional methods look at each data stream separately, missing the connections between different parts of a system, making them less effective for modern, complex systems. To fix this, the authors introduced MTAD-GAT (Multivariate Time-series Anomaly Detection via Graph Attention Network). This method identifies how different features and time points are connected using two types of graph attention layers: feature-based and time-based. It also combines prediction and renewal models to improve how time series data is understood. MTAD-GAT performed better than methods like LSTM-VAE and GAN-Li on datasets like SMAP, TSA and MSL. It achieved the highest average f1-score (86.91%) and recall (93.05%). It reduces errors by accurately identifying relationships between features. However, it has some drawbacks like sensitivity to irregular training data, difficulty in modeling direct relationships between time series and challenges in handling long-term patterns. Despite these issues, MTAD-GAT is a big step forward for detecting anomalies in complex systems.

This research paper focuses on using decision tree-based models to detect anomalies in network systems forming a network-based intrusion detection system (NIDS). Sarker et al., [11] (2020) discuss how the rapid increase in IoT devices creates vulnerabilities for attacks like DoS, malware distribution and unauthorized access. However, antivirus software fails against zero day attacks which exploit vulnerabilities before developers can address them. The authors proposed detecting unusual patterns in network traffic using a tree based model to identify anomalies. By calculating each feature's importance using the Gini Index, they rank and select the most relevant ones ensuring the model remains efficient. The model's performance was compared with other models like Naive Bayes (NB), Logistic Regression (LR),

K-Nearest Neighbor (KNN) and Support Vector Machines (SVM). The proposed Intrusion Detection Tree (IntruDTree) outperformed all these models with nearly perfect scores of 0.98 for precision, recall and F1 Score.

Zhang et al., [10] (2019) unveils a new strategy for identifying unusual activities in network traffic. UNSW-NB15 and KDD Cup 1999 [1] is used here which includes both labeled and unlabeled data. CSS-SVM(Cooperative Semi-Supervised Support Vector Machine) is developed as the main model in this paper. Other traditional models like Decision Tree, Random Forest, Support Vector Machine are also used here to observe the difference with CSS-SVM. The CSS-SVM has shown an impressive performance with 96.5% accuracy. The paper focuses on various types of attacks like DDoS, port scanning and reconnaissance attacks. These attacks are also categorized based on their characteristics though they are DoS attacks or probing attempts. The limitations of this paper are, the model's efficiency highly depends on the quality and distribution of the training data.

Budiarto et al., [9] (2019) investigated unsupervised anomaly detection in two hospital drug use datasets using K-Means, Local Outlier Factor (LOF) and One-Class Support Vector Machine (OC-SVM). Conducted on Google Colab with Python libraries like numpy, pandas and sklearn, the study normalized data using MinMax Scaler and applied K-Means (5 and 6 clusters via elbow method), LOF (density-based) and OC-SVM (hyperplane-based) to detect anomalies. Metrics included anomaly count, CPU time (s) and memory usage (MB) over ten iterations with thresholds of 0.01 and 0.05. All algorithms detected outliers. OC-SVM excelled in the Cephalexine dataset and the Enxyplex dataset, while LOF showed superior computational performance in the Enxyplex dataset. K-Means identified two anomalies per dataset with CPU times of 3.3 s and 3 s and memory usage of 164 MB and 162.94 MB. No significant CPU performance differences were found. OC-SVM outperformed K-Means and LOF in anomaly detection but performance depends on threshold selection. Limitations include sensitivity to parameter settings and dataset specificity. Future work recommends testing additional datasets and enhancing data quality by addressing anomalies.

Laksono et al., [4] (2015) introduced a method for detecting Distributed Denial of Service (DDoS) attacks using the CURE clustering algorithm, enhanced with an Outlier Removal Clustering (ORC) mechanism to improve outlier handling. The study utilized the KDD Cup 1999 dataset which is widely recognized for intrusion detection research. The method builds on traditional hierarchical clustering by addressing its limitations such as sensitivity to outliers and inability to handle non-spherical clusters. CURE with ORC employs random sampling, partitioning and representative points to capture cluster geometry with a two phase outlier elimination process based on maximum distance and distortion metrics. The experimental setup used a sample size of 8734 from the 311029 KDD Cup 1999 corrected dataset, with parameters including a shrink factor of 0.5, 20 representative points, 20 partitions and an ORC threshold of 0.5. The approach achieved a detection rate of 99.49%, a false positive rate of 1.80% and an accuracy of 91.21%, correctly identifying 5934 true positives (attacks) and 1635 true negatives (normal traffic) with 699 false negatives and 30 false positives. Compared to CURE's native outlier

elimination which removes clusters with few members, ORC's distance-based approach reduced data loss improving detection performance while maintaining low error rates. The method outperformed traditional hierarchical clustering and other methods like K-medoids and density-based clustering in handling outliers and non-spherical clusters. Limitations include potential packet loss from aggressive outlier removal which may discard valid attack or normal traffic data and the lack of a standardized method for determining optimal ORC threshold values impacting detection consistency. The study suggests future work to develop dynamic threshold setting techniques and extend the approach to intrusion prevention systems for real time mitigation, enhancing its applicability in network security.

Rathakrishnan et al., [40] (2024) proposed a multi model framework for synthesizing high fidelity network intrusion data using a CTGAN evaluated on the NSL-KDD dataset. The framework integrates CTGAN for data generation, two ANN based binary classifiers like Train-on-Real, Test-on-Synthetic and Train-on-Synthetic, Test-on-Real [TSTR]) and Local Interpretable Model-agnostic Explanations for explainability. CTGAN addresses challenges like unstable training and mode collapse, effectively handling mixed data types and non Gaussian distributions. The TRTS classifier achieved 98.36% accuracy, classifying 96.19% of synthetic data as intrusions while the TSTR classifier reached 97.95% accuracy identifying 98.59% of real intrusions with a similarity score of 0.92 via Kolmogorov-Smirnov tests. LIME enhances interpretability by identifying key features contributing to intrusion detection. The framework focuses on attacks like smurf, teardrop and neptune. Limitations include scalability challenges with larger datasets and the complexity of interpreting CTGAN directly, necessitating further research into model scalability and direct CTGAN interpretability.

Pu et al., [17] (2021) proposed an unsupervised anomaly detection method, SSC-OCSVM, combining Sub-Space Clustering and One-Class Support Vector Machine for detecting network intrusions without prior knowledge evaluated on the NSL-KDD dataset. The method splits the dataset into subspaces q=2 features and applies OCSVM to identify anomalies based on a dissimilarity vector, achieving high detection rates and low false alarm rates. SSC-OCSVM outperformed K-means, DBSCAN and SSC-EA with DR/FAR pairs of (0.99, 0.066) for Probe, (0.85, 0.078) for DoS, (0.52, 0.0895) for R2L, (0.90, 0.0905) for U2R and (0.89, 0.080) for mixed attacks. Feature selection via F-test and one-hot encoding expanded 41 features to 132 enhancing accuracy. Limitations include higher computational time due to sequential subspace processing and the need for effective feature selection methods. Future work focuses on parallelizing subspace clustering and improving feature selection for scalability.

Ige et al., [34] (2024) conducted an in-depth evaluation of Zero-shot, Single-shot and Few-shot learning approaches for detecting out-of-distribution zero-day malware attacks. Malimg and Malevis datasets were used for the experiments with Malevis representing in-distribution and Malimg used for out-of-distribution malware families. The study used static properties of malware converting RGB images to grayscale for consistency. The deep learning model was tested under three settings like Zero-shot (0 samples), Single-shot (1 sample) and Few-shot (5 samples) for the

Dinwod malware family which was excluded from training distribution. Zero correct predictions were recorded in both Zero-shot and Single-shot setups while only 3 out-of-distribution samples were correctly classified in the Few-shot scenario. These results highlight critical limitations in current state of the art N-shot learning models under zero-day attack conditions. The paper suggests that models must be more intelligent and robust in detecting unseen threats, recommending future exploration into combining Bayesian algorithms with deep learning for improved detection performance. Limitations include extremely poor generalization to unseen malware and ineffective learning even when a few samples are introduced. Further innovations are needed to close this performance gap.

Saurabh et al., [37] (2024) proposed a zero-shot based hybrid deep learning system combining Convolutional Neural Networks and Long Short-Term Memory networks to enhance zero-day attack detection. The CICIoT2023 dataset was used containing 33 types of attacks grouped into seven major categories like DDoS, DoS, Mirai, Recon, Spoofing, Web-based and BruteForce. The proposed approach simulates zero-day attack scenarios by withholding one attack class during training and reintroducing it during testing. This zero-shot learning technique allows the system to detect previously unseen attacks. The hybrid CNN-LSTM model first uses CNN layers to extract spatial features from the input network traffic followed by LSTM layers to capture temporal dependencies. The architecture includes Conv1D layers with ReLU activations, max pooling, dense layers and softmax for multi-class classification. Evaluation metrics were based on detection rate, precision, recall and F1 score computed from confusion matrices and ROC-AUC analysis. Performance results demonstrate that the CNN-LSTM hybrid significantly outperforms standalone CNN and traditional Stacked Ensemble Learning. It achieved detection rates of 94.95% (DDoS), 85.23% (DoS), 99.76% (Recon), 95.88% (Spoofing) and 94.98% (Mirai). It also achieved outstanding F1 scores for Recon (0.9979), Spoofing (0.9589) and Mirai (0.9495). The recall for Recon was perfect (1.000) indicating all Recon attacks were detected. The model also recorded a high AUC of 0.92 for DDoS detection. The study highlighted the effectiveness of combining spatial and temporal features in zero-day attack detection and demonstrated strong generalization performance across multiple unseen attack types. However, limitations include slightly reduced performance for DoS attacks and the exclusion of BruteForce and Web-based attacks due to class imbalance. The authors suggest future improvements through better hyperparameter tuning, advanced optimization techniques and refined feature engineering to further boost robustness and generalization in real world security contexts.

Chapter 3

Methodology

3.1 Dataset Pre-Processing

3.1.1 Data Collection

The dataset used in this research is the UNSW-NB15 dataset [29] , obtained from the University of New South Wales research platform. This dataset was developed by the Australian Centre for Cyber Security (ACCS) to provide a comprehensive and realistic benchmark for network intrusion detection systems (NIDS). Generated using the IXIA PerfectStorm tool, the dataset captures realistic network traffic under various conditions, including both benign and malicious activities.

The UNSW-NB15 dataset contains 49 features that encompass a wide range of network traffic and flow-based attributes. These include protocol-specific details, packet-level data and time-based information, enabling the analysis of traffic patterns and behavior. The dataset consists of over 2.5 million records, organized into training and testing subsets, with each record labeled to indicate whether it represents normal traffic or one of several attack categories. These attack categories include modern threats such as Denial-of-Service (DoS), reconnaissance, worms, exploits, backdoors and more.

The capacity of this dataset to address important issues in anomaly detection research lead to its selection. Its labeled data makes it easier for supervised machine learning to identify known attack patterns, while its features complexity and diversity make it ideal for unsupervised learning techniques like clustering. The design of the UNSW-NB15 dataset makes it possible to identify minor attack behaviors, making it a great experiment for assessing the suggested hybrid clustering-based anomaly detection algorithm.

The dataset is a strong resource for academics, guaranteeing repeatability and comparability of results due to its freely available nature and thorough description. The goal of this initiative intends to improve anomaly detection methods and aid in the development by utilizing the UNSW-NB15 dataset.

3.2 Data Pre-processing and Cleaning

3.2.1 Initial Data Cleaning

- **'ct_ftp_cmd' Feature**: Empty spaces were replaced with 0 to ensure the column contained valid data.
- **'attack_cat' Feature**:
  1. Empty values were filled with 'normal'.
  2. All values were converted to lowercase for consistency.
  3. The term 'backdoors' was standardized to 'backdoor' to correct inconsistencies in naming.
- **'ct_flw_http_mthd Feature**: Missing values were filled with 0.
- **'is_ftp_login' Feature**:
  1. Missing values were replaced with 0.
  2. Any values greater than 1 were transformed at 1.
- **'service' Feature**: The value '-' was replaced with 'None' to handle missing service information.

3.2.2 Pipeline Configuration

To have the consistent transformation of the dataset, a dictionary structure (saved_dict) was implemented to store essential transformation parameters:

- Column names
- Categorical column identifiers
- Binary column indicators

This configuration was saved as a pickle file to ensure that the same transformations could be applied to the test dataset, maintaining consistency across both training and testing phases.

3.2.3 Feature Engineering

New 21 features were created to enhance the predictive power of the dataset:

- **Duration Features**: The time difference between 'ltime' and 'stime' was calculated to capture the flow duration.
- **Network Flow Ratios**: Features such as byte ratio (sbytes/dbytes), packet ratio (spkts/dpkts) and load ratio (sload/dload) were computed to understand the network traffic dynamics.

- **Aggregate Features**: Total bytes, packets, load, jitter, inter-packet time and TCP setup time were aggregated to generate additional flow-related insights.
- **Interaction and Statistical Features**: Interaction features like byte-packet interactions, load-jitter interactions and statistical features such as mean packet size were derived to capture complex relationships between variables.

3.2.4 Transformation Pipeline Implementation

Three key functions were developed to apply transformations to the test dataset:

- **clean_data()**: Applied all cleaning operations, such as missing value imputation and standardization.
- **generate_features()**: Created additional engineered features to enhance the dataset's predictive capacity.
- **cat_feature_drop()**: Removed unnecessary categorical 7 columns from the dataset to focus on relevant predictors.
- **coor_feature_drop()**: Removed highly correlated features from the dataset focusing on its correlation with the target column.

3.2.5 Final Pre-processing Steps

Supervised Pre-processing

- **Target Encoding**: The target variable ('attack_cat') was encoded using LabelEncoder to convert categorical labels into numerical form.
- **Feature Scaling**: Standard Scaling was applied to normalize the feature values, ensuring that all features were on the same scale and preventing the model from being biased by features with larger numerical ranges.

Unsupervised Pre-Processing

- **Target Encoding**: The target variable ('attack_cat') was encoded using LabelEncoder to convert categorical labels into numerical form.
- **Weighted Average Approach for Balanced Dataset Representation**: To mitigate the class imbalance issue in the dataset where the "normal" class dominates with 87.35% of the data, we used a class balancing technique based on the weighted average of the target class distribution ("attack_cat"). This method ensures that minority classes are better represented without losing valuable information from the majority class. The weighted average is calculated using the formula [23]:

$$\text{Weighted Average} = \sum_{i=1}^{n} p_i \cdot w_i \tag{3.1}$$

where p is the proportion of each class and w is the weight assigned to each class, inversely proportional to its frequency. By using this approach, we could effectively perform CTGAN for balancing both the minority classes and the majority classes to create a balanced dataset.

- **Avoiding the Mean Value Approach**: We decided against using the mean value for balancing the dataset because, after multiplying it by 6, it becomes significantly large. Specifically, applying the mean (291,387) would have resulted in an excessively large representation of the majority class ("normal" traffic), which is crucial for accurately distinguishing between benign and malicious activities. To preserve the integrity of the dataset, we instead employed a weighted average approach. This method ensured that the "normal" class remained adequately represented while effectively addressing the class imbalance, maintaining a robust and representative training set for model development.

- **Class Imbalance Handling Using CTGAN**: To address class imbalance, CTGAN is used to create synthetic data for minority classes in a dataset to solve the problem of class imbalance. The process starts by identifying minority classes (in our case "exploits", "backdoor", "shellcode" or a broader set including "generic", "fuzzers" etc.). For each minority class, the algorithm filters the real data for that class and calculates how many additional samples are needed to reach a target sample size. This target size is determined either by a custom weighted average, calculated using predefined values and weights to balance the dataset or by a percentage based formula. The weighted average, computed as the sum of values multiplied by their weights divided by the sum of weights. It ensures a tailored sample size for balancing. To handle the majority class, the algorithm undersamples it by randomly selecting a subset of samples to match the target size to reduce its dominance. For minority classes, the algorithm over samples by using CTGAN to generate synthetic samples when the current sample count is below the target. The CTGAN model initialized with metadata automatically detected from the dataset then it is trained on the real data for each minority class. After training, CTGAN generates synthetic samples to fulfill the required sample count. These synthetic samples are combined into a single dataframe and merged with a downsampled majority class and shuffled to create a balanced dataset. CTGAN uses a generative adversarial network framework, where a generator creates synthetic data mimicking the real data and a discriminator checks its authenticity, iteratively improving the synthetic data quality to effectively balance the dataset.

- **Feature Scaling**: Standard Scaling was applied to normalize the feature values, ensuring that all features were on the same scale and preventing the model from being biased by features with larger numerical ranges.

This systematic preprocessing approach ensured consistency across both the training and test datasets, preventing data leakage and preparing the dataset for optimal model performance. Pre-processing figure is attached below in Figure 3.1.

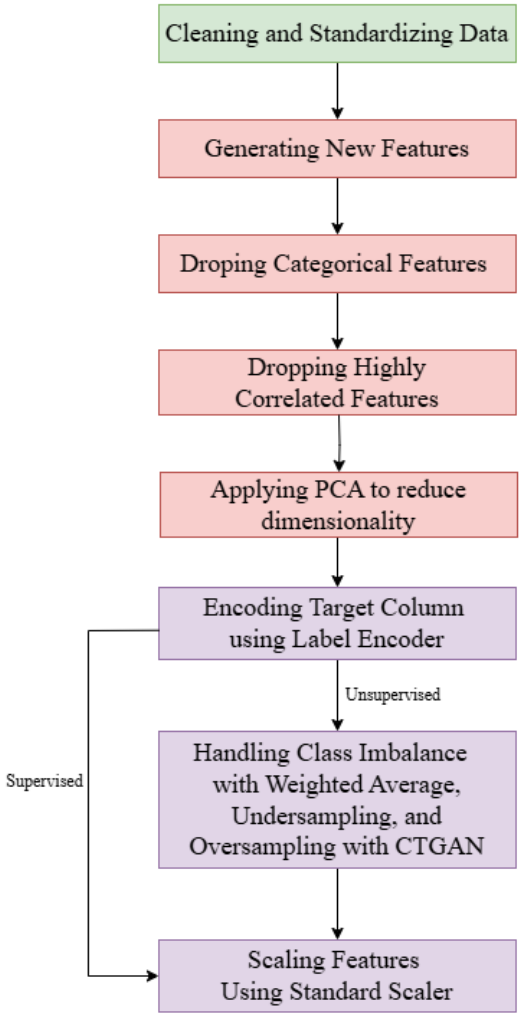


Figure 3.1: Data Pre-processing

3.3 Data Analysis

This section provides a detailed analysis of the UNSW-NB15 dataset [29], including its characteristics, feature distributions and class distributions. This analysis informs subsequent preprocessing and modeling decisions, ensuring that the dataset is optimized for building a robust zero day attack detection system.

3.3.1 Dataset Overview

The UNSW-NB15 dataset consists of 2,540,046 network traffic records, each representing a network flow with 49 distinct features. These features describe basic flow identifiers, traffic characteristics and derived statistical features. The dataset includes both normal traffic and various types of cyberattacks, making it a comprehensive dataset for intrusion detection research.

3.3.2 Feature Categorization

The features in the dataset can be grouped into three primary categories:

- Basic Features (12 features):
    1. Flow identifiers (srcip, sport, dstip, dsport)
    2. Protocol information (proto, state)
    3. Basic flow metrics (dur, sbytes, dbytes)
    4. Time-to-live values (sttl, dttl)
    5. Service type
- Content Features (26 features):
    1. TCP/IP characteristics
    2. Flow statistics (sload, dload)
    3. Packet counts (spkts, dpkts)
    4. Window sizes (swin, dwin)
- Time-based Features (11 features):
    1. Connection timing (stime, ltime)
    2. Jitter measurements (sjit, djit)
    3. Inter-packet arrival times

3.3.3 Class Distribution Analysis

UNSW NB15 dataset includes a large number of records including 1778032 datapoints in train set and 762014 datapoints in test set after splitting at 70:30 ratio, but the class distribution reveals significant class imbalance. Class distribution is shown in **Table 3.1**.

Attack Category	Train Count	Test Count	Total Count	Total Percentage
normal	1,553,176	665,584	2,218,763	87.35%
Generic	150,793	64,688	215,481	8.48%
Exploits	31,158	13,367	44,525	1.75%
Fuzzers	16,954	7,292	24,246	0.95%
DoS	11,480	4,873	16,353	0.64%
Reconnaissance	9,787	4,200	13,987	0.55%
Analysis	1,841	836	2,677	0.11%
Backdoor	1,656	673	2,329	0.09%
Shellcode	1,054	457	1,511	0.06%
Worms	130	44	174	0.02%

Table 3.1: Class Distribution

3.3.4 Analysis of Class Dominance

In the dataset, the "normal" class continues to dominate, making up 87.35% of the total records. The "Generic" class follows with 8.48%,"Exploits" contributes 1.75% and while the rest of the attack categories contribute less than 1% each.
The significant dominance of the "normal" class and the underrepresentation of attack categories, such as "Worms" (0.02%), "Shellcode" (0.06%) and "Backdoor" (0.09%), suggests that the dataset is heavily imbalanced. This class imbalance can lead to models that are biased toward predicting the "normal" class, potentially underperforming in detecting less frequent attack categories.

3.3.5 Implications for Preprocessing

Preprocessing for Supervised Approach

- Feature Selection and Engineering:
    1. Highly correlated features($|\rho| > 0.89$) were removed to reduce redundancy and improve model performance.
    2. Features with higher correlations to the target variable were prioritized.
    3. PCA was employed to eliminate redundant features and enhance model performance by transforming the dataset into a lower-dimensional space to retain the most significant features.
- Stratified Sampling:
    1. This approach was employed for validation to ensure that each fold of the cross-validation process contained a representative proportion of each class.

Preprocessing for Unsupervised Approach

Given the identified class imbalance, several preprocessing techniques were implemented to address this issue:

- Class Balancing:

1. Balancing techniques, such as CTGAN, were applied to the both minority classes and majority classes to balance the dataset.
2. For the supervised models, balancing techniques are only used in the train set but for the unsupervised approaches, balancing techniques are used in the train and test set both.

- Feature Selection and Engineering:

1. Highly correlated features($|\rho| > 0.89$) were removed to reduce redundancy and improve model performance.
2. Features with higher correlations to the target variable were prioritized.
3. PCA was employed to eliminate redundant features and enhance model performance by transforming the dataset into a lower-dimensional space to retain the most significant features.

- Stratified Sampling:

1. This approach was employed for validation to ensure that each fold of the cross-validation process contained a representative proportion of each class.

3.3.6 Importance of Class Balancing for Unsupervised Approach

Class balancing was crucial in improving model performance. Without it, the model would have been biased toward the "normal" class, given its overwhelming presence in the dataset (87.35%). The balancing techniques involved oversampling the minority classes and undersampling the majority class based on the weighted average [23] of the class distribution. These methods ensured that the models could effectively detect less frequent attack categories. This led to improve the performance of the models without sacrificing the size of the dataset or losing valuable data from the majority class.

3.3.7 Class Balancing for Supervised Approach

In all supervised models, the train set is balanced by assigning class weights to handle class imbalance. For the Decision Tree and Random Forest models, the parameter "class_weight='balanced' " is used, which automatically calculates weights inversely proportional to class frequencies. It gives higher importance to minority classes during training. In the FNN, class weights are computed manually using "compute_class_weight('balanced', ...)", then passed to the model via the "class_weight" argument in the "fit()" function. This ensures that the model treats all classes more equally.

But the test set was not balanced intentionally because the goal of testing is to evaluate how well the trained model performs on real-world, unseen data, which is often naturally imbalanced. Balancing the test set would artificially distort the actual distribution of classes and provide an unrealistic assessment of the model's effectiveness. By keeping the test set imbalanced, we ensure that performance metrics such

as accuracy, precision, recall, F1-score and MCC truly reflect the model's ability to detect both majority and minority classes as they occur in practice. This helps to identify whether the model is biased toward frequent classes or can generalize well across all categories.

3.4 Class Distribution for Supervised Approach

At first, the supervised models were trained with all the 10 classes (including the "normal" class and the 9 attack categories) which means there was no unknown anomaly. For this, per class distribution for train set and test set shown in **Table 3.2** below:

Category	**Train**	**Test**
normal	1,553,121	665,639
Generic	150,804	64,677
Exploits	31,152	13,373
Fuzzers	17,087	7,159
DoS	11,469	4,884
Reconnaissance	9,761	4,226
Analysis	1,847	830
Backdoor	1,639	690
Shellcode	1,034	477
Worms	116	58

Table 3.2: Class distribution for no unknown class(Supervised)

Train Set: 17,78,030 samples

Test Set: 7,62,013 samples

To check the model performance for the unknown anomaly detection, we trained the supervised models with 6 classes (including the "normal" class and the 5 attack categories), which means 4 randomly picked unknown classes ("dos", "generic", "exploits", "fuzzers") were there. For this, per class distribution for train set and test set shown in **Table 3.3** below:

Category	Train	Test
normal	1,553,121	665,639
Generic	N/A	64,677
Exploits	N/A	13,373
Fuzzers	N/A	7,159
DoS	N/A	4,884
Reconnaissance	9,761	4,226
Analysis	1,847	830
Backdoor	1,639	690
Shellcode	1,034	477
Worms	116	58

Table 3.3: Class distribution for 6 known 4 unknown classes(Supervised)

Train Set: 15,67,518 Samples

Test Set: 7,62,013 Samples

The class distribution for both the train and test sets maintained the same proportions splitting (a 70:30 ratio). This consistency is critical for preventing data leakage and ensuring reliable model evaluation.

3.4.1 Feature Engineering and Selection Process(Supervised)

Initially, the dataset contained 49 features.

- We dropped the 'label' column because we are working with the 'attack cat' for multiclass detection, reducing the count of features to 48.
- Next, we added 21 additional features, increasing the total number of features to 69.
- After that, we removed 7 categorical features, bringing the feature count down to 62.
- Also, we dropped 27 highly correlated features to avoid multicollinearity.
- Finally, we dropped 13 features applying PCA to reduce dimensionality, resulting in a final feature set of 22 features for both scenarios.

This stepwise feature selection process helped ensure that the model would perform optimally by retaining the most informative features and reducing noise from irrelevant or redundant data.

3.5 Class Distribution for Unsupervised Approach

At first, we had trained the model with 10 known classes (including the "normal" class and the 9 attack categories) no unknown classes. For this, per class distribution for train set and test set shown in **Table 3.4** below:

Category	Train	Test
Generic	39,435	10,140
normal	39,435	10,140
Worms	39,319	10,082
Shellcode	38,401	9,663
Backdoor	37,796	9,450
Analysis	37,588	9,310
Reconnaissance	29,674	5,914
DoS	27,966	5,256
Fuzzers	22,348	2,981
Exploits	8,283	10,140

Table 3.4: Class distribution for no unknown class(Unsupervised)

Train Set: 3,20,245 Samples

Test Set: 83,076 Samples

After that, we had trained the model with 6 known classes (including the "normal" class and the 5 attack categories) and 4 randomly picked unknown classes ("dos", "generic", "exploits", "fuzzers"), which were the same combination we tried in the supervised models. It helped us to visualize the comparison of results of the supervised models and unsupervised approaches. For this, per class distribution for train set and test set shown in **Table 3.5** below:

Category	Train	Test
normal	95,538	24,567
Backdoor	95,538	23,877
Analysis	93,899	23,737
Reconnaissance	93,691	20,341
Shellcode	85,777	24,090
Worms	64,386	24,509
Generic	N/A	24,567
DoS	N/A	19,683
Fuzzers	N/A	17,408
Exploits	N/A	11,194

Table 3.5: Class distribution for 6 known 4 unknown classes(Unsupervised)

Train Set: 5,28,829 Samples

Test Set: 2,13,973 Samples

Then we had tried another combination to see how the unsupervised approaches was performing in other environments too. So, we had trained the model with 7 known classes (including the "normal" class and the 6 attack categories) and 3 randomly picked unknown classes ("worms", "exploits", "analysis") were there. For this, per class distribution for train set and test set shown in **Table 3.6** below:

Category	Train	Test
Generic	72,263	19,922
normal	72,263	19,922
Shellcode	71,229	19,445
Backdoor	70,624	19,232
Reconnaissance	62,502	15,696
DoS	60,794	15,038
Fuzzers	55,176	12,763
Worms	N/A	19,864
Analysis	N/A	19,092
Exploits	N/A	6,549

Table 3.6: Class distribution for 7 known 3 unknown classes(Unsupervised)

Train Set: 4,64,851 Samples

Test Set: 1,67,523 Samples

To evaluate more, another combination was tried by us. Now, we had trained the model with 8 known classes (including the "normal" class and the 7 attack categories) and 2 randomly picked unknown classes ("fuzzers", "worms") were there. For this, per class distribution for train set and test set shown in **Table 3.7** below:

Category	Train	Test
normal	57,603	17,311
Generic	57,603	17,311
Shellcode	56,569	16,834
Backdoor	55,964	16,621
Analysis	55,756	16,481
Reconnaissance	47,842	13,085
DoS	46,134	12,427
Exploits	26,451	3,938
Worms	N/A	17,253
Fuzzers	N/A	10,152

Table 3.7: Class distribution for 8 known 2 unknown classes(Unsupervised)

Train Set: 4,03,922 Samples

Test Set: 1,41,413 Samples

The class distribution for both the train and test sets maintained the same proportions splitting (a 70:30 ratio). This consistency is critical for preventing data leakage and ensuring reliable model evaluation.

In our work, we also could trained the models with 9 known classes and 1 randomly picked unknown classes but we did not do that. Because, our research objective is to detect multiclass anomaly. For this we tried to include more than one unknown

anomaly and experimented with 6:4 ratio, 7:3 ratio and 8:2 ratio but not the 9:1 ratio.

3.5.1 Feature Engineering and Selection Process(Unsupervised Approach)

Initially, the dataset contained 49 features.

- We dropped the 'label' column for all the scenarios (6:4, 7:3, 8:2, 10:0) because we are working with the 'attack cat' for multiclass detection, reducing the count of features to 48.
- Next, we added 21 additional features for all the scenarios (6:4, 7:3, 8:2, 10:0), increasing the total number of features to 69.
- After that, we removed 7 categorical features from all the scenarios (6:4, 7:3, 8:2, 10:0), bringing the feature count down to 62.
- Also, we dropped 27 highly correlated features to avoid multicollinearity.
- Finally, we dropped features applying PCA to reduce dimensionality. resulting in a final feature set of 26 features for 6:4 scenario, 24 features for 7:3 scenario, 26 features for 8:2 scenario and 30 features for the 10:0 scenario.

This stepwise feature selection process helped ensure that the model would perform optimally by retaining the most informative features and reducing noise from irrelevant or redundant data.

3.6 Model Information for Conventional Approaches

3.6.1 FNN

A Feedforward Neural Network [35] is a simple artificial neural network architecture in which information flows unidirectionally from input to output layers through one or more hidden layers. It learns complex patterns by adjusting weights using back-propagation to minimize prediction errors. For regression or classification problems, FNNs are excellent at encoding non-linear relationships in high-dimensional data. However, FNN requires extensive computational resources and careful tuning of hyperparameters such as layer size and learning rate to achieve optimal performance.

3.6.2 Random Forest

Random Forest [21] is an ensemble learning algorithm that builds many decision trees at training time and combines their predictions, often through majority vote for classification or averaging for regression. Through the addition of randomness during feature selection and data sampling, it prevents overfitting and improves robustness over a single tree. Random Forest works well with varying datasets and can identify intricate patterns, but it requires high computational power and might be less interpretable than more straightforward models.

3.6.3 Decision Tree

A Decision Tree [21] is a prediction modeling technique that recursively splits input data into subsets of the data according to feature values and forms a tree-like decision structure to finally output final predictions. Each internal node represents a decision based on features and each leaf node is a decision outcome. Decision Trees are clear and interpretable. The model is ideal for classification and regression. However, they can overfit noisy or complex data. So, pruning or combining models helps them work better

3.6.4 Isolation Forest

Isolation Forest [38] model is an unsupervised method for detecting anomalies by isolating data points. It works by randomly selecting features and splitting data into smaller groups and creating a forest of decision trees after that. Anomalies are identified as points that are easier to isolate. It results in shorter paths in the trees. This approach is robust to noise and highly efficient for large, high-dimensional datasets.

3.6.5 LOF

The Local Outlier Factor [5] algorithm considers anomalies when the density of a given data point's neighborhood is different from that of its neighbors. In outlier cases, less dense regions consisting of only a handful of neighboring points are formed. It is beneficial for datasets with varying density but can be computationally very complex for larger datasets.

3.6.6 One-Class SVM

One-Class Support Vector Machine [3] is an unsupervised model that learns a boundary around normal data points in a high-dimensional space. Data points falling outside this boundary are classified as anomalies. It performs well with complex and high-dimensional data but is sensitive to parameter settings and may not scale well for very large datasets.

3.6.7 Auto Encoder

An autoencoder [7] is a neural network designed to learn a compressed representation of the input data and then reconstruct the same. For anomaly detection, it considers data points that have a large reconstruction error as outliers because they deviate from the normal patterns that it has learnt. It is good at capturing complex and non-linear relationships in data but is computationally expensive and has many parameters to tune.

3.6.8 Siamese Neural Network

A Siamese Neural Network [14] is a neural architecture made of two identical subnetworks that share weights to learn feature similarity between input pairs. We used one-shot learning approach based on Siamese Neural Network (SNN) which consists

of two identical subnetworks sharing weights to learn feature similarity. One-shot learning [15] is an approach that learns from a very small number of examples. It uses Siamese neural networks or metric learning for comparison based tasks. In anomaly detection, it identifies outliers by comparing new data to a single or small set of normal examples. It is ideal for scenarios with limited labeled data but relies on strong feature extraction methods.

3.7 Architecture and Configuration for Unsupervised Approaches

3.7.1 Hybrid CURE algorithm

The Hybrid CURE algorithm is a special method for finding unusual patterns, or anomalies, in large and complex datasets like those used in cybersecurity. It builds on the CURE clustering algorithm but adds tools like MiniBatchKMeans, BallTree and Local Outlier Factor (LOF) to work faster and better with big data. The algorithm works in two steps: training and testing.

In the training step, it takes data from different categories like 'normal', 'dos', 'reconnaissance' etc and uses a set of numerical features. For each category, it starts by grouping data into smaller clusters using MiniBatchKMeans. It helps to handle large datasets easier. Then it combines these clusters based on how close they are using a tool called BallTree, until it reaches the desired number of clusters. It is set by a parameter called n_clusters. It picks key points, which are called representative points for each cluster. Then it applies a "shrinkage factor" (called alpha) to make the clusters more resistant to outliers meaning unusual data points. The algorithm finds and removes outliers using two methods: Z-score thresholding which checks if data points are too far from the average and LOF which looks for points that don't fit well with their neighbors. Finally, it saves information about each cluster, like its center (centroid) and size (radius), which describe the cluster's boundaries.

In the testing step, the algorithm checks new data points to see if they belong to a cluster. It uses a function called analyze_and_separate_test_data, which assigns each point to the nearest cluster based on its distance. If a point is within a cluster's radius, it is labeled as 'normal' if the cluster is from the 'normal' category, or 'classified_attack' if it is from an attack category like 'dos'. If a point doesn't fit into any cluster, it is labeled 'anomaly'. The algorithm also gives a report showing how many points were labeled normal, classified_attack or anomaly, along with their percentages. The algorithm uses a few key parameters like the number of clusters (n_clusters), representative points (n_representatives), shrinkage factor (alpha), outlier threshold (outlier_threshold) and LOF neighbors (lof_neighbors). These parameters help the algorithm work accurately while staying fast. It produces clean training data with unusual points removed, information about the clusters and clear results for the test data. It shows how many points are in each group like normal, classified_attack or anomaly and their percentages.

This method is great for spotting unusual patterns in cybersecurity data because it

handles complex data well, removes outliers effectively and gives clear results. Using MiniBatchKMeans and BallTree makes it fast for big datasets, while LOF improves its ability to find outliers, making it a strong tool for large, complex datasets. The process that we explained can be found in **Algorithm 1**.

3.7.2 Second filter

The lof_novelty_anomaly_detection function enhances anomaly detection by combining Local Outlier Factor (LOF) with cosine similarity to distinguish between "known" and "unknown" anomalies in a cybersecurity dataset. It takes as input an classified_attack DataFrame (classified_attack_df), a list of numerical features (numerical_columns), attack categories (attack_categories) and precomputed cluster metadata (cluster_metadata) from a prior clustering process the OptimizedCURE algorithm. The function gives two DataFrames: known_anomalies_df and another one is unknown_anomalies_df and a dictionary of classification statistics.

Data Preprocessing: The input data (called classified_attack_df) is adjusted using a tool called StandardScaler to make sure all numbers are on the same scale. This helps the algorithm work smoothly for calculations involving Local Outlier Factor (LOF) and similarity checks.

Local Outlier Factor (LOF) Analysis: The algorithm uses Local Outlier Factor (LOF) to find unusual points. It has parameters like n_neighbors and contamination, which guesses how many points might be outliers. LOF gives each point a score where higher scores mean a point is more likely an anomaly. These scores are adjusted to fit between 0 and 1 for consistency.

Cosine Similarity to "Known" Clusters: The algorithm looks at the centers (centroids) of "known" attack clusters like 'dos' or 'reconnaissance' from the cluster_metadata, ignoring the 'normal' class. These centroids are adjusted to match the test data's scale. It then calculates how similar each test point is to these attack centers using a method called cosine similarity. The highest similarity score shows how close a point is to a "known" attack pattern.

Classification Logic: The algorithm combines the normalized LOF score with the opposite of the highest similarity score, using a weight called unknown_bias. This weight makes the algorithm more likely to label points as "unknown" unless they strongly match a "known" attack. If a point's similarity score is above the similarity_threshold and its combined score is below a certain limit, it is labeled as a "known" attack and matched to the closest attack category. Otherwise, it is labeled as "unknown".

Output and Statistics: The function returns known_attacks_df: points classified as "known" attacks with their matched attack category, Unknown_anomalies_df: points classified as novel(new) anomalies with merged anomaly_df from Hybrid CURE algorithm.

The second filter process can be found in **Algorithm 2**.

3.7.3 Pseudocode

Algorithm 1 Hybrid Algorithm

START
$data \leftarrow$ Input training data X, $test\_points \leftarrow test\_df[numerical\_columns]$
$initial\_clusters \leftarrow \min(100, \max(10, n\_samples/20))$
$subcluster\_labels \leftarrow$ MiniBatchKMeans$(data, initial_clusters)$
$clusters \leftarrow$ {set of point indices for each subcluster}
$active\_clusters \leftarrow \{0, 1, \ldots, initial\_clusters - 1\}$
while $|active\_clusters| > n\_clusters$ **do**
    $tree \leftarrow$ BallTree$(active_cluster_centers)$
    find closest pair (i, j) using $tree.query()$
    $clusters[i] \leftarrow clusters[i] \cup clusters[j]$
    $active\_clusters \leftarrow active\_clusters \setminus \{j\}$
    update $subcluster\_centers[i] \leftarrow$ mean$(data[clusters[i]])$
end while
$labels \leftarrow$ assign cluster IDs to all data points
for each $cluster\_id$ **do**
    $cluster\_points \leftarrow data[labels == cluster\_id]$
    $centroid \leftarrow$ mean$(cluster_points)$
    $outlier\_mask \leftarrow |zscore(distances)| >$ threshold
    if $|cluster\_points| > lof\_neighbors$ **then**
        $outlier\_mask \leftarrow outlier\_mask \vee$ LocalOutlierFactor()
    end if
    $core\_points \leftarrow cluster\_points[\neg outlier\_mask]$
    $reps \leftarrow$ select $n\_representatives$ farthest from centroid
    $shrunk\_reps \leftarrow \alpha \times centroid + (1 - \alpha) \times reps$
    $radius \leftarrow$ max distance from $core\_points$ to nearest $shrunk\_rep$
    $cluster\_info[cluster\_id] \leftarrow \{class, centroid, radius\}$
end for
$all\_clusters \leftarrow$ combine $cluster\_info$ from all classes
for each point in $test\_points$ **do**
    $nearest\_cluster \leftarrow$ find cluster with minimum distance to centroid
    if $distance \leq nearest\_cluster.radius$ **then**
        if $nearest\_cluster.class == normal\_class$ **then**
            $classification \leftarrow$ 'normal'
        else
            $classification \leftarrow$ 'classified_attack'
        end if
    else
        $classification \leftarrow$ 'anomaly'
    end if
end for
return $labels$, $anomaly\_df$, $normal\_df$, $classified\_attacks\_df$, $stats$
END

Algorithm 2 Optimized Filtered Algorithm

START
$anomaly\_data \leftarrow$ Input classified_attack dataframe,
$features \leftarrow numerical\_columns$
$attack\_cats \leftarrow attack\_categories$, $metadata \leftarrow cluster\_metadata$
$k \leftarrow 20$, $contamination \leftarrow 0.5$, $sim\_threshold \leftarrow 0.95$, $bias \leftarrow 0.9$
if $|anomaly\_data| = 0$ **then**
    return empty dataframes and zero statistics
end if
$X \leftarrow anomaly\_data[features].values$
$X_{scaled} \leftarrow$ StandardScaler().$fit\_transform(X)$
$lof \leftarrow$ LocalOutlierFactor($k, contamination, novelty = False$)
$lof\_scores \leftarrow lof.fit\_predict(X_{scaled})$
$raw\_scores \leftarrow -lof.negative\_outlier\_factor\_$
$normalized\_scores \leftarrow$ normalize(raw_scores) to $[0, 1]$
$centroids \leftarrow []$, $valid\_categories \leftarrow []$
for each $category$ in $attack\_cats$ **do**
    if $category \in metadata$ and $metadata[category] \neq \emptyset$ **then**
        $valid\_categories.append(category)$
        for each $info$ in $metadata[category].values()$ **do**
            $centroids.append(info[centroid])$
        end for
    end if
end for
if $centroids$ is empty **then**
    $result \leftarrow anomaly\_data.copy()$
    $result[anomaly\_type] \leftarrow unknown$
    return $\emptyset$, $result$, statistics
end if
$centroids_{scaled} \leftarrow$ scaler.$transform(centroids)$
$similarity\_matrix \leftarrow$ cosine_similarity($X_{scaled}, centroids_{scaled}$)
$max\_similarities \leftarrow \max(similarity\_matrix, axis = 1)$
for each anomaly i **do**
    $combined\_score_i \leftarrow bias \times normalized\_scores[i] + (1 - bias) \times (1 - max\_similarities[i])$
    if $max\_similarities[i] \geq sim\_threshold$ and $combined\_score_i <$ threshold **then**
        $classification_i \leftarrow$ 'known'
    else
        $classification_i \leftarrow$ 'unknown'
    end if
end for
$known\_attacks \leftarrow$ filter anomalies with classification = 'known'
$unknown\_anomalies \leftarrow$ filter anomalies with classification = 'unknown'
$stats \leftarrow$ compile statistics from results
return $known\_attacks$, $unknown\_anomalies$, $stats$
END

Chapter 4

Result and Analysis

4.1 Supervised Models: Approaches and Limitations

Initially, we explored the application of supervised learning models for network anomaly detection. These models were meticulously trained on a dataset encompassing all 10 distinct attack classes and subsequently tested under the same comprehensive class framework. This means there were no unknown attack classes in the test. In this scenario, the supervised models performed very well. The following results were obtained in **Table 4.1** below:

Model/Results	Accuracy	Precision	Recall	F1-Score	MCC
Decision Tree	0.98	0.99	0.98	0.98	0.91
Random Forest	0.98	0.99	0.98	0.99	0.93
FNN	0.97	-	-	-	-

Table 4.1: Performance of Supervised Models without Unknown Classes

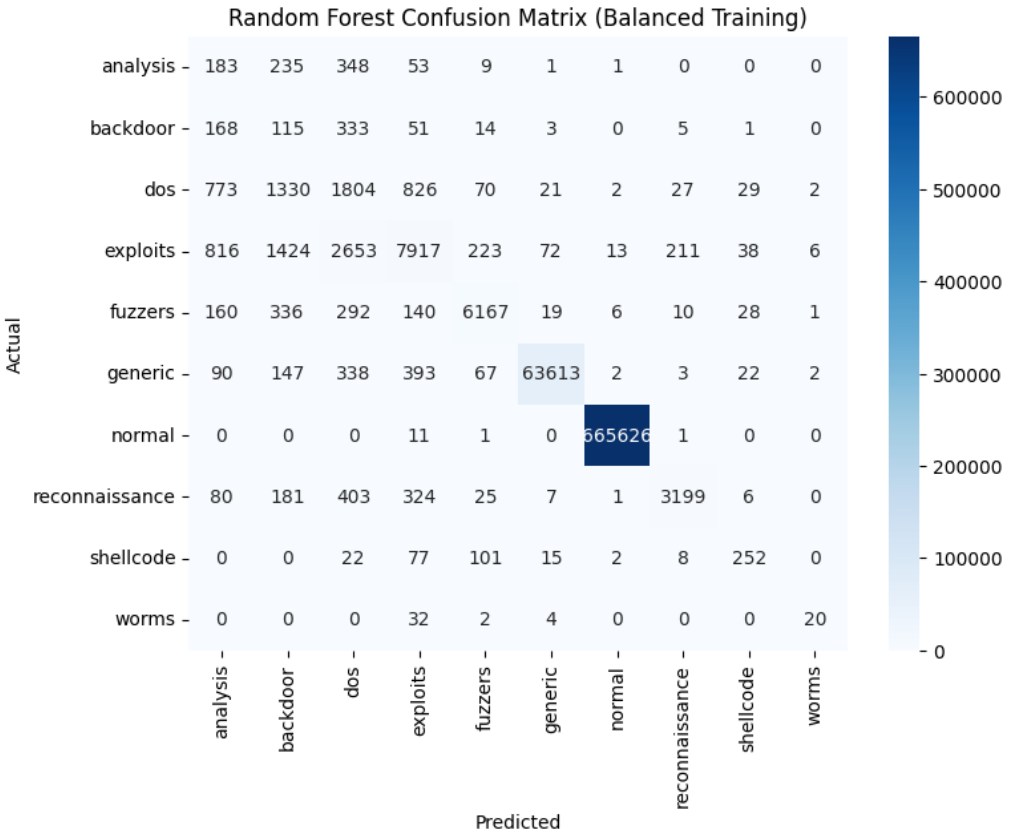
The **Random Forest** model exhibited the best performance among the evaluated models by achieving an accuracy of 98%, precision of 99%, recall of 98% and an F1-score of 99%. Additionally, its Matthews Correlation Coefficient (MCC) of 0.93 underscores its reliability in distinguishing between classes. These results highlights the model's superior ability to accurately identify true positives while maintaining low error rates, making it the preferred choice for this study.

Then, the models were trained using a dataset with six known attack classes. But the testing phase incorporated all ten attack classes, including four previously unseen classes. This approach allowed us to evaluate the model's ability to generalize to both familiar and novel attack types.

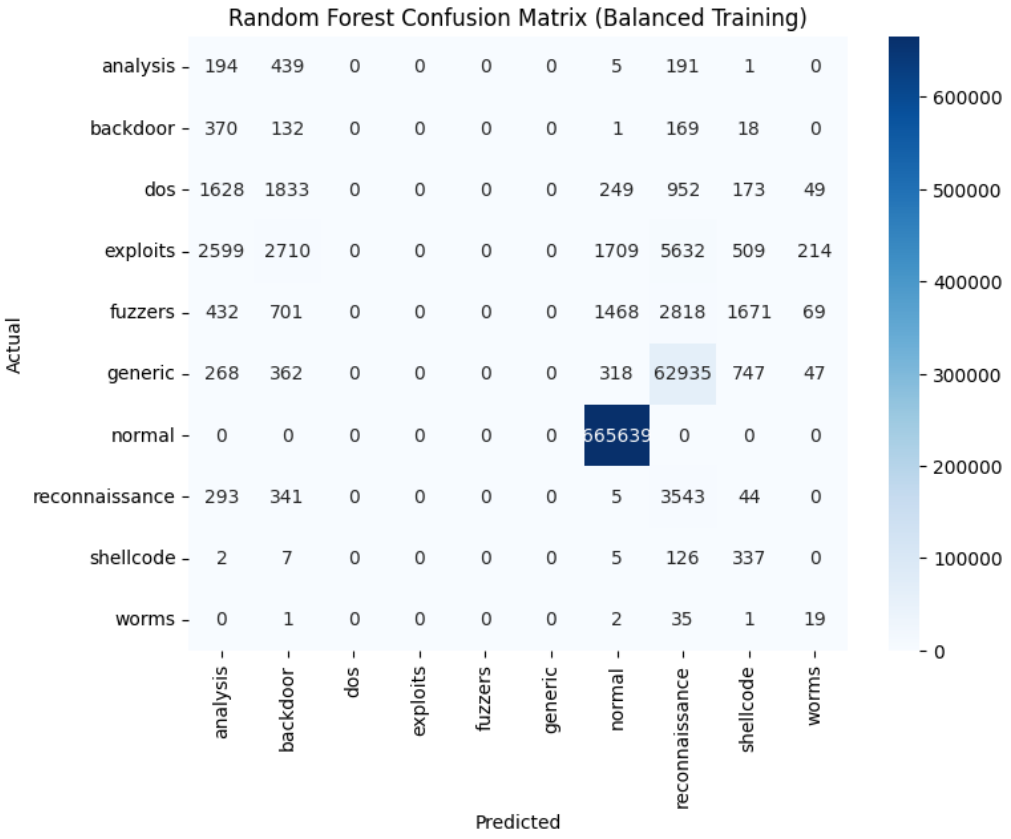
Model/Results	Accuracy	Precision	Recall	F1-Score	MCC
Decision Tree	0.88	0.86	0.88	0.87	0.47
Random Forest	0.88	0.87	0.88	0.87	0.50
FNN	0.87	-	-	-	-

Table 4.2: Performance of Supervised Models with Unknown Classes

The above **Table 4.2** Showed that the **Random Forest** model demonstrated the best performance among the evaluated models for the dataset with 6 known and 4 unknown classes with an accuracy of 88%. Because of the normal class dominance the results were good in accuracy, precision, recall and f1-score but lag behind in MCC. Now the model's can not properly distinguishing between classes. So more accuracy cannot define the models performance untill we cross examined with MCC score. So the supervised models had significant limitations in handling this multiclass classification task. The models struggle to identify attacks not encountered during training, resulting in poor generalization to unknown classes and an inability to predict the correct attack class. This suggests the need for either conventional model optimization or exploration of alternative approaches to improve robustness against both known and unknown attack types. Confusion matrices for **Random Forest** is attached below in Figure 4.1, Confusion matrices for **Decision Tree** is attached below in Figure 4.2 and Confusion matrices for **FNN** is attached below in Figure 4.3.

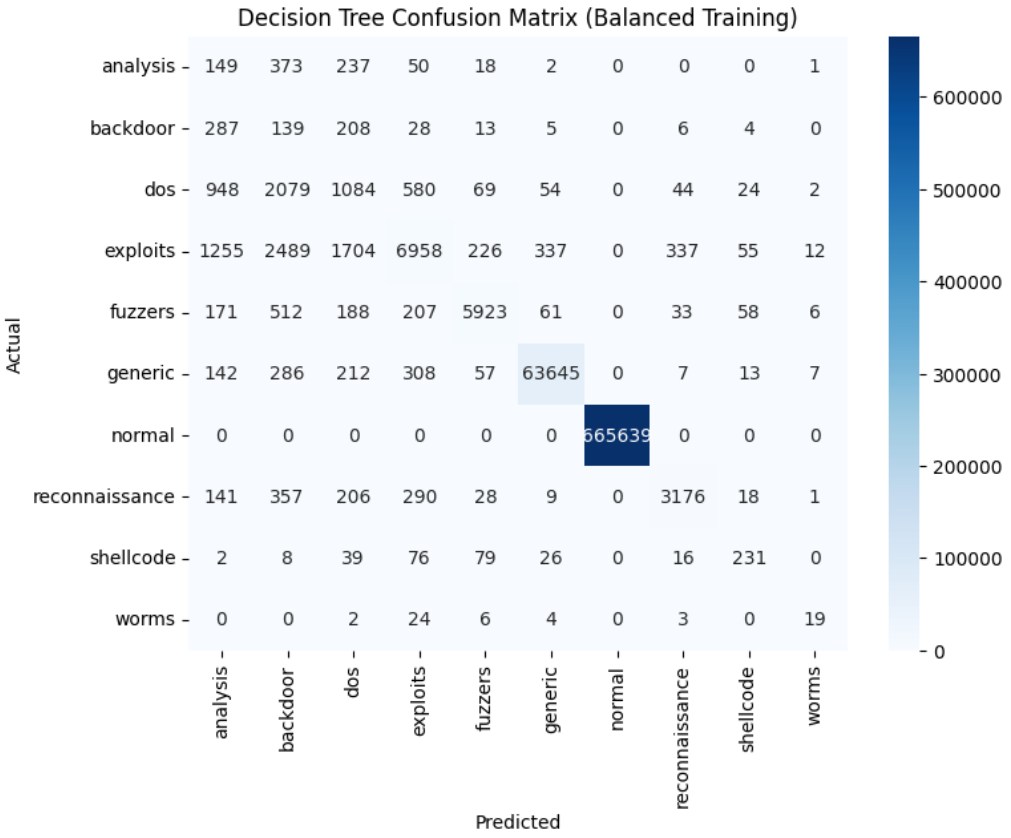


(a) Random Forest (All Known Classes)

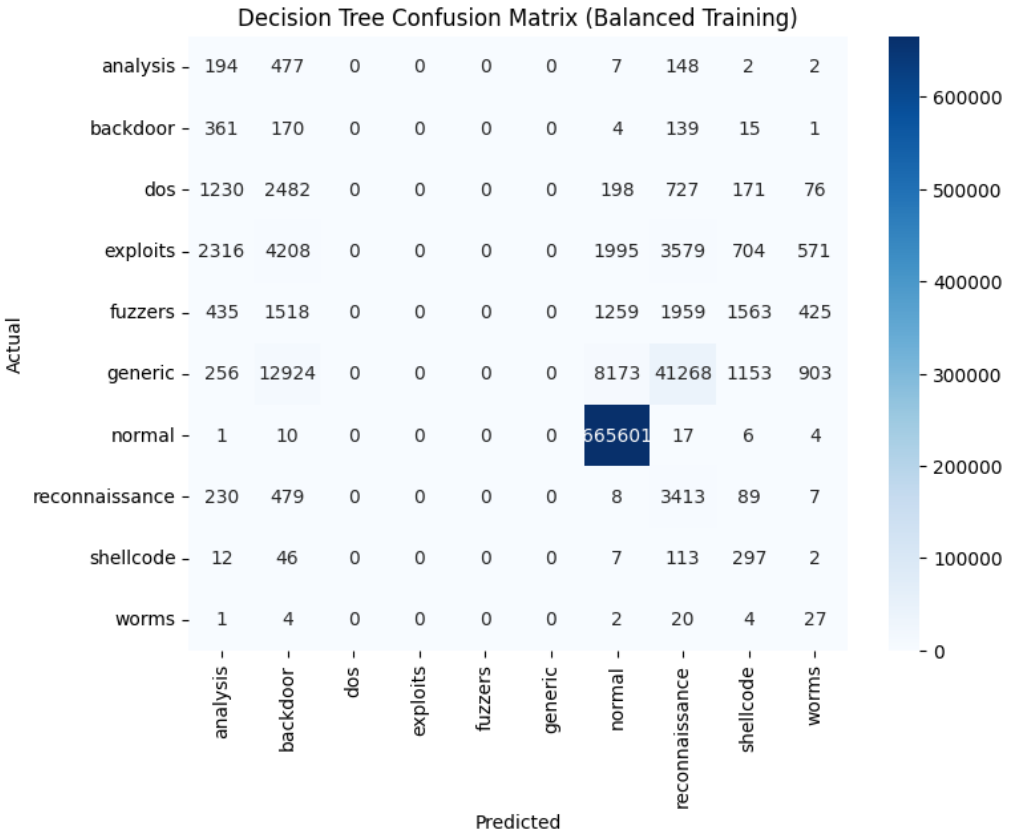


(b) Random Forest (6 Known, 4 Unknown)

Figure 4.1: Confusion Matrices of Random Forest Model

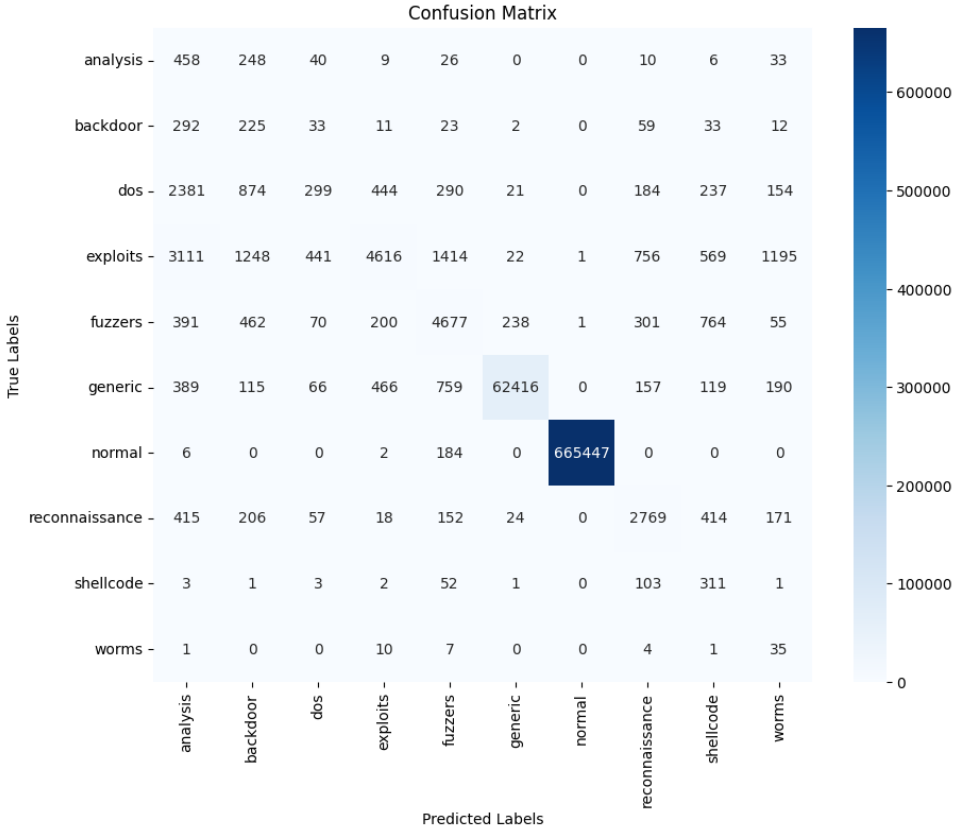


(a) Decision Tree (All Known Classes)

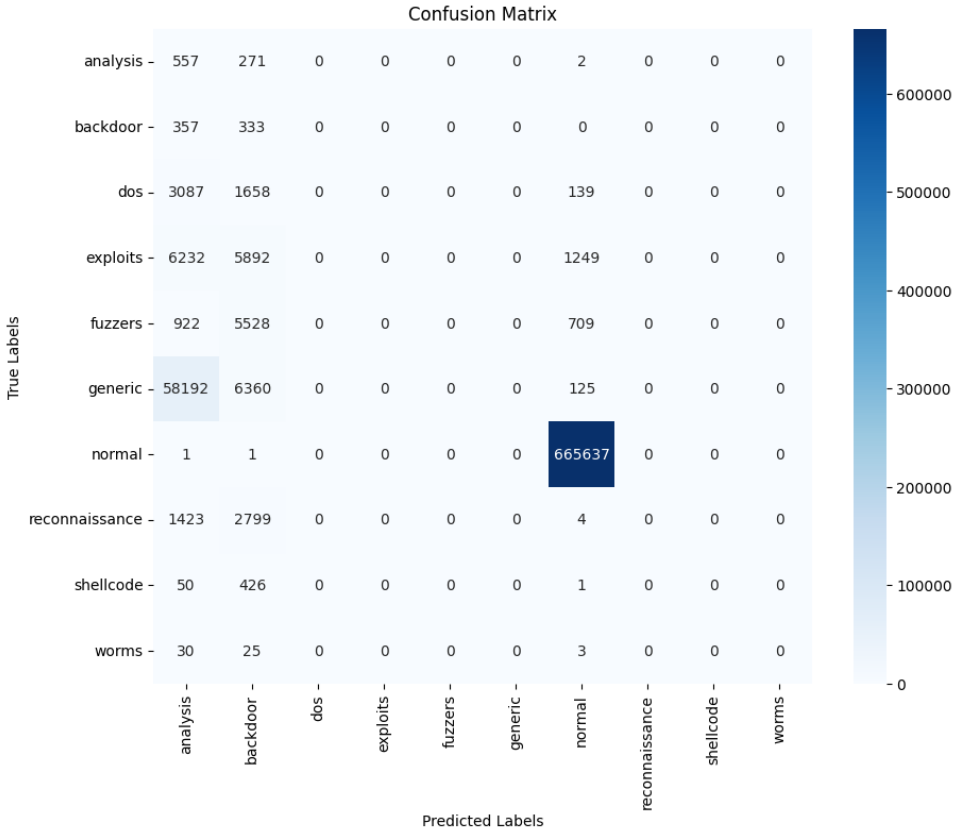


(b) Decision Tree (6 Known, 4 Unknown)

Figure 4.2: Confusion Matrices of Decision Tree Model



(a) FNN (All Known Classes)



(b) FNN (6 Known, 4 Unknown)

Figure 4.3: Confusion Matrices of Feedforward Neural Network (FNN)

4.2 Conventional Anomaly Detection Models: Strategies and Constraints

Thereafter, we explored some traditional models like **Isolation Forest, Autoencoder, One class SVM, Local Outlier Factor and One shot learning** by training the same six known attack classes and testing all ten attack classes, including four previously unseen classes. Here, by Captured Percentage we mean only the unknown captured samples in percentage, true positive is when a test or model correctly identifies a positive case and false positive is when a test or model incorrectly identifies a positive case. The following results were obtained for Conventional Anomaly Detection Models in **Table 4.3** below:

- **Isolation Forest** captured 26,946 instances out of a total of 85,690 points. Among these, 26,946 were classified as true positives out of 42,202 total true points, while rest of 15,256 were identified as false positives.
- **Autoencoder** captured 35,847 instances out of a total of 85,690 points. Among these, 35,847 were classified as true positives out of 47,590 total true points, while rest of 12,095 were identified as false positives.
- **LOF** captured 46,509 instances out of a total of 85,690 points. Among these, 46,509 were classified as true positives out of 42,202 total true points, while rest of 26,522 were identified as false positives.
- **One-Class SVM** captured 30,482 instances out of a total of 85,690 points. Among these, 30,482 were classified as true positives out of 47,590 total true points, while rest of 17,108 were identified as false positives.
- **One-Shot** captured 85,676 instances out of a total of 85,690 points. Among these, 85,676 were classified as true positives out of 2,01,732 total true points, while rest of 1,16,056 were identified as false positives.

Models/Results	Captured Percentage	True Positive	False Positive
Isolation Forest	31.45%	63.58%	36.15%
Autoencoder	41.83%	74.77%	25.23%
One-Class SVM	35.57%	64.05%	35.95%
LOF	54.28%	63.68%	36.32%
One-Shot Learning	99.98%	42.47%	57.53%

Table 4.3: Performance of Conventional Models with Unknown Classes

4.2.1 Constraints

Among the evaluated models, the **One-Shot** Learning model captured percentage is highest at 99.98%. It demonstrates superior capability in identifying potential anomalies. However, The model has three major drawbacks. Firstly, the model cannot identify the normal classes. It is a major drawback for anomaly detection models. Secondly, its high false positive rate of 57.53% significantly undermines its precision, leading to frequent misclassification of non-anomalous instances. Lastly,

the model's dependence on prior anomaly data conflicts with the core research aim of recognizing previously unseen threats effectively as zero-day attacks are novel by nature.

4.3 Unsupervised Approach: Strategies and Constraints

To address these limitations, we developed an unsupervised hybrid algorithm incorporating Cure clustering algorithm, Local Outlier Factor (LOF) and cosine similarity analysis for enhanced network anomaly detection. The algorithm was evaluated through multiple training and testing scenarios. At first, we tried multi class detection where all the classes are present in the train set and no unknown class is there. In this case the unsupervised approach performed really well with an outstanding Captured Percentage result in 89.84% of normal class detection and 99.16% for attack detection. By Captured Percentage we mean only the unknown captured samples in percentage. For all known classes the results in the **Table 4.4** below:

Class Name/Results	**Total**	**Captured**	**Captured Percentage**
Attack	72,936	72,327	99.16%
normal	10,140	9,110	89.84%

Table 4.4: Performance matrices for all known classes

Initially, we have used **Hybrid CURE Algorithm** and **Optimized Filtered Algorithm** where it was trained on six known attack classes and tested on all ten classes, including four previously unseen attack types. To validate the robustness of our approach, we further experimented with seven and eight known attack class combinations. The classes for known and unknown attacks were chosen randomly. The results were analyzed from two perspectives. The first aspect was anomaly detection rate and the second one was normal class detection. This dual evaluation framework allowed us to comprehensively assess both anomaly detection capability and normal class detection. Results for 6 known and 4 unknown classes at **Table 4.5** , 7 known and 3 unknown classes at **Table 4.6** , 8 known and 2 unknown classes at **Table 4.7** showed below respectively:

Anomaly

- For 6 known and 4 unknown combination, it captured 66,013 instances out of a total of 85,690 points. Among these, 66,013 were classified as true positives out of 1,04,469 total true points, while rest of 38,456 were identified as false positives.
- For 7 known and 3 unknown combination, it captured 38,398 instances out of a total of 45,505 points. Among these, 38,398 were classified as true positives out of 1,01,064 total true points, while rest of 62,666 were identified as false positives.

- For 8 known and 2 unknown combination, it captured 25,186 instances out of a total of 27,405 points. Among these, 25,186 were classified as true positives out of 74,734 total true points, while rest of 49,548 were identified as false positives.

Normal

- For 6 known and 4 unknown combination, it captured 15,614 instances out of a total of 18,820 points. Among these, 15,614 were classified as true positives out of 17,521 total true points, while rest of 1,907 were identified as false positives.
- For 7 known and 3 unknown combination, it captured 15,813 instances out of a total of 18,820 points. Among these, 15,813 were classified as true positives out of 18,435 total true points, while rest of 2,622 were identified as false positives.
- For 8 known and 2 unknown combination, it captured 20,437 instances out of a total of 24,567 points. Among these, 20,437 were classified as true positives out of 21,450 total true points, while rest of 1,013 were identified as false positives.

Class Name/Results	**Captured Percentage**	**True Positive**	**False Positive**
Anomaly	77.04%	63.19%	36.81%
normal	82.96%	89.12%	10.88%

Table 4.5: Performance metrics for 6 known and 4 unknown (Hybrid Algorithm)

Class Name/Results	**Captured Percentage**	**True Positive**	**False Positive**
Anomaly	84.38%	37.99%	62.01%
normal	84.02%	85.78%	14.22%

Table 4.6: Performance metrics for 7 known and 3 unknown (Hybrid Algorithm)

Class Name/Results	**Captured Percentage**	**True Positive**	**False Positive**
Anomaly	91.9%	33.7%	66.3%
normal	83.18%	95.27%	4.73%

Table 4.7: Performance metrics for 8 known and 2 unknown (Hybrid Algorithm)

Overall, the unsupervised approach excels in identifying the normal class, demonstrating a higher Captured Percentage with true positives consistently ranging from 82.78% to 95.27% across all scenarios, showcasing its robust capability to accurately recognize benign network traffic. By Captured Percentage we mean only the unknown captured samples in percentage, This consistent performance highlights the algorithm's strength in distinguishing normal activity from potential threats, even as the number of unknown attack classes varies. Its reliability in this aspect provides a solid foundation for practical deployment in network security applications.

4.3.1 Constraints

Figure 4.4 and Figure 4.5 demonstrate substantial feature range overlap between multiple feature pairs in both the training and test sets following PCA transformation. The training set exhibits a maximum overlap of 99.7%, while the test set shows a high overlap of 97.4%, indicating severely limited feature separability. This extensive PCA feature range overlap directly translates into significant cluster overlapping among the known attack classes during the initial clustering phase of the proposed approach. When the clustering algorithm tries to form separate groups for each class, the feature ranges often overlap. This causes the clusters to mix together and become hard to separate. As a result, the algorithm cannot clearly tell the difference between normal and attack patterns. This overlap leads to more false positives in detecting anomalies. When the overlap is severe, it's very hard to tell known attacks apart or detect new unknown attacks. The model gets confused between known and actual unknown behavior. The overlapping issues for the training set are illustrated in Figure 4.4, while the test set overlapping patterns are shown in Figure 4.5 and the cluster overlaps among the training set, which consists of six clusters, are visualized in Figure 4.6

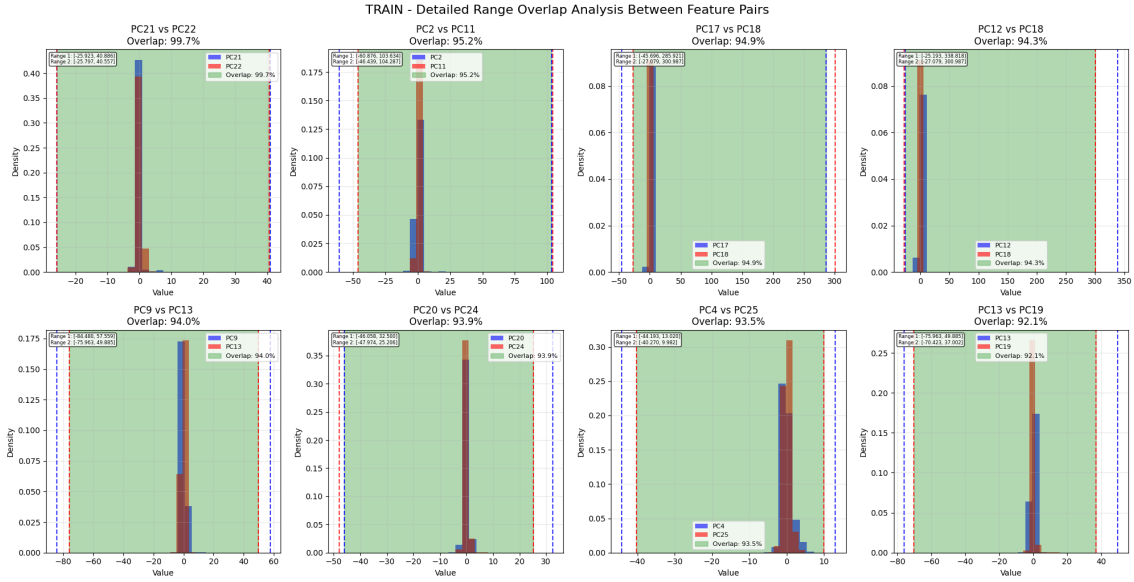


Figure 4.4: Range Overlapping in Features for Train Set

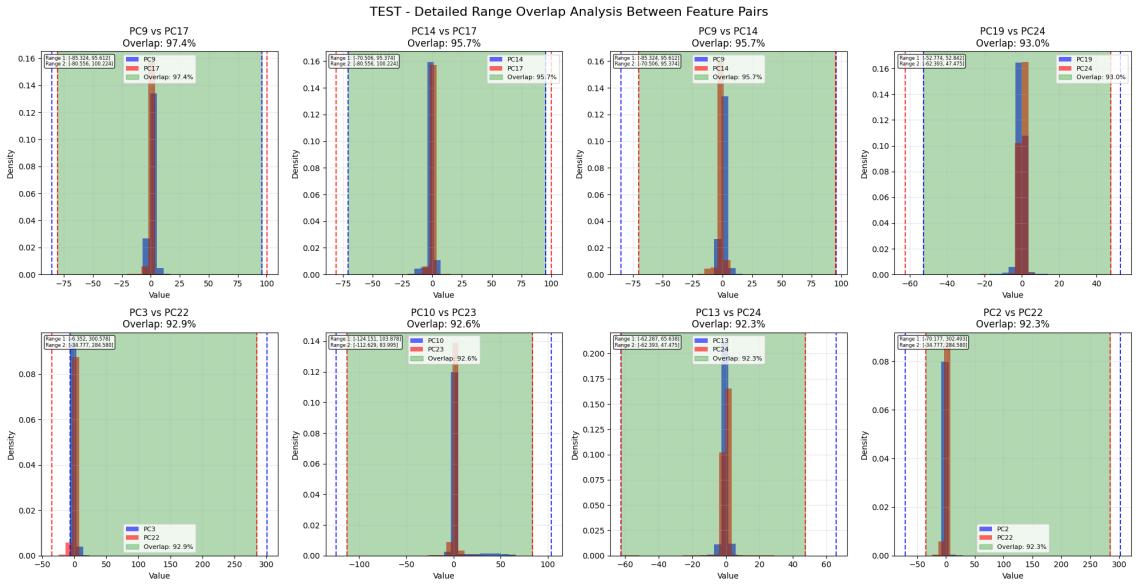


Figure 4.5: Range Overlapping in Features for Test Set

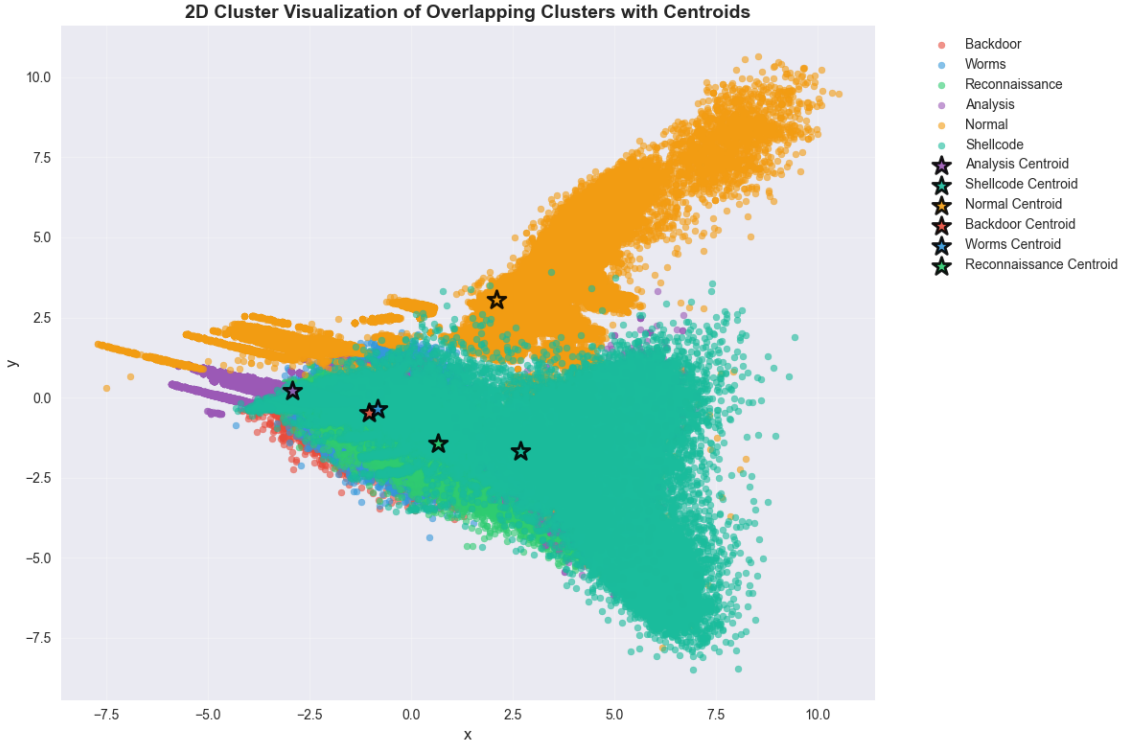


Figure 4.6: Overlapping Clusters

To address the overlapping cluster issue observed in both the train and test sets, we implemented a filtering strategy to ensure class separation and improve the purity of the clusters. Specifically, we removed data points from the training set that exhibited ambiguous boundaries between "known" and "unknown" classes, particularly those near the cluster edges where feature similarities could lead to misclassification. This was done by examining cluster radius and neighborhood densities using LOF and cosine similarity. By excluding overlapping points, we ensured that the

"known" classes used for training were better defined, which significantly reduced false positives in the test results. The revised dataset allowed the hybrid algorithm to more accurately distinguish between "normal", "known" and "unknown".

To substantiate our claims, we performed further experiments with the algorithm across various combinations, resolving the issue of cluster overlap. The overlapping clusters were mitigated through enhanced data preprocessing techniques. Subsequent simulations of the combination scenarios demonstrated improved model performance.

Anomaly

- For 6 known and 4 unknown combination, it captured 83,678 instances out of a total of 85,690 points. Among these, 83,678 were classified as true positives out of 93,313 total true points, while rest of 9,635 were identified as false positives.
- For 7 known and 3 unknown combination, it captured 41,435 instances out of a total of 45,505 points. Among these, 41,435 were classified as true positives out of 46,341 total true points, while rest of 4,907 were identified as false positives.
- For 8 known and 2 unknown combination, it captured 26,724 instances out of a total of 27,405 points. Among these, 26,724 were classified as true positives out of 29,723 total true points, while rest of 2,999 were identified as false positives.

Normal

- For 6 known and 4 unknown combination, it captured 21,339 instances out of a total of 24,331 points. Among these, 21,339 were classified as true positives out of 22,559 total true points, while rest of 1,220 were identified as false positives.
- For 7 known and 3 unknown combination, it captured 18,350 instances out of a total of 19,922 points. Among these, 18,350 were classified as true positives out of 19,050 total true points, while rest of 700 were identified as false positives.
- For 8 known and 2 unknown combination, it captured 15,327 instances out of a total of 17,311 points. Among these, 15,327 were classified as true positives out of 15,500 total true points, while rest of 173 were identified as false positives.

Class Name/Results	Captured Percentage	True Positive	False Positive
Anomaly	97.65%	89.67%	10.33%
normal	87.7%	94.59%	5.41%

Table 4.8: Performance without overlapping classes for 6 known and 4 unknown

Class Name/Results	Captured Percentage	True Positive	False Positive
Anomaly	91.06%	89.41%	10.59%
normal	92.1%	96.33%	3.67%

Table 4.9: Performance without overlapping classes for 7 known and 3 unknown

Class Name/Results	Captured Percentage	True Positive	False Positive
Anomaly	97.52%	89.91%	10.09%
normal	88.53%	98.89%	1.11%

Table 4.10: Performance without overlapping classes for 8 known and 2 unknown

To resolve the cluster overlap issue we had to remove some highly overlapping classes("backdoor", "analysis", "reconnaissance") from the dataset to visualize our hybrid algorithm performance without the overlapping classes. Then the unsupervised approach demonstrated significantly enhanced performance across scenarios with 6 known and 4 unknown classes at **Table 4.8**, 7 known and 3 unknown at **Table 4.9** and 8 known and 2 unknown classes at **Table 4.10**. For unknown attack classes, the algorithm achieved Captured Percentages from 91.06% to 97.65%, true positive rates between 89.41% and 89.91% and false positive rates from 10.08% to 10.59%. For the normal class, Captured Percentages ranged from 87.70% to 92.10%, with true positive rates up to 98.88% and false positive rates as low as 1.11%. By Captured Percentage we mean only the unknown captured samples in percentage. These results show that the algorithm is now better at detecting attack class apart from normal class. The very low number of false positives proves it can be trusted for real world cybersecurity use.

4.4 Comparison of Results with Existing Works

Confusion Matrix		Predicted Category	
		Known Attacks	**Unknown Attacks**
Actual Category	**Known Attacks**	TN	FP
	Unknown Attacks	FN	TP

Table 4.11: Evaluation metrics

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4.2}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4.3}$$

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.4}$$

The **Table 4.11** gives a detailed performance evaluation of the suggested cluster-based unsupervised technique in comparison to established baselines and recent research in cybersecurity attack detection. The proposed methodology uses an unsupervised clustering technique in which performance metrics are manually calculated from clustering results by determining true positives **(TP)**, true negatives **(TN)**, false positives **(FP)** and false negatives **(FN)**, allowing for precise computation of all performance measures. At first, the captured percentage was used as the main evaluation metric. Later analysis showed that it matched the recall formula, confirming it follows standard machine learning evaluation methods. The results show that the proposed approach performs effectively in both multiclass attack classification and the identification of unknown attacks.

Approaches/Results	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Pal et al. [39]	78.16%	84%	78%	79%
Zhang et al. [31]	–	76.4%	76.8%	–
Mebawondu et al. [36]	79%	63.6%	63.8%	61.2%
Zhang et al. [26]	88.60%	88.58%	88.42%	88.03%
Proposed Approach	93.02%	97%	95.9%	96.5%

Table 4.12: Comparative analysis of Existing Works and Proposed Approach for Multiclass Detection

In the multiclass detection evaluation **Table 4.12**, the proposed approach shows strong performance achieving 93.02% accuracy, 97% precision, 95.9% recall and a 96.5% f1-score. Compared to earlier studies, it marks a clear improvement. Pal et al., [39](2025) reached 78.16% accuracy with 84% precision and 78% recall, resulting in a 79% f1-score. Zhang et al., [31](2023) reported 76.4% precision and 76.8% recall. Mebawondu et al., [36](2024) achieved 79% accuracy but had lower precision 63.6% and recall 63.8% with f1-score of 61.2%. Another study by Zhang et al., [26](2022) reported improved results with 88.60% accuracy, 88.58% precision, 88.42% recall and 88.03% f1-score. However, these figures are still notably lower than those achieved by the proposed approach. Moreover, as an unsupervised learning technique, the proposed approach is inherently more flexible, making it suitable for diverse and evolving threat landscapes.

Approaches/Results	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Akata et al. [2]	77.1%	59.34%	68.61%	61.41%
Kodirov et al. [6]	51.5%	44.24%	43.87%	34.55%
Gao et al. [32]	93.6%	94.3%	86.01%	89.42%
Proposed Approach (Overlapping)	72.83%	63.19%	77.04%	69.43%
Proposed Approach (Non-Overlapping)	94.55%	89.67%	97.65%	93.49%

Table 4.13: Comparative analysis of Existing Works and Proposed Approach for Unknown Attack Detection

The proposed cluster-based unsupervised method for unknown attack detection **Table 4.13** addresses the important challenge of identifying unknown attacks. The proposed approach evaluation involved 4 unknown class attacks. It means 6 known attack classes were used during training and while testing was conducted on 10

total classes. It works in both overlapping and non-overlapping cases. It focuses on improving the detection rate of previously unseen threats. The recall metric matches exactly with the Captured Percentage formula. In this work, "Captured Percentage" is used to highlight the focus on identifying unknown attacks. By manually calculating TP, TN, FP and FN from the clustering results, the metrics were measured accurately. Special attention was given to recall as it is the most important metric for finding unknown attacks. In the overlapping scenario, where attack patterns share similar features, the proposed approach achieved 72.83% accuracy, 63.19% precision, 77.04% recall and 69.43% f1-score. These results reflect the difficulty caused by overlapping patterns while still showing a decent ability to detect attacks. In contrast, the non-overlapping scenario performs much better, with 94.55% accuracy, 89.67% precision, 97.65% recall and 93.49% f1-score. This shows the approach's strong ability to detect unknown attacks with very few false negatives. The high recall of 97.65% shows that the proposed method can successfully detect almost all unknown attack classes. This is important in cybersecurity where missing an unknown attack can lead to serious consequences. Compared to earlier studies, the non-overlapping scenario of the proposed approach performs much better.

In the existing works, they worked on 8 known classes and tried to test on 1 new attack including 8 known classes, total 9 classes. They have removed the class "worms" because of its low number of samples compared to the other attack types. Akata et al., [2](2013) reached a recall of 68.61%, Kodirov et al., [6](2017) achieved 43.87% and Gao et al., [32](2024) reported 86.01%. In contrast, the 97.65% detection rate of the proposed approach marks a clear improvement, offering more reliable unknown attack detection while keeping precision at an acceptable level. This performance of the proposed approach is very important in cybersecurity where missing an unknown attack can be far more damaging than dealing with a few false positives. As a result, the proposed approach is highly suitable for practical and real world use.

Chapter 5

Conclusion

In this work, we explored supervised, unsupervised and hybrid approaches for network anomaly detection using the UNSW-NB15 dataset. The dataset was extensively preprocessed through cleaning, class balancing using CTGAN and feature engineering to improve the accuracy and robustness of detection models. The supervised models such as Random Forest, Decision Tree and Feedforward Neural Network performed well on known attack types. But their effectiveness dropped significantly when faced with unknown or unseen anomalies in the test set. The unsupervised models such as Isolation Forest and LOF offered limited precision due to struggles with the complex structure of the data. One Shot Learning gave good results but it needs prior label information. Then the proposed hybrid approach integrating a CURE-based clustering algorithm, achieved a detection rate of up to 91.9% for unknown anomalies and for known class scenarios up to 99.16% detection rate is obtained significantly outperforming conventional methods in open world scenarios. During evaluation, we also encountered overlapping issues with certain attack types. Despite this challenge, our findings confirm that this hybrid learning strategy provides a more flexible, adaptive and effective solution for anomaly detection in dynamic cybersecurity environments.

One immediate avenue we aim to explore is enhancing the CURE based clustering algorithm at the core of our detection system. While we have already integrated LOF and cosine similarity to refine anomaly boundaries, we plan to introduce dynamic determination of the number of clusters to make the algorithm more adaptable to varying data distributions. We also intend to implement adaptive shrinkage based on intra-cluster variance which will allow the model to better fit clusters of differing densities. Additionally, we will investigate the incorporation of density based metrics to improve the detection of irregular or subtle attack behaviors. During our current evaluation, we faced overlapping issues between some clusters which we temporarily resolved by removing overlapping classes to achieve the best results. However, this is not an ideal practice. In the future, we aim to develop mechanisms to effectively mitigate overlapping without excluding valuable data. Looking forward, we are interested in integrating attention mechanisms or reinforcement learning to enable the CURE algorithm to dynamically prioritize high risk regions within complex network environments. Furthermore, future work will focus on reducing computational costs by implementing parallel processing for large scale datasets and optimizing the Ball-Tree and MiniBatchKMeans algorithms for faster convergence. We will also explore

adaptive parameter tuning for the shrinkage factor (alpha) and outlier thresholds to further enhance clustering efficiency and accuracy.

Bibliography

[1] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009, pp. 1–6. DOI: 10.1109/CISDA.2009.5356528.

[2] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid, "Label-embedding for attribute-based classification," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 819–826. DOI: 10.1109/CVPR.2013.111.

[3] Y. Guerbai, Y. Chibani, and B. Hadjadji, "The effective use of the one-class svm classifier for reduced training samples and its application to handwritten signature verification," in *2014 International Conference on Multimedia Computing and Systems (ICMCS)*, 2014, pp. 362–366. DOI: 10.1109/ICMCS.2014.6911221.

[4] M. A. T. Laksono, Y. Purwanto, and A. Novianty, "Ddos detection using cure clustering algorithm with outlier removal clustering for handling outliers," in *2015 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC)*, 2015, pp. 12–18. DOI: 10.1109/ICCEREC.2015.7337029.

[5] Z. Li and L. Zeng, "A hybrid vertex outlier detection method based on distributed representation and local outlier factor," in *2015 IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom)*, 2015, pp. 512–516. DOI: 10.1109/UIC-ATC-ScalCom-CBDCom-IoP.2015.104.

[6] E. Kodirov, T. Xiang, and S. Gong, "Semantic autoencoder for zero-shot learning," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4447–4456. DOI: 10.1109/CVPR.2017.473.

[7] J.-N. Lee and K.-C. Kwak, "A performance comparison of auto-encoder and its variants for classification," in *2017 International Conference on Signals and Systems (ICSigSys)*, 2017, pp. 207–211. DOI: 10.1109/ICSIGSYS.2017.7967042.

[8] G. Mohi-ud-din, *Nsl-kdd*, 2018. DOI: 10.21227/425a-3e55. [Online]. Available: https://dx.doi.org/10.21227/425a-3e55.

[9] E. H. Budiarto, A. Erna Permanasari, and S. Fauziati, "Unsupervised anomaly detection using k-means, local outlier factor and one class svm," in *2019 5th International Conference on Science and Technology (ICST)*, vol. 1, 2019, pp. 1–5. DOI: 10.1109/ICST47872.2019.9166366.

[10] L. Zhang, H. Du, and Y. Zhang, "Network anomaly detection based on cooperative semi-supervised support vector machine," in *2019 International Conference on Networking and Network Applications (NaNA)*, 2019, pp. 171–180. DOI: 10.1109/NaNA.2019.00039.

[11] I. Sarker, Y. Abushark, F. Alsolami, and A. Khan, "Intrudtree: A machine learning based cyber security intrusion detection model," *Symmetry*, vol. 12, no. 5, p. 754, 2020. DOI: 10.3390/sym12050754.

[12] K. Shaukat, S. Luo, S. Chen, and D. Liu, "Cyber threat detection using machine learning techniques: A performance evaluation perspective," in *International Conference on Cyber Warfare and Security (ICCWS)*, 2020, pp. 1–6. DOI: 10.1109/ICCWS48432.2020.9292388.

[13] H. Zhao *et al.*, "Multivariate time-series anomaly detection via graph attention network," in *IEEE International Conference on Data Mining (ICDM)*, 2020, pp. 841–850. DOI: 10.1109/ICDM50108.2020.00093.

[14] N. I. Ahmad Sabri and S. Setumin, "One-shot learning for facial sketch recognition using the siamese convolutional neural network," in *2021 IEEE 11th IEEE Symposium on Computer Applications Industrial Electronics (ISCAIE)*, 2021, pp. 307–312. DOI: 10.1109/ISCAIE51753.2021.9431773.

[15] G. Kowadlo, A. Ahmed, and D. Rawlinson, "One-shot learning for the long term: Consolidation with an artificial hippocampal algorithm," in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021, pp. 1–7. DOI: 10.1109/IJCNN52387.2021.9534193.

[16] B. Min, J. Yoo, S. Kim, D. Shin, and D. Shin, "Network anomaly detection using memory-augmented deep autoencoder," *IEEE Access*, vol. 9, pp. 104 695–104 706, 2021. DOI: 10.1109/access.2021.3100087.

[17] G. Pu, L. Wang, J. Shen, and F. Dong, "A hybrid unsupervised clustering-based anomaly detection method," *Tsinghua Science and Technology*, vol. 26, no. 2, pp. 146–153, 2021. DOI: 10.26599/TST.2019.9010051.

[18] K. Qin, Y. Zhou, B. Tian, and R. Wang, "Attentionae: Autoencoder for anomaly detection in attributed networks," in *IEEE International Conference on Natural Language Analysis and the New Artificial Intelligence*, 2021. DOI: 10.1109/nana53684.2021.00089.

[19] A. Z. Alalmaie, P. Nanda, and X. He, "Zero trust-nids: Extended multi-view approach for network trace anonymization and auto-encoder cnn for network intrusion detection," in *IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 2022, pp. 449–456. DOI: 10.1109/TrustCom56396.2022.00069.

[20] H. Anand, B. S. Sammuli, K. E. J. Olofsson, and D. A. Humphreys, "Real-time magnetic sensor anomaly detection using autoencoder neural networks on the diii-d tokamak," *IEEE Transactions on Plasma Science*, vol. 50, no. 11, pp. 4126–4130, 2022. DOI: 10.1109/tps.2022.3181548.

[21] C. Islam, M. A. Babar, R. Croft, and H. Janicke, "Smartvalidator: A framework for automatic identification and classification of cyber threat data," *Journal of Network and Computer Applications*, 2022. DOI: 10.1016/j.jnca.2022.103370.

[22] S. Al-Jeshi, A. Tarfa, H. Al-Aswad, W. Elmedany, and C. Balakrishna, "A blockchain enabled system for enhancing fintech industry of the core banking systems," in *International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, 2022, pp. 209–213. DOI: 10.1109/3ICT56508.2022.9990875.

[23] Z. Shu, "Analysis on ordered weighted averaging operators in different types and applications for decision making," in *2022 7th International Conference on Intelligent Computing and Signal Processing (ICSP)*, 2022, pp. 353–359. DOI: 10.1109/ICSP54964.2022.9778323.

[24] X. Wang and X. Qiu, "A novel semi-supervised anomaly detection method for network intrusion detection," in *2022 IEEE 22nd International Conference on Communication Technology (ICCT)*, 2022, pp. 1276–1280. DOI: 10.1109/ICCT56141.2022.10073098.

[25] M. Wazid, A. Das, V. Chamola, and Y. Park, "Uniting cyber security and machine learning: Advantages, challenges and future research," *ICT Express*, vol. 8, no. 3, 2022. DOI: 10.1016/j.icte.2022.04.007.

[26] Y. Zhang, Y. Gandhi, Z. Li, and Z. Xiao, "Improving the classification effectiveness of network intrusion detection using ensemble machine learning techniques and deep neural networks," in *2022 International Conference on Intelligent Data Science Technologies and Applications (IDSTA)*, 2022, pp. 117–123. DOI: 10.1109/IDSTA55301.2022.9923205.

[27] J. Bharadiya, "Machine learning in cybersecurity: Techniques and challenges," *European Journal of Technology*, vol. 7, no. 2, pp. 1–14, 2023. DOI: 10.47672/ejt.1486.

[28] Y. E. Tadesse and Y.-J. Choi, *Cse-cic-ids2018 and nslkdd image dataset*, 2023. DOI: 10.21227/acha-tc06. [Online]. Available: https://dx.doi.org/10.21227/acha-tc06.

[29] x. chen xinpeng, *Cicids2017 and unbsw-nb15*, 2023. DOI: 10.21227/ykpn-jx78. [Online]. Available: https://dx.doi.org/10.21227/ykpn-jx78.

[30] H. Zhang, N. Kumar, S. Wu, C. Wu, J. Wang, and P. Zhang, "Anomaly detection with memory-augmented adversarial autoencoder networks for industry 5.0," *IEEE Transactions on Consumer Electronics*, 2023. DOI: 10.1109/tce.2023.3319131.

[31] H. Zhang, Z. Xiao, J. Gu, *et al.*, "A network anomaly detection algorithm based on semi-supervised learning and adaptive multiclass balancing," *The Journal of Supercomputing*, vol. 79, pp. 20 445–20 480, 2023. DOI: 10.1007/s11227-023-05474-y.

[32] X. Gao, K. Chen, Y. Zhao, P. Zhang, L. Han, and D. Zhang, "A zero-shot learning-based detection model against zero-day attacks in iot," in *2024 9th International Conference on Electronic Technology and Information Science (ICETIS)*, 2024, pp. 309–314. DOI: 10.1109/ICETIS61828.2024.10593684.

[33] S. Gunupusala and S. Kaila, "Multi-class network anomaly detection using machine learning techniques," *Contemporary Mathematics*, pp. 5–22, Jun. 2024. DOI: 10.37256/cm.5220243723.

[34] T. Ige, C. Kiekintveld, A. Piplai, A. Wagler, O. Kolade, and B. Hafiz Matti, "An in-depth investigation into the performance of state-of-the-art zero-shot, single-shot, and few-shot learning approaches on an out-of-distribution zero-day malware attack detection," in *2024 International Symposium on Networks, Computers and Communications (ISNCC)*, 2024, pp. 1–6. DOI: 10.1109/ISNCC62547.2024.10758952.

[35] T. Li, "Optimization of algorithm for network traffic anomaly detection using convolutional neural networks (cnn)," in *2024 International Conference on Intelligent Algorithms for Computational Intelligence Systems (IACIS)*, 2024, pp. 1–6. DOI: 10.1109/IACIS61494.2024.10721912.

[36] O. J. Mebawondu, B. Ajisafe-Badeji, J. O. Mebawondu, O. C. Akinduyite, O. B. Abiola, and T. G. Oluwatoki, "A multi-class intrusion detection model using an ensemble of deep learning techniques," in *2024 IEEE 5th International Conference on Electro-Computing Technologies for Humanity (NIGERCON)*, 2024, pp. 1–5. DOI: 10.1109/NIGERCON62786.2024.10927048.

[37] K. Saurabh, U. Singh, A. Mishra, R. Vyas, and O. Vyas, "Zero-shot based hybrid neural network system for enhancing zero-day attack detection," in *2024 IEEE 21st India Council International Conference (INDICON)*, 2024, pp. 1–6. DOI: 10.1109/INDICON63790.2024.10958486.

[38] R. Sharma and M. Grover, "Enhancing cybersecurity with machine learning: Evaluating the efficacy of isolation forests and autoencoders in anomaly detection," in *2024 7th International Conference on Circuit Power and Computing Technologies (ICCPCT)*, vol. 1, 2024, pp. 1017–1021. DOI: 10.1109/ICCPCT61902.2024.10673338.

[39] K. K. Pal, A. V. Eriksen, and N. Dinh, "Xgboost feature selection for multi-class and binary classification on unsw-nb15 dataset," in *2025 IEEE International Conference on Consumer Electronics (ICCE)*, 2025, pp. 1–6. DOI: 10.1109/ICCE63647.2025.10930023.

[40] M. Rathakrishnan, S. Gayan, S. Edirisinghe, and H. Inaltekin, "A multi-model framework for synthesizing high-fidelity network intrusion data using generative ai," in *2025 5th International Conference on Advanced Research in Computing (ICARC)*, 2025, pp. 1–6. DOI: 10.1109/ICARC64760.2025.10963129.